Deep Priority Hashing

Zhangjie Cao, Ziping Sun, Mingsheng Long(✉), Jianmin Wang, and Philip S. Yu

School of Software, Tsinghua University, Beijing 100084, China

Beijing National Research Center for Information Science and Technology

{caozhangjie14,sunziping2016}@gmail.com,{mingsheng,jimwang}@tsinghua.edu.cn,psyu@uic.edu

ABSTRACT

Deep hashing enables image retrieval by end-to-end learning of deep representations and hash codes from training data with pairwise similarity information. Subject to the distribution skewness underlying the similarity information, most existing deep hashing methods may underperform for imbalanced data due to misspecified loss functions. This paper presents Deep Priority Hashing (DPH), an end-to-end architecture that generates compact and balanced hash codes in a Bayesian learning framework. The main idea is to reshape the standard cross-entropy loss for similarity-preserving learning such that it down-weighs the loss associated to highly-confident pairs. This idea leads to a novel *priority* cross-entropy loss, which prioritizes the training on uncertain pairs over confident pairs. Also, we propose another *priority* quantization loss, which prioritizes hard-to-quantize examples for generation of nearly lossless hash codes. Extensive experiments demonstrate that DPH can generate high-quality hash codes and yield state-of-the-art image retrieval results on three datasets, ImageNet, NUS-WIDE, and MS-COCO.

CCS CONCEPTS

• **Information systems** → **Image search**; • **Computing methodologies** → **Neural networks**;

KEYWORDS

Deep hashing, Image search, Priority loss

1 INTRODUCTION

Multimedia data has been ubiquitous in search engines and online communities, while its efficient retrieval is important to enhance user experience. The major challenges in multimedia retrieval reside in the large-scale and high-dimension of multimedia data. To enable accurate retrieval under efficient computation, approximate nearest neighbors (ANN) search has attracted increasing attention. Parallel to the traditional indexing methods [15] for candidates pruning,

another advantageous solution is hashing methods [32] for data compression, which transform high-dimensional media data into compact binary codes while similar binary codes are generated for similar data items. In this paper, we focus on the learning to hash methods [32], which build data-dependent hash encoding schemes for efficient image retrieval. These methods can capture the underlying data distributions to achieve better performance than traditional data-independent hashing methods, e.g. Locality-Sensitive Hashing (LSH) [7].

A fruitful line of learning to hash methods have been designed to enable efficient ANN search, where the efficiency comes from the compact binary codes that are orders of magnitude smaller than the original high-dimensional feature descriptors. Ranking these binary codes in response to each query entails only a few computations of the Hamming distance between the query and each database item. Hash lookup further reduces the search to constant time by early pruning of irrelevant candidates falling out of a small Hamming ball. The literature can be divided into supervised and unsupervised paradigms [6, 8, 13, 20, 22–24, 31, 35]. Recently, deep learning to hash methods [1, 5, 14, 16, 19, 28, 34, 36] have shown that deep neural networks can be used as nonlinear hash functions to enable end-to-end learning of deep representations and hash codes. These deep hashing methods have shown state-of-the-art results. In particular, it proves crucial to jointly learn similarity-preserving representations and control quantization error of converting continuous representations to binary codes [1, 16, 19, 36].

Most of the existing methods are tailored to image retrieval scenarios with balanced or nearly balanced data. In other words, they weigh equally each data pair, no matter they are similar pairs or dissimilar pairs. Thus, they can maximize retrieval performance on average per-instance accuracy. However, due to the well-known long-tail law, multimedia data with skewed distribution are prevalent in many online image search systems. The data skewness may either stem from the imbalanced numbers of similar and dissimilar pairs associated with each query, or from the diversity of popular and rare classes, or even from the variations in easy and difficult pairs of images. Such data skewness will severely affect the retrieval performance, especially when one needs to trade off the precision (weigh dissimilar pairs more in order to discard irrelevant results) from recall (weigh similar pairs more in order to include potentially relevant results). Therefore, how to address various data skewness problems simultaneously remains an open problem.

This work presents Deep Priority Hashing (DPH), a novel deep hashing model that generates compact binary codes to enable effective and efficient image retrieval under data skewness problems. DPH is formalized as a Bayesian learning framework, providing two novel loss functions motivated by the success of the focal loss in object detection problem [17]. One is a *priority* cross-entropy loss for similarity-preserving learning, which prioritizes difficult image

pairs over easy image pairs to learn prioritized deep representations. The other is a *priority* quantization loss, which prioritizes hard-to-quantize examples for generating nearly lossless hash codes. Both loss functions are well-specified to similarity retrieval of highly skew image data. The proposed DPH model is an end-to-end architecture that can be trained by standard back-propagation. Extensive experiments demonstrate that DPH can generate high-quality hash codes and yield state-of-the-art image retrieval performance on three benchmark datasets, ImageNet, NUS-WIDE, and MS-COCO.

2 RELATED WORK

Learning to hash has become an important research direction in multimedia retrieval, which trades off efficacy from efficiency. Wang et al. [32] has provided a comprehensive literature survey that covers most of important methods and latest advances.

Existing hashing methods can be divided into unsupervised hashing and supervised hashing. Unsupervised hashing methods learn hash functions that encode data points to binary codes by training solely from unlabeled data. Typical learning criteria include reconstruction error minimization [8, 10, 27] and graph structure preservation [21, 33]. While unsupervised methods are more general and can be trained without semantic labels or relevance information, they are subject to the semantic gap dilemma [30] that high-level semantic description of an object differs from low-level feature descriptors. Supervised methods can incorporate semantic labels or relevance information to mitigate the semantic gap and improve the hashing quality. Typical supervised methods include Binary Reconstruction Embedding (BRE) [13], Minimal Loss Hashing (MLH) [24], Hamming Distance Metric Learning [25], and Supervised Hashing with Kernels (KSH) [20], which generate hash codes by minimizing the Hamming distances across similar pairs and maximizing the Hamming distances across dissimilar pairs.

As deep convolutional networks [9, 12] yield sharp performance on many computer vision tasks, deep learning to hash has attracted attention recently. CNNH [34] adopts a two-stage strategy in which the first stage learns hash codes and the second stage learns a deep network to map input images to the hash codes. DNNH [14] improved the two-stage CNNH with a simultaneous feature learning and hash coding pipeline such that representations and hash codes can be optimized in a joint optimization process. DHN [36] further improves DNNH by a cross-entropy loss and a quantization loss which preserve the pairwise similarity and control the quantization error simultaneously. DHN obtains state-of-the-art performance on several benchmarks. DPSH [16] and DSH [19] follow similar framework as DHN, thus yielding similar retrieval performance. HashNet [1] improves DHN by balancing the positive and negative pairs in training data to trade off precision vs. recall, and by the continuation technique to yield exactly binary codes with the lowest quantization error. HashNet obtains state-of-the-art performance on several benchmark datasets.

