Towards Optimal Discrete Online Hashing with Balanced Similarity

Mingbao Lin,[1] **Rongrong Ji,**[1,2*] **Hong Liu,**[1] **Xiaoshuai Sun,**[1] **Yongjian Wu,**[3] **Yunsheng Wu**[3]
[1]Fujian Key Laboratory of Sensing and Computing for Smart City, Department of Cognitive Science,
School of Information Science and Engineering, Xiamen University, China
[2] Peng Cheng Laboratory, China
[3]Tencent Youtu Lab, Tencent Technology (Shanghai) Co., Ltd, China
lmbxmu@stu.xmu.edu.cn, rrji@xmu.edu.cn, lynnliu.xmu@gmail.com,
xiaoshuaisun.hit@gmail.com, {littlekenwu, simonwu}@tencent.com

Abstract

When facing large-scale image datasets, online hashing serves as a promising solution for online retrieval and prediction tasks. It encodes the online streaming data into compact binary codes, and simultaneously updates the hash functions to renew codes of the existing dataset. To this end, the existing methods update hash functions solely based on the new data batch, without investigating the correlation between such new data and the existing dataset. In addition, existing works update the hash functions using a relaxation process in its corresponding approximated continuous space. And it remains as an open problem to directly apply discrete optimizations in online hashing. In this paper, we propose a novel supervised online hashing method, termed **B**alanced **S**imilarity for **O**nline **D**iscrete **H**ashing (BSODH), to solve the above problems in a unified framework. BSODH employs a well-designed hashing algorithm to preserve the similarity between the streaming data and the existing dataset via an asymmetric graph regularization. We further identify the "data-imbalance" problem brought by the constructed asymmetric graph, which restricts the application of discrete optimization in our problem. Therefore, a novel *balanced similarity* is further proposed, which uses two equilibrium factors to balance the similar and dissimilar weights and eventually enables the usage of discrete optimizations. Extensive experiments conducted on three widely-used benchmarks demonstrate the advantages of the proposed method over the state-of-the-art methods.

Introduction

With the increasing amount of image data available on the Internet, hashing has been widely applied to approximate nearest neighbor (ANN) search (Wang et al. 2016; 2018). It aims at mapping real-valued image features to compact binary codes, which merits in both low storage and efficient computation on large-scale datasets. One promising direction is online hashing (OH), which has attracted increasing attentions recently. Under such an application scenario, data are often fed into the system via a streaming fashion, while traditional hashing methods can hardly accommodate this configuration. In OH, the online streaming data is encoded into compact binary codes, while the hash functions are simultaneously updated in order to renew codes of the existing data.

In principle, OH aims to analyze the streaming data while preserving structure of the existing dataset[1]. In the literature, several recent works have been proposed to handle OH. The representative works include, but not limited to, OKH (Huang, Yang, and Zheng 2013), SketchHash (Leng et al. 2015), AdaptHash (Fatih and Sclaroff 2015), OSH (Fatih, Bargal, and Sclaroff 2017), FROSH (Chen, King, and Lyu 2017) and MIHash (Fatih et al. 2017). However, the performance of OH is still far from satisfactory for real-world applications. We attribute it to two open issues, *i.e.*, *updating imbalance* and *optimization inefficiency*.

In terms of the updating imbalance, the existing OH schemes update hash functions solely based on the newly coming data batch, without investigating the correlation between such new data and the existing dataset. To that effect, an asymmetric graph can be constructed to preserve similarity between the new data and the existing dataset as shown in Fig.1. Under online setting, the similarity matrix is usually sparse and unbalanced, *i.e.*, data-imbalance phenomenon, since most image pairs are dissimilar and only a few are similar. The updating imbalance issue, if not well addressed, might cause the learned binary codes ineffective for both the new data and the existing data, and hence lead to severe performance degeneration for OH schemes.

In terms of the optimization inefficiency, the existing OH schemes still rely on the traditional relaxation (Gong and Lazebnik 2011; Datar et al. 2004; Jiang and Li 2015; Liu et al. 2018; Lin et al. 2018) over the approximated continuous space to learn hash functions, which often makes the produced hash functions less effective, especially when the code length increases (Liu et al. 2014; Shen et al. 2015b). Despite the recent advances in direct discrete optimizations in offline hashing (Ji et al. 2017; Jiang and Li 2018) with discrete cyclic coordinate descent (DCC) (Shen et al. 2015b), such discrete optimizations can not be directly applied to online case that contains serious data-imbalance problem, since the optimization heavily relies on the dissimilar pairs, and thus lose the information of similar pairs.

[*]Corresponding Author.

[1]The streaming data is usually in a small batch, which can be processed easily to pursue a better tradeoff among computation, storage, and accuracy.

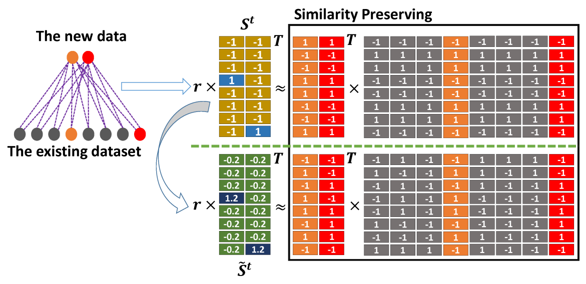

Figure 1: An example of data-imbalance problem and the learned binary codes. The similarity matrix $\mathbf{S}^t$ is highly sparse under online setting and thus tends to generate consistent binary codes, which are indiscriminate and uninformative. With the introduction of the balanced similarity $\tilde{\mathbf{S}}^t$, codes of similar items are tightened while codes of dissimilar items are expanded. By combining with discrete optimizations, advanced retrieval results are obtained.

We argue that, the above two issues are not independent. In particular, to conduct discrete optimizations, the existing offline methods typically adopt an asymmetric graph regularization to preserve the similarity between training data. Constructing the asymmetric graph consumes both time and memory. Note that, since the streaming data is in a small batch, such an asymmetric graph between the streaming data and the existing dataset can be dynamically constructed under online setting. However, as verified both theoretically and experimentally later, it still can not avoid the generation of consistent codes (most bits are the same) due to the data-imbalance problem brought by the constructed asymmetric graph in online learning, as illustrated in Fig.1.

In this paper, we propose a novel supervised OH method, termed Balanced Similarity for Online Discrete Hashing (BSODH) to handle the updating imbalance and optimization inefficiency problems in a unified framework. First, unlike the previous OH schemes, the proposed BSODH mainly considers updating the hash functions with correlation between the online streaming data and the existing dataset. Therefore, we aim to adopt an asymmetric graph regularization to preserve the relation in the produced Hamming space. Second, we further integrate the discrete optimizations into OH, which essentially tackles the challenge of quantization error brought by the relaxation learning. Finally, we present a new similarity measurement, termed *balanced similarity*, to solve the problem of data-imbalance during the discrete binary learning process. In particular, we introduce two equilibrium factors to balance the weights of similar and dissimilar data, and thus enable the discrete optimizations. Extensive experimental results on three widely-used benchmarks, *i.e.*, CIFAR10, Places205 and MNIST, demonstrate the advantages of the proposed BSODH over the state-of-the-art methods.

