DistillHash: Unsupervised Deep Hashing by Distilling Data Pairs

Erkun Yang[1,2], Tongliang Liu[2], Cheng Deng[1*], Wei Liu[3*], Dacheng Tao[2]
[1] School of Electronic Engineering, Xidian University, Xian 710071, China
[2] UBTECH Sydney AI Centre, School of Computer Science, FEIT, University of Sydney, Darlington, NSW 2008, Australia, [3] Tencent AI Lab, Shenzhen, China
ekyang@stu.xidian.edu.cn, tongliang.liu@sydney.edu.au, chdeng.xd@gmail.com, wl2223@columbia.edu, dacheng.tao@sydney.edu.au

Abstract

Due to the high storage and search efficiency, hashing has become prevalent for large-scale similarity search. Particularly, deep hashing methods have greatly improved the search performance under supervised scenarios. In contrast, unsupervised deep hashing models can hardly achieve satisfactory performance due to the lack of reliable supervisory similarity signals. To address this issue, we propose a novel deep unsupervised hashing model, dubbed DistillHash, which can learn a distilled data set consisted of data pairs, which have confidence similarity signals. Specifically, we investigate the relationship between the initial noisy similarity signals learned from local structures and the semantic similarity labels assigned by a Bayes optimal classifier. We show that under a mild assumption, some data pairs, of which labels are consistent with those assigned by the Bayes optimal classifier, can be potentially distilled. Inspired by this fact, we design a simple yet effective strategy to distill data pairs automatically and further adopt a Bayesian learning framework to learn hash functions from the distilled data set. Extensive experimental results on three widely used benchmark datasets show that the proposed DistillHash consistently accomplishes the state-of-the-art search performance.

1. Introduction

The explosive growth of visual data (e.g., photos and videos) has led to renewed interest in efficient indexing and searching algorithms [6, 9, 19, 44, 48, 53, 57, 58, 65–67], among which, hashing-based approximate nearest neighbor (ANN) searching, which balances retrieval quality and computational cost, has attracted increasing attention.

Generally, hashing methods can be divided into supervised and unsupervised models. The supervised hashing models [7, 35, 40, 62], which aim to learn hash functions with semantic labels, have shown remarkable performance. However, existing supervised hashing methods, especially deep hashing rely on massive labeled data examples to train their models. Thus, when there exist no enough training examples, their performance may be dramatically degraded caused by over-fitting to those training examples.

To address this challenge, unsupervised hashing methods usually adopt learning frameworks without requiring any supervised information. Traditional unsupervised hashing methods [3, 21, 25, 39] with hand-crafted features cannot well preserve the similarities of real-world data samples due to the low model capacity and the separated representation and binary codes optimization processes. To take advantages of the recent progress of deep learning [31, 56, 68], unsupervised deep hashing methods [11, 15, 17, 30, 36], which adopt neural networks as hash functions, have also been proposed. These deep hashing models are usually trained by minimizing either the quantization loss or data reconstruction loss. However, since these objectives fail to exploit the semantic similarities between data points, they can hardly achieve satisfactory results.

In this paper, we propose a novel unsupervised deep hashing model, dubbed DistillHash, which addresses the absence of supervisory signals by distilling data pairs with confident semantic similarity relationships. In particular, we first exploit the local structure of training data points to assign an initial similarity label for each data pair. If we treat the semantic similarity labels as true labels, these initial similarity labels then contain label- and instance-dependent label noise, because many of them fail to represent semantic similarities. By assuming that we know the probability of a semantic similarity label given a pair of the data points, the Bayes optimal classifier will assign the semantic similarity label to the data pair which has a higher probability (or has a probability greater than 0.5). Based on these results, we give strict analysis on the relationship between the noisy labels and the labels assigned by the Bayes

*Corresponding author

optimal classifier. Inspired by the framework of [8], we show that, under a mild assumption, data pairs with confident semantic labels can be potentially distilled. Furthermore, we theoretically give the criteria to select distilled data pairs and also provide a simple but effective method to collect distilled data pairs automatically. Finally, given the distilled data pair set, we design a deep neural network and adopt a Bayesian learning framework to perform the representation and hash code learning simultaneously.

Our main contributions can be summarized as follows:

- By considering signals learned from deep features as noisy pairwise labels, we successfully apply noisy label learning techniques to our method. This shows that data pairs, of which labels are consistent with those assigned by the Bayes optimal classifier, can be potentially distilled.
- We theoretically give the criteria to select distilled data pairs for hash learning and further provide a simple but effective method to collect distilled data pairs automatically.
- Experiments on three popular benchmark datasets show that our method can outperform current state-of-the-art unsupervised hashing methods.

The rest of this paper is organized as follows. We review the relevant literature in Section 2. We present our novel DistillHash in Section 3. Section 4 details the experiments, after which concluding remarks are presented in Section 5.

2. Related Work

Recently, the amount of literatures have grown up considerably around the theme of hashing [12, 13, 32, 33, 42, 42, 43]. According to whether supervised information are involved in the learning phase, existing hashing models can be divided into two categories: supervised hashing methods and unsupervised hashing methods.

Supervised hashing methods [5, 14, 22, 35, 40, 49, 55, 61, 64] aim to learn hash functions that can map data points to Hamming space where the semantic similarity can be preserved. Kernel-based supervised hashing (KSH) [40] uses inner products to approximate the Hamming distance and learns hash functions by perserving semantic similarities in Hamming space. Fast supervised discrete hashing (FSDH) [22] uses a simple yet effective regression from the class labels of training data points to the corresponding hash code to accelerate the learning process. Convolutional neural networks-based hashing (CNNH) [61] decomposes the hash function learning into two stages. Firstly, a pairwise similarity matrix is constructed and decomposed into the product of approximate hash codes. Secondly, CNNH simultaneously learns representations and hash functions by training the model to predict the learned hash codes as well as the discrete image class labels. Deep Cauchy hashing (DCH) [5] adopts Cauchy distribution to continue to optimize data pairs in a relatively small Hamming ball.

Unsupervised hashing methods [3, 21, 25, 39, 41] try to encode original data into binary codes by training with unlabeled data points. Iterative quantization (ITQ) [21] first uses principal component analysis (PCA) to map the data to a low dimensional space and then exploits an alternating minimization scheme to find a rotation matrix, which maps the data to binary codes with minimum quantization error. Discrete graph hashing (DGH) [39] casts the graph hashing problem into a discrete optimization framework and explicitly deals with the discrete constraints, so it can directly output binary codes. Spherical hashing (SpH) [25] minimizes the spherical distance between the original real-valued features and the learned binary codes. Anchor graph hashing (AGH) [41] utilizes anchor graphs to obtain tractable low-rank adjacency matrices and approximate the data structure. Though current traditional unsupervised hashing methods have made great progress, they usually depend on predefined features and cannot simultaneously optimize the feature and hash code learning processes, thus missing an opportunity to learn more effective hash codes.

