Rescuing Deep Hashing from Dead Bits Problem

Shu Zhao[1,2] , **Dayan Wu**[1*] , **Yucan Zhou**[1] , **Bo Li**[1,2] and **Weiping Wang**[1,2]
[1]Institute of Information Engineering, Chinese Academy of Sciences
[2]School of Cyber Security, University of Chinese Academy of Sciences
{zhaoshu, wudayan, zhouyucan, libo, wangweiping}@iie.ac.cn

Abstract

Deep hashing methods have shown great retrieval accuracy and efficiency in large-scale image retrieval. How to optimize discrete hash bits is always the focus in deep hashing methods. A common strategy in these methods is to adopt an activation function, e.g. $\mathrm{sigmoid}(\cdot)$ or $\tanh(\cdot)$, and minimize a quantization loss to approximate discrete values. However, this paradigm may make more and more hash bits stuck into the wrong saturated area of the activation functions and never escaped. We call this problem "Dead Bits Problem (DBP)". Besides, the existing quantization loss will aggravate DBP as well. In this paper, we propose a simple but effective gradient amplifier which acts before activation functions to alleviate DBP. Moreover, we devise an error-aware quantization loss to further alleviate DBP. It avoids the negative effect of quantization loss based on the similarity between two images. The proposed gradient amplifier and error-aware quantization loss are compatible with a variety of deep hashing methods. Experimental results on three datasets demonstrate the efficiency of the proposed gradient amplifier and the error-aware quantization loss.

1 Introduction

Hashing has been widely used in image retrieval [Cao *et al.*, 2017; Jiang and Li, 2018; Wu *et al.*, 2019; Zhao *et al.*, 2020; Wu *et al.*, 2018; Zhang *et al.*, 2020], video retrieval [Yuan *et al.*, 2020], and cross-modal retrieval [Jiang and Li, 2017] due to its high computation efficiency and low storage cost. Traditional hashing methods are based on hand-crafted features. The representative methods include LSH [Gionis *et al.*, 1999], ITQ [Gong *et al.*, 2012] and SDH [Shen *et al.*, 2015].

In recent years, deep hashing methods have greatly improved the retrieval performance due to their powerful representation ability. Large number of deep hashing methods have been proposed, including single-label hashing methods [Gui *et al.*, 2017; Zhao *et al.*, 2020; Li *et al.*, 2020] and multi-label hashing methods [Zhao *et al.*, 2015; Lai *et al.*, 2016; Wu *et al.*, 2017]. However, for all these deep hashing methods, the optimization is an intractable problem because of the discrete property of the binary hash codes.

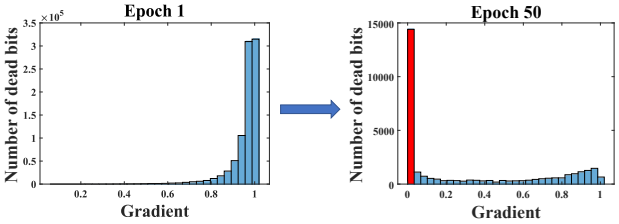

Figure 1: Existing deep hashing methods will push more and more dead bits (red bar) into the saturated area of activation functions and these bits will never escape from it during training phase, leading to the retrieval performance degradation. Best viewed in color.

Previous methods usually adopt binary approximation to tackle the above problem. A common strategy in these methods is to adopt an activation function, e.g. $\mathrm{sigmoid}(\cdot)$ or $\tanh(\cdot)$, and minimize a quantization loss to approximate the discrete values. However, this paradigm may make more and more wrong hash bits stuck into the saturated area of the activation functions, and these wrong hash bits will never escape (as shown in Figure 1). We call this problem "Dead Bits Problem (DBP)". Besides, we call the wrong bits (red bar) stuck in the saturated area as "Dead Bits", because the gradients are nearly zero. What's more, the number of dead bits will increase with the training epoch. This is because different bits have different change trends, i.e. *uncertainty* [Fu *et al.*, 2020]. As a result, the direction of optimization of each hash bit is oscillatory. Some bits may "incautiously" fall into the saturated area and hard to escape.

Moreover, the quantization loss widely used in hashing methods will also aggravate DBP, such as L_2 quantization loss [Li *et al.*, 2015] or $\log\cosh$ quantization loss [Zhu *et al.*, 2016]. These quantization loss functions drag bits into -1 or $+1$ simply according to their current sign, which may conflict with the similarity preserving requirement. For example, if the hash bits of two similar images fall into different sides of zero, the quantization loss will push them apart, while the similarity preserving loss will pull them together.