To summarize, the main contributions of the proposed BSODH in this paper include:

- To capture the data correlation between online streaming data and the existing dataset, we introduce an asymmetric graph regularization to preserve such correlation in the produced Hamming space.
- To reduce the quantization error in the Hamming space, we design a customized discrete optimization algorithm. It handles the optimization inefficiency issue in the existing OH scheme, making discrete learning feasible for the first time in the online framework.
- We propose a balanced similarity matrix to handle the data-imbalance problem, which further prevents the generation of consistent binary codes, *i.e.*, a phenomenon that previously occurred when directly applying discrete optimizations in online setting.

Related Work

In this section, we briefly review the existing OH methods. OH merits in efficiently updating the hash functions by using the streaming data online, which can be further subdivided into two categories: SGD-based OH methods, and matrix sketch-based OH methods.

For SGD-based methods, Online Kernel Hashing (OKH) (Huang, Yang, and Zheng 2013) is the first attempt to learn hash functions via an online passive-aggressive strategy (Crammer et al. 2006), which updates hash functions to retain important information while embracing information from new pairwise input. Adaptive Hashing (AdaptHash) (Fatih and Sclaroff 2015) adopts a hinge loss to decide which hash function to be updated. Similar to OKH, labels of pairwise similarity are needed for AdaptHash. Inspired by Error Correcting Output Codes (ECOCs) (Dietterich and Bakiri 1995), Online Supervised Hashing (OSH) (Fatih, Bargal, and Sclaroff 2017) adopts a more general two-step hash learning framework, where each class is firstly deployed with a vector from ECOCs, and then an convex function is further exploited to replace the $0/1$ loss. In (Fatih et al. 2017), an OH with Mutual Information (MIHash) is developed which targets at optimizing the mutual information between neighbors and non-neighbors.

Motivated by the idea of "data sketching" (Clarkson and Woodruff 2009), skech-based methods provide a good alternative for unsupervised online binary coding, via which a large dataset is summarized by a much smaller data batch. Leng *et al.* proposed the Online Sketching Hashing (SketchHash) (Leng et al. 2015), which adopts an efficient variant of SVD decomposition to learn hash functions. More recently, Subsampled Randomized Hadamard Transform (SRHT) is adopted in FasteR Online Sketching Hashing (FROSH) (Chen, King, and Lyu 2017) to accelerate the training process of SketchHash.

However, existing sketch-based algorithms are based on unsupervised learning, and their retrieval performance is mostly unsatisfactory without fully utilizing label information. Although most SGD-based algorithms aim to preserve the label information via online hash function learning, the relaxation process is adopted to update the hash functions, which contradicts with the recent advances in offline hashing where discrete optimizations are adopted directly, such as Discrete Graph Hashing (Liu et al. 2014) and Discrete

Supervised Hashing (Shen et al. 2015b). In this paper, we are the first to investigate OH with discrete optimizations, which have shown superior performance compared with the quantization-based schemes.

The Proposed Method

Problem Definition

Given a dataset $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_n] \in \mathbb{R}^{d\times n}$ with its corresponding labels $\mathbf{L} = [l_1, ..., l_n] \in \mathbb{N}^n$, where $\mathbf{x}_i \in \mathbb{R}^d$ is the i-th instance with its class label $l_i \in \mathbb{N}$. The goal of hashing is to learn a set of k-bit binary codes $\mathbf{B} = [\mathbf{b}_1, ..., \mathbf{b}_n] \in \{-1, +1\}^{k\times n}$, where $\mathbf{b}_i$ is the binary vector of $\mathbf{x}_i$. A widely-adopted hash function is the linear hash mapping (Gong and Lazebnik 2011; Fatih, Bargal, and Sclaroff 2017), *i.e.*,

$$\mathbf{B} = F(\mathbf{X}) = \mathrm{sgn}(\mathbf{W}^T\mathbf{X}), \tag{1}$$

where $\mathbf{W} = [\mathbf{w}_1, ..., \mathbf{w}_k] \in \mathbb{R}^{d\times k}$ is the projection matrix to be learned with $\mathbf{w}_i$ being responsible for the i-th hash bit. The sign function $\mathrm{sgn}(x)$ returns $+1$ if input variable $x > 0$, and returns -1 otherwise.

For the online learning problem, the data is coming in a streaming fashion. Therefore $\mathbf{X}$ is not available once for all. Without loss of generality, we denote $\mathbf{X}_s^t = [\mathbf{x}_{s1}^t, ..., \mathbf{x}_{sn_t}^t] \in \mathbb{R}^{d\times n_t}$ as the input streaming data at t-stage, and denote $\mathbf{L}_s^t = [l_{s1}^t, ..., l_{sn_t}^t] \in \mathbb{N}^{n_t}$ as the corresponding label set, where n_t is the size of the batch. We denote $\mathbf{X}_e^t = [\mathbf{X}_s^1, ..., \mathbf{X}_s^{t-1}] = [\mathbf{x}_{e1}^t, ..., \mathbf{x}_{em_t}^t] \in \mathbb{R}^{d\times m_t}$, where $m_t = n_1 + ... + n_{t-1}$, as the previously existing dataset with its label set $\mathbf{L}_e^t = [\mathbf{L}_s^1, ..., \mathbf{L}_s^{t-1}] = [l_{e1}^t, ..., l_{em_t}^t] \in \mathbb{N}^{m_t}$. Correspondingly, we denote $\mathbf{B}_s^t = \mathrm{sgn}(\mathbf{W}^{t^T}\mathbf{X}_s^t) = [\mathbf{b}_{s1}^t, ..., \mathbf{b}_{sn_t}^t] \in \mathbb{R}^{k\times n_t}$, $\mathbf{B}_e^t = \mathrm{sgn}(\mathbf{W}^{t^T}\mathbf{X}_e^t) = [\mathbf{b}_{e1}^t, ..., \mathbf{b}_{em_t}^t] \in \mathbb{R}^{k\times m_t}$ as the discretely learned binary codes for $\mathbf{X}_s^t$ and $\mathbf{X}_e^t$, respectively. Under online setting, the parameter matrix $\mathbf{W}^t$ should be updated based on the newly coming batch $\mathbf{X}_s^t$ instead of the existing dataset $\mathbf{X}_e^t$.