Unsupervised deep hashing methods [11, 17, 30, 36, 37, 52, 63] adopt deep architectures to extract features and perform hash mapping. Semantic hashing [52] uses pre-trained restricted Boltzmann machines (RBMs) [46] to construct an auto-encoder network, which is then used to generate efficient hash codes and reconstruct the original inputs. Deep binary descriptors (DeepBit) [36] considers original images and the corresponding rotated images as similar pairs and tries to learn hash codes to preserve this similarity. Stochastic generative hashing (SGH) [11] utilizes a generative mechanism to learn hash functions through the minimum description length principle. The hash codes are optimized to maximally compress the dataset as well as to regenerate the inputs. Semantic structure-based unsupervised deep hashing (SSDH) [63] takes advantage of the semantic information in deep features and learns semantic structures based on the pairwise distances and a Gaussian estimation. The semantic structure is then used to guide the hash code learning process. By integrating the feature and hash code learning processes, deep unsupervised hashing methods usually produce better results.

Training classifiers from noisy labels is also a closely related task. We refer the noisy labels to the setting where the labels of data points are corrupted [4, 23, 24, 69]. Since in many situations, it is both expensive and difficult to obtain reliable labels, a growing body of literature has been devoted to learning with noisy labels. Those methods can be organized into two major groups: label noise-tolerant classification [2, 45] and label noise cleansing meth-

ods [8, 38, 47, 50]. The former adopts the strategies like decision trees or boosting-based ensemble techniques, while the latter tries to filter the label noise by exploiting the prior information from training samples. For a comprehensive understanding, we recommend readers to read [20]. By treating the initial similarity relationship as noisy labels, our method can explicitly model the relationship between noisy labels and labels assigned by the Bayes optimal classifiers, which then enables us to extract data pairs with confident similarity signals.

3. Approach

Let $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^N$ denote the training set with N instances, deep hashing aims to learn nonlinear hash functions $h : \boldsymbol{x} \mapsto \boldsymbol{b} \in \{-1, 1\}^K$, which can encode original data points $\boldsymbol{x}$ to compact K-bit hash codes.

Traditional supervised deep hashing methods usually accept data pairs $\{(\boldsymbol{x}_i, \boldsymbol{x}_j), S_{ij}\}$ as inputs, where $S_{ij} \in \{+1, -1\}$ is a binary label to indicate whether $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are similar or not. However, due to the laborious labeling process and the need of requisite domain knowledge, it's not feasible to directly acquire labels in many tasks. Thus, in this paper, we study the hashing problem under unsupervised settings.

Inspired by the Bayesian classifier theory [18], reliable labels for data pairs can be confidently assigned by an Bayes optimal classifier, i.e.,

$$S_{ij} = \begin{cases} 1, & \text{if } \eta(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0.5, \\ -1, & \text{if } \eta(\boldsymbol{x}_i, \boldsymbol{x}_j) < 0.5, \end{cases} \tag{1}$$

where $\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) = P(S_{ij} = +1|\boldsymbol{x}_i, \boldsymbol{x}_j)$. This Bayes optimal classifier implies that if we have access to $\eta(\boldsymbol{x}_i, \boldsymbol{x}_j)$, we can infer the true data labels with Eq. 1. However, under unsupervised settings, we cannot access $\eta(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

For unsupervised learning, some recent works [34, 51, 63] demonstrate that local structures learned from original features can help to capture the similarity relationship between points. Motivated by this, we can roughly label the training data pairs based on their local structures and construct a similarity matrix $\tilde{\boldsymbol{S}}$ as

$$\tilde{S}_{ij} = \begin{cases} 1, & \text{if } d(i, j) \leq t_1, \\ -1, & \text{if } d(i, j) > t_2, \end{cases} \tag{2}$$

where $d(i, j)$ denotes the distance of extracted features for $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, t_1 and t_2 are the thresholds for the distance. However, since $\tilde{\boldsymbol{S}}$ is only constructed from local structures, they are unreliable and may contain label noise.

Note that, based on $\tilde{\boldsymbol{S}}$ we can learn an estimation of the conditional probability $\tilde{\eta}(\boldsymbol{x}_i, \boldsymbol{x}_j) = P(\tilde{S}_{ij} = +1|\boldsymbol{x}_i, \boldsymbol{x}_j)$. And, there exists a relationship between $\tilde{\eta}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and $\eta(\boldsymbol{x}_i, \boldsymbol{x}_j)$ as

$$\begin{aligned} &\tilde{\eta}(\boldsymbol{x}_i, \boldsymbol{x}_j) = P(\tilde{S}_{ij} = +1|\boldsymbol{x}_i, \boldsymbol{x}_j) \\ &= P(\tilde{S}_{ij} = +1|\boldsymbol{x}_i, \boldsymbol{x}_j, S_{ij} = +1)P(S_{ij} = +1|\boldsymbol{x}_i, \boldsymbol{x}_j) \\ &\quad + P(\tilde{S}_{ij} = +1|\boldsymbol{x}_i, \boldsymbol{x}_j, S_{ij} = -1)P(S_{ij} = -1|\boldsymbol{x}_i, \boldsymbol{x}_j) \\ &= (1 - \rho_{+1}(\boldsymbol{x}_i, \boldsymbol{x}_j))\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ &\quad + \rho_{-1}(\boldsymbol{x}_i, \boldsymbol{x}_j)(1 - \eta(\boldsymbol{x}_i, \boldsymbol{x}_j)), \end{aligned} \tag{3}$$

where $\rho_{S_{ij}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = P(\tilde{S}_{ij} = -S_{ij}|\boldsymbol{x}_i, \boldsymbol{x}_j, S_{ij})$ denotes the flip rate between true labels and noisy labels on given data pair $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and their label S_{ij}. If we know the values of $\rho_{S_{ij}}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and $\tilde{\eta}(\boldsymbol{x}_i, \boldsymbol{x}_j)$, the values of $\eta(\boldsymbol{x}_i, \boldsymbol{x}_j)$ can be easily inferred. However, the values of $\rho_{S_{ij}}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ are unknown. From Eq. 3 we can further get that, when the flip rates $\rho_{+1}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and $\rho_{-1}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ are relatively small, if $\tilde{\eta}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is large, $\eta(\boldsymbol{x}_i, \boldsymbol{x}_j)$ should also be large, and vice versa. In the following subsection, we show that it is possible to infer whether $\eta(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is smaller or larger than 0.5 based on some weak information[1] of $\rho_{S_{ij}}(\boldsymbol{x}_i, \boldsymbol{x}_j)$, which means we may potentially achieve the reliable labels for some data pairs. We define those data pairs of which reliable labels can be recovered from $\tilde{S}$ as distilled data pairs.

In the following subsection, we theoretically prove that distilled data pairs can be extracted under a mild assumption. And we further provide a method to collect distilled data pairs automatically.

3.1. Collecting Distilled Data Pairs Automatically

To collect distilled data pairs, we first give the following assumption.

Assumption 1. *For any data pairs* $\{(\boldsymbol{x}_i, \boldsymbol{x}_j), i, j = 1, ...N\}$*, we have*

$$0 \leq \rho_{+1}(\boldsymbol{x}_i, \boldsymbol{x}_j) + \rho_{-1}(\boldsymbol{x}_i, \boldsymbol{x}_j) \leq 1. \tag{4}$$

This assumption implies that label noise is not too heavy. Note that, if the number of correctly labeled data pairs is considered larger than that of mislabeled data pairs, the flip rate $\rho_{S_{ij}}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ will be bounded by 0.5. We can see that Assumption 1 is much weaker than the above assumption.

It is hard to prove that the noisy labels constructed by exploiting local structures satisfy Assumption 1. However, the experimental results on three widely used benchmark datasets empirically verify that the assumption applies well to the constructed noisy labels. In the rest of this paper, we always suppose Assumption 1 holds.