To address these issues, we propose a simple but effective deep hashing component named Gradient Amplifier which

*Corresponding author

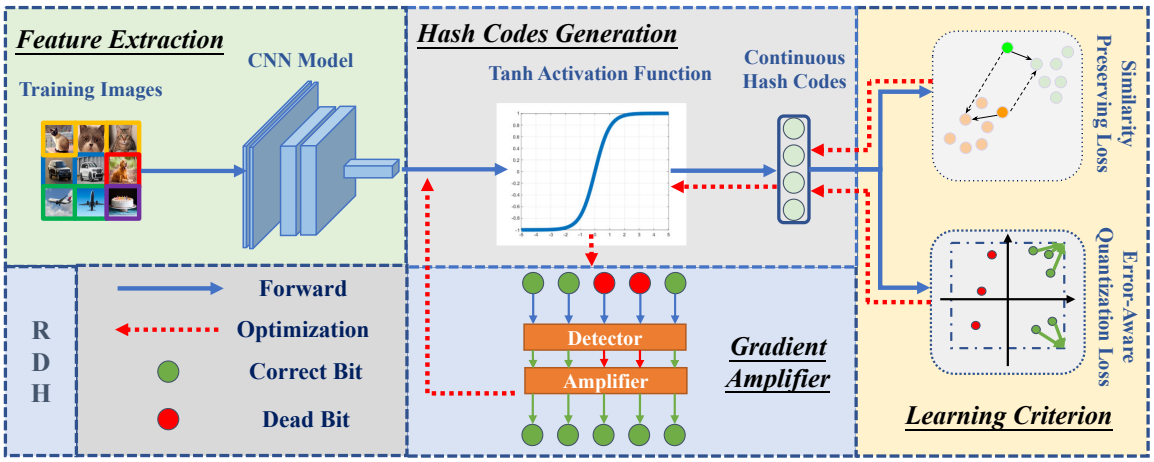

Figure 2: Framework of our proposed method. It consists of two components: a Gradient Amplifier and an Error-Aware Quantization Loss. The Gradient Amplifier aims to detect and rescue the dead bits. The Error-Aware Quantization Loss adaptively selects the bits that can be safely quantified and delegates others to the similarity preserving loss. Best viewed in color.

can detect and rescue dead bits by amplifying their gradients. Moreover, we devise an error-aware quantization loss function. For correct bits, it adaptively reduce their quantization loss to generate high-quality hash codes. For dead bits, they are propagated to the similarity preserving loss. This will avoid the negative effect brought by the original quantization loss.

The main contributions of this work can be summarized as follows:

- To the best of our knowledge, we are the first to formalize the "Dead bits Problem": the saturated area of activation function and existing quantization loss will "kill" more and more hash bits.
- We propose a gradient amplifier which detects the dead bits and amplifies their gradients to rescue them. Furthermore, we design an error-aware quantization loss, which will further alleviate the DBP. The proposed gradient amplifier and error-aware quantization loss can be compatible with a various of deep hashing methods.
- Extensive experiments on three datasets demonstrate the efficiency of the proposed gradient amplifier and the error-aware quantization loss.

2 Related Work

Learning to hash aims to project data from high-dimensional space into low-dimensional Hamming space and can be categorized into traditional hashing methods and deep hashing methods.

Traditional hashing methods leverage hand-crafted features to learn hash function. LSH [Gionis *et al.*, 1999] generates a random projection matrix to map the features into hash codes. ITQ [Gong *et al.*, 2012] uses PCA to perform dimension reduction. SDH [Shen *et al.*, 2015] adopts discrete cyclic coordinate descent (DCC) to optimize hash codes directly.

