A Probabilistic Code Balance Constraint with Compactness and Informativeness Enhancement for Deep Supervised Hashing

Qi Zhang1,2,4, Liang Hu3,4, Longbing Cao1, Chongyang Shi2∗, Shoujin Wang5, Dora D. Liu4

1Data Science Lab, University of Technology Sydney
2School of Computer Science and Technology, Beijing Institute of Technology
3College of Electronic and Information Engineering, Tongji University
4Deepblue Academy of Sciences
5School of Computing Technologies, RMIT University

qi.zhang-13@student.uts.edu.au, lianghu@tongji.edu.cn, longbing.cao@uts.edu.au

cy_shi@bit.edu.cn, shoujin.wang@mq.edu.au, liudongmei_0506@163.com

Abstract

Building on deep representation learning, deep supervised hashing has achieved promising performance in tasks like similarity retrieval. However, conventional code balance constraints (i.e., bit balance and bit uncorrelation) imposed on avoiding overfitting and improving hash code quality are unsuitable for deep supervised hashing owing to their inefficiency and impracticality of simultaneously learning deep data representations and hash functions. To address this issue, we propose probabilistic code balance constraints on deep supervised hashing to force each hash code to conform to a discrete uniform distribution. Accordingly, a Wasserstein regularizer aligns the distribution of generated hash codes to a uniform distribution. Theoretical analyses reveal that the proposed constraints form a general deep hashing framework for both bit balance and bit uncorrelation and maximizing the mutual information between data input and their corresponding hash codes. Extensive empirical analyses on two benchmark datasets further demonstrate the enhancement of compactness and informativeness of hash codes for deep supervised hash to improve retrieval performance (code available at: https://github.com/mumuxi/dshwr).

1 Introduction

Due to the explosive growth of high-dimensional and large-scale data in real applications, hashing has attracted significant attention and has been widely utilized for fast information search and retrieval tasks in recent years. Intending to improve storage and search efficiency, hashing encodes high-dimensional data into compact binary codes which preserve the similarities of original data. Parallel to traditional data-independent hashing, e.g., locality sensitive hashing (LSH) [Datar et al., 2004] applying random projections as hash functions, this paper focuses on data-dependent hashing (specifically supervised hashing). It aims to learn task-specific hash functions to guarantee the similarities in hash coding space as close as those in the original space and is roughly categorized into supervised and unsupervised hashing [Wang et al., 2016; Wang et al., 2018].

Benefiting from the advances of deep neural models in nonlinear end-to-end representation learning, deep hashing has enjoyed wide attention in similarity retrieval [Wang et al., 2021]. Especially, deep supervised hashing (DSH) methods jointly learn deep representation and hash codes through given similarity/label supervision, achieving state-of-the-art retrieval performance [Wu et al., 2019]. Recent DSH methods focus primarily on refining or customizing objective functions for preserving similarity, e.g., reducing quantization loss caused by continuous relaxation [Gattupalli et al., 2019].

Figure 1: Visualization illustration using T-SNE. Figures (a) and (d) are the visualization of original images and CNN features extracted via a pre-trained AlexNet [Krizhevsky et al., 2012]. Figures (b) and (c) visualize the hash codes generated by DHN [Zhu et al., 2016] and DCH [Cao et al., 2018] respectively, while figures (e) and (f) visualize the hash codes from DHN and DCH considering code balance constraint [Wang et al., 2018] (suffixed “+CB”)

∗Corresponding authors
weighting training pairs to tackle the label imbalance issue [Cao et al., 2018] and introducing class-wise learning objective [Wang et al., 2020]. However, these methods hardly involve code balance constraints to improve hash quality, e.g., compactness and informativeness of hash codes. In fact, a large number of works have demonstrated that code balance can effectively avoid learning collapse, i.e., generating the same hash codes for all similar datapoints, and facilitate generating compact and informative hash codes [Weiss et al., 2008; Chen et al., 2019]. Figure 1 illustrates an example of random 5,000 images from the CIFAR-10 dataset. We can observe: figure (b) is overly intensive and figure (c) shows excessive dispersion and intra-cluster overlaps (fewer points in figure), and both two figures show massive inter-cluster overlaps. In contrast, figures (e) and (f) show better performance due to preserving both inter-cluster and intra-cluster discriminability, indicating that code balance promotes code compactness and informativeness.

