Unifying Training Objectives for Deep Hashing: Learning Orthogonal Hash Codes with Cosine Similarity

Anonymous Author(s)
Affiliation
Address
email

Abstract

In this work, we propose to unify the training objectives of deep hashing under only *a single objective*, which is by maximizing the cosine similarity between the continuous codes and *binary orthogonal target* under *a single classification objective* with cross entropy loss. First, we reformulate the problem of deep hashing in the lens of cosine similarity. Second, we show that our method can perform end-to-end training to learn highly discriminative hash codes which can theoretically achieve maximum inter-class Hamming distance without sophisticated training objectives. Moreover, we leverage the concept of Label Smoothing which allow us to train multi-labels classification with cross entropy loss and Batch Normalization (BN) layer for code balancing. Extensive experiments show that our method can achieve comparable state-of-the-art results on ImageNet100, NUS-WIDE and MS-COCO. Yet, our method can achieve state-of-the-art in large-scale instance-level retrieval on GLDv2, $\mathcal{R}$Oxf and $\mathcal{R}$Par by at most 0.6%, 6.4% and 9.2% improvement respectively.

1 Introduction

To hash image with high efficiency and accuracy, deep *learning to hash* (deep hashing) methods [47, 21] have shown great improvement in large-scale image retrieval as compared with conventional hashing methods [45, 13, 14, 20, 35, 36, 17], mainly because of the capability of deep neural network (DNN) of simultaneously learning both highly discriminative feature representations [22] and hash codes. Typically, deep hashing methods can be categorized as pointwise [48, 53, 39, 10, 49], pairwise [24, 21, 4, 3], triplet-wise [44, 31], listwise [51]. While non pointwise methods exhibit computational complexity with at least $O(N^2)$ for N data points and impractical when N becomes large [48], pointwise methods often based on classification objectives and enjoy the benefit of $O(N)$ computational complexity, which are faster to compute, easier to optimize and yet scalable to large-scale datasets [48].

Nevertheless, end-to-end training of deep hashing is difficult due to the vanishing gradient problem from the sign function which is to apply on the output of network to get binary hash codes. If the continuous codes are not under some constraints during training, a simple classification objective (i.e. cross entropy (CE) loss) will not easily learn discriminative hash codes in which the hash codes are not optimal and expressing high quantization error (see Figure 1). Although it can be solved by relaxation schemes [4, 21, 24], but these schemes still end up with suboptimal hash codes due to the introduction of quantization error. Hence, most deep hashing methods [40, 24, 3, 52, 49] will include quantization error minimization as a side objective which usually in the form of $\|\mathbf{v} - \mathbf{b}\|_2^2$ where $\mathbf{v}$ is the continuous codes and $\mathbf{b}$ is the hash codes. To better ensure the quality of hash

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

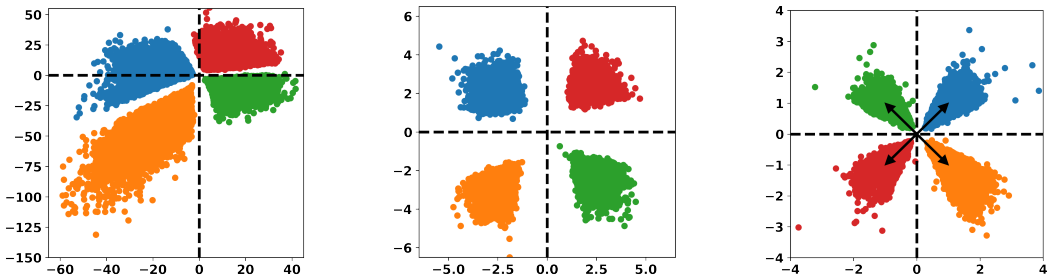


Figure 1: We trained a simple CNN model on CIFAR10 with only first 4 classes and 2-bits, then we visualize the continuous codes $\mathbf{v}$ before *sgn*. The xy-axis are the value of continuous codes. **(Left)** The model is trained with **CE** only, although able to separate the classes in the Euclidean space, but the output is not bounded, expressing high quantization error, and not optimal in Hamming space. **(Middle)** By appending batch normalization (BN) layer after $\mathbf{v}$, zero-mean ensure that the hash codes are balanced, yet the output is bounded by variance of 1. Hence getting good retrieval performance. **(Right)** The black arrows are the **binary orthogonal target**, $\mathbf{o}$ for each class, the model is trained to maximize the cosine similarity between $\mathbf{v}$ and its corresponding binary target $\mathbf{o}$. The continuous codes exhibit lower intra-class variance and quantization error as compared with **CE+BN** model.

codes, these methods include even more sophisticated objectives such as bit balance loss [52, 48, 39], weights constraints to maximize Hamming distance [53], code orthogonality [30, 31] or bypassing the vanishing gradient problem [40, 39, 26] which eventually increased the complexity of the training.

Thus, one question worthy of study arises: Can we unify training objectives of deep hashing such that by minimizing only *a single objective* can maximize the performance of retrieval[1] with hash codes? To this end, we propose to approach the problem under *a single classification objective* which is to maximize the cosine similarity between the continuous codes and *binary orthogonal target* with CE loss, we name our method **OrthoHash**.

In Section 3.1, we reformulate the problem of deep hashing (i.e. Hamming distance based retrieval and minimization of quantization error) in the lens of cosine similarity. Therefore, maximizing the cosine similarity between the L_2-normalized continuous codes and *binary orthogonal target* can actually achieve maximal inter-class Hamming distance and minimize the quantization error simultaneously under *a single classification objective*. We can leverage the benefit of margin [41, 8] to further improve the intra-class variance. Moreover, conventional CE loss only works for single-label classification, we leverage the concept of Label Smoothing [37] to modify the CE loss to adapt multi-labels classification. In Section 3.3, we propose to use batch normalization [15] (BN) layer for code balancing. Empirically, we show that appending a BN layer after the continuous codes can maximize bit entropy and improved performance for about 5-20% compared with a CE model without BN (see Section 4.2). In Section 4, extensive experiments show that our method achieves comparable state-of-the-art (SOTA) performance on three benchmark datasets, i.e. ImageNet100, NUS-WIDE and MS-COCO. Yet, to evaluate the efficiency of different deep hashing methods on large-scale instance-level retrieval, our method also achieves SOTA in GLDv2, $\mathcal{R}$Oxf and $\mathcal{R}$Paris by at most 0.6%, 6.4% and 9.2% improvement respectively.