Deep hashing methods involve CNN models into the learning of hash codes. CNNH [Xia *et al.*, 2014] is a two-stage hashing method that utilizes CNN model to generate hash codes. DNNH [Lai *et al.*, 2015] is the first end-to-end deep hashing method that learns features and hash codes simultaneously. DPSH [Li *et al.*, 2015] leverages pairwise supervised information to learn hash function. And DSDH [Li *et al.*, 2020] introduces classification information into the training process and optimizes database codes by DCC [Shen *et al.*, 2015]. Recently, asymmetric architecture of deep hashing has shown great potential to improve the retrieval performance. DAPH [Shen *et al.*, 2017] adopts two different CNN model to learn a hash function simultaneously. ADSH [Jiang and Li, 2018] utilizes a CNN for query images, while the database codes are learned directly by DCC. DIHN [Wu *et al.*, 2019] aims to learn a hash function incrementally. CCDH [Zhao *et al.*, 2020] leverages variational autoencoder to update the CNN model and database codes efficiently.

For all the deep hashing methods, the optimization for binary hash codes is remain an intractable problem. To approximate the non-differentiable $\mathrm{sign}(\cdot)$ function, $\tanh(\cdot)$ and quantization loss are introduced and widely used in hashing methods [Zhu *et al.*, 2016; Cao *et al.*, 2017; Wu *et al.*, 2019; Zhao *et al.*, 2020]. However, few research focuses at the problem leading by the saturated area of activation function and the quantization loss in pairwise learning. They will push bits toward wrong direction and can not escape from the saturated area.

3 Preliminaries: Deep Hashing Models

Assume we have N training images denoted as $\mathbb{X} = \{\mathbf{x}_i\}_{i=1}^{N}$, and an $N \times N$ similarity matrix $\mathbf{S}$, where $\mathbf{S}_{ij} = 1$ if $\mathbf{x}_i$ and $\mathbf{x}_j$ have same label and otherwise $\mathbf{S}_{ij} = 0$. Deep hashing model aims to learn a hash function $\mathrm{F}(\mathbf{x}; \theta)$, mapping an input image $\mathbf{x}$ into a K-dimension embedding, where θ is the parameters of the model. Subsequently, we apply $\mathrm{sign}(\cdot)$ function to the embeddings for obtaining binary hash codes $\mathbf{B} = \{\mathbf{b}_i\}_{i=1}^{N} \in \{-1, +1\}^{N \times K}$. However, because the gradient of $\mathrm{sign}(\cdot)$ function is always 0 expects $x = 0$, we can not back-propagate it directly. To address this issue, a common strategy in these methods is to adopt $\mathrm{sigmoid}(\cdot)$ or

$\tanh(\cdot)$ function and minimize a quantization loss, i.e. L_2 quantization loss, to approximate it. Without loss of generality, we utilize the $\tanh(\cdot)$ to illustrate our approach. In the next sections, we firstly characterize a common and serious problem in these saturated functions and quantization losses, then propose a simple but effective method to solve it.

4 Dead Bits Problem

To study DBP, we firstly investigate the gradient of hash loss functions. Without loss of generality, here we choose a representative hash loss, which is formulated as:

$$\begin{aligned}\mathcal{L} = &\sum_{i=1}^{N}\sum_{j=1}^{N}\left(\log\left(1+e^{\Theta_{ij}}\right) - \mathbf{S}_{ij}\Theta_{ij}\right) \\ &+ \eta\sum_{i=1}^{N}\|\mathbf{h}_i - \mathrm{sign}(\mathbf{h}_i)\|_2^2,\end{aligned} \tag{1}$$

where $\mathbf{h}_i \in [-1,1]^K$ is the relaxed continuous codes, $\Theta_{ij} = \frac{\mathbf{h}_i^T\mathbf{h}_j}{2}$, η is the hyper-parameter to control the balance between similarity loss (first term) and quantization loss (second term). This form of hash loss is widely used in hash methods [Li *et al.*, 2015; Zhu *et al.*, 2016; Cao *et al.*, 2017; Chen *et al.*, 2019; Li *et al.*, 2020].

Next, we calculate the derivative of Eq. (1) w.r.t. $\mathrm{F}^k(\mathbf{x}_i;\theta)$, where k is the k-th dimension of embedding.

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\,\mathrm{F}^k(\mathbf{x}_i;\theta)} &= \frac{\partial\mathcal{L}}{\partial\mathbf{h}_i^k}\frac{\partial\mathbf{h}_i^k}{\partial\,\mathrm{F}^k(\mathbf{x}_i;\theta)} \\ &= \left\{\sum_{j=1}^{N}\left[\sigma(\Theta_{ij}) - \mathbf{S}_{ij}\right]\mathbf{h}_j^k + 2\eta\left[\mathbf{h}_i^k - \mathrm{sign}(\mathbf{h}_i^k)\right]\right\} \\ &\quad * \left(1-(\mathbf{h}_i^k)^2\right),\end{aligned} \tag{2}$$

where $\mathbf{h}_i^k = \tanh(\mathrm{F}^k(\mathbf{x}_i;\theta))$ is the k-th bit of $\mathbf{h}_i$, $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function. The first term in Eq. (2) is the derivate of hash loss and the second is the derivate of $\tanh(\cdot)$.