The Proposed Framework

Ideally, if data $\mathbf{x}_i$ and $\mathbf{x}_j$ are similar, the Hamming distance between their binary codes should be minimized, and vice versa. This is achieved by minimizing the quantization error between the similarity matrix and the Hamming similarity matrix (Liu et al. 2012). However, considering the streaming batch data alone does not reflect the structural relationship of all data samples. Therefore, following (Shen et al. 2015a; Jiang and Li 2018), we resort to preserve the similarity in the Hamming space between new data batch $\mathbf{X}_s^t$ and the existing dataset $\mathbf{X}_e^t$ at t-stage with an asymmetric graph as shown in Fig.1. To that effect, we minimize the Frobenius norm loss between the supervised similarity and the inner products of $\mathbf{B}_s^t$ and $\mathbf{B}_e^t$ as follows:

$$\begin{aligned} &\min_{\mathbf{B}_s^t, \mathbf{B}_e^t} \|\mathbf{B}_s^{t^T}\mathbf{B}_e^t - k\mathbf{S}^t\|_{\mathcal{F}}^2 \\ &s.t. \quad \mathbf{B}_s^t \in \{-1,1\}^{k\times n_t}, \mathbf{B}_e^t \in \{-1,1\}^{k\times m_t}. \end{aligned} \tag{2}$$

where $\mathbf{S}^t \in \mathbb{R}^{n_t\times m_t}$ is the similarity matrix between $\mathbf{X}_s^t$ and $\mathbf{X}_e^t$. Note that $s_{ij}^t = 1$ iff both $\mathbf{x}_{si}^t$ and $\mathbf{x}_{ej}^t$ share the same label, *i.e.*, $l_{si}^t = l_{ej}^t$. Otherwise, $s_{ij}^t = -1$[2]. And $\|\cdot\|_{\mathcal{F}}$ denotes the Frobenius norm.

Besides, we aim to learn the hash functions by minimizing the error term between the linear hash functions F in Eq.1 and the corresponding binary codes $\mathbf{B}_s^t$, which is constrained by $\|\mathbf{B}_s^t - F(\mathbf{X}_s^t)\|_{\mathcal{F}}^2$. It can be easily combined with the above asymmetric graph that can be seen as a regularizer for learning the hash functions, which is rewritten as:

$$\begin{aligned} &\min_{\mathbf{B}_s^t, \mathbf{B}_e^t, \mathbf{W}^t} \underbrace{\|\mathbf{B}_s^{t^T}\mathbf{B}_e^t - k\mathbf{S}^t\|_{\mathcal{F}}^2}_{\text{term 1}} + \sigma^t \underbrace{\|F(\mathbf{X}_s^t) - \mathbf{B}_s^t\|_{\mathcal{F}}^2}_{\text{term 2}} + \\ &\lambda^t \underbrace{\|\mathbf{W}^t\|_{\mathcal{F}}^2}_{\text{term 3}} \quad s.t. \, \mathbf{B}_s^t \in \{-1,1\}^{k\times n_t}, \mathbf{B}_e^t \in \{-1,1\}^{k\times m_t}, \end{aligned} \tag{3}$$

where σ^t and λ^t serve as two constants at t-stage to balance the trade-offs among the three learning parts.

We analyze that using such a framework can learn better coding functions. Firstly, in term 2, $\mathbf{W}^t$ is optimized based on the dynamic streaming data $\mathbf{X}_s^t$, which makes the hash function more adaptive to unseen data. Secondly, As in Eq.7, the training complexity for $\mathbf{X}_s^t$-based learning $\mathbf{W}^t$ is $\mathcal{O}(d^2 n_t + d^3)$, while it is $\mathcal{O}(d^2 m_t + d^3)$ for the learnt $\mathbf{W}^t$ based on $\mathbf{X}_e^t$. Therefore, updating $\mathbf{W}^t$ based on $\mathbf{X}_e^t$ is impractical when $m_t \gg n_t$ with the increasing number of new data batch. Further, it also violates the basic principle of OH that $\mathbf{W}^t$ can only be updated based on the newly coming data. Last but not least, with the asymmetric graph loss in term 1, the structural relationship in the original space can be well preserved in the produced Hamming space, which makes the learned binary codes $\mathbf{B}_s^t$ more robust. The above discussion will be verified in the subsequent experiments.

The Data-Imbalance Issue

As shown in Fig.1, the similarity matrix $\mathbf{S}^t$ between the streaming data and the existing dataset is very sparse[3]. That is to say, there exists a severe data-imbalance phenomenon, *i.e.*, most of image pairs are dissimilar and few pairs are similar. Due to this problem, the optimization will heavily rely on the dissimilar information and miss the similar information, which leads to performance degeneration.

As a theoretical analysis, we decouple the whole sparse similarity matrix into two subparts, where similar pairs and dissimilar pairs are separately considered. Term 1 in Eq.3 is then reformulated as:

$$\begin{aligned} \text{term 1} = &\underbrace{\sum_{i,j,\mathbf{S}_{ij}^t=1} (\mathbf{b}_{si}^{t^T}\mathbf{b}_{ej}^t - k)^2}_{\text{term } \mathcal{A}} + \underbrace{\sum_{i,j,\mathbf{S}_{ij}^t=-1} (\mathbf{b}_{si}^{t^T}\mathbf{b}_{ej}^t + k)^2}_{\text{term } \mathcal{B}} \\ &s.t. \quad \mathbf{b}_{si}^t \in \{-1,1\}^k, \mathbf{b}_{ej}^t \in \{-1,1\}^k. \end{aligned} \tag{4}$$

Analysis 1. We denote $\mathbf{S}_1^t = \{\mathbf{S}_{ij}^t \in \mathbf{S}^t | \mathbf{S}_{ij}^t = 1\}$, *i.e.*, the set of similar pairs and $\mathbf{S}_2^t = \{\mathbf{S}_{ij}^t \in \mathbf{S}^t | \mathbf{S}_{ij}^t = -1\}$, *i.e.*,

[2]At each stage, $\mathbf{S}^t$ is calculated on-the-fly.

[3]Here, "sparse" denotes the vast majority of elements in a matrix are -1.

the set of dissimilar pairs. In online setting, when $n_t \ll m_t$ with the increase of new data batch, the similarity matrix $\mathbf{S}^t$ becomes a highly sparse matrix, *i.e.*, $|\mathbf{S}_1^t| \ll |\mathbf{S}_2^t|$. In other words, term 1 suffers from a severe data-imbalance problem. Furthermore, since term 1 $\gg$ term 2 in Eq.3 and term $\mathcal{B} \gg$ term $\mathcal{A}$ in Eq.4, the learning process of $\mathbf{B}_s^t$ and $\mathbf{B}_e^t$ heavily relies on term $\mathcal{B}$.