2 Related Work

Hashing methods. Typically hashing can be categorized into many streams. Data-independent methods such as Locality-sensitive Hashing (LsH) [14, 11], and its kernelized version (KLsH) [20] have built up many fundamental components for hashing such as the requirement of **code balance**, **uncorrelated bit**, and **similarity preserving**. Furthermore, data-dependent methods such as [45, 19, 13, 17, 35, 36] have always shown improvement over data-independent methods as the learned hash codes can be more compact and more specific over the datasets. Recently, deep learning to hash methods [24, 47, 21] even shown better improvement over conventional data-dependent methods due to the superior learning ability of DNN, able to compute a better feature representation from the input, hence better hash codes. These methods defined objective functions for DNN to learn suitable hash codes over the datasets. The objective functions include 1) task learning objective

[1] usually the mean average precision (mAP).

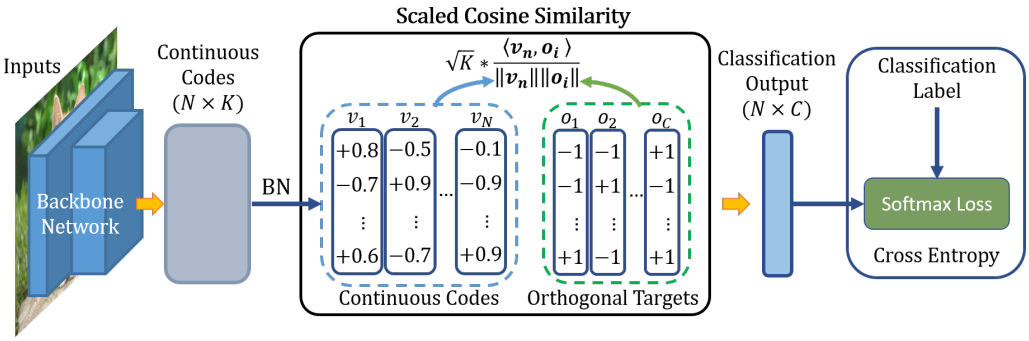


Figure 2: We first obtain continuous codes $\mathbf{V} \in \mathbb{R}^{N \times K}$ from our backbone network, it is then passing through batch normalization (BN) layer to obtain zero-mean continuous codes. Next, we compute scaled cosine similarity between the continuous codes and binary orthogonal target $\mathbf{O} \in \{-1, +1\}^{C \times K}$ where $\mathbf{C}$ is the number of classes. Finally, the scaled cosine similarity act as classification output and we minimize the softmax cross entropy loss.

which can further be categorized into pointwise [48, 53, 39, 40, 10, 49], pairwise [4, 24, 21], triplet-wise [44, 31], listwise [51] and unsupervised [26, 12]; 2) quantization error minimization such as minimize the p-norm (usually $p = 2$) between continuous codes and hash codes; 3) information entropy maximization to achieve code balance [26, 39]. We refer readers to learning to hash survey [43, 42, 9] for better history and details in hashing.

Binary optimization. Hashing is a NP-hard binary optimization problem [45], and also causing vanishing gradient problem due to the discrete and non-differentiable binary hash functions. At the early year, this problem was first solved by discarding the discrete constraints (e.g. design a penalty loss term to generate feature as binary as possible [24, 21]). Some methods utilized coordinate descent method in the training [27, 23]. Then, [4] proposed to solve with continuous relaxation, that is to optimize in a continuous space using sigmoid or tanh for approximation, and then quantize the continuous values to obtain hash codes. Nevertheless, these methods have increased the complexity of learning to hash due to the tuning of hyper-parameters, and different datasets might lead to different scale hyper-parameters in order to get best performance.

Modifying computational graph. Greedy Hash [40] designed a new coding layer which uses the sign function in the forward pass to generate binary codes, and gradients are backpropagated using straight-through estimator [1] during optimization. [26] designed a parameter-free coding layer – *Bi-half*, to maximize the bit capacity (so that each bit can have a 50% chance of being $+1$ or -1) by shifting the network output by *median*.

Learning hash codes with pre-defined target. Deep Polarized Network (DPN) [10] uses random assignment scheme to generate target vectors with maximal inter-class distance, then optimized with their hinge-like polarized loss. Central Similarity Quantization (CSQ) [49] uses Hadamard matrix as "hash centers", then optimized with binary cross entropy. Both methods have similar objective, which is the continuous codes is learned to be as similar as the target vectors (or "hash centers").

Cosine similarity. While most works focus on how to hash an image with various constraints, we reformulate the problem of deep hashing in the lens of cosine similarity. As inspired by [50, 12] which utilize cosine similarity to find closest approximate binary or ternary representation, we also reformulate the quantization error in terms of cosine similarity between continuous codes and hash codes. Moreover, under the framework of deep hypersphere embedding learning (SphereFace [34], CosFace [41] and ArcFace [8]), they imposed discriminative constraints on a hypersphere manifold and proposed to improve decision boundary by cosine or angular margin. We leverage the benefit of margin to improve intra-class variance.

3 Unifying Training Objectives for Deep Hashing

In this section, we describe our approach for deep hashing. In Section 3.1, we reformulate the problem of deep hashing in the lens of cosine similarity, i.e. interpreting Hamming distance retrieval and quantization error using cosine similarity. In Section 3.2, we propose to maximize cosine similarity

between the continuous codes and binary orthogonal target under a single classification objective (for both single-label and multi-labels classification). Finally, we describe why appending batch normalization layer after the continuous codes can achieve code balance in Section 3.3. Figure 2 summarizes our approach and the intuitive outcome.

We will introduce some notations. Let d-dimensional data, $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^{N} \in \mathbb{R}^{N\times d}$ where N is number of training samples, and $\mathbf{Y} = \{\mathbf{y}_n\}_{n=1}^{N}$ as one-hot training labels of C classes (for multi-labels, $y_{ni} = 1$ represents i-th class is assigned to the n-th sample). Our objective is to learn a set of K-bit binary codes $\mathbf{B} = \{\mathbf{b}_n\}_{n=1}^{N} \in \{-1, 1\}^{N\times K}$ for each training point $\mathbf{x}_n$, which is converted from the continuous codes $\mathbf{v}_n$ through a *sgn* function. $\mathbf{v}$ can be computed by a latent layer $\mathcal{H}(\mathbf{x}) = \mathbf{W}\phi(\mathbf{x}) \in \mathbb{R}^K$, ϕ is a deep neural network (backbone network) to compute q-dimensional nonlinear feature representation $\mathbf{f} = \phi(\mathbf{x}) \in \mathbb{R}^q$, $\mathbf{W} \in \mathbb{R}^{K\times q}$ is the weights of the latent layer and $sgn(\mathbf{v}_{nk}) = 1$ if k-th bit of $\mathbf{v}_n \geq 0$ and -1 otherwise. We use a or A to represent scalar, $\mathbf{a}$ to represent vector and $\mathbf{A}$ to represent matrix.