Unfortunately, traditional code balance constraints, such as bit balance and bit uncorrelation, imposed on whole data population [Weiss et al., 2008], are unsuitable for DSH methods due to three key issues: 1) finding the zero-mean-thresholded hash functions that achieve bit balance is difficult, especially when building deep hash functions [Wang et al., 2012]; 2) obtaining the hash codes for all datapoints once is both inefficient and impractical when simultaneously learning deep data representation and hash functions; 3) traditional code balance constraints are unsuitable for the conventional batch training in deep models, since guaranteeing code balance across the entire data population hardly hold same balance conditions in each data batch. Due to the above issues, DSH methods rarely consider the benefits obtainable from code balance, which leads to low quality and representability of hash codes.

To tackle the above issues, we propose a novel probabilistic code balance constraint suitable for DSH scenarios. Specifically, we force each hash code to independently satisfy a discrete uniform distribution on $\{-1, 1\}^K$, i.e., $\text{Uni}(\{-1, 1\}^K)$. Our theoretical analysis indicates that it not only covers the traditional bit balance and bit uncorrelation constraints but maximizes the mutual information between the original data and the corresponding hash codes. The deep insights reveal that the constraint introduces random noise to uniformly scatter datapoints into hash space and to improve hash robustness and avoid overfitting. In addition, we propose a Wasserstein regularization that utilizes the Wasserstein-1 distance to measure the distance between the hash code distribution and the target discrete uniform distribution and minimize the regularization to achieve the constraint. The contributions can be summarized as follows:

- We propose a novel probabilistic code balance constraint suitable for DSH and theoretically analyze the effectiveness and insights of the constraint for the first time.
- We introduce Wasserstein regularization to achieve the constraint via Wasserstein-1 distance and propose an empirical estimate of the Wasserstein regularization.
- We conduct extensive experiments using different types of SotA DSH baselines on two benchmark datasets in terms of metrics on retrieval performance and informativeness. The results show the constraint not only enhances code compactness for promoting retrieval performance but improves the informativeness of hash codes.

2 Related Work

Supervised hashing aims to learn similarity-preserving hash functions via the given/computed similarity relation in the original space [Wang et al., 2018]. With the emergence of early supervised hashing methods, e.g., spectral hashing [Weiss et al., 2008], graph hashing [Liu et al., 2014] and kernel hashing [Liu et al., 2012], which learn hash projection vectors rather than random projections in the data-independent hash, supervised hashing has attracted an increasing amount of research interest and achieved significantly better performance than the data-independent hash [Shi et al., 2017]. To improve hash quality, the non-deep supervised hashing adopts code balance constraints – bit balance and bit uncorrelation – to avoid learning collapse and facilitate generating compact hash codes [Luo et al., 2020].

Deep supervised hashing methods, leveraging the capability of deep learning in representation learning, outperform non-deep hashing methods and has thus been widely applied [Dizaji et al., 2018; Wu et al., 2019]. Most DSH methods focus on refining or customizing objective functions, e.g., quantization loss [Zhu et al., 2016; Gattupalli et al., 2019], weighted pairwise loss [Cao et al., 2018] and triple loss [Liu et al., 2018], for preserving similarity. A couple of works, e.g., pointwise methods [Jiang et al., 2018; Su et al., 2018], further introduce label supervision to capture global position relationship, which is influenced by the quality of classification results. Recently, some works introduce class centers as proxies of classes to ensure continuous semantic similarity, which improves the discriminability of hash codes [Yuan et al., 2020; Wang et al., 2020]. Although these methods can achieve excellent performance, code balance constraints, which are beneficial to improving hash quality, are generally ignored in the DSH context. Quite a few previous methods such as [Chen et al., 2019] consider code balance to improve hash quality. However, those methods often utilize discrete optimization algorithms to solve the objective functions and act on the whole data population, which thus is inefficient under large-scale data and unsuitable for DSH. To tackle these issues, this work proposes a probabilistic code balance suitable for DSH to facilitate the learning of compact and informative hash codes.