A suitable way to minimize term $\mathcal{B}$ is to have ${\mathbf{b}_{si}^t}^T \mathbf{b}_{ej}^t = -k$, *i.e.*, $\mathbf{b}_{si}^t = -\mathbf{b}_{ej}^t$. Similarly, for any $\mathbf{b}_{eg}^t \in \mathbf{B}_e^t$ with $g \neq j$, we have $\mathbf{b}_{si}^t = -\mathbf{b}_{eg}^t$. It is easy to see that $\mathbf{b}_{ej}^t = \mathbf{b}_{eg}^t$. In other words, each item in $\mathbf{B}_e^t$ shares consistent binary codes. Similarly, each item in $\mathbf{B}_s^t$ also shares consistent binary codes which are opposite with $\mathbf{B}_e^t$. Fig.1 illustrates such an extreme circumstance. However, as can be seen from term 2 in Eq.3, the performance of hash functions deeply relies on the learned $\mathbf{B}_s^t$. Therefore, such a data-imbalance problem will cause all the codes produced by $\mathbf{W}^t$ to be biased, which will seriously affect the retrieval performance.

Balanced Similarity

To solve the above problem, a common method is to keep a balance between term 1 and term 2 in Eq.3 by scaling up the parameter σ^t. However, as verified later in our experiments (see Fig.5), such a scheme still suffers from unsatisfactory performance and will get stuck in how to choose an appropriate value of σ^t from a large range[4]. Therefore, we present another scheme to handle this problem, which expands the feasible solutions for both $\mathbf{B}_e^t$ and $\mathbf{B}_s^t$. Concretely, we propose to use a balanced similarity matrix $\tilde{\mathbf{S}}^t$ with each element defined as follows:

$$\tilde{\mathbf{S}}_{ij}^t = \begin{cases} \eta_s \mathbf{S}_{ij}^t, & \mathbf{S}_{ij}^t = 1, \\ \eta_d \mathbf{S}_{ij}^t, & \mathbf{S}_{ij}^t = -1, \end{cases} \tag{5}$$

where η_s and η_d are two positive equilibrium factors used to balance the similar and dissimilar weights, respectively. When setting $\eta_s > \eta_d$, the Hamming distances among similar pairs will be reduced, while the ones among dissimilar pairs will be enlarged.

Analysis 2. With the balanced similarity, the goal of term $\mathcal{B}$ in Eq.4 is to have ${\mathbf{b}_{si}^t}^T \mathbf{b}_{ej}^t \approx -k\eta_d$. The number of common hash bits between $\mathbf{b}_{si}^t$ and $\mathbf{b}_{ej}^t$ is at least $\lfloor \frac{k(1-\eta_d)}{2} \rfloor$[5]. Therefore, by fixing $\mathbf{b}_{si}^t$, the cardinal number of feasible solutions for $\mathbf{b}_{ej}^t$ is at least $\binom{k}{\lfloor \frac{k(1-\eta_d)}{2} \rfloor}$. Thus, the balanced similarity matrix $\tilde{\mathbf{S}}^t$ can effectively solve the problem of generating consistent binary codes, as showed in Fig.1.

By replacing the similarity matrix $\mathbf{S}^t$ in Eq.3 with the balanced similarity matrix $\tilde{\mathbf{S}}^t$, the overall objective function can be written as:

$$\min_{\mathbf{B}_s^t, \mathbf{B}_e^t, \mathbf{W}^t} \underbrace{\|{\mathbf{B}_s^t}^T \mathbf{B}_e^t - k\tilde{\mathbf{S}}^t\|_{\mathcal{F}}^2}_{\text{term 1}} + \sigma^t \underbrace{\|F(\mathbf{X}_s^t) - \mathbf{B}_s^t\|_{\mathcal{F}}^2}_{\text{term 2}} + \lambda^t \underbrace{\|\mathbf{W}^t\|_{\mathcal{F}}^2}_{\text{term 3}} \quad s.t.\, \mathbf{B}_s^t \in \{-1,1\}^{k \times n_t}, \mathbf{B}_e^t \in \{-1,1\}^{k \times m_t}. \tag{6}$$

[4]Under the balanced similarity, we constrain σ^t to [0, 1].

[5]$\lfloor \cdot \rfloor$ denotes the operation of rounding down.

The Optimization

Due to the binary constraints, the optimization problem of Eq.6 is still non-convex with respect to $\mathbf{W}^t, \mathbf{B}_s^t, \mathbf{B}_e^t$. To find a feasible solution, we adopt an alternative optimization approach, *i.e.*, updating one variable with the rest two fixed until convergence.

1) $\mathbf{W}^t$-step: Fix $\mathbf{B}_e^t$ and $\mathbf{B}_s^t$, then learn hash weights $\mathbf{W}^t$. This sub-optimization of Eq.6 is a classical linear regression that aims to find the best projection coefficient $\mathbf{W}^t$ by minimizing term 2 and term 3 jointly. Therefore, we update $\mathbf{W}^t$ with a close-formed solution as:

$$\mathbf{W}^t = \sigma^t(\sigma^t \mathbf{X}_s^t {\mathbf{X}_s^t}^T + \lambda^t \mathbf{I})^{-1} \mathbf{X}_s^t {\mathbf{B}_s^t}^T, \tag{7}$$

where $\mathbf{I}$ is a $d \times d$ identity matrix.

2) $\mathbf{B}_e^t$-step: Fix $\mathbf{W}^t$ and $\mathbf{B}_s^t$, then update $\mathbf{B}_e^t$. Since only term 1 in Eq.6 contains $\mathbf{B}_e^t$, we directly optimize this term via a discrete optimization similar to (Kang, Li, and Zhou 2016), where the squared Frobenius norm in term 1 is replaced with the L_1 norm. The new formulation is:

$$\min_{\mathbf{B}_e^t} \|{\mathbf{B}_s^t}^T \mathbf{B}_e^t - k\tilde{\mathbf{S}}^t\|_1 \quad s.t. \quad \mathbf{B}_e^t \in \{-1,1\}^{k \times m_t}. \tag{8}$$

Similar to (Kang, Li, and Zhou 2016), the solution of Eq.8 is as follows:

$$\mathbf{B}_e^t = sgn(\mathbf{B}_s^t \tilde{\mathbf{S}}^t). \tag{9}$$

3) $\mathbf{B}_s^t$-step: Fix $\mathbf{B}_e^t$ and $\mathbf{W}^t$, then update $\mathbf{B}_s^t$. The corresponding sub-problem is:

$$\min_{\mathbf{B}_s^t} \|{\mathbf{B}_s^t}^T \mathbf{B}_e^t - k\tilde{\mathbf{S}}^t\|_{\mathcal{F}}^2 + \sigma^t \|{\mathbf{W}^t}^T \mathbf{X}_s^t - \mathbf{B}_s^t\|_{\mathcal{F}}^2 \quad s.t. \quad \mathbf{B}_s^t \in \{-1,1\}^{k \times n_t}. \tag{10}$$