3.1 Reformulate Deep Hashing in the lens of Cosine Similarity

3.1.1 Interpreting Hamming Distance as Cosine Similarity

We start by interpreting the hamming distance as cosine similarity. Typically, the Hamming distance can be computed using logical *xor* operation between $\mathbf{b}_i$ and $\mathbf{b}_j$, followed by *popcount*. If $\mathbf{b}$ is represented by $\{-1, +1\}^K$, then Hamming distance can also computed mathematically as:

$$D(\mathbf{b}_i, \mathbf{b}_j) = \frac{K - \langle \mathbf{b}_i \,, \mathbf{b}_j \rangle}{2} \tag{1}$$

in which $\langle \cdot \rangle$ is dot product between two vectors. Geometrically, the dot product can be interpreted as:

$$\langle \mathbf{b}_i \,, \mathbf{b}_j \rangle = \|\mathbf{b}_i\| \, \|\mathbf{b}_j\| \cos\theta_{ij} \tag{2}$$

in which $\|\cdot\|$ is Euclidean norm and θ_{ij} is the angle between $\mathbf{b}_i$ and $\mathbf{b}_j$. As both $\|\mathbf{b}_i\|$ and $\|\mathbf{b}_j\|$ are constant (i.e. $\|\mathbf{b}\| = \sqrt{K}$), equation (1) can then be viewed as:

$$D(\mathbf{b}_i, \mathbf{b}_j) = \frac{K - K\cos\theta_{ij}}{2} = \frac{K}{2}(1 - \cos\theta_{ij}) \tag{3}$$

Since $\frac{K}{2}$ is a constant, we can see that the retrieval is now only based on the angle between two hash codes i.e. more similar hash codes will have a similar direction, and yield lower angle between them, hence lower hamming distance.

3.1.2 Interpreting Quantization Error as Cosine Similarity

Often, converting to binary codes $\mathbf{b}$ from continuous codes $\mathbf{v}$ will often lose a lot of information, which is called quantization error. Therefore, most hashing methods include quantization error minimization in their learning objective such as L_1-norm, L_2-norm and p-norm (e.g. $p = 3$ in Greedy Hash [40]), usually in the form of:

$$\min L + \lambda Q \tag{4}$$

in which L is supervised learning objective such as Cross Entropy and Q is the quantization error between $\mathbf{v}$ and $\mathbf{b}$. However, it is difficult to control the scale λ, as lower λ might not be effective, and higher λ might lead to underfitting. Careful tuning is required and yet the tuned λ vary in different tasks. Even so, we can first interpret quantization error geometrically:

$$\min \|\mathbf{v} - \mathbf{b}\|^2 \;\; \text{s.t.} \;\; \mathbf{b} \in \{-1, 1\}^K \tag{5}$$

in which $\mathbf{v}$ is in continuous space, $\mathbf{b} = sgn(\mathbf{v})$ is in binary space. We expand equation (5) to get:

$$\begin{aligned} \|\mathbf{v} - \mathbf{b}\|^2 &= \langle \mathbf{v} - \mathbf{b} \,, \mathbf{v} - \mathbf{b} \rangle \\ &= \|\mathbf{v}\|^2 - \langle \mathbf{v} \,, \mathbf{b} \rangle - \langle \mathbf{b} \,, \mathbf{v} \rangle + \|\mathbf{b}\|^2 \\ &= \|\mathbf{v}\|^2 - 2\langle \mathbf{v} \,, \mathbf{b} \rangle + \|\mathbf{b}\|^2 \\ &= \|\mathbf{v}\|^2 + \|\mathbf{b}\|^2 - 2\,\|\mathbf{v}\|\,\|\mathbf{b}\| \cos\theta_{vb} \end{aligned} \tag{6}$$

According to equation (3), retrieval is only based on the similarity in the direction of two hash codes. Hence, we can ignore the magnitude of $\mathbf{v}$ by normalizing it to have the same norm with $\mathbf{b}$, i.e., $\|\mathbf{v}\| = \sqrt{K}$ and interpret quantization error as only the angle θ_{vb} between $\mathbf{v}$ and $\mathbf{b}$:

$$\begin{aligned}\|\mathbf{v}-\mathbf{b}\|^2 &= \|\mathbf{v}\|^2 + \|\mathbf{b}\|^2 - 2\|\mathbf{v}\|\|\mathbf{b}\|\cos\theta_{vb} \\ &= \sqrt{K}^2 + \sqrt{K}^2 - 2\sqrt{K}\sqrt{K}\cos\theta_{vb} \\ &= 2K - 2K\cos\theta_{vb} = 2K(1-\cos\theta_{vb}) \end{aligned} \tag{7}$$

As $2K$ is a constant, we can then conclude that maximize cosine similarity between $\mathbf{v}$ and $\mathbf{b}$ leads to lower quantization error, hence the better the approximation in hash codes.

3.2 Discriminative Hash Codes with Orthogonal Target

According to [5], the probability of two samples $\mathbf{x}_i$ and $\mathbf{x}_j$ getting same hash code under a family $\mathcal{F}$ of hash functions using random hyperplane technique can be described as $\mathbf{Pr}_{h\in\mathcal{F}}[h(\mathbf{x}_i) = h(\mathbf{x}_j)] = 1 - \frac{\theta_{ij}}{\pi}$, where $h()$ is a hash function and θ_{ij} is the angle between $\mathbf{x}_i$ and $\mathbf{x}_j$. Hence, we can also apply to if the two continuous codes $\mathbf{v}_i$ and $\mathbf{v}_j$ from latent layer $\mathcal{H}$ have high cosine similarity, then the hash codes $\mathbf{b}_i$ and $\mathbf{b}_j$ should have high chance of getting same hash codes. Furthermore, cosine similarity can be used to justify the performance of retrieval using hash codes (see Section 3.1.1) and quantization error between continuous codes and hash codes (see Section 3.1.2), we therefore propose to maximize the cosine similarity of the continuous codes $\mathbf{v}$ and its corresponding **binary orthogonal target** $\mathbf{o}$, this can be done by maximizing the posterior probability of the grounth-truth class using softmax (cross-entropy) loss:

$$L = -\frac{1}{N}\sum_{n=1}^{N}\log\frac{\exp(\mathbf{o}_{y_n}\mathbf{v_n})}{\sum_i^C \exp(\mathbf{o}_i\mathbf{v_n})} \tag{8}$$

where $\mathbf{v}_n$ denotes the deep continuous codes of the n-th samples from DNN ϕ and both $\mathbf{o}_{y_n}$ and $\mathbf{o}_i \in \mathbb{R}^K$ denotes the ground-truth class y_n and the i-th row of the binary orthogonal target $\mathbf{o} \in \mathbb{R}^{C\times K}$. For simplicity, we omit the bias term from the equation (8). Then, under the framework of deep hypersphere embedding [34, 41, 8], we can transform the logit $\mathbf{o}_i\mathbf{v}_n = \|\mathbf{o}_i\|\|\mathbf{v}_n\|\cos\theta_i$ where θ_i is the angle between the continuous codes $\mathbf{v}_n$ and the binary orthogonal target $\mathbf{o}_i$. Next, we perform L_2 normalization on $\mathbf{v}_n$ so that $\|\mathbf{v}_n\| = 1$, and $\|\mathbf{o}_i\| = \sqrt{K}$ since it is in binary form, hence we can re-define our loss function as:

$$L = -\frac{1}{N}\sum_{n=1}^{N}\log\frac{\exp(\sqrt{K}\cos\theta_{y_n})}{\exp(\sqrt{K}\cos\theta_{y_n}) + \sum_{i=1,i\neq y_n}^{C}\exp(\sqrt{K}\cos\theta_i)} \tag{9}$$

Instead of including quantization error minimization in the training objective as shown in the equation (4), our method unify both learning objective and quantization error minimization under *a single classification objective* as shown in the loss function (equation (9)). Since the binary orthogonal targets achieve maximal inter-class Hamming distance, and our loss function aim to minimize the intra-class variance, we can leverage the benefit of cosine or angular margin[2] as introduced by CosFace [41] and ArcFace [8] to further improve the minimization of intra-class variance (we set $m = 0.2$ in all of our experiments unless mention explicitly). As a result, our method can perform end-to-end training to learn highly discriminative hash codes without sophisticated training objectives or by modifying computational graph.

3.2.1 Binary Orthogonal Target

The maximization of the expectation of inter-class Hamming distance will help to increase recall rate during retrieval as lesser chance to retrieve incorrect items. Suppose K-bit Hamming space $\mathbb{H}^K \in \{-1,+1\}^K$, for any two binary vectors $\mathbf{b}_i, \mathbf{b}_j$ sampled with probability p for $+1$ on each bit, the expectation of Hamming distance is $\mathbb{E}[D(\mathbf{b}_i,\mathbf{b}_j)] = 2\cdot K\cdot p(1-p)$ and it achieves the upper bound of $\frac{K}{2}$ with $p = 0.5$ [10, 49][3]. Hence, hash codes $\mathbf{b}_i$ and $\mathbf{b}_j$ must be orthogonal so that we can get $D(\mathbf{b}_i,\mathbf{b}_j) = \frac{K}{2}$ in equation (3).

[2] cosine margin will transform $\exp(\sqrt{K}\cos\theta_{y_n})$ to $\exp(\sqrt{K}(\cos(\theta_{y_n}) - m))$ and angular margin will transform the same to $\exp(\sqrt{K}\cos(\theta_{y_n} + m))$

[3] see supplementary material for detailed explanation

Orthogonal Targets Generation. Orthogonal binary targets can be sampled from Bernoulli distribution $Bern(0.5)$, sign of Gaussian distribution $\mathcal{N}(0,1)$ or sign of Uniform distribution $\mathcal{U}(-1,1)$, as long as having $p = 0.5$ for $+1$ on each bit. Moveover, orthogonal binary targets can be generated from Hadamard matrix [49, 28], if number of rows in the generated Hadamard matrix is less than number of classes, then the binary targets for remaining classes are generated from $Bern(0.5)$. For simplicity, we use $Bern(0.5)$ to generate binary targets for all classes.

3.2.2 Multi-labels Hash Codes Learning

As conventional cross-entropy loss only works for single-label classification, we leverage the concept of Label Smoothing [37] to generate labels for multi-labels classification. A standard cross entropy loss is mathematically formulated as:

$$L = -\frac{1}{N}\sum_{i}^{C} y_{ni}\log(p(y_{ni}|\mathbf{x}_n)) \tag{10}$$

where y_{ni} is the one-hot target label which i-th class is assigned to the n-th sample and $\mathbf{x}$ is the given input sample. In [37], the target label become soft-target such that non-target class has a small "smoothing" value, in our work, we set $y_{ni} = 1/|\mathbf{y}_n|$ where $|\mathbf{y}_n|$ is total number of assigned classes for n-th sample, e.g. $\mathbf{y}_n = [0, 0.5, 0, 0.5]$ when 2nd and 4th classes are the assigned classes. Our motivation is that the model should maximize the probabilities of the target classes, which can optimize the hash codes to be as similar as the binary targets from assigned classes (Nevertheless, we can not guarantee that the final hash codes are the center of hash codes of the target classes, instead we let the optimization algorithm to find the best hash codes). During our experiment, we found out that replacing *softmax* with *sigmoid* for multi-labels are not effective, because *softmax* will intrinsically suppress the lower activated class unit (i.e. scaled cosine similarity) with lower probability and increase the highly activated class unit with higher probability, while *sigmoid* will treat each class unit as an individual unit, hence maximizing probability of a class might not lead to minimizing the probability of other classes. Therefore, we propose to leverage the concept of Label Smoothing to generate labels so that we can use cross entropy loss for learning.