3 Preliminary: Supervised Hashing

We first outline the general supervised hashing settings used to achieve similarity-preserving hash codes. Let $\mathcal{X} \subset \mathbb{R}^D$ and $\mathcal{Y} \subset \{-1, 1\}^K$ be the input domain and binary hash domain respectively, where $D$ and $K$ denote their respective dimensions. We have pairwise supervision of similarity information $S \in \{0, 1\}^{n \times n}$ for $n$ datapoints where $s_{ij} = 1$ if datapoints $x_i$ and $x_j$ in $\mathcal{X}$ are semantically similar and $s_{ij} = 0$ otherwise. Supervised hashing aims to learn a mapping function $H_\phi : \mathcal{X} \rightarrow \mathcal{Y}$ with parameters $\phi$ (e.g., a neural network) by minimizing the gap between the similarities $S$ in the input
domain \( \mathcal{X} \) and those calculated in the hash domain \( \mathcal{Y} \). We then introduce the two traditional code balance constraints.

**Code Balance.** To avoid severe overfitting and guarantee high-quality hash codes [Wang et al., 2016; Wang et al., 2018], bit balance and bit uncorrelation are usually considered from the information-theoretic perspective. Let \( \mathbf{X} := \{x_i \in \mathcal{X}\}_{i=1}^{n} \) and \( \mathbf{Y} := \{y_i = H_{\phi}(x_i), y_i \in \mathcal{Y}\}_{i=1}^{n} \).

- **Bit Balance:** To generate compacted hash codes, it is desirable to maximize the information contained in each hash bit. According to the maximum entropy principle, hash bits that provide balanced partitioning of \( \mathbf{X} \), i.e., \( \sum_{i=1}^{n} y_i = 0 \), have maximum information.

- **Bit Uncorrelation:** A general method of obtaining informative hash codes is to maximize the informativeness in hash codes by forcing the different hash bits to be uncorrelated (mutually orthogonal), i.e., \( \sum_{i=1}^{n} y_i y_i^T = nI \) (where \( I \) is an identity matrix of size \( n \)).

### 4 Probabilistic Code Balance Constraint

We propose a novel code balance constraint that suits DSH scenarios. More specifically, we force each generated hash code independently to satisfy a discrete uniform distribution on \((-1, 1)^K\), e.g., \( y \sim \text{Uni}(-1, 1)^K \), where we can prove that the constraint over \( \mathbf{Y} \) covers the aforementioned bit balance and bit uncorrelation constraint:

**Theorem 1** (Coverage of Bit Balance and Bit Uncorrelation). For any \( y_i \in \mathcal{Y} \), if \( y_i \sim \text{Uni}(-1, 1)^K \), it satisfies \( \mathbb{E}(\sum_{i=1}^{n} y_i) = 0 \) and \( \mathbb{E}(\sum_{i=1}^{n} y_i y_i^T) = nI \).

Theorem 1 (proved in Appendix A) indicates that \( y \sim \text{Uni}(-1, 1)^K \) can achieve both effects of the above two code balance constraints. Unlike the two constraints performing summation over the hash codes of all datapoints in a symmetrical manner, our proposed asymmetrical constraint forces the hash codes to satisfy the independent discrete uniform distribution and has no need to obtain the hash codes for all datapoints beforehand. It can effectively avoid the difficulty in achieving the zero-mean threshold in the bit balance.