We then extend the noisy label learning techniques in [8] to pairwise labels and present the following key theorem, which gives the basic criteria to collect distilled data pairs.

[1] As many of the labels in $\tilde{S}$ are correct, we show later that upper bounds for $\rho_{S_{ij}}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ can be easily obtained.

Theorem 1. *For any data pairs* $\{(\boldsymbol{x}_i, \boldsymbol{x}_j), i, j = 1, ...N\}$, *we have*

if $\tilde{\eta}(\boldsymbol{x}_i, \boldsymbol{x}_j) < \frac{1-\rho_{+1}(\boldsymbol{x}_i, \boldsymbol{x}_j)}{2}$, *then* $\{(\boldsymbol{x}_i, \boldsymbol{x}_j), s_{ij} = -1\}$ *is a distilled data pair;*

if $\tilde{\eta}(\boldsymbol{x}_i, \boldsymbol{x}_j) > \frac{1+\rho_{-1}(\boldsymbol{x}_i, \boldsymbol{x}_j)}{2}$, *then* $\{(\boldsymbol{x}_i, \boldsymbol{x}_j), s_{ij} = +1\}$ *is a distilled data pair.*

Proof. According to Eq. 3, for any data pair $\{(\boldsymbol{x}_i, \boldsymbol{x}_j) | \eta(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0.5, i, j = 1, ..., N\}$, we have

$$\begin{aligned} \tilde{\eta}(\boldsymbol{x}_i, \boldsymbol{x}_j) &= (1 - \rho_{+1}(\boldsymbol{x}_i, \boldsymbol{x}_j))\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ &\quad + \rho_{-1}(\boldsymbol{x}_i, \boldsymbol{x}_j)(1 - \eta(\boldsymbol{x}_i, \boldsymbol{x}_j)) \\ &= \eta(\boldsymbol{x}_i, \boldsymbol{x}_j)(1 - \rho_{+1}(\boldsymbol{x}_i, \boldsymbol{x}_j) - \rho_{-1}(\boldsymbol{x}_i, \boldsymbol{x}_j)) \\ &\quad + \rho_{-1}(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ &\geq \frac{1 - \rho_{+1}(\boldsymbol{x}_i, \boldsymbol{x}_j) + \rho_{-1}(\boldsymbol{x}_i, \boldsymbol{x}_j)}{2} \\ &\geq \frac{1 - \rho_{+1}(\boldsymbol{x}_i, \boldsymbol{x}_j)}{2}. \end{aligned} \tag{5}$$

The first inequality holds since $\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0.5$ and $\rho_{+1}(\boldsymbol{x}_i, \boldsymbol{x}_j) + \rho_{-1}(\boldsymbol{x}_i, \boldsymbol{x}_j) \leq 1$. Based on Eq. 5, we have

$$\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0.5 \Rightarrow \tilde{\eta}(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq \frac{1 - \rho_{+1}(\boldsymbol{x}_i, \boldsymbol{x}_j)}{2},$$

which implies that

$$\tilde{\eta}(\boldsymbol{x}_i, \boldsymbol{x}_j) < \frac{1 - \rho_{+1}(\boldsymbol{x}_i, \boldsymbol{x}_j)}{2} \Rightarrow \eta(\boldsymbol{x}_i, \boldsymbol{x}_j) < 0.5.$$

Combining this result with Eq. 1, we can label data pair $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ with $S_{ij} = -1$, if $\tilde{\eta}(\boldsymbol{x}_i, \boldsymbol{x}_j) < \frac{1-\rho_{+1}(\boldsymbol{x}_i, \boldsymbol{x}_j)}{2}$. Similarly, we can prove that data pairs $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ with $\tilde{\eta}(\boldsymbol{x}_i, \boldsymbol{x}_j) > \frac{1+\rho_{-1}(\boldsymbol{x}_i, \boldsymbol{x}_j)}{2}$ can be labeled with $S_{ij} = +1$. □

The trade-off for selecting distilled data pairs is the need of estimating the conditional probability $\tilde{\eta}$ and the flip rate $\rho_{S_{ij}}(\boldsymbol{x}_i, \boldsymbol{x}_j)$. To estimate $\tilde{\eta}$, we adopt a probabilistic classification method. Specifically, we design a deep network to map data pairs to probabilities. Since this objective is similar to the hash code learning, we explore the same architecture for the estimation of $\tilde{\eta}$ and hash code learning. The detailed description of this deep network will be presented in the next subsection.

For the estimation of the flip rate $\rho_{S_{ij}}(\boldsymbol{x}_i, \boldsymbol{x}_j)$, most existing works [38, 50] assume the noise to be label- and instance-independent or instance-independent. While in our method, the flip rate should be label- and instance-dependent, so most existing methods are not suitable for the current problem. Considering the difficulty to directly estimate the flip rate, we alternatively propose a method to obtain an upper bound. Formally, we give the following proposition.

Proposition 1. *Given the conditional probability* $\tilde{\eta}(\boldsymbol{x}_i, \boldsymbol{x}_j)$, *the following inequations holds*

$$\begin{aligned} \rho_{-1}(\boldsymbol{x}_i, \boldsymbol{x}_j) &\leq \tilde{\eta}(\boldsymbol{x}_i, \boldsymbol{x}_j), \\ \rho_{+1}(\boldsymbol{x}_i, \boldsymbol{x}_j) &\leq 1 - \tilde{\eta}(\boldsymbol{x}_i, \boldsymbol{x}_j). \end{aligned} \tag{6}$$

Proof. According to Eq. 3, we can get

$$\begin{aligned} \tilde{\eta}(\boldsymbol{x}_i, \boldsymbol{x}_j) &= (1 - \rho_{+1}(\boldsymbol{x}_i, \boldsymbol{x}_j))\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ &\quad + \rho_{-1}(\boldsymbol{x}_i, \boldsymbol{x}_j)(1 - \eta(\boldsymbol{x}_i, \boldsymbol{x}_j)) \\ &= \eta(\boldsymbol{x}_i, \boldsymbol{x}_j)(1 - \rho_{+1}(\boldsymbol{x}_i, \boldsymbol{x}_j) - \rho_{-1}(\boldsymbol{x}_i, \boldsymbol{x}_j)) \\ &\quad + \rho_{-1}(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq \rho_{-1}(\boldsymbol{x}_i, \boldsymbol{x}_j). \end{aligned} \tag{7}$$

The inequality holds because $\rho_{+1}(\boldsymbol{x}_i, \boldsymbol{x}_j) + \rho_{-1}(\boldsymbol{x}_i, \boldsymbol{x}_j) \leq 1$. Similarly, it gives $\rho_{+1}(\boldsymbol{x}_i, \boldsymbol{x}_j) \leq 1 - \tilde{\eta}(\boldsymbol{x}_i, \boldsymbol{x}_j)$. □

However, if we directly combine Proposition 1 and Theorem 1, no distilled data pairs can be selected. So, here we further assume the flip rate to be local invariant, and thus obtain the flip rate upper bounds as

$$\begin{aligned} &\rho_{-1max}(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ &= min\{\tilde{\eta}(\boldsymbol{x}_k, \boldsymbol{x}_l)|, \boldsymbol{x}_k \in nn_o(\boldsymbol{x}_i), \boldsymbol{x}_l \in nn_o(\boldsymbol{x}_j)\}, \\ &\rho_{+1max}(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ &= min\{(1 - \tilde{\eta}(\boldsymbol{x}_k, \boldsymbol{x}_l))|, \boldsymbol{x}_k \in nn_o(\boldsymbol{x}_i), \boldsymbol{x}_l \in nn_o(\boldsymbol{x}_j)\}, \end{aligned} \tag{8}$$

where $nn_o(\boldsymbol{x}_i)$ indicates the set of top o nearest neighbors for $\boldsymbol{x}_i$.