By expanding each term in Eq.10, we get the sub-optimal problem of $\mathbf{B}_s^t$ by minimizing the following formulation:

$$\begin{aligned} \min_{\mathbf{B}_s^t} & \|{\mathbf{B}_e^t}^T \mathbf{B}_s^t\|_{\mathcal{F}}^2 + \underbrace{\|k\tilde{\mathbf{S}}^t\|_{\mathcal{F}}^2}_{const} - 2tr(k\tilde{\mathbf{S}}^t {\mathbf{B}_e^t}^T \mathbf{B}_s^t) \\ & + \sigma^t(\underbrace{\|{\mathbf{W}^t}^T \mathbf{X}_s^t\|_{\mathcal{F}}^2}_{const} + \underbrace{\|\mathbf{B}_s^t\|_{\mathcal{F}}^2}_{const} - 2tr({\mathbf{X}_s^t}^T \mathbf{W}^t \mathbf{B}_s^t)) \\ & s.t. \quad \mathbf{B}_s^t \in \{-1,1\}^{k \times n_t}, \end{aligned} \tag{11}$$

where the “const” terms denote constants. The optimization problem of Eq.11 is equivalent to

$$\min_{\mathbf{B}_s^t} \| \underbrace{{\mathbf{B}_e^t}^T \mathbf{B}_s^t}_{\text{term I}} \|_{\mathcal{F}}^2 - 2tr(\underbrace{\mathbf{P}^T \mathbf{B}_s^t}_{\text{term II}}) \quad s.t.\, \mathbf{B}_s^t \in \{-1,1\}^{k \times n_t}, \tag{12}$$

where $\mathbf{P} = k\mathbf{B}_e^t {\tilde{\mathbf{S}}^t}^T + \sigma^t {\mathbf{W}^t}^T \mathbf{X}_s^t$ and $tr(\cdot)$ is trace norm.

The problem in Eq.12 is NP-hard for directly optimizing the binary code matrix $\mathbf{B}_s^t$. Inspired by the recent advance on binary code optimization (Shen et al. 2015b), a closed-form solution for one row of $\mathbf{B}_s^t$ can be obtained while fixing all the other rows. Therefore, we first reformulate term I and term II in Eq.12 as follows:

$$\text{term I} = {\tilde{\mathbf{b}}_{er}^t}{}^T \tilde{\mathbf{b}}_{sr}^t + {\tilde{\mathbf{B}}_e^t}{}^T \tilde{\mathbf{B}}_s^t, \tag{13}$$

Algorithm 1 Balanced Similarity for Online Discrete Hashing (BSODH)

```
Require: Training data set $\mathbf{X}$ with its label space $\mathbf{L}$, the number of hash bits $k$, the parameters $\sigma$ and $\lambda$, the total number of streaming data batches $T$.
Ensure: Binary codes $\mathbf{B}$ for $\mathbf{X}$ and hash weights $\mathbf{W}$.
1: for $t = 1 \to T$ do
2:   Denote the newly coming data batch as $\mathbf{X}_s^t$;
3:   if $t = 1$ then
4:     Initialize $\mathbf{W}^t$ with normal Gaussian distribution;
5:     Compute $\mathbf{B}_s^t = sgn(\mathbf{W}^{t^T}\mathbf{X}_s^t)$;
6:   else
7:     Compute $\mathbf{S}^t$ based on the label sets $\mathbf{L}_s^t$ and $\mathbf{L}_e^t$;
8:     Compute $\tilde{\mathbf{S}}^t$ via Eq.5;
9:     Initialize $\mathbf{B}_s^t = sgn(\mathbf{W}^{t^T}\mathbf{X}_s^t)$;
10:    Update $\mathbf{W}^t$ via Eq.7 and $\mathbf{B}_e^t$ via Eq.9;
11:    repeat
12:      for $r = 1 \to k$ do
13:        Update $\tilde{\mathbf{b}}_{sr}^t$ via Eq.17;
14:      end for
15:    until (convergency or reaching maximum iterations)
16:  end if
17:  Set $\mathbf{X}_e^t = [\mathbf{X}_e^t; \mathbf{X}_s^t]$ and $\mathbf{B}_e^t = [\mathbf{B}_e^t; \mathbf{B}_s^t]$;
18: end for
19: Set $\mathbf{W} = \mathbf{W}^t$;
20: Compute $\mathbf{B} = sgn(\mathbf{W}^T\mathbf{X})$;
21: Return $\mathbf{W}$ and $\mathbf{B}$.
```

$$\text{term II} = \tilde{\mathbf{p}}_r^T \tilde{\mathbf{b}}_{sr}^t + \tilde{\mathbf{P}}^T \tilde{\mathbf{B}}_s^t, \tag{14}$$

where $\tilde{\mathbf{b}}_{er}^t$, $\tilde{\mathbf{b}}_{sr}^t$ and $\tilde{\mathbf{p}}_r$ stand for the r-row of $\mathbf{B}_e^t$, $\mathbf{B}_s^t$ and $\mathbf{P}$, respectively. Also, $\tilde{\mathbf{B}}_e^t$, $\tilde{\mathbf{B}}_s^t$ and $\tilde{\mathbf{P}}$ represent the matrix of $\mathbf{B}_e^t$ excluding $\tilde{\mathbf{b}}_{er}^t$, the matrix of $\mathbf{B}_s^t$ excluding $\tilde{\mathbf{b}}_{sr}^t$ and the matrix of $\mathbf{P}$ excluding $\tilde{\mathbf{p}}_r$, respectively.