**Theorem 2** (Mutual Information Maximization). Given \( \mathbf{X} \) and \( \mathbf{Y} \) where \( \mathbf{Y} \) is a deterministic function of \( \mathbf{X} \), if \( \mathcal{P}_X \) is a uniform distribution on the space of \( \mathbf{Y} \); then mutual information between \( \mathbf{X} \) and \( \mathbf{Y} \), i.e., \( I(\mathbf{X}, \mathbf{Y}) \), is the maximum.

The key insight underlying the probabilistic code balance constraint is that of maximizing the mutual information between the original data and the corresponding hash codes (as illustrated in Theorem 2 proved in Appendix B). The obtained hash codes, therefore, preserve not only the similarity relationships but also more information from the original data. Intuitively, the constraint uniformly scatters datapoints into hash space via introducing random noises, which facilitates avoiding overfitting training data and improves the generalization robustness of hash functions. In addition, we can easily find that our proposed constraint is irrelevant to the scale of data since the proposed constraint is imposed on every single bit. It is therefore consistent to impose the constraint on data batches and the whole data population, indicating that the constraint is suitable for batch training. In summary, the proposed code balance tackles the three key issues in adopting the traditional code balance constraints and is suitable for DSH. Next, we minimize the distance between the distribution of the generated hash codes and the target discrete uniform distribution to achieve the proposed constraint.

#### 4.1 Wasserstein Regularization

We propose a Wasserstein regularization to estimate the distance between the distribution of generated hash codes and the target discrete uniform distribution, thereby achieving the proposed probabilistic code balance constraint via minimizing the estimated distance. We here denote the discrete uniform distribution as \( \mathcal{P}_r : \text{Uni}([-1, 1]^K) \), and the distribution of hash codes as \( \mathcal{P}_\phi \), i.e., \( y_i \sim \mathcal{P}_\phi \) which is generated by the specific deep hash function \( H_{\phi} \). To achieve \( y_i \sim \text{Uni}([-1, 1]^K) \), we minimize the distance between the two distributions \( \mathcal{P}_r \) and \( \mathcal{P}_\phi \). Accordingly, we introduce the Wasserstein-1 distance (a.k.a. Earth-Mover distance), which is a distance function between probability distributions defined on the same metric space \( \Omega \), i.e., \( \mathcal{P}_r \) and \( \mathcal{P}_\phi \) defined on space \( \Omega = \{-1, 1\}^K \) in our case:

\[
W(\mathcal{P}_r, \mathcal{P}_\phi) = \inf_{\gamma \in \Gamma(\mathcal{P}_r, \mathcal{P}_\phi)} \mathbb{E}(y, y') \sim_{\gamma} ||y - y'||, \tag{1}
\]

where \( \Gamma(\mathcal{P}_r, \mathcal{P}_\phi) \) denotes the set of all joint distributions \( \gamma(y, y') \), the marginal distributions of which are \( \mathcal{P}_r \) and \( \mathcal{P}_\phi \) respectively. Intuitively, we can understand the definition via considering the optimal transport problem: in the given space \( \Omega \), the Wasserstein-1 distance reflects the minimal cost of transporting mass from \( \mathcal{P}_r \) to \( \mathcal{P}_\phi \) in order to transform the distribution \( \mathcal{P}_r \) to the distribution \( \mathcal{P}_\phi \). Analogously, we propose an empirical estimate of the Wasserstein-1 distance in a way that makes it unnecessary to directly estimate the distribution of hash codes. More specifically, given input data \( \mathbf{X} \in \mathbb{R}^{n \times D} \), we first randomly sample \( n \) binary target vectors denoted by \( \mathbf{A} \), the elements of which follow \( \mathcal{P}_r \):

\[
\mathbf{A} \in \{-1, 1\}^{n \times K}, \text{ s.t. } \mathbf{a} \in \{-1, 1\}^K, \mathbf{a} \sim \mathcal{P}_r. \tag{2}
\]