However, existing deep hashing methods do not consider the data skewness problem. In other words, they let all the training pairs contribute equally to the loss function, where easy pairs will overwhelm the loss function so that the difficult ones cannot be trained sufficiently. To address these problems, we propose a novel Deep Priority Hashing (DPH) model. We design a novel priority cross-entropy loss to concentrate on difficult pairs more than on easy pairs. We design another priority quantization loss to concentrate on hard-to-quantize examples for generating less lossy hash codes. This work is among the earliest endeavors on deep hashing with the *prioritization* towards different data skewness scenarios.

3 PRELIMINARY ON FOCAL LOSS

The Focal Loss was introduced by He et al. [17], which yields state-of-the-art performance for object detection. It is designed to address the problem of an extreme imbalance between examples of different classes (e.g. foreground and background classes) during training. The Focal Loss has a close connection to the cross-entropy loss for binary classification. The Cross-Entropy (CE) loss is defined as

$$\text{CE}(p, y) = \begin{cases} -\log(p) & \text{if } y = 1 \\ -\log(1-p) & \text{otherwise.} \end{cases} \quad (1)$$

In the above $y \in \{\pm 1\}$ specifies the ground-truth class and $p \in [0, 1]$ is the model's estimated probability for the class of label $y = 1$. For notational convenience, He et al. [17] defined p_t as

$$p_t = \begin{cases} p & \text{if } y = 1 \\ 1-p & \text{otherwise.} \end{cases} \quad (2)$$

and rewrote $\text{CE}(p, y) = \text{CE}(p_t) = -\log(p_t)$.

One notable property of the cross-entropy loss is that even for easily classified examples ($p_t \gg 0.5$) there incurs a loss with non-trivial magnitude. When summed over a large number of easy examples, these small loss values can accumulate to overwhelm the rare class with difficult examples. Towards this problem, He et al. [17] proposes to reshape the cross-entropy loss into Focal Loss, which down-weighs easy examples and focuses on difficult examples.

The Focal Loss (FL) adds a modulating factor $(1 - p_t)^\gamma$ to the cross-entropy loss with a tunable focusing parameter $\gamma \geq 0$:

$$\text{FL}(p_t) = -(1 - p_t)^\gamma \log(p_t). \quad (3)$$

Focal Loss has two nice properties. 1) When an example is misclassified and p_t is small, the modulating factor is near 1 and the loss is unaffected. As $p_t \rightarrow 1$, the factor goes to 0 and the loss for well-classified examples is down-weighed. 2) The focusing parameter γ smoothly adjusts the rate at which easy examples are down-weighed. When $\gamma = 0$, FL is equivalent to CE, and as γ increases the effect of the modulating factor is likewise increase.

Intuitively, the modulating factor reduces the contribution from easy examples to the loss and enlarges the loss gap between easy and difficult examples. This in turn increases the force to correct misclassified examples.

In practice, we prefer an α-balanced variant of the focal loss:

$$\text{FL}(p_t) = -\alpha_t (1 - p_t)^\gamma \log(p_t). \quad (4)$$

This variant of the focal loss can further address the class imbalance problem between large classes and rare classes, which is a typical option for practical applications.

A limitation of the focal loss is that the modulating factor $(1-p_t)^\gamma$ is strictly determined by classification uncertainty p_t. In many real scenarios, we may need to consider other measures to quantify the *difficulty* of image pairs. The flexibility in choosing different modulating factors for prioritization is the motivation of our work.
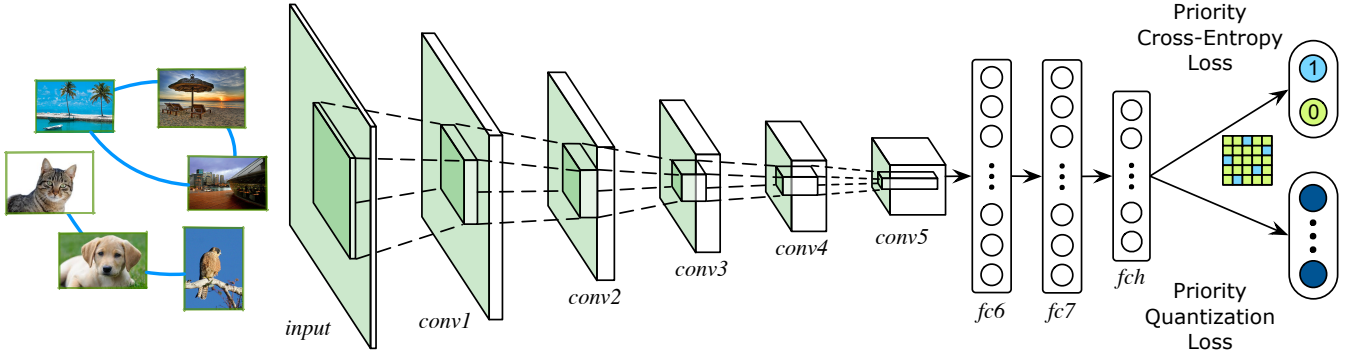
Figure 1: The architecture of Deep Priority Hashing (DPH) consists of four components: 1) a convolutional network (CNN) for learning deep representation of each point x_i, 2) a fully-connected hash layer (*fch*) for transforming the deep representation into K-bit hash code h_i, 3) a priority cross-entropy loss that prioritizes difficult pairs over easy pairs for similarity-preserving learning, and 4) a priority quantization loss that prioritizes hard-to-quantize points for controlling the hashing quality.

4 DEEP PRIORITY HASHING

In similarity retrieval, we are given a training set of N points $\{x_i\}_{i=1}^N$, each represented by a D-dimensional feature vector $x_i \in \mathbb{R}^D$. Some pairs of points x_i and x_j are provided with similarity labels s_{ij}, where $s_{ij} = 1$ if x_i and x_j are similar while $s_{ij} = 0$ if x_i and x_j are dissimilar. Deep hashing learns a nonlinear hash function $f : x \mapsto h \in \{-1, 1\}^K$ from input space $\mathbb{R}^D$ to Hamming space $\{-1, 1\}^K$ with deep network. It encodes each point x into K-bit hash code $h = f(x)$ such that the similarity in the training pairs $\{(x_i, x_j, s_{ij}) : s_{ij} \in \mathcal{S}\}$ can be preserved in the Hamming space. The similarity information $\mathcal{S} = \{s_{ij}\}$ can be collected from semantic labels or relevance feedback in online search systems.

This paper presents Deep Priority Hashing (**DPH**), an end-to-end architecture to enable efficient image retrieval, as shown in Figure 1. The proposed deep architecture accepts pairwise input images $\{(x_i, x_j, s_{ij})\}$ and processes them through a pipeline of deep representation learning and binary hash coding: 1) a convolutional network (CNN) for learning deep representation of each image x_i, 2) a fully-connected hash layer (*fch*) for transforming the deep representation into K-bit hash code $h_i \in \{1, -1\}^K$, 3) a priority cross-entropy loss that prioritizes difficult pairs over easy pairs for similarity-preserving learning, and 4) a priority quantization loss that prioritizes hard-to-quantize images for controlling the binarization error due to continuous relaxation in the optimization.