With the flip rate upper bounds $\rho_{+1max}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and $\rho_{-1max}(\boldsymbol{x}_i, \boldsymbol{x}_j)$, we have

$$\begin{aligned} \frac{1 - \rho_{+max}(\boldsymbol{x}_i, \boldsymbol{x}_j)}{2} &\leq \frac{1 - \rho_{+1}(\boldsymbol{x}_i, \boldsymbol{x}_j)}{2} \\ \frac{1 + \rho_{-max}(\boldsymbol{x}_i, \boldsymbol{x}_j)}{2} &\geq \frac{1 + \rho_{-1}(\boldsymbol{x}_i, \boldsymbol{x}_j)}{2}. \end{aligned} \tag{9}$$

The conditional probability $\tilde{\eta}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ can be estimated by the adopted deep networks, and the noisy rate upper bound can be acquired with Eq. 8. Combining these results with Eq. 9 and Theorem 1, we can find that the distilled data pairs can be successfully collected by picking out every pairs $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ that satisfy $\tilde{\eta}(\boldsymbol{x}_i, \boldsymbol{x}_j) > \frac{1+\rho_{-1max}(\boldsymbol{x}_i, \boldsymbol{x}_j)}{2}$ and assigning label $S_{ij} = +1$, and picking out every pairs $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ that satisfy $\tilde{\eta}(\boldsymbol{x}_i, \boldsymbol{x}_j) < \frac{1-\rho_{+1max}(\boldsymbol{x}_i, \boldsymbol{x}_j)}{2}$ and assigning label $S_{ij} = -1$. The distilled data pair set can be represented as $\{(\boldsymbol{x}_i, \boldsymbol{x}_j, S_{ij}), i, j = 1, ...m\}$, where m is the number of distilled data pairs.

After obtaining the distilled data pair set, we can perform hash code learning, which is similar to the learning process for supervised hashing. Specifically, we adopt a Bayesian learning framework, which is elaborated in the following subsection.

3.2. Bayesian Learning Framework

In this subsection, we propose a Bayesian learning framework to perform deep hashing learning and also estimate the conditional probability $\tilde{\eta}(\boldsymbol{x}_i, \boldsymbol{x}_j)$. We first introduce the framework for hash code learning and then show how to apply it for the estimation of $\tilde{\eta}(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

By representing the hash codes for distilled data as $\boldsymbol{B} = [\boldsymbol{b}_1, ..., \boldsymbol{b}_m]$, the Maximum Likelihood (ML) estimation of the hash codes can be defined as:

$$\log P(S|\boldsymbol{B}) = \frac{1}{m^2} \sum_{i=1}^{m} \sum_{j=1}^{m} \log P(S_{ij}|\boldsymbol{b}_i, \boldsymbol{b}_j), \qquad (10)$$

where $P(S_{ij}|\boldsymbol{b}_i, \boldsymbol{b}_j)$ is the conditional probability of similarity label S_{ij} given the hash codes $\boldsymbol{b}_i$ and $\boldsymbol{b}_j$, which can be naturally approximated by a pairwise logistic function

$$\log P(S_{ij}|\boldsymbol{b}_i, \boldsymbol{b}_j) = \begin{cases} \sigma(\langle \boldsymbol{b}_i, \boldsymbol{b}_j \rangle) & S_{ij} = 1, \\ 1 - \sigma(\langle \boldsymbol{b}_i, \boldsymbol{b}_j \rangle) & S_{ij} = -1, \end{cases} \qquad (11)$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function and $\langle \boldsymbol{b}_i, \boldsymbol{b}_j \rangle$ denotes the inner product of the hash codes $\boldsymbol{b}_i$ and $\boldsymbol{b}_j$. Here, we adopt the inner product, since as indicated in [40], the Hamming distance $dist_H(\cdot, \cdot)$ of hash codes can be inferred from the inner product $\langle \cdot, \cdot \rangle$ as: $dist_H(\boldsymbol{b}_i, \boldsymbol{b}_j) = \frac{1}{2}(K - \langle \boldsymbol{b}_i, \boldsymbol{b}_j \rangle)$. Hence, the inner product can well reflect the Hamming distance for binary hash codes.

Similar to logistic regression, we can find that the smaller the Hamming distance $dist_H(\boldsymbol{b}_i, \boldsymbol{b}_j)$ is, the larger the inner product results $\langle \boldsymbol{b}_i, \boldsymbol{b}_j \rangle$ and the conditional probability $P(1|\boldsymbol{b}_i, \boldsymbol{b}_j)$ will be. Otherwise, the larger the conditional probability $P(-1|\boldsymbol{b}_i, \boldsymbol{b}_j)$ will be. These results imply that similar data points will be enforced to have small Hamming distance and dissimilar data points will be enforced to have large Hamming distance, which are expected for Hamming space similarity search. So, learning with Eq. 10, effective hash codes can be obtained.

After training the model, given a data point, we can obtain its hash codes by directly forward propagating it through the adopted network, and obtain the final binary codes by the following sign function

$$\text{sign}(x) = \begin{cases} 1 & \text{if } x \geq 0, \\ -1 & \text{if } x < 0. \end{cases} \qquad (12)$$

The whole learning algorithm is summarized in Algorithm 1.

Since this framework maps data pairs to similarity probabilities, we can also use it to estimate the conditional probability. The main difference lies in that, for hash code learning we use distilled data pairs as inputs, while for conditional probability estimation, we use the data pairs constructed from local structures as inputs.

Algorithm 1: DistillHash

Training Stage

Input: Training images $\mathbf{X}$, code length K, mini-batch size t, hyper-parameters o and p.
Procedure:
1. Construct initial noisy similarity labels with Eq. (2).
2. Estimate the conditional noisy label probability $\tilde{\eta}(\cdot, \cdot)$ for all training data pairs.
3. Estimate the flip rate upper bounds for all training data pairs with Eq. (8).
4. Distill data pairs with Theorem 1.
repeat
  3.1 Randomly sample t data pairs from the distilled data pair set as inputs.
  3.2 Calculate the outputs by forward-propagation through the adopted networks.
  3.3 Update parameters of the network by minimizing Eq. (10).
until *convergence*;

Testing Stage

Input: Image query $\mathbf{q}_i$, parameters for the adopted network.
Procedure:
1. Calculate the output of the neural network by directly forward-propagating the input images.
2. Obtain hash codes with the sign function.

4. Experiments

We evaluate our method on three popular benchmark datasets, **FLICKR25K**, **NUSWIDE**, and **CIFAR10**, and provide extensive evaluations to demonstrate its performance. In this section, we first introduce the datasets and then present our experimental results.