We then optimally pair the hash code of each datapoint with each target vector respectively such that the sum of the distances between all pairs is minimal. To obtain the optimal pairing matrix \( \mathbf{P} \in \{0, 1\}^{n \times n} \), we first define a set of constraints (denoted \( \mathcal{P} \)) for all possible pairing matrices:

\[
\mathbb{P}_n = \{\mathbf{P} \in \{0, 1\}^{n \times n}; |\mathbf{P}_{1_n}| = 1_n, |\mathbf{P}^T 1_n| = 1_n\}, \tag{3}
\]

where \( 1_n \) denotes a \( n \)-sized vector with all 1s. Given hash codes \( \mathbf{Y} \) of \( \mathbf{X} \), we then optimize the following objective:

\[
\min_{\mathbf{P} \in \mathbb{P}_n} \frac{1}{2} \|\mathbf{Y} - \mathbf{P}\mathbf{A}\|_F^2 = \min_{\mathbf{P} \in \mathbb{P}_n} -\text{tr}(\mathbf{P}\mathbf{A}\mathbf{A}^T), \tag{4}
\]

where we use a squared \( \ell_2 \) distance and \( \text{tr}(\cdot) \) denotes the trace function. Once an optimal pairing matrix \( \mathbf{P} \) is found, the above distance can be regarded as an estimate of the Wasserstein-1 distance between \( \mathcal{P}_r \) and \( \mathcal{P}_\phi \), that is:

\[
\inf_{\mathcal{P} \in \Gamma(\mathcal{P}_r, \mathcal{P}_\phi)} \mathbb{E}(y, y') \sim_{\gamma} ||y - y'|| \approx \min_{\mathbf{P} \in \mathbb{P}_n} -\text{tr}(\mathbf{P}\mathbf{A}\mathbf{A}^T). \tag{5}
\]

Accordingly, the learning objective of DSH equipped with Wasserstein Regularization (WR) contains two components:
the similarity loss used to preserve the similarity $S$ in the original space, and the Wasserstein Regularization which enhances the compactness and informativeness of hash codes:

$$J(\phi) = \min_{\phi} \ell(S, YY^T) + \beta \min_{\phi} \min_{P \in \mathcal{P}_n} -tr(PAY^T),$$

where $Y = H\phi(X)$, and $\beta > 0$ denotes a balance weight adjusting the importance of Wasserstein Regularization. The similarity loss is calculated via the loss function $\ell$ and can be specified to the similarity loss of a certain DSH method.

### 4.2 Optimization

Following the common continuous relaxation treatment [Li et al., 2017; Cao et al., 2018], we approximate the $\text{sgn}$ function with a squashing function (e.g., tanh), the output $\tilde{Y}$ of which is within $(-1, 1)$. We thus obtain a differentiable neural function $H'_\phi := X \rightarrow \tilde{Y}$ and the corresponding learning objective $J'(\phi)$ updated as below, of which the parameters can be solved using gradient-based back propagation algorithm.

$$J'(\phi) = \min_{\phi} \ell(S, \tilde{Y}\tilde{Y}^T) - \min_{\phi} \min_{P \in \mathcal{P}_n} tr(P\tilde{A}\tilde{Y}^T).$$

Apart from $A$ which is randomly sampled from the distribution $\mathcal{P}_r$, we further need to obtain the optimal pairing matrix $P$. Obtaining $P$ is a linear assignment problem, which can be solved exactly via Hungarian algorithm [Kuhn, 1955]. Under the batch gradient updates, the algorithm can efficiently perform under the restriction of one batch, significantly reducing the time complexity from $O(n^3)$ to $O(nb^2)$, where $b$ is the number of samples in one batch.