4.1 Deep Architecture

Figure 1 illustrates the architecture of the proposed Deep Priority Hashing. We extend from AlexNet [12], a deep convolutional neural network (CNN) with five convolutional layers *conv1–conv5* and three fully-connected layers *fc6–fc8*. We replace the classifier layer *fc8* with a new hash layer *fch* of K hidden units, which transforms the representation of the *fc7* layer into K-dimensional continuous code $z_i \in \mathbb{R}^K$. We can obtain hash code h_i through the sign thresholding $h_i = \text{sgn}(z_i)$. However, we adopt the hyperbolic tangent (tanh) function to squash the continuous code to be within $[-1, 1]$ instead of using sign function. To further guarantee the quality of hash codes for efficient image retrieval, we preserve the similarity

in training pairs $\{(x_i, x_j, s_{ij}) : s_{ij} \in \mathcal{S}\}$ by designing a *priority* cross-entropy loss that prioritizes difficult pairs over easy pairs and control the quantization error by designing another *priority* quantization loss. Both loss functions can be derived in the Maximum a Posteriori (MAP) estimation framework. Though following most work to use AlexNet in Figure 1, we can easily replace the backbone network in our architecture with any classification network since we only replace the last classifier layer while the other layers can be inherited from the backbone network.

4.2 Model Formulation

This paper enables deep hashing from skew data with both easy and difficult examples by a Bayesian learning framework. The framework jointly preserves similarity information of pairwise images and controls the quantization error of continuous relaxation. For a pair of hash codes h_i and h_j, their Hamming distance $d_H(\cdot, \cdot)$ and their inner product $\langle \cdot, \cdot \rangle$ satisfy $d_H(h_i, h_j) = \frac{1}{2}(K - \langle h_i, h_j \rangle)$, indicating that the Hamming distance and inner product can be used interchangeably for binary codes. Thus we adopt inner product to quantify the pairwise similarity. Given training image pairs with pairwise similarity labels as $\{(x_i, x_j, s_{ij}) : s_{ij} \in \mathcal{S}\}$, the logarithm Weighted Maximum a Posteriori (WMAP) estimation of the hash codes $H = [h_1, \ldots, h_N]$ for N training images can be defined as

$$\log p(H|\mathcal{S}) \propto \log p(\mathcal{S}|H) p(H)$$
$$= \sum_{s_{ij} \in \mathcal{S}} w_{ij} \log p(s_{ij}|h_i, h_j) + \sum_{i=1}^N w_i' \log p(h_i), \quad (5)$$

where $p(\mathcal{S}|H) = \prod_{s_{ij} \in \mathcal{S}} \left[p(s_{ij}|h_i, h_j) \right]^{w_{ij}}$ is the weighted likelihood function for pairwise data, and w_{ij} is the weight for each training pair (x_i, x_j, s_{ij}). This is extended from the weighted maximum likelihood on pointwise data [3]. Another difference from [3] is the weighted prior $p(H)$, and w_i' is the weight for an image x_i. In this paper, we propose the above Weighted Maximum a Posteriori (WMAP) estimation over pairwise data with different skewness scenarios. This is a general framework for instantiating specific learning to hash models, by choosing well-specified probability functions and weighting schemes.

4.2.1 Priority Cross-Entropy Loss. We first derive the priority cross-entropy loss for similarity-preserving learning. For each image pair with label $(\boldsymbol{x}_i, \boldsymbol{x}_j, s_{ij})$, $p(s_{ij}|\boldsymbol{h}_i, \boldsymbol{h}_j)$ is the conditional probability of the similarity label s_{ij} given the pair of corresponding hash codes $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$, which is defined by the logistic function:

$$p\left(s_{ij}|\boldsymbol{h}_i, \boldsymbol{h}_j\right) = \begin{cases} \sigma\left(\langle \boldsymbol{h}_i, \boldsymbol{h}_j \rangle\right), & s_{ij} = 1 \\ 1 - \sigma\left(\langle \boldsymbol{h}_i, \boldsymbol{h}_j \rangle\right), & s_{ij} = 0 \end{cases}$$
$$= \sigma(\langle \boldsymbol{h}_i, \boldsymbol{h}_j \rangle)^{s_{ij}} (1 - \sigma\left(\langle \boldsymbol{h}_i, \boldsymbol{h}_j \rangle\right))^{1-s_{ij}} \quad (6)$$

where $\sigma(x) = 1/(1 + e^{-\beta x})$ is an adaptive variant of the sigmoid function with parameter β to control its bandwidth. As the sigmoid function with larger β has larger saturation zone, we usually require $\beta < 1$ to perform back-propagation with more gradients.

Motivated by the Focal Loss (FL) [17], we use the weight combined by α_t-scaling and modulating factor $(1-q)^\gamma$ to simultaneously model the class diversity (including imbalance in both similar and dissimilar pairs) and the variation in easy and difficult examples. But unlike the focal loss, in the modulating factor we use a different measure q instead of the classification uncertainty p to quantify the *difficulty* of each image pair. This new design relaxes the restriction in the focal loss that the "difficulty" of each image pair and the classification uncertainty should be consistent. In this way, we can adopt more flexible choices for the modulating factor. Furthermore, in deep hashing we do not have images with pointwise class labels, and thus need to model image pairs with pairwise labels $(\boldsymbol{x}_i, \boldsymbol{x}_j, s_{ij})$. We propose a novel *priority* weighting scheme as follows,

$$w_{ij} = \alpha_{ij}\left(1 - q_{ij}\right)^\gamma, \quad (7)$$

where w_{ij} is the weight for each training pair $(\boldsymbol{x}_i, \boldsymbol{x}_j, s_{ij})$. It consists of a scaling part to weigh for class imbalance problem, and a modulating part to weigh for easy and hard examples. Specifically, the measure of *difficulty* for the modulating part, i.e. $q_{ij} = q\left(s_{ij}|\boldsymbol{h}_i, \boldsymbol{h}_j\right)$ is defined as

$$q\left(s_{ij}|\boldsymbol{h}_i, \boldsymbol{h}_j\right) = \begin{cases} \frac{1+\cos(\boldsymbol{h}_i, \boldsymbol{h}_j)}{2}, & s_{ij} = 1 \\ \frac{1-\cos(\boldsymbol{h}_i, \boldsymbol{h}_j)}{2}, & s_{ij} = 0 \end{cases}$$
$$= \left(\frac{1 + \cos\left(\boldsymbol{h}_i, \boldsymbol{h}_j\right)}{2}\right)^{s_{ij}} \left(\frac{1 - \cos\left(\boldsymbol{h}_i, \boldsymbol{h}_j\right)}{2}\right)^{1-s_{ij}} \quad (8)$$

where q_{ij} indicates how difficult an image pair $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is classified as similar when $s_{ij} = 1$, or classified as dissimilar when $s_{ij} = 0$. With these difficulty quantities q_{ij}, we can put different weights on easy and hard examples to prioritize on more difficult image pairs. An important motivation of using q_{ij} as the measure of "difficulty" is that q_{ij} is *magnitude-invariant*, consistent with the fact that hash codes have the same magnitude for different images. Therefore, q_{ij} can potentially eliminate the variations in code magnitudes due to skew data distributions.