3.3 Code Balance

Although binary orthogonal target can achieve code balance, since every bit has 50% chance being $+1$ or -1, but it is still an optimization problem which will not guarantee that the model can learn to output balance code. Therefore, we propose to **append a batch normalization (BN) layer** after the continuous codes v to ensure the code balance. If $\sum_n \mathbf{v}_{nk} = 0$, then we can see that $\sum_n \mathbf{b}_{nk} = 0$ for k-th bit. Because the distribution of $\mathbf{v}$ has been normalized to have zero-mean and variance of 1, therefore, $\mathbf{b} = sgn(\mathbf{v})$, the hash codes $\mathbf{b}$ will follow a uniform binary distribution with about 50% on $+1$ and 50% on -1. Empirically, it improved the retrieval performance in ImageNet100 by about 5-20% as compared with a model with normal cross entropy loss (see Table 1). See Figure 1 for the effect of appending batch normalization layer. In Bi-half [26], they proposed to shift the continuous codes by median and then converting the continuous codes to binary codes for optimization, but they will have to modify the computational graph in order to have a proxy derivative to solve vanishing gradient problem. However, appending BN layer will not modify the computational graph, hence maintaining end-to-end training.

4 Experiment

Training Setup. We have implemented[4] 7 different deep hashing methods for comparison (5 point-wise, 1 pair-wise and 1 triplet-wise). For a fair comparison, we are using same learning rate of 0.0001, *Adam* optimizer [16] and 100 epochs for all methods[5] and present their results. For **SDH-C** [30], we have modified it from pair-wise objective to point-wise objective, while all penalty terms are remained (i.e. quantization loss, bit variance loss and orthogonality on projection weights).

Evaluation Datasets. We follow the settings from previous works [4, 10, 40, 32, 39, 47, 21, 33] and choose 3 datasets **ImageNet100** [7], **NUS-WIDE** [6] and **MS-COCO** [29] as our benchmark

[4] some taken directly from authors' released code repository

[5] we have summarized the detailed training settings in the supplementary material

Methods	**ImageNet100** (mAP@1K)				**NUS-WIDE** (mAP@5K)				**MS COCO** (mAP@5K)			
	16	32	64	128	16	32	64	128	16	32	64	128
HashNet[2] [4]	0.343	0.480	0.573	0.612	0.814	0.831	0.842	0.847	0.663	0.693	0.713	0.727
DTSH[3] [44]	0.442	0.528	0.581	0.612	**0.816**	**0.836**	**0.851**	**0.862**	0.699	0.732	0.753	0.770
SDH-C[1] [30]	0.584	0.649	0.664	0.662	0.763	0.792	0.816	0.832	0.671	0.710	0.733	0.742
GreedyHash[1] [40]	0.570	0.639	0.659	0.659	0.771	0.797	0.815	0.832	0.677	0.722	0.740	0.746
JMLH[1] [39]	0.517	0.621	0.662	0.678	0.791	0.825	0.836	0.843	0.689	0.733	0.758	0.768
CSQ[1] [49]	0.586	0.666	0.693	0.700	0.797	0.824	0.835	0.839	0.693	**0.762**	0.781	0.789
DPN[1] [10]	0.592	0.670	0.703	0.714	0.783	0.818	0.838	0.842	0.668	0.721	0.752	0.773
CE[1]	0.350	0.379	0.406	0.445	0.744	0.770	0.796	0.813	0.602	0.639	0.658	0.676
CE+BN[1]	0.533	0.586	0.612	0.617	0.801	0.814	0.823	0.825	0.697	0.721	0.729	0.726
CE+Bihalf[1] [26]	0.541	0.630	0.661	0.662	0.802	0.825	0.836	0.839	0.674	0.728	0.755	0.757
OrthoCos[1]	0.583	0.660	0.702	0.714	0.795	0.826	0.842	0.851	0.690	0.745	0.772	0.784
OrthoCos+Bihalf[1]	0.562	0.656	0.698	0.711	0.804	0.834	0.846	0.852	0.690	0.746	0.775	0.782
OrthoCos+BN[1]	0.606	0.679	**0.711**	**0.717**	0.804	**0.836**	0.850	0.856	**0.709**	**0.762**	**0.787**	**0.797**
OrthoArc+BN[1]	**0.614**	**0.681**	0.709	0.714	0.806	0.833	0.850	0.856	0.708	**0.762**	0.785	0.794

Table 1: Performance of different methods for 4 different bits on different benchmark datasets. The order of previous hashing methods are sorted by their average performance overall datasets and bits. All results are run by us. The superscript [1], [2] and [3] indicates point-wise, pair-wise and triplet-wise method respectively. **Bold** values indicate best performance in the column.

datasets. Furthermore, to understand the effectiveness of different hashing methods in large-scale instance-level retrieval (i.e. tremendous number of classes & unseen classes), we propose to evaluate on **GLDv2** [46], $\mathcal{R}$**Oxf** and $\mathcal{R}$**Par** [38]. We will describe each dataset in details in the supplementary material.

Architecture. Follow the settings in [4, 40, 39, 10], we use pre-trained AlexNet [18] as network backbone initialization (our method can work on any deep network architecture), the output from last fully-connected with ReLU (4096-dimension vector) act input to the latent layer, then train the model with various supervised deep hashing methods. The image input to the network backbone is 224×224. For instance-level retrieval, due to the expensive cost of training from scratch, we use pre-trained model[6] (**R50-DELG-GLDv2-clean**) from DELG [2] to compute the 2048-dimension global descriptors, then we train a latent layer $\mathcal{H}$ to compute hash codes where inputs are the global descriptors. For GLDv2, the images input are 512×512. For $\mathcal{R}$Oxf and $\mathcal{R}$Par, we use 3 scales $\{\frac{1}{\sqrt{2}}, 1, \sqrt{2}\}$ to produce multi-scale representations, and followed by L_2 normalization, then average-pooled to obtain a single descriptor as done by [2], and we GLDv2-trained latent layer to compute hash codes for unseen classes evaluation.

4.1 Retrieval Performance

To evaluate the performance on benchmark datasets, we use mean average precision (mAP@R) which is mean of average precision of top R retrieved items. Table 1 summarizes the performance comparison between all previous hashing methods and our methods. **CE** denotes model trained with cross entropy only, the hash codes are computed from sign of continuous codes. **CE+BN** denotes **CE** model with BN layer [15] appended after the latent layer. **CE+Bihalf** denotes **CE** model with Bihalf [26] layer appended after the latent layer. **OrthoCos** denotes model trained with cosine margin and binary orthogonal target. **OrthoCos+Bihalf** denotes a variant of **OrthoCos**, and with Bihalf layer appended. **OrthoCos+BN** denotes a variant of **OrthoCos**, and with BN layer appended. **OrthoArc+BN** denotes a variant of **OrthoCos+BN**, trained with angular margin.