4.1 Why does DBP Appear?

For simplicity, assume we have a pair of hash codes with one bit, $\mathbf{h}_j = 0.9, \mathbf{S}_{ij} = 1, \eta = 1$, and we visualize Eq. (2) w.r.t. $\mathbf{h}_i$, which is illustrated in Figure 3. The x-axis is divided into 4 areas. In area ① (the left of green dashed line), the absolute value of gradient should be large because $\mathbf{h}_i^k$ is far away from $\mathbf{h}_j^k$ now. However, due to the second term in Eq. (2), the saturated portion of $\tanh(\cdot)$ will suppress the gradient, especially when $\mathbf{h}_i^k$ is close to -1. Consequently, these bits can not be optimized. To minimize the loss, network tends to change others bits, which may lead to training instability. In area ② (between green and yellow dashed line), the gradient seems to be decreased. This is because the quantization loss (the second term in Eq. (1)) can not leverage the supervised information and only pushes the bit to $+1$ or -1 (depends on which side of axis it is on). When the bit is close to 0 from the left side, the punishment of quantization loss will

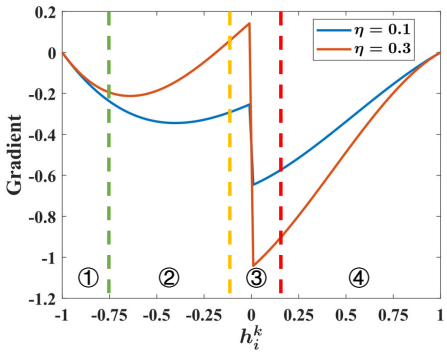

Figure 3: The visualization of hash loss's gradient. X-axis is split into 4 areas. In area ①, the saturated portion of $\tanh$ suppresses gradient; In area ②, the quantization loss is intensified and will prevent the bit from crossing zero point. If the hyper-parameter of quantization loss term is large, it may push the bit toward error direction; In area ③, there is a cliff around zero due to the change of the quantization loss's direction; In area ④, the gradient is decreasing as the distance between $\mathbf{h}_i^k$ and $\mathbf{h}_j^k$ achieves zero. Best viewed in color.

get intense and prevent it from crossing 0 point. If η is large, it may push the bit toward wrong direction, and the network can not be optimized correctly (orange line). In area ③ (between yellow and red dashed line), there is a cliff around 0. The reason is quantization loss changes the direction of punishment around zero point. If $\mathbf{h}_i^k$ approaches 0 from the left side, the quantization loss will push it toward -1. Once it crosses zero point, the quantization pushed it toward $+1$ immediately. In area ④ (the right of read dashed line), the gradient is decreasing as the distance between $\mathbf{h}_i^k$ and $\mathbf{h}_j^k$ achieves zero.

The core reason leading to DBP is the saturated area in $\tanh(\cdot)$. From Eq. (2) and Figure 3, if there is a continuous hash bit $\mathbf{h}_i^k = +1$ or -1, whatever $\mathbf{h}_j^k$ is, the gradient is always 0. Furthermore, there are two reasons that may aggravate the problem. First, in Eq. (2), the first term in braces shows the gradient of $\mathrm{F}^k(\mathbf{x}_i;\theta)$ is depended on the sum of the relations between $\mathbf{h}_i^k$ and $\mathbf{h}_j^k$. Limited by the noise in data and the capacity of CNN model, some bits may be optimized incorrectly. Once they get stuck in the saturated area, they can not escape from it. The other problem may aggravate DBP is the quantization loss. Existing quantization losses, e.g., L_2 quantization loss [Li *et al.*, 2015], $\log\cosh$ quantization loss [Zhu *et al.*, 2016], do not leverage the similarity information. They just push bit to -1 or $+1$ (depend on which side of axis it is on). If the parameter of quantization loss is selected improperly (Figure 3, orange line), it may push the bit toward wrong direction and lead to DBP. Furthermore, these dead bits will disturb the optimization process of others bits due to Eq. (2), and lead to sub-optimal retrieval performance.