4.1. Datasets

FLICKR25K [26] contains 25,000 images collected from the Flickr website. Each image is manually annotated with at least one of the 24 unique labels provided. We randomly select 2,000 images as a test set; the remaining images are used as a retrieval set, from which we randomly select 5,000 images as a training set. **NUSWIDE** [10] contains 269,648 images, each of the images is annotated with multiple labels referring to 81 concepts. The subset containing the 10 most popular concepts is used here. We randomly select 5,000 images as a test set; the remaining images are used as a retrieval set, and 10,500 images are randomly selected from the retrieval set as the training set. **CIFAR10** [29] is a popular image dataset containing 60,000 images in 10 classes. For each class, we randomly select 1,000 images as queries and 500 as training images, result-

ing in a query set containing 10,000 images and a training set made up of 5,000 images. All images except for the query set are used as the retrieval set.

4.2. Baseline Methods

The proposed method is compared with six state-of-the-art traditional unsupervised hashing methods (LSH [3], SH [60], ITQ [21], PCAH [59], DSH [28], and SpH [25]) and three recently proposed deep unsupervised hashing methods (DeeBit [36], SGH [11], and SSDH [63]). All the codes for these methods have been kindly provided by the authors. LSH, SH, ITQ, PCAH, DSH, and SpH are implemented with MATLAB, SGH and SSDH are implemented with TensorFlow [1], and DeepBit is implemented with Caffe [27]. We use TensorFlow when write our code, and run the algorithm in a machine with one Titan X Pascal GPU.

4.3. Evaluation.

To evaluate the performance of our proposed method, we adopt three evaluation criteria: mean of average precision (MAP), topN-precision, and precision-recall. The first two criteria are based on Hamming ranking, which ranks data points based on their Hamming distances to the query; for its part, precision-recall is based on hash lookup. More detailed introductions are given as follows.

MAP is one of the most widely-used criteria for evaluating retrieval accuracy. Given a query and a list of R ranked retrieval results, the average precision (AP) for this query can be computed. MAP is then defined as the average of APs for all queries. For all three datasets, we set R as the number of the retrieval set. **TopN-precision** is defined as the average ratio of similar instances among the top N retrieved instances for all queries in terms of Hamming distance. In our experiments, N is set to 1,000. **Precision-recall** reveals the precisions at different recall levels and is a good indicator of overall performance. Typically, the area under the precision-recall curve is computed. A larger Precision-recall value always indicates better performance.

4.4. Implementation Details

To initialize the noisy similarity matrix in Eq. (2), we select the cosine distance as the distance to measure the local structure of training examples. The threshold t_1 and t_2 are selected as indicated in [63]. For the adopted deep networks, we use the VGG16 architecture [54] and replace the last fully-connected layer with a new fully-connected layer with K units for hash code learning. For the estimation of the conditional probability $\tilde{\eta}$, we set the dimensions of the last fully-connected layer as p, which is 48 in our experiments. To obtain the upper bound of the flip rate, we set o as 4. The parameter sensitivity of our algorithm with regard to o and p are analyzed in Subsection 4.6. Parameters for the new fully connected layer are learned from scratch, while parameters for the preceding layers are fine-tuned from the model pre-trained on ImageNet [16]. We employ the standard stochastic gradient descent algorithm with 0.9 momentum for optimization, min-batch size is set to 64, and the learning rate is fixed to 10^{-3}. Two data points are considered neighbors if they share the same label (for CIFAR10) or share at least one common label (for the multi-label datasets FLICKR25K and NUSWIDE).

For a fair comparison, we adopt the deep features extracted from the last fully-connected layer from the VGG16 network pre-trained on ImageNet for all shallow architecture-based baseline methods. These deep features are also used for the construction of $\tilde{S}$. Since VGG16 accepts images of size 224×224 as inputs, we resize all images to be 224×224 before inputting them into the VGG16 network. Random rotation and flipping are also used for data augmentation.

4.5. Results and Discussion

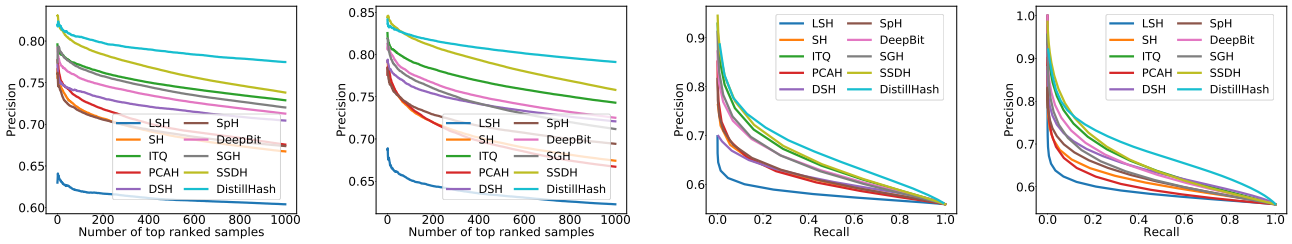
We first present the MAP values for all methods with different hash bit lengths, then draw precision-recall and TopN-precision curves for all methods with 32 and 64 hash code lengths to give a more comprehensive comparison.

Table 1 presents the MAP results for DistillHash and all baseline methods on FLICKR25K, NUSWIDE, and CIFAR10, with hash code numbers varying from 16 to 128. By comparing the data-independent method LSH with other data-dependent methods, we can see that data-dependent hashing methods outperform the data-independent hashing method in most cases. This may be because that data-dependent methods learn hash functions from data, so can better capture the used data structures. By comparing deep hashing methods and no-deep hashing methods, we find that no-deep hashing methods can surpass deep hashing methods DeepBit and SGH in some cases. This may be because that, without proper supervisory signals, deep hashing methods cannot fully exploit the representation ability of deep networks, and may achieve unsatisfactory performance by over-fitting to bad local minima. While, by exploiting local structures, deep hashing methods (SSDH and DistillHash) achieve more promising results.

Concretely, from the MAP results, we can see that DistillHash consistently obtains the best results across different hash bit lengths for all three datasets. Specifically, compared to one of the best non-deep hashing methods, i.e, ITQ, we achieve absolute improvements of 6.89%, 13.97%, and 7.73% in the average MAP for different bits on FLICKR25K, NUSWIDE, and CIFAR10 respectively. Compared to the state-of-the-art deep hashing method SSDH, we achieve absolute improvements of 3.08%, 4.01%, and 2.86% in average MAP for different bits on the three datasets respectively. Note that DeepBit,

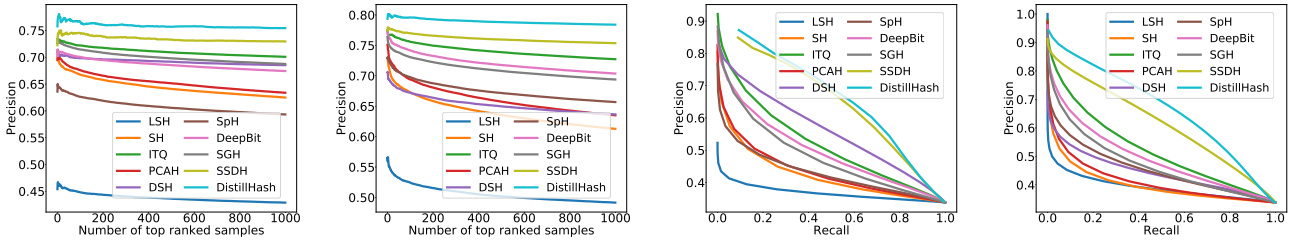
Table 1. Comparison with baselines in terms of MAP. The best accuracy is shown in boldface.