In the original focal loss, the scaling part tackles the imbalance between different classes in a pointwise scenario, which is not applicable to pairwise scenario. In HashNet [1], the number of similar pairs and dissimilar pairs are used to compute the weight for data imbalance. However, such a strategy cannot quantify the imbalance of underlying different classes, since the class information is unavailable in pairwise scenarios. Quantification of the scaling part for the pairwise scenario remains a nontrivial problem unsolved

by previous work. In this paper, we define the scaling part α_{ij} by considering the degrees of each vertex in the similarity graph $\mathcal{S}$ as

$$\alpha_{ij} = \begin{cases} \dfrac{|\mathcal{S}_i||\mathcal{S}_j|}{\sqrt{\left|\mathcal{S}_i^1\right|\left|\mathcal{S}_j^1\right|}}, & s_{ij} = 1 \\ \dfrac{|\mathcal{S}_i||\mathcal{S}_j|}{\sqrt{\left|\mathcal{S}_i^0\right|\left|\mathcal{S}_j^0\right|}}, & s_{ij} = 0 \end{cases} \quad (9)$$

where α_{ij} indicates the pairwise weight characterizing the class imbalance, which is influenced by both images in that pair. $\mathcal{S}_i = \left\{s_{ij} \in \mathcal{S} : \forall j\right\}$ is the set of pairs containing $\boldsymbol{x}_i$. It is divided into two subsets: $\mathcal{S}_i^1 = \left\{s_{ij} \in \mathcal{S} : \forall j, s_{ij} = 1\right\}$ is the subset of similar pairs that contain $\boldsymbol{x}_i$, while $\mathcal{S}_i^0 = \left\{s_{ij} \in \mathcal{S} : \forall j, s_{ij} = 0\right\}$ is the subset of dissimilar pairs that contain $\boldsymbol{x}_i$. $\mathcal{S}_j, \mathcal{S}_j^1$ and $\mathcal{S}_j^0$ are similarly defined. Intuitively, for each image $\boldsymbol{x}_i$, $\left|\mathcal{S}_i^1\right|$ and $\left|\mathcal{S}_i^0\right|$ respectively calculate the numbers of its similar and dissimilar images, which naturally captures the data skewness due to the class imbalance.

By taking the priority weight in Equation (7) into the WMAP estimation in Equation (5) and letting $p_{ij} = p\left(s_{ij}|\boldsymbol{h}_i, \boldsymbol{h}_j\right)$, we can obtain a novel (pairwise) Priority Cross-Entropy Loss as follows:

$$L = -\sum_{s_{ij} \in \mathcal{S}} \alpha_{ij}\left(1 - q_{ij}\right)^\gamma \log\left(p_{ij}\right). \quad (10)$$

The priority cross-entropy loss is a natural extension of the focal loss to the pairwise classification scenario, which inherits all nice properties of the focal loss and broadens the choices of the modulating factor to incorporate more reasonable measures of "difficulty". More specifically, the priority cross-entropy loss will down-weigh confident pairs and prioritize on difficult pairs with low confidence. And with the proposed scaling strategy in Equation (9), the priority cross-entropy loss assigns larger weights to rare classes and smaller weights to popular classes, which for the first time, addresses the class imbalance issue in pairwise scenario for image retrieval.

4.2.2 Priority Quantization Loss. To control the quantization error of the binarization operation, we further derive the priority quantization loss from the MAP framework to yield nearly lossless hash codes. As discrete optimization of Equation (5) with binary constraints $\boldsymbol{h}_i \in \{-1, 1\}^K$ is very challenging, *continuous relaxation* widely adopted by existing hashing methods [32] to apply to the binary constraints for ease of optimization. However, the continuous relaxation will give rise to two important technical issues: 1) uncontrollable quantization error caused by binarizing continuous codes to binary codes and 2) large approximation error by adopting inner product between continuous codes as the surrogate of Hamming distance between binary codes. To control the quantization error and close the gap between Hamming distance and its surrogate, in this paper, we propose an (unnormalized) bimodal Laplacian prior for the continuous codes $\{\boldsymbol{h}_i\}$, which is defined as:

$$p\left(\boldsymbol{h}_i\right) = \frac{1}{2\epsilon} \exp\left(-\frac{\||\boldsymbol{h}_i| - 1\|_1}{\epsilon}\right), \quad (11)$$

where ϵ is the diversity parameter, and let $p_i = p\left(\boldsymbol{h}_i\right)$. We can validate that the prior puts the largest density on the discrete values $\{-1, 1\}$, which enforces that the learned Hamming embeddings $\{\boldsymbol{h}_i\}$ should be assigned to $\{-1, 1\}$ with the largest probability.

Similar to the priority cross-entropy loss, we define the priority weight for the priority quantization loss as

$$w'_i = \alpha_i (1 - q_i)^\gamma, \tag{12}$$

which consists of a scaling part and a modulating part. Note that we need to control the quantization error of all images equally, regardless of whether they are from the rare classes. Thus we set constant scaling $\alpha_i = 1$. The modulating part controls the quantization error under the variations in easy and hard examples, with q_i defined as

$$q_i = q(\boldsymbol{h}_i) = \frac{1 + \cos(|\boldsymbol{h}_i|, 1)}{2}, \tag{13}$$

which indicates how likely a continuous code $\boldsymbol{h}_i$ can be perfectly quantized into binary code sgn$(\boldsymbol{h}_i)$. With probabilities q_i, we can put different weights on easy-to-quantize and hard-to-quantize examples and prioritize hard-to-quantize ones.

By taking the priority weight in Equation (12) into the WMAP in Equation (5), we obtain a novel Priority Quantization Loss as:

$$Q = -\sum_{s_{ij} \in \mathcal{S}} (1 - q_i)^\gamma \log(p_i). \tag{14}$$

The priority quantization loss helps generating nearly lossless hash codes by reducing the quantization error of hard-to-quantized examples more than of easy-to-quantize examples.