5 Rescuing Deep Hashing

To address DBP, we propose Rescuing Deep Hashing (RDH), which consists of two components: Gradient Amplifier and Error-Aware Quantization Loss, as illustrated in Figure 2.

Gradient amplifier aims to amplify the gradient of dead bits

in saturated area. Its formulation is as following:

$$\mathrm{GA}(\mathbf{g}_i^k) = \begin{cases} \mathbf{g}_i^k, & |\mathbf{h}_i^k| < \tau \\ \alpha \cdot \mathbf{g_i^k}, & |\mathbf{h}_i^k| \geq \tau, \mathrm{sign}(\mathbf{h}_i^k) = \mathrm{sign}(\mathbf{g}_i^k), \end{cases} \tag{3}$$

where $\mathrm{GA}(\cdot)$ is the gradient amplifier, $\mathbf{g}_i^k$ is the gradient of $\mathbf{h}_i^k$, $\alpha = \frac{1}{1-\tau^2}$ is the amplification factor and $\tau \in [0, 1)$ is the threshold to determine whether to amplify gradient.

During the forward-propagation phase, gradient amplifier collects continuous hash codes $\mathbf{h}$. At the back-propagation stage, it fetches gradients from later layers and takes out $\mathbf{h}$. If the direction of the gradient is equal to the bit, it means the bit need to be moved toward the opposite direction. Furthermore, the bit is in the saturated area of $\tanh(\cdot)$ if $|\mathbf{h}_i^k| \geq \tau$ and it can not escape from the area. Then gradient amplifier will magnify the gradient of it and help it move toward the correct direction.

Error-aware quantization loss is to solve the problem that ordinary quantization loss [Li *et al.*, 2015; Zhu *et al.*, 2016; Li *et al.*, 2020] can not aware the correct direction of hash bits. The formulation is as following:

$$\mathcal{L}_{\mathbf{EAQ}} = \begin{cases} \|\mathbf{h}_{i/j}^k - \mathrm{sign}(\mathbf{h}_{i/j}^k)\|^2, & \text{if } (-1)^{\mathbf{S}_{ij}} \cdot \delta = -1 \\ 0, & \text{else}, \end{cases} \tag{4}$$

where $\delta = \mathrm{sign}(\mathbf{h}_i^k)\,\mathrm{sign}(\mathbf{h}_j^k)$.

Traditional quantization loss just pushes the continuous bit toward -1, if the bit is on the left of 0; $+1$ otherwise. It does not leverage the similarity information and may push the bit toward the wrong direction. Error-aware quantization loss chooses bits that satisfy 1). $\mathbf{S}_{ij} = 1$ and $\mathrm{sign}(\mathbf{h}_i^k) = \mathrm{sign}(\mathbf{h}_j^k)$; 2). $\mathbf{S}_{ij} = 0$ and $\mathrm{sign}(\mathbf{h}_i^k) \neq \mathrm{sign}(\mathbf{h}_j^k)$. For others bits, we ignore and delegate them to the similarity preserving loss to optimize them, because the number of correct and wrong bits is imbalance and dynamic during training. Optimizing them will introduce extra hyper-parameter and increase the difficulty of tuning.

Due to the flexibility of gradient amplifier and error-aware quantization loss, they can be seamlessly assembled with existing similarity preserving loss in hashing methods [Li *et al.*, 2015; Cao *et al.*, 2017; Chen *et al.*, 2019], we denote these losses as $\mathcal{L}_{\text{sim}}$. Finally, the overall objective function is

$$\min_{\theta} \mathcal{L} = \min_{\theta} (\mathcal{L}_{\text{sim}} + \eta \mathcal{L}_{\text{EQA}}). \tag{5}$$

The learning algorithm is summarized in Algorithm 1.