method	FLICKR25K				NUSWIDE				CIFAR10			
	16 bits	32 bits	64 bits	128 bits	16 bits	32 bits	64 bits	128 bits	16 bits	32 bits	64 bits	128 bits
LSH [3]	0.5831	0.5885	0.5933	0.6014	0.4324	0.4411	0.4433	0.4816	0.1319	0.1580	0.1673	0.1794
SH [60]	0.5919	0.5923	0.6016	0.6213	0.4458	0.4537	0.4926	0.5000	0.1605	0.1583	0.1509	0.1538
ITQ [21]	0.6192	0.6318	0.6346	0.6477	0.5283	0.5323	0.5319	0.5424	0.1942	0.2086	0.2151	0.2188
PCAH [59]	0.6091	0.6105	0.6033	0.6071	0.4625	0.4531	0.4635	0.4923	0.1432	0.1589	0.1730	0.1835
DSH [28]	0.6074	0.6121	0.6118	0.6154	0.5200	0.5227	0.5345	0.5370	0.1616	0.1876	0.1918	0.2055
SpH [25]	0.6108	0.6029	0.6339	0.6251	0.4532	0.4597	0.4958	0.5127	0.1439	0.1665	0.1783	0.1840
DeepBit [36]	0.5934	0.5933	0.6199	0.6349	0.4542	0.4625	0.47616	0.4923	0.2204	0.2410	0.2521	0.2530
SGH [11]	0.6162	0.6283	0.6253	0.6206	0.4936	0.4829	0.4865	0.4975	0.1795	0.1827	0.1889	0.1904
SSDH [63]	0.6621	0.6733	0.6732	0.6771	0.6231	0.6294	0.6321	0.6485	0.2568	0.2560	0.2587	0.2601
DistillHash	**0.6964**	**0.7056**	**0.7075**	**0.6995**	**0.6667**	**0.6752**	**0.6769**	**0.6747**	**0.2844**	**0.2853**	**0.2867**	**0.2895**



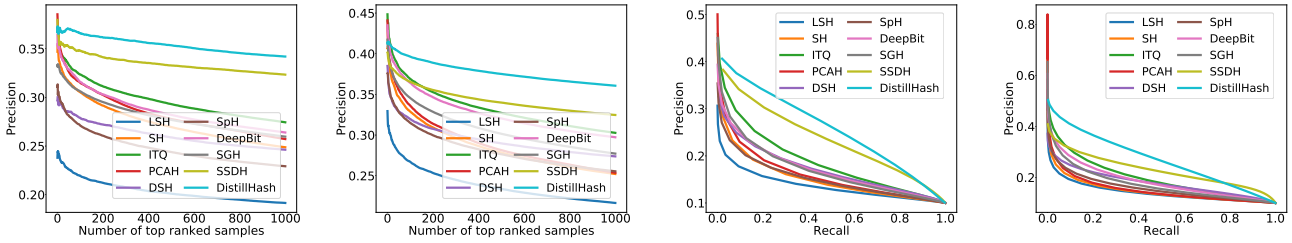
(a) TopN-precision with 16bits (b) TopN-precision with 32bits (c) Precision-recall with 16bits (d) Precision-recall with 32bits

Figure 1. TopN-precision and precision-recall curves on FLICKR25K with 16 and 32 hash bits.



(a) TopN-precision with 16bits (b) TopN-precision with 32bits (c) Precision-recall with 16bits (d) Precision-recall with 32bits

Figure 2. TopN-precision and precision-recall curves on NUSWIDE with 16 and 32 hash bits.



(a) TopN-precision with 16bits (b) TopN-precision with 32bits (c) Precision-recall with 16bits (d) Precision-recall with 32bits

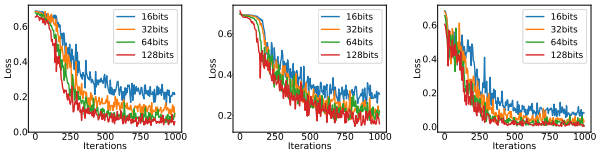
Figure 3. TopN-precision and precision-recall curves on CIFAR10 with 16 and 32 hash bits.

SGH, SSDH, and DistillHash are both deep hashing methods, only SSDH and DistillHash can exploit and preserve the similarity of different data points, thus they can achieve better performance than the other two. Moreover, DistillHash learns more accurate similarity relationships by distilling some data pairs, so can obtain a further performance improvement than SSDH.

The left two subfigures of Figure 1, 2, and 3 present the TopN-precision curves for all methods on each of the three datasets with hash bit lengths of 16 and 32. Consistent with MAP results, we can observe that DistillHash achieves the best results among all approaches. Since MAP values and TopN-precision curves are both Hamming ranking-based metrics, an overview of the above analysis reveals

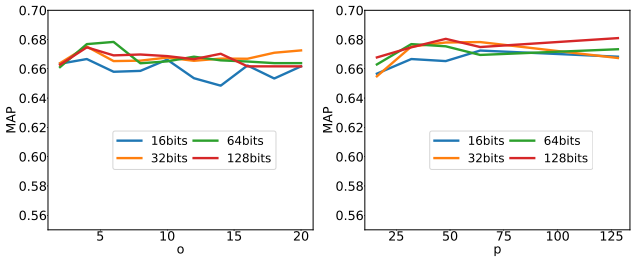
Table 2. MAP results for DistillHash* and DistillHash. The best accuracy is shown in boldface.

method	FLICKR25K				NUSWIDE				CIFAR10			
	16 bits	32 bits	64 bits	128 bits	16 bits	32 bits	64 bits	128 bits	16 bits	32 bits	64 bits	128 bits
DistillHash*	0.6653	0.6633	0.6726	0.6784	0.6322	0.6357	0.6480	0.6451	0.2547	0.2538	0.2573	0.2583
DistillHash	**0.6964**	**0.7056**	**0.7075**	**0.6995**	**0.6667**	**0.6752**	**0.6769**	**0.6747**	**0.2844**	**0.2853**	**0.2867**	**0.2895**



(a) FLICKR25K (b) NUSWIDE (c) CIFAR10

Figure 4. Losses of DistillHash through the training process.



(a) MAP w.r.t. different o. (b) MAP w.r.t. different p.

Figure 5. Parameter sensitive analysis for o and p on NUSWIDE.

that DistillHash can achieve superior performance for Hamming ranking-based evaluations. Moreover, to illustrate the hash lookup results, we plot the precision-recall curves for all methods with hash bit lengths of 16 and 32 in the right two subfigures of Figure 1, 2, and 3. From the results, we can again observe that DistillHash consistently achieves the best performance, which further demonstrates the superiority of our proposed method.

To investigate the change of loss values through the training process, we display the loss values in Figure 4. The results reveal that our methods can converge in all cases within 1,000 iterations.

4.6. Parameter Sensitivity

We next investigate the sensitivity of hyper-parameters o and p. Figure 5 shows the effect of these two hyper-parameters on NUSWIDE dataset with hash code lengths of 16, 32, 64, and 128. We first fix p to 48 and evaluate the MAP by varying o between 2 and 20, the results are presented in Figure 5(a). The performance shows that the algorithm is not sensitive to parameter o in the range of $[2, 20]$, and we can set o as any number in the range of $[2, 20]$. In our experiments, we set o as 4. Figure 5(b) shows the MAP by varying p between 16 and 128 with o fixed to 4. The performance of DistillHash first increases and then keeps at a relatively high level. The result is also not sensitive to p in the range of $[32, 128]$. For other experiments in this paper, we select p as 48.