4.2.3 Hash Function Learning. This paper establishes deep learning to hash for skew data with pairwise similarity information, which constitutes two key components: Priority Cross-Entropy Loss for similarity-preserving learning and Priority Quantization Loss for generating nearly lossless hash codes. The overall optimization problem is an integration of the Priority Cross-Entropy Loss in Equations (10) and the Priority Quantization Loss in Equation (14):

$$\min_{\Theta} L + Q, \tag{15}$$

where Θ is the network parameters efficiently optimized using standard back-propagation with automatic differentiation techniques. Note that, the scale parameter $\frac{1}{\epsilon}$ in Equation (11) introduces a tradeoff hyper-parameter between the two losses.

Based on the WMAP estimation in Equation (15), we can learn compact hash codes by jointly preserving the pairwise similarity and controlling the quantization error. Finally, we can obtain K-bit binary codes by simple sign thresholding $\boldsymbol{h} \leftarrow \text{sgn}(\boldsymbol{h})$, where sgn$(\boldsymbol{h})$ is the sign function on vectors that for $i = 1, \ldots, K$, sgn$(h_i) = 1$ if $h_i > 0$, otherwise sgn$(h_i) = -1$. It is worth noting that, since we have minimized the quantization error in (15) during training, this final binarization step will incur very small loss of retrieval quality.

5 EXPERIMENTS

We conduct extensive experiments to evaluate DPH with several state-of-the-art hashing methods on three benchmark datasets. Codes and datasets are available at https://github.com/thuml/DPH.

5.1 Setup

ImageNet is a benchmark dataset for Large Scale Visual Recognition Challenge (ILSVRC 2015) [26]. It contains over 1.2M images in the training set and 50K images in the validation set, where each image is single-labeled by one of the 1,000 categories. We use the sample of 100 categories organized by HashNet [1]. We use the same

database set and query set but re-sample the training set to make it data-skew. Our training set contains 10,000 images, which consists of three groups: the first group with 1 big class (~1,300 images/class), the second group with 9 middle classes (~400 images/class) and the third group with 90 small classes (~50 images/class). The classes of each group are randomly sampled.

NUS-WIDE[1] [2] is an image dataset containing 269,648 images from Flickr.com. Each image is annotated by some of the 81 ground truth concepts (categories). We follow a slightly different evaluation protocol as HashNet [1]. We randomly sample 50 images per class as query images, with the remaining images used as the database. We further sample 10,000 images randomly from the database as training images.

MS-COCO[2] [18] is an image recognition, segmentation, and captioning dataset. The current release contains 82,783 training images and 40,504 validation images, where each image is labeled by some of the 80 categories. We obtain the 12,2218 images set by combining the training and validation images. As described above in the NUS-WIDE dataset, we randomly sample 50 images per class as queries. We use the rest images as the database and randomly sample 10,000 images from the database as training images.

Following standard evaluation protocol as previous work [1, 14, 34, 36], the similarity information for hash function learning and for ground-truth evaluation is constructed from image labels: if two images i and j share at least one label, they are similar and $s_{ij} = 1$; otherwise, they are dissimilar and $s_{ij} = 0$. Note that, although we use the image labels to construct the similarity information, our proposed DPH model can learn hash codes when only the similarity information is available. These datasets exhibit data skewness problems and thus can be used to evaluate different hashing methods in the data skewness scenario.

We compare **DPH** in terms of the retrieval performance against eleven classical or state-of-the-art hashing methods, including unsupervised hashing methods **LSH** [7], **SH** [33], **ITQ** [8], supervised shallow hashing methods **KSH** [20], **SDH** [28], and supervised deep hashing methods **CNNH** [34], **DNNH** [14], **DPSH** [16], **DSH** [19], **DHN** [36], **HashNet** [1].

We evaluate the retrieval quality on four standard metrics: Mean Average Precision (**MAP**), Precision-Recall curves (**PR**), Precision curves within Hamming distance 2 (**P@H≤2**), and Precision curves with respect to different numbers of top returned samples (**P@N**). For fair comparison, all methods use identical training and test sets. We adopt MAP@1000 for ImageNet and MAP@5000 for the other datasets as in [1]. It is worth noting that, in order to evaluate the retrieval performance equally on large and small classes, we let each class appears nearly equally in the query set for each dataset. This is a reasonable setting under data skewness.

For shallow hashing methods, we use DeCAF[7] features [4]. For deep hashing methods, we use raw images as input. We use AlexNet [12] for all deep hashing methods, and implement DPH in **Caffe** [11]. We fine-tune convolutional layers and fully-connected layers *conv1–fc7* copied from AlexNet pre-trained on ImageNet 2012 and train the hash layer *fch* by back-propagation. As the *fch* layer is trained from scratch, we set its learning rate to be 10 times that

Table 1: Mean Average Precision (MAP) of Hamming Ranking for Different Number of Bits on the Three Image Datasets

Method	ImageNet				NUS-WIDE				MS-COCO			
	16 bits	32 bits	48 bits	64 bits	16 bits	32 bits	48 bits	64 bits	16 bits	32 bits	48 bits	64 bits
DPH	**0.3252**	**0.4803**	**0.5279**	**0.5492**	**0.6689**	**0.7014**	**0.7218**	**0.7312**	**0.7100**	**0.7435**	**0.7544**	**0.7614**
HashNet [1]	0.2959	0.4211	0.4836	0.5056	0.6294	0.6604	0.6785	0.6883	0.6529	0.7025	0.7231	0.7335
DHN [36]	0.2734	0.3821	0.4352	0.4921	0.5940	0.6186	0.6277	0.6343	0.6206	0.6445	0.6641	0.6685
DSH [19]	0.2665	0.3513	0.3963	0.4213	0.5880	0.6102	0.6155	0.6242	0.6242	0.6369	0.6481	0.6511
DPSH [16]	0.1625	0.3037	0.4074	0.4841	0.4041	0.4826	0.5067	0.5379	0.5660	0.6269	0.6587	0.6906
DNNH [14]	0.1189	0.2783	0.3328	0.3456	0.3845	0.4632	0.4901	0.5158	0.5461	0.5923	0.6147	0.6272
CNNH [34]	0.0890	0.2459	0.2903	0.3156	0.3653	0.4432	0.4672	0.4801	0.5659	0.5624	0.5519	0.5687
SDH [28]	0.1023	0.1543	0.1785	0.2043	0.3021	0.4056	0.4329	0.4675	0.4933	0.5323	0.5493	0.5577
KSH [20]	0.0823	0.1121	0.1341	0.1456	0.2007	0.2704	0.3002	0.3257	0.4946	0.5193	0.5224	0.5316
ITQ [8]	0.2546	0.3313	0.3564	0.4023	0.3309	0.4179	0.4461	0.4774	0.5289	0.5824	0.6227	0.6320
SH [33]	0.0917	0.1234	0.1524	0.1623	0.1649	0.1904	0.2115	0.2605	0.4429	0.4910	0.4782	0.5073
LSH [7]	0.0412	0.0623	0.0889	0.1021	0.0557	0.0938	0.1392	0.1724	0.3932	0.4659	0.5301	0.5171