Algorithm 1 The learning algorithm for RDH

Input: training images $\mathbb{X}$; Similarity matrix $\mathbf{S}$; Threshold τ; Hyper-parameter η
Output: CNN model $\mathrm{F}(\mathbf{x}; \theta)$

```
Initialize a new CNN model and replace the last fully connected layer with a new K-dimension fully connected layer followed by tanh(·) activation function.
while Not convergence or not reach maximum iterations do
    Forward propagate and get continuous hash codes h.
    Store h for gradient amplification.
    Recognize bits need to be quantified according to Eq. (4).
    Calculate the loss by Eq. (5).
    Backward propagate to obtain the derivate ∂L/∂h_i^k.
    Recognize dead bits according to Eq. (3).
    Amplify gradients through Gradient Amplifier according to Eq. (3).
    Update model parameters θ.
end while
return CNN model F(x; θ)
```

6 Experiments

6.1 Datasets

The experiments of RDH are conducted on three widely used datasets: **CIFAR-10** [Krizhevsky *et al.*, 2009], **MS-COCO** [Lin *et al.*, 2014], and **NUS-WIDE** [Chua *et al.*, 2009].

- **CIFAR-10**[1] is a single-label dataset, containing 60,000 color images with 32×32 resolution, belonging to 10 classes. Each class has 6,000 images. Following [Li *et al.*, 2015; Wu *et al.*, 2019; Zhao *et al.*, 2020; Li *et al.*, 2020], we randomly sample 1,000 images (100 images per class) as the query set, and the rest are used to form the gallery set. Then we randomly select 5,000 images (500 images per class) from the gallery as the training set.
- **MS-COCO**[2] is a multi-label dataset. It contains 82,783 training images and 40,504 validation images, which belong to 80 classes. We obtain 12,2218 images by combining the training and validation images and pruning images without category annotation. Following [Cao *et al.*, 2017; Jiang and Li, 2018], we randomly sample 5,000 images as the query set, and the rest images are used as the gallery. Furthermore, we random select 10,000 images from the gallery as the training points.
- **NUS-WIDE**[3] is a multi-label dataset, which contains 269,648 images from 81 classes. Following [Lai *et al.*, 2015; Jiang and Li, 2018; Zhao *et al.*, 2020], we use a subset of original dataset which associates with the 21 most frequent classes to conduct our experiment. Specially, we randomly select 2,100 images (100 images per class) as the query set. From the rest images, we randomly choose 10,500 images (500 images per class) to make up the training set.

6.2 Evaluation Methodology

Following [Li *et al.*, 2015; Wu *et al.*, 2019; Zhao *et al.*, 2020; Li *et al.*, 2020], we adopt the Mean Average Precision (MAP) and Precision-Recall (PR) curves to evaluate the retrieval performance. Particularly, for MS-COCO and NUS-WIDE, the

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

[2] https://cocodataset.org/

[3] https://lms.comp.nus.edu.sg/wp-content/uploads/2019/research/nuswide/NUS-WIDE.html

Methods	CIFAR-10				MS-COCO				NUS-WIDE			
	24 bits	32 bits	48 bits	64 bits	24 bits	32 bits	48 bits	64 bits	24 bits	32 bits	48 bits	64 bits
DPSH	0.7465	0.7467	0.7418	0.7466	0.6770	0.6946	0.7137	0.7109	0.8075	0.8137	0.8242	0.8265
+RDH	**0.7621**	**0.7719**	**0.7748**	**0.7731**	**0.6986**	**0.7097**	**0.7189**	**0.7231**	**0.8250**	**0.8332**	**0.8387**	**0.8414**
DHN	0.7426	0.7461	0.7449	0.7387	0.6894	0.6950	0.7081	0.7132	0.8198	0.8278	0.8326	0.8351
+RDH	**0.7653**	**0.7704**	**0.7712**	**0.7651**	**0.7043**	**0.7168**	**0.7294**	**0.7354**	**0.8202**	**0.8307**	**0.8344**	**0.8370**
HashNet	0.7477	0.7568	0.7630	0.7635	0.6877	0.6985	0.7100	0.7172	0.8176	0.8209	0.8303	0.8332
+RDH	**0.7737**	**0.7804**	**0.7859**	**0.7866**	**0.7034**	**0.7167**	**0.7269**	**0.7302**	**0.8256**	**0.8311**	**0.8404**	**0.8455**
DAGH	0.7310	0.7218	0.7248	0.7269	0.6635	0.6720	0.6854	0.6899	0.8185	0.8254	0.8313	0.8354
+RDH	**0.7561**	**0.7579**	**0.7622**	**0.7559**	**0.6897**	**0.7051**	**0.7084**	**0.7131**	**0.8233**	**0.8296**	**0.8374**	**0.8392**
DSDH	0.7584	0.7611	0.7740	0.7703	0.7199	0.7474	0.7728	0.7805	0.8152	0.8206	0.8279	0.8326
+RDH	**0.7767**	**0.7886**	**0.7845**	**0.7887**	**0.7285**	**0.7539**	**0.7764**	**0.7862**	**0.8185**	**0.8297**	**0.8416**	**0.8461**

Table 1: Comparison of MAP with different bits on CIFAR-10, MS-COCO and NUS-WIDE. The best accuracy is shown in boldface.