Overall. It was observed that both **OrthoCos+BN** and **OrthoArc+BN** perform better than recent state-of-the-art, **DPN** [10] and **CSQ** [49], e.g. on ImageNet100 dataset, which 0.711/0.717 and 0.709/0.714 with 64 and 128-bits hash codes. On multi-labeled datasets (i.e. NUS-WIDE and MS COCO), **DTSH** [44] (triplet based method) performed the best with 0.851 and 0.862 with 64 and 128-bits hash codes in NUS-WIDE followed by our method (e.g. **OrthoCos+BN** achieves 0.850 and 0.856 in the same settings), while **OrthoCos+BN** and **OrthoArc+BN** performed the best in MS-COCO with at most 1% improvement over previous deep hashing methods.

[6] `https://github.com/tensorflow/models/tree/master/research/delf`

Methods	**GLDv2** (mAP@100)			$\mathcal{R}$**Oxf-Hard** (mAP@all)			$\mathcal{R}$**Paris-Hard** (mAP@all)		
	128	512	2048	128	512	2048	128	512	2048
HashNet[2] [4]	0.018	0.069	0.111	0.034	0.058	0.307	0.133	0.190	0.490
DPN[1] [10]	0.021	0.089	0.133	0.053	0.184	0.303	0.224	0.399	0.562
GreedyHash[1] [40]	0.029	**0.108**	0.144	0.032	0.251	0.373	0.128	0.531	0.652
CSQ[1] [49]	0.023	0.086	0.114	0.093	0.284	0.398	0.245	0.541	0.649
OrthoCos+BN[1]	**0.035**	**0.108**	**0.149**	**0.123**	**0.348**	**0.436**	**0.337**	**0.597**	**0.667**
R50-DELG-H	-	-	0.125*	-	-	0.471	-	-	0.682
R50-DELG-C	-	-	0.138*	-	-	0.510	-	-	0.715

Table 2: Performance of different methods for 3 different bits on different instance-level benchmark datasets. The order of previous hashing methods are sorted by their average performance overall datasets and bits. All results are run by us. The superscript [1] and [2] indicates point-wise and pair-wise method respectively. **Bold** values indicate best performance in the column. * indicates using 512 $\times$ 512 image inputs, hence different performance as reported by DELG [2]. R50-DELG-H denotes Hamming distance retrieval using the sign of extracted descriptors. R50-DELG-C denotes Cosine distance retrieval using the extracted descriptors.

Code Balance. Although the retrieval performance in **CE** model performed the worst, but by appending BN layer after the latent layer (**CE+BN**), has shown at least 5-20% improvement over all settings (dataset and number of bits). Bihalf [26] layer (zero-median features) has a proxy derivative to learn hash features, hence getting 0.1-4.9% better performance than **CE+BN**. This has shown that without sophisticated training objectives, code balance itself is very important in improving Hamming distance based retrieval. However, **OrthoCos+Bihalf** didn't show significant improvement over **OrthoCos+BN**, and yet comparable with **OrthoCos**, hence we conclude that our method can achieve code balance without explicitly engineering the computational graph.

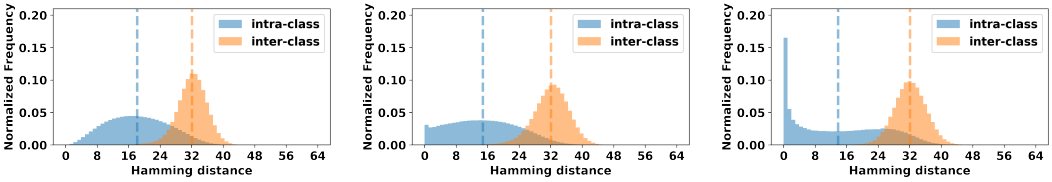
Cosine and Angular Margin. In our experiments, we observed that cosine margin (**OrthoCos+BN**) slightly perform better than angular margin (**OrthoArc+BN**) by about 0.5% on average only.

4.1.1 Instance-Level Retrieval

To evaluate the instance level retrieval performance, we follow the evaluation method released by the authors [2, 38]. The baseline performance of GLDv2, $\mathcal{R}$Oxf-Hard and $\mathcal{R}$Par-Hard from the pre-trained **R50-DELG-GLDv2-clean** are 0.138, 0.510 and 0.715 respectively. Table 2 summarizes the comparison of performance of different deep hashing methods with our method. In all datasets (GLDv2, $\mathcal{R}$Oxf-Hard and $\mathcal{R}$Par-Hard), our method outperforms all the previous deep hashing method over all bits by at most 0.6%, 6.4% and 9.2% respectively. This has shown that our method has better generalization ability on unseen classes than previous deep hashing methods. For particular example, we outperforms previous deep hashing by 0.6%, 3% and 9.2% respectively on 3 datasets with 128-bits hash codes.

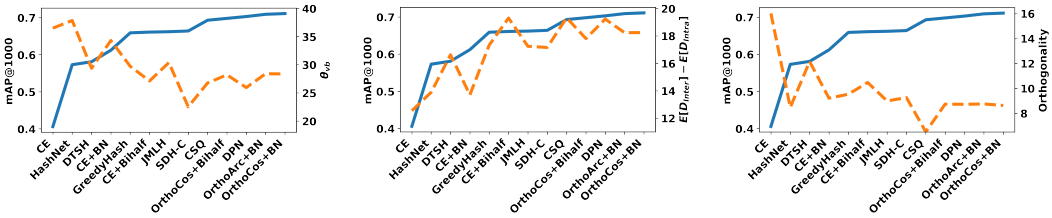
Orthogonal Transformation. For GLDv2 2048-bits hash codes, surprisingly it can achieve even better performance than pre-trained 2048-dimensions descriptors (by 1.1% improvement over R50-DELG-C). We then analyze the separability in cosine distances i.e. the difference in the mean of intra-class cosine distance and the mean of inter-class cosine distance before and after the transformation (similar to Figure 3). We observed that the separability in cosine distances increases after the orthogonal transformation, i.e. before is 0.142 and after is 0.167. Therefore, learning orthogonal hash codes can transform inputs to be more discriminative.