Accordingly, we apply an alternating optimization to solve Equation 7. First, when fixing $\tilde{Y}$, we sample $b$ training samples $X_b$ and calculate its corresponding intermediate matrix $\tilde{Y}_b$. We then randomly sample $b$ target vectors, denoted by $A_b$ from the discrete uniform distribution and solve $P_b \in \mathcal{P}_b$ with the Hungarian algorithm to optimally pair $\tilde{Y}_b$ and $A_b$. Finally, when fixing $A_b$ and $P_b$, we update the parameters $\phi$ via batch gradient descent. The corresponding algorithm is presented in Algorithm 1.

## 5 Experiments and Evaluation

To verify the effectiveness of the proposed probabilistic code balance in promoting retrieval performance and improving hash code quality, we select six state-of-the-art deep hashing baselines and compare the baselines with their variants equipped with WR on two public image datasets.

### 5.1 Experimental Setup

#### Datasets.
We adopt two public image datasets for evaluation. 1) CIFAR-10: the dataset consists of 60,000 $32 \times 32$ images in 10 classes, where each class has 6,000 images. Two images will be treated as a ground-truth similar pair if they share common label. 2) NUS-WIDE: it contains a total of 269,648 images. Similar to [Zhu et al., 2016; Liu et al., 2019], we use its subset of 195,834 images associated with the 21 most frequently concepts, where each concept consists of at least 5000 images, and define two images as a groundtruth similar pair if they share at least one common label.

#### Baselines.
We select different types of Sota DSH methods, including pointwise, pairwise and class-wise methods, for evaluation in our experiments: 1) DSDH [Li et al., 2017] jointly learns a linear classifier based on pointwise groundtruth labels along with the hash functions; 2) HashNet [Cao et al., 2017] tackle the data balance issue by weighting training pairs; 3) DCH [Cao et al., 2018] further introduces a Cauchy cross-entropy loss to measure pairwise similarity; 4) ADSH [Jiang and Li, 2018] directly learns hash codes for all database points asymmetrically and efficiently. 5) CSQ [Yuan et al., 2020], the latest class-wise method, proposes a global similarity metric referring to hash centers. 6) DPAH [Wang et al., 2020] introduces learnable class centers as the global proxies to capture global similarity. Note that non-deep hashing methods are not included in our experiments due to the focus on DSH. To evaluate the effectiveness of our proposed probabilistic code constraint, we construct a WR-enabled variant for each baseline, i.e., adding the Wasserstein regularization to its objective function.

#### Evaluation Protocol.
We follow the experimental settings recommended in [Zhu et al., 2016; Li et al., 2017]. In CIFAR-10 and NUS-WIDE, we randomly sample 100 images per class to form the testing set, with the remaining images used as the database, then randomly sample 100 images and 500 images per class from the database to act as the validation set and training set respectively (we adopt the whole database for training in ADSH for consistency.). To facilitate fair comparison, we adopt the same network (see Appendix C), fix batch size to 256 for all baselines, and evaluate the comparative methods over code length $K \in \{16, 32, 48, 64\}$. We tune the baselines on validation sets to find their optimal configuration. For their WR-enabled variants, we further tune the balance hyperparameter $\beta$ within the range of 0.1 to 1.0 with step 0.1 to obtain the best results. All experiments have been run five times, and the average results are reported. In the experiments, we report the Mean Average Precision (MAP) to evaluate similarity retrieval performance, while Mutual Information Neural Estimation (MINE) [Belghazi et al., 2018] is used to evaluate the informativeness of hash codes.