4.7. Ablation Study

In this subsection, we go deeper to study the efficacy of the proposed distilled data pair learning. More specifically, we investigate DistillHash*, a variant of DistillHash with the same Bayesian learning framework but trained with the initial similarity label $\tilde{\boldsymbol{S}}$. The MAP results of DistillHash* and DistillHash are shown in Table 2, from which we can see that DistillHash consistently outperforms DistillHash* by margins of 3.11%, 4.23%, 3.49% and 2.11% for the FLICKR25K dataset, 3.45%, 3.95%, 2.89% and 2.96% for the NUSWIDE dataset, and 2.97%, 3.15%, 2.94% and 3.12% for the CIFAR10 dataset at hash bit lengths of 16, 32, 64, and 128 respectively. Note that the only difference between DistillHash and DistillHash* lies in that DistillHash is trained with distilled data set and DistillHash* is trained with the initial data set. The performance improvements clearly demonstrate the superiority of the proposed distilled data pair learning.

5. Conclusions

This work presented a new unsupervised deep hashing approach for image search, namely DstilHash. Firstly, we theoretically investigated the relationship between the Bayes optimal classifier and noisy labels learned from local structures, showing that distilled data pairs can be potentially collected. Secondly, with the above understanding, we provided a simple yet effective scheme to automatically distill data pairs. Thirdly, leveraging a distilled data set, we designed a deep hashing model and adopted a Bayesian learning framework to perform the hash code learning. The experimental results on three benchmark datasets demonstrated that the proposed DistillHash surpasses other state-of-the-art methods.

6. Acknowledgements

This work was also supported in part by the National Natural Science Foundation of China under Grant 61572388 and 61703327, in part by the Key R&D Program-The Key Industry Innovation Chain of Shaanxi under Grant 2017ZDCXL-GY-05-04-02, 2017ZDCXL-GY-05-02 and 2018ZDXM-GY-176, in part by the National Key R&D Program of China under Grant 2017YFE0104100, and in part by the Australian Research Council Projects DP-180103424, DE-1901014738, and FL-170100117.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016. 6

[2] Kamal M Ali and Michael J Pazzani. Error reduction through learning multiple descriptions. *Machine Learning*, 24(3):173–202, 1996. 2

[3] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *FOCS*, pages 459–468, 2006. 1, 2, 6, 7

[4] Battista Biggio, Blaine Nelson, and Pavel Laskov. Support vector machines under adversarial label noise. In *Asian Conference on Machine Learning*, pages 97–112, 2011. 2

[5] Yue Cao, Mingsheng Long, Bin Liu, Jianmin Wang, and MOE KLiss. Deep cauchy hashing for hamming space retrieval. In *CVPR*, pages 1229–1237, 2018. 2

[6] Xinyuan Chen, Chang Xu, Xiaokang Yang, and Dacheng Tao. Attention-gan for object transfiguration in wild images. In *ECCV*, pages 164–180, 2018. 1

[7] Zhixiang Chen, Xin Yuan, Jiwen Lu, Qi Tian, and Jie Zhou. Deep hashing via discrepancy minimization. In *CVPR*, June 2018. 1

[8] Jiacheng Cheng, Tongliang Liu, Kotagiri Ramamohanarao, and Dacheng Tao. Learning with bounded instance-and label-dependent label noise. *arXiv preprint arXiv:1709.03768*, 2017. 2, 3

[9] Lianhua Chi, Bin Li, Xingquan Zhu, Shirui Pan, and Ling Chen. Hashing for adaptive real-time graph stream classification with concept drifts. *IEEE Trans. Cybern.*, 48(5):1591–1604, 2018. 1

[10] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. Nus-wide: a real-world web image database from national university of singapore. In *CIVR*, page 48, 2009. 5

[11] Bo Dai, Ruiqi Guo, Sanjiv Kumar, Niao He, and Le Song. Stochastic generative hashing. In *ICML*, 2017. 1, 2, 6, 7

[12] Cheng Deng, Zhaojia Chen, Xianglong Liu, Xinbo Gao, and Dacheng Tao. Triplet-based deep hashing network for cross-modal retrieval. *IEEE Trans. Image Process.*, 27(8):3893–3903, 2018. 2

[13] Cheng Deng, Huiru Deng, Xianglong Liu, and Yuan Yuan. Adaptive multi-bit quantization for hashing. *Neurocomputing*, 151:319–326, 2015. 2

[14] Cheng Deng, Xu Tang, Junchi Yan, Wei Liu, and Xinbo Gao. Discriminative dictionary learning with common label alignment for cross-modal retrieval. *IEEE Trans. Multimedia*, 18(2):208–218, 2016. 2

[15] Cheng Deng, Erkun Yang, Tongliang Liu, Wei Liu, Jie Li, and Dacheng Tao. Unsupervised semantic-preserving adversarial hashing for image search. *IEEE Transactions on Image Processing*, 2019. 1

[16] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255. Ieee, 2009. 6

[17] Kamran Ghasedi Dizaji, Feng Zheng, Najmeh Sadoughi Nourabadi, Yanhua Yang, Cheng Deng, and Heng Huang. Unsupervised deep generative adversarial hashing network. In *CVPR*, 2018. 1, 2

[18] Pedro Domingos and Michael Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine learning*, 29(2-3):103–130, 1997. 3

[19] Yuxuan Du, Min-Hsiu Hsieh, Tongliang Liu, and Dacheng Tao. The expressive power of parameterized quantum circuits. *arXiv preprint arXiv:1810.11922*, 2018. 1

[20] Benoît Frénay and Michel Verleysen. Classification in the presence of label noise: a survey. *IEEE Trans. Neural Netw. Learn. Syst.*, 25(5):845–869, 2014. 3

[21] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(12):2916–2929, 2013. 1, 2, 6, 7

[22] Jie Gui, Tongliang Liu, Zhenan Sun, Dacheng Tao, and Tieniu Tan. Fast supervised discrete hashing. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(2):490–496, 2018. 2

[23] Bo Han, Jiangchao Yao, Gang Niu, Mingyuan Zhou, Ivor Tsang, Ya Zhang, and Masashi Sugiyama. Masking: A new perspective of noisy supervision. In *Advances in Neural Information Processing Systems*, pages 5841–5851, 2018. 2

[24] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Advances in Neural Information Processing Systems*, pages 8536–8546, 2018. 2

[25] Jae-Pil Heo, Youngwoon Lee, Junfeng He, Shih-Fu Chang, and Sung-Eui Yoon. Spherical hashing. In *CVPR*, pages 2957–2964, 2012. 1, 2, 6, 7

[26] Mark J. Huiskes and Michael S. Lew. The mir flickr retrieval evaluation. In *ICMR*, 2008. 5

[27] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM Multimedia*, pages 675–678, 2014. 6

[28] Zhongming Jin, Cheng Li, Yue Lin, and Deng Cai. Density sensitive hashing. *IEEE Trans. Cybern.*, 44(8):1362–1371, 2014. 6, 7