Taking Eq.13 and Eq.14 back to Eq.12 and expanding it, we obtain the following optimization problem:

$$\min_{\tilde{\mathbf{b}}_{sr}^t} \underbrace{\|\tilde{\mathbf{b}}_{er}^{t\,T} \tilde{\mathbf{b}}_{sr}^t\|_{\mathcal{F}}^2}_{const} + \underbrace{\|\tilde{\mathbf{B}}_e^{t\,T} \tilde{\mathbf{B}}_s^t\|_{\mathcal{F}}^2}_{const} + 2tr(\tilde{\mathbf{B}}_s^{t\,T} \tilde{\mathbf{B}}_e^t \tilde{\mathbf{b}}_{er}^{t\,T} \tilde{\mathbf{b}}_{sr}^t)$$
$$- 2tr(\tilde{\mathbf{p}}_r^T \tilde{\mathbf{b}}_{sr}^t) - 2\underbrace{tr(\tilde{\mathbf{P}}^T \tilde{\mathbf{B}}_s^t)}_{const} \quad s.t. \quad \tilde{\mathbf{b}}_{sr}^t \in \{-1, 1\}^{n_t}. \tag{15}$$

Note that $\|\tilde{\mathbf{b}}_{er}^{t\,T} \tilde{\mathbf{b}}_{sr}^t\|_{\mathcal{F}}^2 = k^2$, which is a constant value. The above optimization problem is equivalent to:

$$\min_{\tilde{\mathbf{b}}_{sr}^t} tr((\tilde{\mathbf{B}}_s^{t\,T} \tilde{\mathbf{B}}_e^t \tilde{\mathbf{b}}_{er}^{t\,T} - \tilde{\mathbf{p}}_r^T) \tilde{\mathbf{b}}_{sr}^t) \quad s.t. \quad \tilde{\mathbf{b}}_{sr}^t \in \{-1, 1\}^{n_t}. \tag{16}$$

Therefore, this sub-problem can be solved by the following updating rule:

$$\tilde{\mathbf{b}}_{sr}^t = sgn(\tilde{\mathbf{p}}_r - \tilde{\mathbf{b}}_{er}^t \tilde{\mathbf{B}}_e^{t\,T} \tilde{\mathbf{B}}_s^t). \tag{17}$$

The main procedures of the proposed BSODH are summarized in Alg.1. Note that, in the first training stage, *i.e.*, $t = 1$, we initialize $\mathbf{W}^1$ with normal Gaussian distribution as in line 4 and compute $\mathbf{B}_s^1$ as in line 5. When $t \geq 2$, we initialize $\mathbf{B}_s^t$ in line 9 to fasten the training iterations from line 11 to line 15. By this way, it is quantitatively shown in the experiment that it takes only one or two iterations to get convergence (see Fig.6).

Experiments

Datasets

CIFAR-10 contains 60K samples from 10 classes, with each represented by a 4,096-dimensional CNN feature (Simonyan and Zisserman 2015). Following (Fatih et al. 2017), we partition the dataset into a retrieval set with 59K samples, and a test set with 1K samples. From the retrieval set, 20K instances are adopted to learn the hash functions.

Places205 is a 2.5-million image set with 205 classes. Following (Fatih et al. 2017; Fatih, Bargal, and Sclaroff 2017), features are first extracted from the *fc7* layer of the AlexNet (Krizhevsky, Sutskever, and Hinton 2012), and then reduced to 128 dimensions by PCA. 20 instances from each category are randomly sampled to form a test set, the remaining of which are formed as a retrieval set. 100K samples from the retrieval set are sampled to learn hash functions.

MNIST consists of 70K handwritten digit images with 10 classes, each of which is represented by 784 normalized original pixels. We construct the test set by sampling 100 instances from each class, and form a retrieval set using the rest. A random subset of 20K images from the retrieval set is used to learn the hash functions.

Baselines and Evaluated Metrics

We compare the proposed BSODH with several state-of-the-art OH methods, including Online Kernel Hashing (**OKH**) (Huang, Yang, and Zheng 2013), Online Sketch Hashing (**SketchHash**) (Leng et al. 2015), Adaptive Hashing (**AdaptHash**) (Fatih and Sclaroff 2015), Online Supervised Hashing (**OSH**) (Fatih, Bargal, and Sclaroff 2017) and OH with Mutual Information (**MIHash**) (Fatih et al. 2017).

To evaluate the proposed method, we adopt a set of widely-used protocols including mean Average Precision (denoted as ***m*AP**), mean precision of the top-R retrieved neighbors (denoted as **Precision@R**) and precision within a Hamming ball of radius 2 centered on each query (denoted as **Precision@H2**). Note that, following the work of (Fatih et al. 2017), we only compute *m*AP on the top-1,000 retrieved items (denoted as ***m*AP@1,000**) on Places205 due to its large scale. And for SketchHash (Leng et al. 2015), the batch size has to be larger than the size of hash bits. Thus, we only report its performance when the hash bit is 32.

Quantitative Results

We first show the experimental results of *m*AP (*m*AP@1,000) and Precision@H2 on CIFAR-10, Places205 and MNIST. The results are shown in Tab.1 and Tab.2. Generally, the proposed BSODH is consistently better in these two evaluated metrics on all three benchmarks. For a depth analysis, in terms of *m*AP, compared with the second

Table 1: *m*AP (*m*AP@$1,000$) and Precision@H2 comparisons on CIFAR-10 and Places205 with hash bits of 32, 64 and 128.

Method	CIFAR-10						Places205					
	*m*AP			Precision@H2			*m*AP-$1,000$			Precision@H2		
	32-bit	64-bit	128-bit	32-bit	64-bit	128-bit	32-bit	64-bit	128-bit	32-bit	64-bit	128-bit
OKH	0.223	0.268	0.350	0.100	0.175	0.372	0.122	0.114	0.258	0.026	0.217	0.075
SketchHash	0.302	-	-	0.385	-	-	0.202	-	-	0.220	-	-
AdaptHash	0.216	0.305	0.293	0.185	0.166	0.164	0.195	0.222	0.229	0.012	0.021	0.022
OSH	0.129	0.127	0.125	0.137	0.083	0.038	0.022	0.043	0.164	0.012	0.030	0.059
MIHash	0.675	0.667	0.664	0.657	0.500	0.413	0.244	**0.308**	0.332	0.204	0.202	0.069
BSODH	**0.689**	**0.709**	**0.711**	**0.691**	**0.690**	**0.602**	**0.250**	**0.308**	**0.337**	**0.241**	**0.212**	**0.101**

Table 2: *m*AP (*m*AP@$1,000$) and Precision@H2 comparisons on MNIST with hash bits of 32, 64 and 128.

Method	*m*AP			Precision@H2		
	32-bit	64-bit	128-bit	32-bit	64-bit	128-bit
OKH	0.224	0.301	0.404	0.457	0.522	0.124
SketchHash	0.348	-	-	0.691	-	-
AdaptHash	0.319	0.292	0.208	0.535	0.163	0.168
OSH	0.130	0.146	0.143	0.192	0.109	0.019
MIHash	0.744	0.713	0.681	0.814	0.720	0.471
BSODH	**0.747**	**0.766**	**0.760**	**0.826**	**0.814**	**0.643**

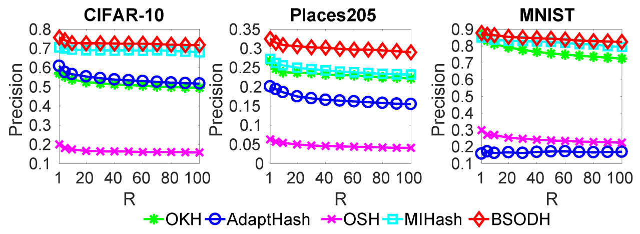

Figure 2: Precision@R curves of compared algorithms on three datasets with hash bit of 64.

best method, *i.e.*, MIHash, the proposed method achieves improvements of 5.11%, 1.40%, and 6.48% on CIFAR-10, Places-205 and MNIST, respectively. As for Precision@H2, compared with MIHash, the proposed method acquires 29.97%, 2.63% and 9.2% gains on CIFAR-10, Places-205 and MNIST, respectively.