### 5.2 Results and Discussion

#### Retrieval Performance.
The MAP results of each baseline and its corresponding WR-enabled variant are reported in Table 1. As we can observe from the table, the WR-enabled

1 http://www.cs.toronto.edu/kriz/cifar.html

2 http://lms.comp.nus.edu.sg/research/NUS-WIDE.htm
variants outperform their corresponding baselines. More specifically, the WR-enabled variants perform much better on longer code lengths, i.e., 32 bits and 48 bits, where the MAP improvement of each WR-enabled variant over its baseline is over 1.74% and can reach 3.87%. This is intuitively attributable to the fact that the learning of the similarity-preserving objective is more easily distorted, and more vulnerable to the noise introduced by WR, as smaller code lengths. Moreover, we find that the improvement on ADSH, CSQ, and DPAH is larger than that on the others. This is attributable to the fact that WR-enabled ADSH trained on the database points improves the quality of the hash codes on these points, while the Wasserstein regularization can further improve the discriminability of the hash codes (especially for similar data points) generated from the class-wise methods, i.e., CSQ and DPAH. The above results are impressive since the Wasserstein regularization promotes the retrieval performance on different kinds of DSH methods, including point-wise (label-based), pairwise and class-based methods. In addition, we investigate the approximation quality of Wasserstein-1 distance (i.e., Equation 5) and observe that the WR-enabled variants get better performance with the increase of batch size and the best MAP when batch size $b = 512$ (see Appendix D). These results demonstrate that Wasserstein regularization introducing random noises is beneficial to improving the robustness of hash functions and avoiding overfitting training data.

**Training Complexity.** We ran the baselines and their WR-enabled variants using a single GTX-1080 GPU. The average training time costs on different training batch sizes are reported in Table 2; $\Delta$ denotes the percentage of additional training time cost required by WR-enabled variants over their corresponding baselines. The results show that the time cost for computing Wasserstein regularization increases as the batch size increases. This is because the bulk of the time cost associated with WR computation is linked with predicting hash codes for batch data, i.e., $O(bT_h)$ where $T_h$ denotes the time complexity for a single datapoint, as well as that for applying the Hungarian algorithm, i.e., $O(n^2)$. In addition, the percentage is less than 13% when the batch size $b <= 256$, indicating that the additional time cost is affordable and worthwhile considering the benefits of WR.

---

### Table 1: MAP evaluation of the six baselines and their WR-enabled variants on two public datasets.

| Method | WR | 16 bits | 32 bits | 48 bits | 64 bits | Δ | 16 bits | 32 bits | 48 bits | 64 bits | Δ |
|--------|----|---------|---------|---------|---------| |        |---------|---------|---------|---------| |
| DSDH   | /  | 0.7407  | 0.7523  | 0.7478  | 0.7486  | 2.52 | 0.7038 | 0.721 | 0.7208  | 0.7215  | 1.37 |
|        | +  | 0.7493* | 0.7737* | 0.7704* | 0.7713* |        | 0.7127* | 0.7315* | 0.7301* | 0.7321* |        |
| HashNet| /  | 0.6628  | 0.691   | 0.6903  | 0.6853  | 2.29 | 0.7017 | 0.7323 | 0.7418  | 0.7378  | 1.75 |
|        | +  | 0.6771* | 0.7045* | 0.7083* | 0.7021* |        | 0.7115* | 0.7467* | 0.7539* | 0.7526* |        |
| DCH    | /  | 0.7451  | 0.7484  | 0.7491  | 0.7253  | 1.77 | 0.7182 | 0.7555 | 0.7593  | 0.7413  | 3.08 |
|        | +  | 0.7515* | 0.7626* | 0.7587* | 0.7477* |        | 0.7291* | 0.7718* | 0.7887* | 0.7763* |        |
| ADSH   | /  | 0.6438  | 0.7612  | 0.764   | 0.7614  | 3.67 | 0.7007 | 0.7102 | 0.7182  | 0.7022  | 2.61 |
|        | +  | 0.6702* | 0.7869* | 0.7911* | 0.7893* |        | 0.7144* | 0.7273* | 0.7358* | 0.7278* |        |
| CSQ    | /  | 0.7436  | 0.7691  | -       | 0.755  | 2.05 | 0.76  | 0.7761 | -      | 0.7812  | 1.93 |
|        | +  | 0.7587* | 0.7871* | -       | 0.7685* |        | 0.7721* | 0.7896* | -      | 0.8003* |        |
| DPAH   | /  | 0.7129  | 0.7217  | 0.7347  | 0.7329  | 2.33 | 0.7567 | 0.7832 | 0.7912  | 0.7819  | 2.11 |
|        | +  | 0.7287* | 0.7381* | 0.7521* | 0.751*  |        | 0.7691* | 0.7978* | 0.8097* | 0.802*  |        |