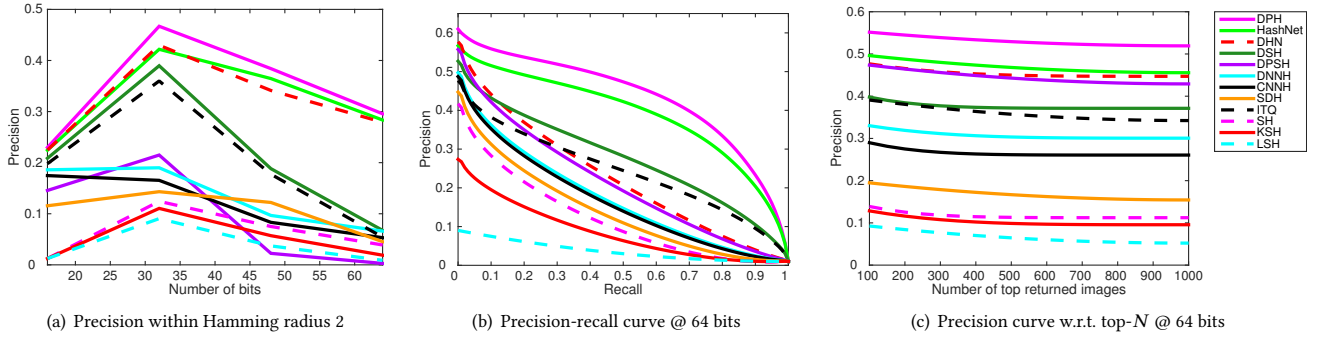(a) Precision within Hamming radius 2   (b) Precision-recall curve @ 64 bits   (c) Precision curve w.r.t. top-N @ 64 bits

Figure 2: The experimental results of DPH and comparison methods on the ImageNet dataset under three evaluation metrics.

(a) Precision within Hamming radius 2   (b) Precision-recall curve @ 64 bits   (c) Precision curve w.r.t. top-N @ 64 bits
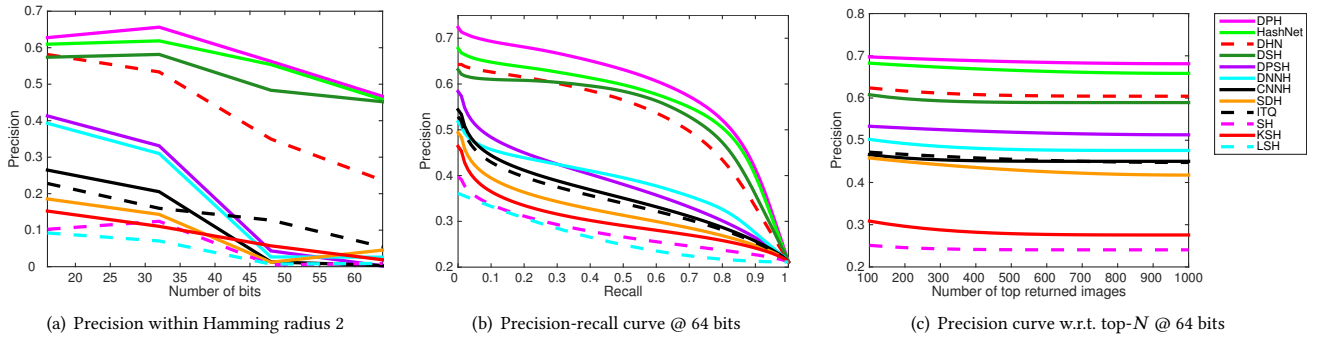
Figure 3: The experimental results of DPH and comparison methods on the NUS-WIDE dataset under three evaluation metrics.

of the lower layers. We use mini-batch stochastic gradient descent (SGD) with 0.9 momentum and the learning rate annealing strategy implemented in Caffe, and cross-validate the learning rate from 10^{-5} to 10^{-3} with a multiplicative step-size $10^{\frac{1}{2}}$. We fix the mini-batch size of images as 256 and the weight decay parameter as 0.0005. We select the hyper-parameters of all methods through three-fold cross-valuation.

5.2 Results

Table 1 shows the **MAP** results. DPH outperforms all comparison methods substantially. Compared to ITQ, the best shallow hashing method using deep features, we achieve absolute boosts of 13.45%, 28.78%, and 15.08% in average MAP for different bits on ImageNet, NUS-WIDE, and MS-COCO, respectively. Compared to HashNet, the state-of-the-art deep hashing method, we achieve
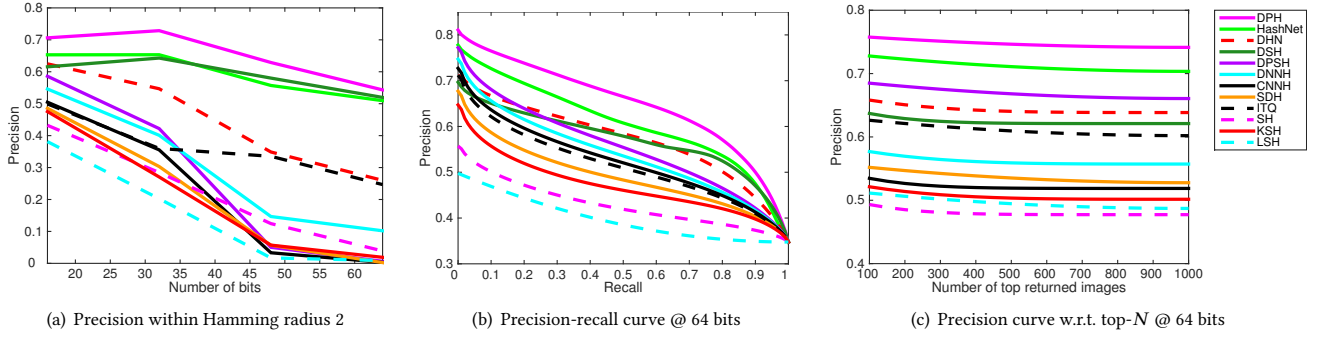
(a) Precision within Hamming radius 2   (b) Precision-recall curve @ 64 bits   (c) Precision curve w.r.t. top-N @ 64 bits

Figure 4: The experimental results of DPH and comparison methods on the MS-COCO dataset under three evaluation metrics.

Table 2: MAP on the Balanced ImageNet Dataset (left) and on the Three Imbalanced Image Datasets with VGG-16 (right)

Method	ImageNet			
	16 bits	32 bits	48 bits	64 bits
DPH	**0.5168**	**0.6409**	**0.6723**	**0.6967**
HashNet [1]	0.5059	0.6306	0.6633	0.6835

Method	ImageNet		NUS-WIDE		MS-COCO	
	16 bits	64 bits	16 bits	64 bits	16 bits	64 bits
DPH	**0.3455**	**0.6395**	**0.7107**	**0.7439**	**0.7252**	**0.7958**
HashNet [1]	0.3252	0.5891	0.6853	0.7163	0.6835	0.7581

absolute boosts of 4.41%, 4.17%, 3.93% in average MAP for different bits on the three datasets, respectively.