We also evaluate Precision@R with R ranging from 1 to 100 under the hash bit of 64. The experimental results are shown in Fig.2, which verifies that the proposed BSODH also achieves superior performance on all three benchmarks.

Parameter Sensitivity

The following experiments are conducted on MNIST with the hash bit fixed to 64.

Sensitivities to λ^t and σ^t. The left two figures in Fig.3 present the effects of the hyper-parameters λ^t and σ^t. For simplicity, we regard λ^t and σ^t as two constants across the whole training process. As shown in Fig.3, the performance of the proposed BSODH is sensitive to the values of σ^t and λ^t. The best combination for (λ^t, σ^t) is $(0.6, 0.5)$. By conducting similar experiments on CIFAR-10 and Places-205, we finally set the tuple value of (λ^t, σ^t) as $(0.3, 0.5)$ and $(0.9, 0.8)$ for these two benchmarks.

Necessity of $\tilde{\mathbf{S}}^t$. We validate the effectiveness of the proposed balanced similarity $\tilde{\mathbf{S}}^t$ by plotting the Precision@H2 curves with respect to the two positive equilibrium factors, *i.e.*, η_s and η_d. As shown in the right two figures of Fig.3, the performance stabilizes when $\eta_s \geq 1$ and $\eta_d \leq 0.3$. When $\eta_d = 1$ and $\eta_s = 1$, $\tilde{\mathbf{S}}^t$ degenerates into an un-balanced version $\mathbf{S}^t$. However, as observed from the rightmost chart in Fig.3, when $\eta_s = 1$, the proposed method suffers from severe performance loss. Precisely, the Precision@H2 shows the best of 0.814 when $\eta_s = 1.2$ and $\eta_d = 0.2$, while it is only 0.206 when $\eta_s = 1$ and $\eta_d = 1$. Compared with the un-balanced $\mathbf{S}^t$, the proposed balanced similarity $\tilde{\mathbf{S}}^t$ gains a 295.15% increase, which effectively shows the superiority of the proposed balanced similarity $\tilde{\mathbf{S}}^t$. In our experiment, we set the tuple (η_s, η_d) as $(1.2, 0.3)$ on MNIST. Similarly, it is set as $(1.2, 0.2)$ on CIFAR-10 and $(1, 0)$ on Places205.

To verify the aforementioned **Analysis 1** and **Analysis 2**, we further visualize the learned binary codes in the last training stage via t-SNE (Maaten and Hinton 2008). As shown in Fig.4, (a), (b) and (c) are derived under un-balanced similarity $\mathbf{S}^t$ with $\eta_s = 1$ and $\eta_d = 1$. And Fig.4 (d), (e) and (f) are obtained under balanced similarity $\tilde{\mathbf{S}}^t$ with $\eta_s = 1.2$ and $\eta_d = 0.2$.

Though the discretely optimized binary codes $\mathbf{B}_e^t$ (a), $\mathbf{B}_s^t$ (b) and linearly mapped binary codes $sgn(\mathbf{W}^{t^T}\mathbf{X}_s^t)$ (c) are clustered, each cluster is mixed with items from different classes and only four out of ten clusters are formed with each close to each other. That is to say, the majorities of Hamming codes are the same, which conforms with **Analysis 1**. However, under the balanced setting, both $\mathbf{B}_e^t$ and $\mathbf{B}_s^t$ are formed into ten separated clusters without mixed items in each clusters, which conforms with **Analysis 2**. Under such a situation, the hash functions $\mathbf{W}^t$ are well deduced by $\mathbf{B}_s^t$, with the hash codes in Fig.4 (f) more discriminative.

Scaling up σ^t. As aforementioned, an alternative approach to solving the data-imbalance problem in **Analysis 1** is to keep a balance between term 1 and term 2 in Eq.3 via scaling up the parameter σ^t. To test the feasibility of this scheme, we plot the values of Precision@H2 with σ^t varying in a large scale in Fig.5. Intuitively, scaling up σ^t affects the performance quite a lot. Quantitatively, when the value of σ^t is set as $10,000$, Precision@H2 achieves the best, *i.e.,* 0.341. We argue that this scheme shows its drawbacks in two aspects. First, it suffers from the unsatisfactory performance. As shown in Tab.2, when hash bit is 64, the proposed

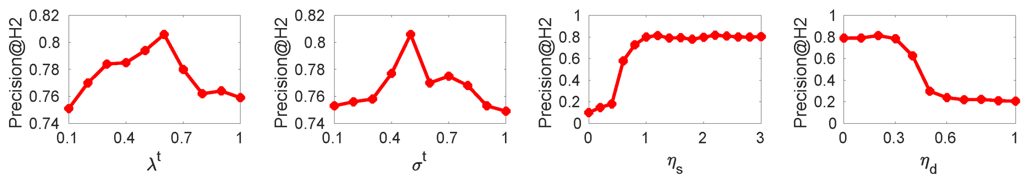
Figure 3: Precision@H2 with respect to varying values of λ^t, σ^t, η_s and σ_d.

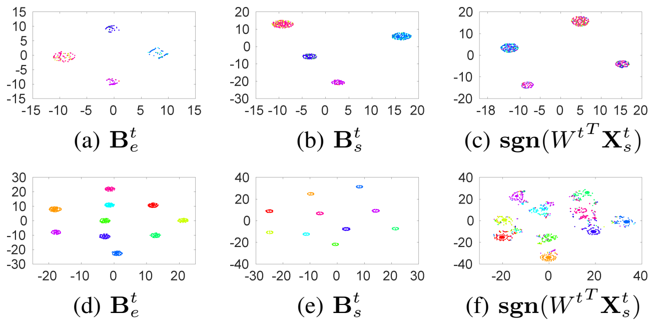
Figure 4: The t-SNE visualization of hash codes. The top row shows the un-balanced results. The bottom row shows the balanced results. Given 10 data clusters, only four are formed for un-balanced results due to the existence of data-imbalance problem. It can be solved by the proposed balanced similarity with more clusters being formed.