---

### Figure 2: MAP evaluation of the WR-enabled variants with different $\beta$ on CIFAR-10 (a-c) and NUS-WIDE (d-f).

---

### Table 2: Average training time cost (seconds per epoch) over six baselines with $K = 48$ on CIFAR in terms of different batch sizes.

<table>
<thead>
<tr>
<th>WR</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>128</th>
<th>256</th>
<th>512</th>
</tr>
</thead>
<tbody>
<tr>
<td>/</td>
<td>5.9</td>
<td>4.14</td>
<td>3.23</td>
<td>2.77</td>
<td>2.57</td>
<td>2.47</td>
</tr>
<tr>
<td>+</td>
<td>6.1</td>
<td>4.29</td>
<td>3.43</td>
<td>2.95</td>
<td>2.89</td>
<td>2.96</td>
</tr>
<tr>
<td>Δ</td>
<td>3.39</td>
<td>3.62</td>
<td>6.19</td>
<td>6.5</td>
<td>12.45</td>
<td>19.84</td>
</tr>
</tbody>
</table>
Performance Varying with $\beta$. We also investigate the retrieval performance of WR-enabled variants under varying values of $\beta$ on the two datasets. Note that results on $K = 48$ are not reported due to space limitations. Comparing the figures of different bits, we find that the WR-enabled variants perform better on longer code lengths and worse on shorter code lengths, confirming the results in Table 1. In addition, the performance of most WR-enabled variants decreases as the values of $\beta$ increase from 0.3 to 0.9, especially on $K = 16$ and $K = 32$. This is reasonable because Wasserstein regularization – i.e., the probabilistic code balance – improves hash quality by introducing random noise and avoiding DSH overfitting from arising during the supervision of similarity in training data. However, when $\beta$ increases, a larger weight is allocated to Wasserstein regularization; this may result in the hash function being highly biased to noises and greatly degrade the retrieval performance.

5.3 Code Compactness and Informativeness

In this section, we evaluate the compactness and informativeness of the hash codes generated in the first experiment. We first calculate the mutual information between the original data (images) and the corresponding hash codes via MINE by a $100 - 100 - 1$ sized three-layer fully-connected neural network, as in Table 3. The results show that WR enables large amounts of mutual information between each original data and its hash codes, which indicates that the Wasserstein regularization improves the informativeness of the generated hash codes. We further visualize the hash codes via T-SNE, as shown in Figure 3. We find that WR-enabled variants achieve good performance, and even outperform their baselines, in terms of similarity preservation. In addition, when comparing WR-enabled variants with their baselines, we easily observe that hash codes generated by WR-enabled variants are scattered and exhibit less overlap, indicating the higher compactness and informativeness of the hash codes. These results demonstrate the effectiveness of the probabilistic code balance and reveal that WR, with its introduction of random noise to scatter hash codes, improves the compactness and informativeness of hash codes and provides benefits for model generalization, i.e., avoiding training data overfitting.

6 Conclusions

To improve the quality of hash codes in deep supervised hashing, we propose a novel probabilistic code balance constraint, which forces each hash bit to independently satisfy a discrete uniform distribution on $\{-1,1\}^K$ ($K$ is hash code length). We prove and analyze the effectiveness and insights of the proposed constraint, and further incorporate the Wasserstein regularization to implement the constraint. Extensive experiments by comparing six DSH baselines and their WR-enabled variants on two image benchmark datasets demonstrate the probabilistic code balance can effectively promote retrieval performance and improve the compactness and informativeness of hash codes.
References