Domain shifting with BN. As the model is trained with GLDv2, hence the running mean and variance in the BN layer might experience domain shifting problem [25] when we inference on different datasets (e.g. $\mathcal{R}$Oxf and $\mathcal{R}$Par). During experiment, we observed that using running mean and variance from GLDv2 caused large performance drop in Hamming distance retrieval. One simple solution is to recompute the mean and variance from all continuous codes in the database, then update the running mean and variance with the computed mean and variance. The performances of $\mathcal{R}$Oxf and $\mathcal{R}$Par in Table 2 are obtained with running mean and variance of the respective database continuous codes.



(a) HashNet [4] (b) GreedyHash [40] (c) OrthoCos+BN

Figure 3: Histogram of intra-class and inter-class Hamming distance with 64-bits ImageNet100. We normalized the frequency so that sum of all bins equal to 1.



(a) Quantization error in angle θ_{vb} (b) $\mathbb{E}[D_{inter}] - \mathbb{E}[D_{inter}]$ (c) Orthogonality

Figure 4: Analysis of retrieval peformance of 64-bits ImageNet100. Blue solid line denotes mean average precision (mAP@1000) and orange dotted line denotes respective analysis score.

4.2 Empirical Analysis

Histogram of Hamming Distances. Figure 3 summarizes the histogram of intra-class and inter-class distances. We compare our method **OrthoCos+BN** with pair-wise method HashNet [4] and point-wise classification based GreedyHash [40]. Although the distribution of inter-class distances are about the same for the 3 methods (close to Hamming distance of $K/2 = 32$), but we can see that the larger the difference in the mean of intra-class distance (the blue dotted line) with the mean of inter-class distance (the orange dotted line), the better the performance.

Performance Improvement Analysis. We further analyze the reason of performance improvement in different deep hashing methods and summarizes the analysis in Figure 4. There are 3 main reasons that contribute to the improvement in deep hashing methods: 1) quantization error; 2) the separability in Hamming distances; 3) orthogonality in hash centers. For quantization error, we measure the angle θ_{vb} between the continuous codes $\mathbf{v}$ and the hash codes $\mathbf{b}$ i.e. $\theta_{vb} = deg(\arccos(\frac{v^\intercal b}{\|v\|\|b\|}))$. For the separability, we measure the difference in the average inter-class distances and the average intra-class distances, i.e. $\mathbb{E}[D_{inter}] - \mathbb{E}[D_{intra}]$. For orthogonality, we first compute the hash centers $\mathbf{H} \in \{-1, +1\}^{C \times K}$ for every class (by taking the sign of average hash codes in every class), then we measure the orthogonality with $\left\| \frac{1}{K}\mathbf{H}\mathbf{H}^\intercal - \mathbb{I} \right\|^2$. When the quantization error reduced, the separability increases and the hash centers has better orthogonality, then the performance improved.

5 Conclusion & Future Work

In this work, we proposed to unify training objectives of deep hashing under a single classification objective by maximizing the cosine similarity between the continuous codes and binary orthogonal target with cross entropy loss. We reformulated the problem of deep hashing in the lens of cosine similarity which we showed that both retrieval performance and quantization error can be interpreted with cosine similarity. We further showed that if we perform L_2-normalization on the continuous codes, then end-to-end training of deep hashing is possible without extra sophisticated constraints. Yet, maximum inter-class Hamming distance is important in order to improve the discriminative-ness of hash codes, which can be achieved by learning with binary orthogonal target. Moreover, we leverage the concept of Label Smoothing to train multi-labels classification with cross-entropy loss and batch normalization for code balancing. Extensive experiments has proven that the efficiency of our method in both general retrieval benchmarks and instance-level retrieval benchmarks. As in future research direction, we are exploring how to learn better feature representations to improve retrieval performance by using hash codes through unsupervised learning.

Broader Impact

As the DNN become advance everyday and more retrieval applications are based on DNN, but with millions of images, it is normal that we first select some candidates by fast and approximate retrieval with hash codes, then only perform re-ranking based on better local features. However, many methods are difficult to train and reproduce, especially methods with many hyperparameters to train and multi-stage methods. This work is to unify the learning to hash objectives under a single objective, yet without modifying computational graph during backpropagation. Instead of looking for sophisticated constraints or modifications, we think that retrieval with hash codes can ignore the norm and based on only the cosine similarity, hence we come out with a solution that is in the lens of cosine similarity, which makes end-to-end deep hashing possible. Therefore, this work not only to demonstrate that a single classification objective can achieve SOTA but also to remind future deep hashing method to be as simple as possible.

References

[1] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[2] Bingyi Cao, Andre Araujo, and Jack Sim. Unifying deep local and global features for image search. In *ECCV 2020*, 2020.

[3] Yue Cao, Mingsheng Long, Bin Liu, and Jianmin Wang. Deep cauchy hashing for hamming space retrieval. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1229–1237, 2018.

[4] Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Philip S. Yu. Hashnet: Deep learning to hash by continuation. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5609–5618, 2017.

[5] Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388, 2002.

[6] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yan-Tao Zheng. Nus-wide: A real-world web image database from national university of singapore. In *Proc. of ACM Conf. on Image and Video Retrieval (CIVR'09)*, Santorini, Greece., July 8-10, 2009.

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[8] Jiankang Deng, Jia Guo, Xue Niannan, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *CVPR*, 2019.

[9] Shiv Ram Dubey. A decade survey of content based image retrieval using deep learning, 2020.

[10] Lixin Fan, Kam Woh Ng, Ce Ju, Tianyu Zhang, and Chee Seng Chan. Deep polarized network for supervised learning of accurate binary hashing codes. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 825–831. International Joint Conferences on Artificial Intelligence Organization, 7 2020. Main track.

[11] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al. Similarity search in high dimensions via hashing. In *Vldb*, volume 99, pages 518–529, 1999.

[12] Yunchao Gong, Sanjiv Kumar, Vishal Verma, and Svetlana Lazebnik. Angular quantization-based binary codes for fast similarity search. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[13] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE transactions on pattern analysis and machine intelligence*, 35(12):2916–2929, 2012.

[14] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, page 604–613, New York, NY, USA, 1998. Association for Computing Machinery.

[15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR.

[16] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[17] Weihao Kong and Wu-jun Li. Isotropic hashing. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, page 1097–1105, Red Hook, NY, USA, 2012. Curran Associates Inc.

[19] Brian Kulis and Trevor Darrell. Learning to hash with binary reconstructive embeddings. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc., 2009.

[20] Brian Kulis and Kristen Grauman. Kernelized locality-sensitive hashing for scalable image search. In *2009 IEEE 12th international conference on computer vision*, pages 2130–2137. IEEE, 2009.