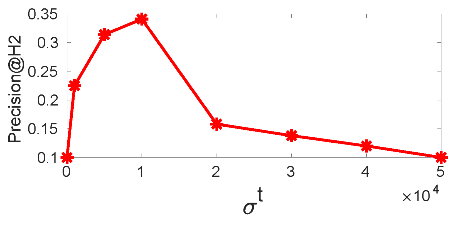
Figure 5: Precision@H2 results when scaling up σ^t.

BSODH gets 0.814 in term of Precision@H2 on MNIST. Compared with scaling up σ^t, the proposed method achieves more than 2.5 times better performance. Second, scaling up σ^t also easily gets stuck in how to choose an appropriate value due to the large range of σ^t. To decide a best value, extensive experiments have to be repeated, which is infeasible in online learning. However, σ^t is limited to $[0, 1]$ under the proposed BSODH. It is much convenient to choose an appropriate value for σ^t.

Convergence of $\mathbf{B}_s^t$. Each time when the new streaming data arrives, $\mathbf{B}_s^t$ is updated based on iterative process, as shown in lines $11-15$ in Alg.1. Fig.6 shows the convergence ability of the proposed BSODH on the input streaming data at t-stage. As can be seen, when $t \leq 2$, it merely takes two

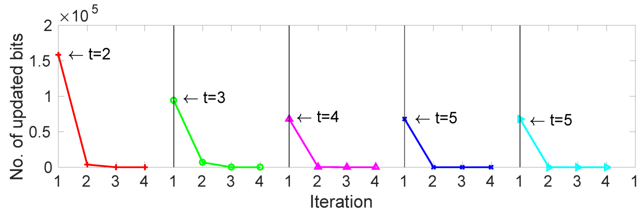
Figure 6: Convergence of the proposed BSODH.

iterations to get convergence. What's more, it costs only one iteration to finish updating $\mathbf{B}_s^t$ when $t > 2$, which validates not only the convergence ability, but also the efficiency of the proposed BSODH.

Conclusions

In this paper, we present a novel supervised OH method, termed BSODH. The proposed BSODH learns the correlation of binary codes between the newly streaming data and the existing database via a discrete optimization, which is the first to the best of our knowledge. To this end, first we use an asymmetric graph regularization to preserve the similarity in the produced Hamming space. Then, to reduce the quantization error, we mathematically formulate the optimization problem and derive the discrete optimal solutions. Finally, to solve the data-imbalance problem, we propose a balanced similarity, where two equilibrium factors are introduced to balance the similar/dissimilar weights. Extensive experiments on three benchmarks demonstrate that our approach merits in both effectiveness and efficiency over several state-of-the-art OH methods.

Acknowledge

This work is supported by the National Key R&D Program (No. 2017YFC0113000, and No. 2016YFB1001503), Nature Science Foundation of China (No. U1705262, No. 61772443, and No. 61572410), Post Doctoral Innovative Talent Support Program under Grant BX201600094, China Post-Doctoral Science Foundation under Grant 2017M612134, Scientific Research Project of National Language Committee of China (Grant No. YB135-49), and Nature Science Foundation of Fujian Province, China (No. 2017J01125 and No. 2018J01106).

References

Chen, X.; King, I.; and Lyu, M. R. 2017. Frosh: Faster online sketching hashing. In *Proceedings of the UAI*.

Clarkson, K. L., and Woodruff, D. P. 2009. Numerical linear algebra in the streaming model. In *Proceedings of the ACM STOC*.

Crammer, K.; Dekel, O.; Keshet, J.; Shalev-Shwartz, S.; and Singer, Y. 2006. Online passive-aggressive algorithms. *JMLR*.

Datar, M.; Immorlica, N.; Indyk, P.; and Mirrokni, V. S. 2004. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the ASCG*.

Dietterich, T. G., and Bakiri, G. 1995. Solving multiclass learning problems via error-correcting output codes. *JAIR*.

Fatih, C., and Sclaroff, S. 2015. Adaptive hashing for fast similarity search. In *Proceedings of the ICCV*.

Fatih, C.; Bargal, S. A.; and Sclaroff, S. 2017. Online supervised hashing. *CVIU*.

Fatih, C.; He, K.; Bargal, S. A.; and Sclaroff, S. 2017. Mihash: Online hashing with mutual information. In *Proceedings of the ICCV*.

Gong, Y., and Lazebnik, S. 2011. Iterative quantization: A procrustean approach to learning binary codes. In *Proceedings of the CVPR*.

Huang, L.; Yang, Q.; and Zheng, W. 2013. Online hashing. In *Proceedings of the IJCAI*.

Ji, R.; Liu, H.; Cao, L.; Liu, D.; Wu, Y.; and Huang, F. 2017. Toward optimal manifold hashing via discrete locally linear embedding. *IEEE TIP*.

Jiang, Q., and Li, W. 2015. Scalable graph hashing with feature transformation. In *Proceedings of the IJCAI*.

Jiang, Q., and Li, W. 2018. Asymmetric deep supervised hashing. In *Proceedings of the AAAI*.

Kang, W.; Li, W.; and Zhou, Z. 2016. Column sampling based discrete supervised hashing. In *Proceedings of the AAAI*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Proceedings of the NIPS*.

Leng, C.; Wu, J.; Cheng, J.; Bai, X.; and Lu, H. 2015. Online sketching hashing. In *Proceedings of the CVPR*.

Lin, S.; Ji, R.; Chen, C.; Tao, D.; and Luo, J. 2018. Holistic cnn compression via low-rank decomposition with knowledge transfer. *IEEE TPAMI*.

Liu, W.; Wang, J.; Ji, R.; Jiang, Y.; and Chang, S.-F. 2012. Supervised hashing with kernels. In *Proceedings of the CVPR*.

Liu, W.; Mu, C.; Kumar, S.; and Chang, S. 2014. Discrete graph hashing. In *Proceedings of the NIPS*.

Liu, H.; Ji, R.; Wang, J.; and Shen, C. 2018. Ordinal constraint binary coding for approximate nearest neighbor search. *IEEE TPAMI*.

Maaten, L. v. d., and Hinton, G. 2008. Visualizing data using t-sne. *JMLR*.

Shen, F.; Liu, W.; Zhang, S.; Yang, Y.; and Tao Shen, H. 2015a. Learning binary codes for maximum inner product search. In *Proceedings of the ICCV*.

Shen, F.; Shen, C.; Liu, W.; and Shen, H. T. 2015b. Supervised discrete hashing. In *Proceedings of the CVPR*.

Simonyan, K., and Zisserman, A. 2015. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the ICLR*.

Wang, J.; Liu, W.; Kumar, S.; and Chang, S.-F. 2016. Learning to hash for indexing big data – a survey. *Proceedings of the IEEE*.

Wang, J.; Zhang, T.; Sebe, N.; and Shen, H. T. 2018. A survey on learning to hash. *IEEE TPAMI*.