[29] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, 2009. 5

[30] Alex Krizhevsky and Geoffrey E Hinton. Using very deep autoencoders for content-based image retrieval. In *ESANN*, 2011. 1, 2

[31] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012. 1

[32] Chao Li, Cheng Deng, Ning Li, Wei Liu, Xinbo Gao, and Dacheng Tao. Self-supervised adversarial hashing networks for cross-modal retrieval. In *CVPR*, pages 4242–4251, 2018. 2

[33] Ning Li, Chao Li, Cheng Deng, Xianglong Liu, and Xinbo Gao. Deep joint semantic-embedding hashing. In *IJCAI*, pages 2397–2403, 2018. 2

[34] Siyang Li, Bryan Seybold, Alexey Vorobyov, Alireza Fathi, Qin Huang, and C-C Jay Kuo. Instance embedding transfer to unsupervised video object segmentation. In *CVPR*, pages 6526–6535, 2018. 3

[35] Wu-Jun Li, Sheng Wang, and Wang-Cheng Kang. Feature learning based deep supervised hashing with pairwise labels. In *IJCAI*, pages 1711–1717, 2016. 1, 2

[36] Kevin Lin, Jiwen Lu, Chu-Song Chen, and Jie Zhou. Learning compact binary descriptors with unsupervised deep neural networks. In *CVPR*, pages 1183–1192, 2016. 1, 2, 6, 7

[37] Venice Erin Liong, Jiwen Lu, Gang Wang, Pierre Moulin, Jie Zhou, et al. Deep hashing for compact binary codes learning. In *CVPR*, volume 1, page 3, 2015. 2

[38] Tongliang Liu and Dacheng Tao. Classification with noisy labels by importance reweighting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(3):447–461, 2016. 3, 4

[39] Wei Liu, Cun Mu, Sanjiv Kumar, and Shih-Fu Chang. Discrete graph hashing. In *NIPS*, pages 3419–3427, 2014. 1, 2

[40] W. Liu, J. Wang, R. Ji, Y. Jiang, and S.-F. Chang. Supervised hashing with kernels. In *CVPR*, pages 2074–2081, 2012. 1, 2, 5

[41] Wei Liu, Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Hashing with graphs. In *ICML*, pages 1–8, 2011. 2

[42] Xianglong Liu, Cheng Deng, Bo Lang, Dacheng Tao, and Xuelong Li. Query-adaptive reciprocal hash tables for nearest neighbor search. *IEEE Trans. Image Process.*, 25(2):907–919, 2016. 2

[43] Xianglong Liu, Bowen Du, Cheng Deng, Ming Liu, and Bo Lang. Structure sensitive hashing with adaptive product quantization. *IEEE Trans. Cybern.*, 46(10):2252–2264, 2016. 2

[44] Yu Liu, Jingkuan Song, Ke Zhou, Lingyu Yan, Li Liu, Fuhao Zou, and Ling Shao. Deep self-taught hashing for image retrieval. *IEEE Trans. Cybern.*, 2018. 1

[45] Prem Melville, Nishit Shah, Lilyana Mihalkova, and Raymond J Mooney. Experiments on ensembles with missing and noisy data. In *MCS Workshop*, pages 293–302, 2004. 2

[46] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, pages 807–814, 2010. 2

[47] Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. Learning with noisy labels. In *NIPS*, pages 1196–1204. 2013. 3

[48] Wing WY Ng, Xing Tian, Witold Pedrycz, Xizhao Wang, and Daniel S Yeung. Incremental hash-bit learning for semantic image retrieval in nonstationary environments. *IEEE Trans. Cybern.*, (99), 2018. 1

[49] M. Norouzi and D. M Blei. Minimal loss hashing for compact binary codes. In *ICML*, pages 353–360, 2011. 2

[50] Curtis G Northcutt, Tailin Wu, and Isaac L Chuang. Learning with confident examples: Rank pruning for robust classification with noisy labels. In *UAI*, 2017. 3, 4

[51] Pedro O Pinheiro and AI Element. Unsupervised domain adaptation with similarity learning. In *CVPR*, pages 8004–8013, 2018. 3

[52] Ruslan Salakhutdinov and Geoffrey Hinton. Semantic hashing. *Int. J. Approx. Reasoning.*, 50(7):969–978, 2009. 2

[53] Yuming Shen, Li Liu, Fumin Shen, and Ling Shao. Zero-shot sketch-image hashing. In *CVPR*, pages 3598–3607, 2018. 1

[54] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 6

[55] Jingkuan Song, Yi Yang, Xuelong Li, Zi Huang, and Yang Yang. Robust hashing with local models for approximate similarity search. *IEEE Trans. Cybern.*, 44(7):1225–1236, 2014. 2

[56] Chaoyue Wang, Chang Xu, Xin Yao, and Dacheng Tao. Evolutionary generative adversarial networks. *IEEE Trans. Evol. Comput.*, 2019. 1

[57] Hao Wang, Yanhua Yang, Erkun Yang, and Cheng Deng. Exploring hybrid spatio-temporal convolutional networks for human action recognition. *Multimedia Tools and Applications*, 76(13):15065–15081, 2017. 1

[58] Shengnan Wang, Chunguang Li, and Hui-Liang Shen. Distributed graph hashing. *IEEE Trans. Cybern.*, (99):1–13, 2018. 1

[59] Xin-Jing Wang, Lei Zhang, Feng Jing, and Wei-Ying Ma. Annosearch: Image auto-annotation by search. In *CVPR*, volume 2, pages 1483–1490, 2006. 6, 7

[60] Yair Weiss, Antonio Torralba, and Rob Fergus. Spectral hashing. In *NIPS*, pages 1753–1760, 2009. 6, 7

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

[62] Erkun Yang, Cheng Deng, Chao Li, Wei Liu, Jie Li, and Dacheng Tao. Shared predictive cross-modal deep quantization. *IEEE Trans. Neural Netw. Learn. Syst.*, 2018. 1

[63] Erkun Yang, Cheng Deng, Tongliang Liu, Wei Liu, and Dacheng Tao. Semantic structure-based unsupervised deep hashing. In *IJCAI*, pages 1064–1070, 2018. 2, 3, 6, 7

[64] Erkun Yang, Cheng Deng, Wei Liu, Xianglong Liu, Dacheng Tao, and Xinbo Gao. Pairwise relationship guided deep hashing for cross-modal retrieval. In *AAAI*, pages 1618–1625, 2017. 2

[65] Erkun Yang, Tongliang Liu, Cheng Deng, and Dacheng Tao. Adversarial examples for hamming space search. *IEEE Trans. Cybern.*, 2018. 1

[66] Xu Yang, Cheng Deng, Xianglong Liu, and Feiping Nie. New l2, 1-norm relaxation of multi-way graph cut for clustering. In *AAAI*, pages 4374–4381, 2018. 1

[67] Shan You, Chang Xu, Yunhe Wang, Chao Xu, and Dacheng Tao. Privileged multi-label learning. In *IJCAI*, pages 3336–3342, 2017. 1

[68] Shan You, Chang Xu, Chao Xu, and Dacheng Tao. Learning from multiple teacher networks. In *ACM SIGKDD*, pages 1285–1294. ACM, 2017. 1

[69] Xiyu Yu, Tongliang Liu, Mingming Gong, Kun Zhang, and Dacheng Tao. Transfer learning with label noise. *arXiv preprint arXiv:1707.09724*, 2017. 2