[21] Hanjiang Lai, Yan Pan, Ye Liu, and Shuicheng Yan. Simultaneous feature learning and hash coding with deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3270–3278, 2015.

[22] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[23] Qi Li, Zhenan Sun, Ran He, and Tieniu Tan. Deep supervised discrete hashing. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[24] Wu-Jun Li, Sheng Wang, and Wang-Cheng Kang. Feature learning based deep supervised hashing with pairwise labels. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, page 1711–1717. AAAI Press, 2016.

[25] Yanghao Li, Naiyan Wang, Jianping Shi, Xiaodi Hou, and Jiaying Liu. Adaptive batch normalization for practical domain adaptation. *Pattern Recognition*, 80:109–117, 2018.

[26] Yunqiang Li and Jan van Gemert. Deep unsupervised image hashing by maximizing bit entropy. *AAAI*, 2021.

[27] Guosheng Lin, Chunhua Shen, David Suter, and Anton van den Hengel. A general two-step approach to learning-based hashing. In *2013 IEEE International Conference on Computer Vision*, pages 2552–2559, 2013.

[28] Mingbao Lin, Rongrong Ji, Hong Liu, and Yongjian Wu. Supervised online hashing via hadamard codebook learning. In *Proceedings of the 26th ACM International Conference on Multimedia*, MM '18, page 1635–1643, New York, NY, USA, 2018. Association for Computing Machinery.

[29] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing.

[30] Venice Erin Liong, Jiwen Lu, Gang Wang, Pierre Moulin, and Jie Zhou. Deep hashing for compact binary codes learning. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2475–2483, 2015.

[31] Bin Liu, Yue Cao, Mingsheng Long, Jianmin Wang, and Jingdong Wang. Deep triplet quantization. In *Proceedings of the 26th ACM international conference on Multimedia*, pages 755–763, 2018.

[32] Haomiao Liu, Ruiping Wang, Shiguang Shan, and Xilin Chen. Deep supervised hashing for fast image retrieval. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2064–2072, 2016.

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

[34] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphereface: Deep hypersphere embedding for face recognition. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6738–6746, 2017.

[35] Mohammad Norouzi and David J. Fleet. Minimal loss hashing for compact binary codes. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 353–360, Madison, WI, USA, 2011. Omnipress.

[36] Mohammad Norouzi, David J Fleet, and Russ R Salakhutdinov. Hamming distance metric learning. In *Advances in neural information processing systems*, pages 1061–1069, 2012.

[37] Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*, 2017.

[38] Filip Radenovic, Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. Revisiting oxford and paris: Large-scale image retrieval benchmarking. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5706–5715, 2018.

[39] Yuming Shen, Jie Qin, Jiaxin Chen, Li Liu, Fan Zhu, and Ziyi Shen. Embarrassingly simple binary representation learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, Oct 2019.

[40] Shupeng Su, Chao Zhang, Kai Han, and Yonghong Tian. Greedy hash: Towards fast optimization for accurate hash coding in cnn. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[41] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. Cosface: Large margin cosine loss for deep face recognition. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5265–5274, 2018.

[42] Jingdong Wang, Ting Zhang, jingkuan song, Nicu Sebe, and Heng Tao Shen. A survey on learning to hash. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):769–790, 2018.

[43] Jun Wang, Wei Liu, Sanjiv Kumar, and Shih-Fu Chang. Learning to hash for indexing big data - a survey, 2015.

[44] Xiaofang Wang, Yi Shi, and Kris M Kitani. Deep supervised hashing with triplet labels. *Asian Conference on Computer Vision*, 2016.

[45] Yair Weiss, Antonio Torralba, and Rob Fergus. Spectral hashing. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc., 2009.

[46] Tobias Weyand, Andre Araujo, Bingyi Cao, and Jack Sim. Google Landmarks Dataset v2 - A Large-Scale Benchmark for Instance-Level Recognition and Retrieval. In *Proc. CVPR*, 2020.

[47] Rongkai Xia, Yan Pan, Hanjiang Lai, Cong Liu, and Shuicheng Yan. Supervised hashing for image retrieval via image representation learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 28, 2014.

[48] Huei-Fang Yang, Kevin Lin, and Chu-Song Chen. Supervised learning of semantics-preserving hash via deep convolutional neural networks. *IEEE transactions on pattern analysis and machine intelligence*, 40(2):437–451, 2017.

[49] Li Yuan, Tao Wang, Xiaopeng Zhang, Francis EH Tay, Zequn Jie, Wei Liu, and Jiashi Feng. Central similarity quantization for efficient image and video retrieval. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3083–3092, 2020.

[50] Tianyu Zhang, Lei Zhu, Qian Zhao, and Kilho Shin. Neural networks weights quantization: Target none-retraining ternary (tnt), 2019.

[51] Fang Zhao, Yongzhen Huang, Liang Wang, and Tieniu Tan. Deep semantic ranking based hashing for multi-label image retrieval. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1556–1564, 2015.

[52] Xiangtao Zheng, Yichao Zhang, and Xiaoqiang Lu. Deep balanced discrete hashing for image retrieval. *Neurocomputing*, 403:224–236, 2020.

[53] Chang Zhou, Lai-Man Po, Wilson Y. F. Yuen, Kwok Wai Cheung, Xuyuan Xu, Kin Wai Lau, Yuzhi Zhao, Mengyang Liu, and Peter H. W. Wong. Angular deep supervised hashing for image retrieval. *IEEE Access*, 7:127521–127532, 2019.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] We discussed one limitation in Section 4.1.1 "Domain shifting with BN". But we also provided a solution for it.
   (c) Did you discuss any potential negative societal impacts of your work? [No] Our research is to hash image in to binary codes and hence we do not think that this will lead to any potential negative societal impacts.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 3.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Section 3 and supplementary material.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We did not include full code, but our pseudocode are enough to reproduce experimental results. Please see supplementary material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See supplementary material.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] Our results are averaged from 3 runs, but we did not report the error bars because all results are having similar low standard deviation (about maximum of 0.001 only).
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See supplementary material.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] Yes we have cited the creators, see supplementary material for details.
   (b) Did you mention the license of the assets? [Yes] We tried to mention many as possibles, see supplementary material for details.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No] No, we did not include any new assets.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] Most implementations are referred from open-source GitHub, we will list down the URLs in supplementary material.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] No, we do not think that the data we are using contains personally identifiable information or offensive content.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A] We did not use any crowdsourcing results.