HashNet uses Weighted Maximum Likelihood to tackle the data imbalance problem but it only uses the weight between positive and negative pairs, which cannot capture the the more fine-grained class imbalance issue. This class imbalance issue substantially deteriorates the average performance on all the classes since previous hashing methods will unavoidably focus on large classes and underperform on small classes. Thus, with each class appears nearly equally in the query set, the results on each dataset are worse than those reported by the original paper. However, our DPH puts larger weights on the samples of small classes to focus more on difficult examples (usually samples of small classes). Thus, DPH performs relatively well on all the classes and yields the best performance.

The performance in terms of Precision within Hamming radius 2 (**P@H=2**) is important for efficient retrieval with hash codes since such Hamming ranking only requires $O1)$ time for each query. As shown in Figures 2(a), 3(a) and 4(a), DPH achieves the highest P@H=2 results on all three datasets. In particular, P@H=2 of DPH with 32 bits is better than that of HashNet with any bits. This validates that DPH can learn more compact hash codes than HashNet. When using longer codes, the Hamming space will become sparse and few data points fall within the Hamming ball with radius 2 [6]. This is why most hashing methods achieve best accuracy with relatively shorter code lengths.

The retrieval performance on the three datasets in terms of Precision-Recall curves (**PR**) and Precision curves with respect to different numbers of top returned samples (**P@N**) are shown in Figures 2(b)~4(b) and Figures 2(c)~4(c), respectively. DPH outperforms comparison methods by large margins. In particular, DPH achieves much higher precision at lower recall levels or when the number of top results is small. This is desirable for precision-first retrieval, which is widely implemented in practical systems.

To enable direct comparison with published papers, we also test our method on the Balanced ImageNet dataset as in HashNet [1]. As shown in the left part of Table 2, our DPH outperforms HashNet even on the balanced ImageNet dataset, which proves that our method can perform well even on balanced dataset.

As the network structure can influence the performance of general-purpose computer vision models, we test our model on the three datasets with VGG-16 Net [29]. Observing from the right part of Table 2, our DPH still outperforms HashNet, which demonstrates that our method is robust to the base network architecture.

5.3 Discussion

5.3.1 Ablation Study. We dive deeper into the efficacy of our DPH model. We study three variants of DPH: 1) **DPH-F**, replace the modulating factor to that of the focal loss, i.e. using p instead of q in the modulating factor; 2) **DPH-W**, variant without priority weight, i.e. $w_{ij} = 1$; 3) **DPH-Q**, variant without priority quantization loss, i.e. $\epsilon = \infty$. We compare these variants in Table 3.

As expected, DPH substantially outperforms DPH-W by large margins of 10.31%, 5.33% and 2.54% in average MAP for different bits on ImageNet, NUS-WIDE and MS-COCO, respectively. The classical pairwise cross-entropy loss (without priority weighting) has been widely adopted in previous work [34, 36]. However, this classical loss does not account for the class imbalance and for the variations in easy and hard examples. Thus it may suffer from performance drop when training data is highly skew (e.g. NUS-WIDE) or has large variations of easy and hard examples. In contrast, our DPH model uses the proposed priority cross-entropy loss, which is a principled solution to these data skewness problems.

Furthermore, we notice that DPH outperforms DPH-F, which demonstrates the sub-optimality in using classification uncertainty p in the focal loss. Our priority weight can relax the restriction on the choices of the difficulty measure in the modulating factor.

Table 3: MAP Results of DPH and Its Variants, DPH-F, DPH-W, and DPH-Q on the Three Image Datasets

Method	ImageNet				NUS-WIDE				MS-COCO			
	16 bits	32 bits	48 bits	64 bits	16 bits	32 bits	48 bits	64 bits	16 bits	32 bits	48 bits	64 bits
DPH	**0.3252**	**0.4803**	**0.5279**	**0.5592**	0.6689	0.7014	0.7218	0.7312	0.7100	**0.7435**	**0.7544**	**0.7614**
DPH-F	0.3115	0.4683	0.5139	0.5487	0.6451	0.6798	0.6918	0.7012	0.7100	0.7335	0.7434	0.7555
DPH-W	0.1919	0.3645	0.4473	0.4767	0.6210	0.6535	0.6652	0.6703	0.6826	0.7180	0.7314	0.7356
DPH-Q	0.3102	0.4656	0.5162	0.5431	0.6132	0.6557	0.6723	0.6898	0.6978	0.7329	0.7475	0.7555

DPH outperforms DPH-Q by 1.44%, 4.81%, and 0.89% in average MAP for different code lengths on ImageNet, NUS-WIDE, and MS-COCO respectively. These results validate that the proposed priority quantization loss (14) can control the quantization error caused by continuous relaxation and generate less lossy binary codes.
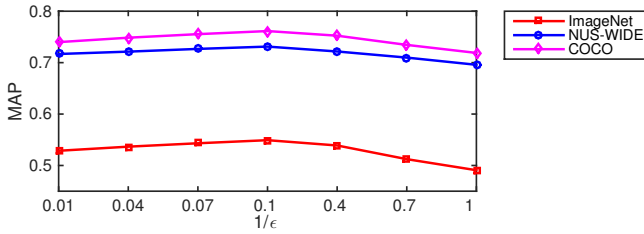

Figure 5: Sensitivity of $\frac{1}{\epsilon}$ for DPH on the three datasets.

5.3.2 Parameter Sensitivity. We investigate the sensitivity of the tradeoff hyper-parameter $\frac{1}{\epsilon}$ between the Priority Cross-Entropy Loss and the Priority quantization Loss on ImageNet, NUS-WIDE and MS-COCO datasets. From Figure 5, we can observe that the MAP results fluctuate slightly as $\frac{1}{\epsilon}$ increases from 0.01 to 1. This demonstrates that DPH is not sensitive to the scale of the tradeoff hyper-parameter and can perform stably in a wide range of $\frac{1}{\epsilon}$.
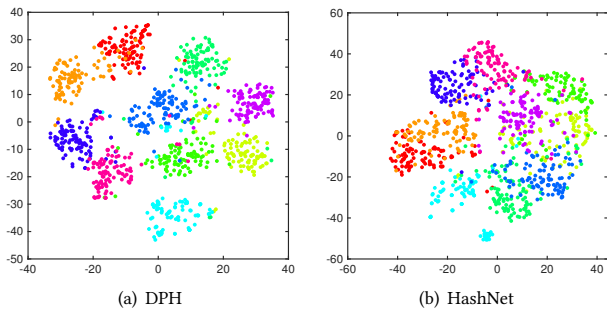

Figure 6: t-SNE of hash codes learned by DPH and HashNet.

5.3.3 Visualization. We visualize the t-SNE [4] of hash codes generated by HashNet and DPH on ImageNet in Figure 6 (we sample 10 categories for ease of visualization). We observe that the hash codes generated by DPH show clear discriminative structures in that different categories are well separated, while those generated by HashNet do not show such discriminative structures.

Figure 7: Examples of the top 10 images and Precision@10.

Figure 7 illustrates the top 10 returned images of DPH and the best deep hashing baseline HashNet [1] for three query images on the three datasets ImageNet, NUS-WIDE, and MS-COCO, respectively. DPH yields much more relevant and user-desired retrieval results than the state-of-the-art method.

6 CONCLUSION

This paper studies deep learning to hash approaches to establish efficient and effective image retrieval under a variety of data skewness scenarios. The proposed Deep Priority Hashing (DPH) approach generates compact and balanced hash codes by jointly optimizing a novel priority cross-entropy loss and a priority quantization loss in a single Bayesian learning framework. The overall model can be trained end-to-end with well-specified loss functions. Extensive experiments demonstrate that DPH can yield state-of-the-art image retrieval performance under skewness on three benchmark datasets, ImageNet, NUS-WIDE, and MS-COCO.

In the future, we plan to extend the probabilistic framework to support image retrieval with relative similarity information, i.e. there is only similarity information on whether an image is more similar to another image than a third image.

7 ACKNOWLEDGEMENTS

REFERENCES

[1] Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Philip S. Yu. 2017. HashNet: Deep Learning to Hash by Continuation. In *The IEEE International Conference on Computer Vision (ICCV)*.

[2] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yan-Tao Zheng. 2009. NUS-WIDE: A Real-World Web Image Database from National University of Singapore. In *ICMR*. ACM.

[3] Jacek P Dmochowski, Paul Sajda, and Lucas C Parra. 2010. Maximum likelihood in cost-sensitive learning: Model specification, approximations, and upper bounds. *Journal of Machine Learning Research (JMLR)* 11, Dec (2010), 3313–3332.

[4] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. 2014. DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. In *ICML*.

[5] Venice Erin Liong, Jiwen Lu, Gang Wang, Pierre Moulin, and Jie Zhou. 2015. Deep Hashing for Compact Binary Codes Learning. In *CVPR*. IEEE, 2475–2483.

[6] D. J. Fleet, A. Punjani, and M. Norouzi. 2012. Fast search in Hamming space with multi-index hashing. In *CVPR*. IEEE.

[7] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al. 1999. Similarity search in high dimensions via hashing. In *VLDB*, Vol. 99. ACM, 518–529.

[8] Yunchao Gong and Svetlana Lazebnik. 2011. Iterative quantization: A procrustean approach to learning binary codes. In *CVPR*. 817–824.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. *CVPR* (2016).

[10] H. Jegou, M. Douze, and C. Schmid. 2011. Product Quantization for Nearest Neighbor Search. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 33, 1 (Jan 2011), 117–128.

[11] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. 2014. Caffe: Convolutional Architecture for Fast Feature Embedding. In *ACM Multimedia Conference*. ACM.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *NIPS*.

[13] Brian Kulis and Trevor Darrell. 2009. Learning to hash with binary reconstructive embeddings. In *NIPS*. 1042–1050.

[14] Hanjiang Lai, Yan Pan, Ye Liu, and Shuicheng Yan. 2015. Simultaneous Feature Learning and Hash Coding with Deep Neural Networks. In *CVPR*. IEEE.

[15] Michael S. Lew, Nicu Sebe, Chabane Djeraba, and Ramesh Jain. 2006. Content-based Multimedia Information Retrieval: State of the Art and Challenges. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 2, 1 (Feb. 2006), 1–19. https://doi.org/10.1145/1126004.1126005

[16] Wu-Jun Li, Sheng Wang, and Wang-Cheng Kang. 2016. Feature learning based deep supervised hashing with pairwise labels. In *IJCAI*.

[17] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. 2017. Focal Loss for Dense Object Detection. In *The IEEE International Conference on Computer Vision (ICCV)*.

[18] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *ECCV*. Springer, 740–755.

[19] Haomiao Liu, Ruiping Wang, Shiguang Shan, and Xilin Chen. 2016. Deep supervised hashing for fast image retrieval. In *CVPR*. 2064–2072.

[20] Wei Liu, Jun Wang, Rongrong Ji, Yu-Gang Jiang, and Shih-Fu Chang. 2012. Supervised hashing with kernels. In *CVPR*. IEEE.

[21] Wei Liu, Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. 2011. Hashing with Graphs. In *ICML*. ACM.

[22] Xianglong Liu, Junfeng He, Cheng Deng, and Bo Lang. 2014. Collaborative hashing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2139–2146.

[23] Xianglong Liu, Junfeng He, Bo Lang, and Shih-Fu Chang. 2013. Hash bit selection: a unified solution for selection problems in hashing. In *CVPR*. IEEE, 1570–1577.

[24] Mohammad Norouzi and David M Blei. 2011. Minimal loss hashing for compact binary codes. In *ICML*. ACM, 353–360.

[25] Mohammad Norouzi, David M Blei, and Ruslan R Salakhutdinov. 2012. Hamming distance metric learning. In *NIPS*. 1061–1069.

[26] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)* 115, 3 (2015), 211–252. https://doi.org/10.1007/s11263-015-0816-y

[27] Ruslan Salakhutdinov and Geoffrey E Hinton. 2007. Learning a nonlinear embedding by preserving class neighbourhood structure. In *AISTATS*. 412–419.

[28] Fumin Shen, Chunhua Shen, Wei Liu, and Heng Tao Shen. 2015. Supervised Discrete Hashing. In *CVPR*. IEEE.

[29] K. Simonyan and A. Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *ICLR, 2015 (arXiv:1409.1556v6)*.

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

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

[32] Jingdong Wang, Ting Zhang, Jingkuan Song, Nicu Sebe, and Heng Tao Shen. 2018. A Survey on Learning to Hash. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40, 4 (Feb. 2018), 769–790.

[33] Yair Weiss, Antonio Torralba, and Rob Fergus. 2009. Spectral Hashing. In *NIPS*.

[34] Rongkai Xia, Yan Pan, Hanjiang Lai, Cong Liu, and Shuicheng Yan. 2014. Supervised hashing for image retrieval via image representation learning. In *AAAI*. AAAI, 2156–2162.

[35] Peichao Zhang, Wei Zhang, Wu-Jun Li, and Minyi Guo. 2014. Supervised hashing with latent factor models. In *SIGIR*. ACM, 173–182.

[36] Han Zhu, Mingsheng Long, Jianmin Wang, and Yue Cao. 2016. Deep Hashing Network for Efficient Similarity Retrieval. In *AAAI*. AAAI.