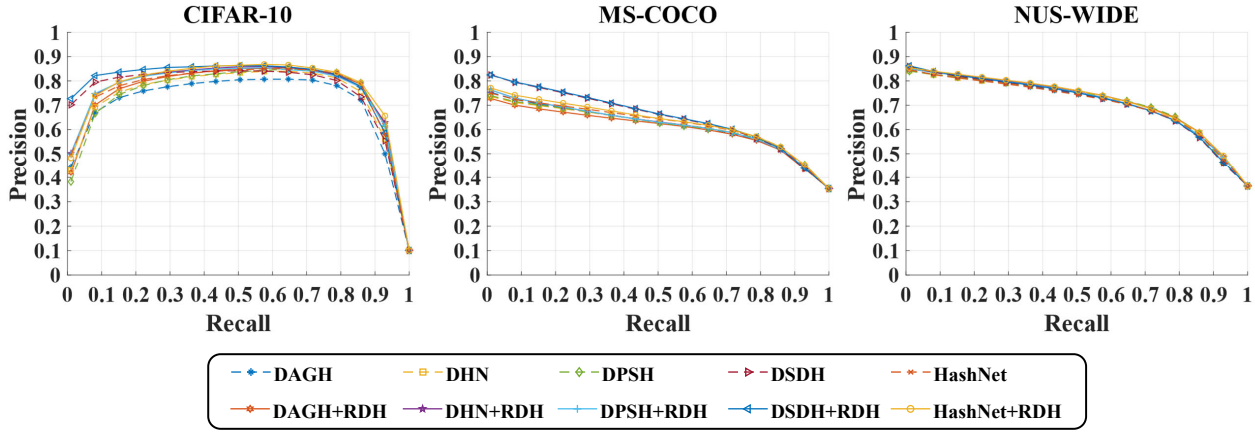

Figure 4: Precision-Recall curves with a code length of 64 on three datasets. Best viewed in color.

MAP is calculated based on the Top-5K returned samples. For multi-label dataset, e.g. MS-COCO, two images are considered to be similar if they share at least one common label. All experiments are conducted 5 times, and we report the average results of them.

6.3 Experimental Details

As a plug-in module, RDH can be adopted with various methods trained with different models and loss functions. We selected some representative deep hashing methods, e.g. DPSH [Li *et al.*, 2015], DHN [Zhu *et al.*, 2016], HashNet [Cao *et al.*, 2017], DAGH [Chen *et al.*, 2019], DSDH [Li *et al.*, 2020] as the baseline methods. All of these methods are implemented on PyTorch [Paszke *et al.*, 2019]. For a fair comparison with other state-of-the-art methods, we retrain all the baseline methods, employing AlexNet [Krizhevsky *et al.*, 2017] pretrained on ImageNet [Deng *et al.*, 2009] as the backbone. The last fully connected layer is removed, and replaced with a new one, where the dimension of the outputs is the hash code length. In RDH, SGD is utilized as the optimizer with $1e{-5}$ weight decay. The initial learning rate is set to $1e{-2}$. Cosine annealing the learning rate scheduler [Loshchilov and Hutter, 2016] is leveraged to gradually reduce learning rate to zero. The batch size is set to 128. We set $\tau = 0.99$. η is set to 1 for CIFAR-10 and 0.1 for the others.

All the experiments are conducted on a single NIVDIA RTX 2080ti GPU.

6.4 Accuracy Comparison

In this section, we compare the performance between with and without RDH on CIFAR-10, MS-COCO and NUS-WIDE. Table 1 shows the MAP of different methods with various hash code lengths. "+RDH" represents training model with the gradient amplifier and the error-aware quantization loss.

Figure 4 illustrate the Precision-Recall curves. The results indicates RDH has ability to improve the retrieval performance of the baseline methods and can be used as a plug-and-play component to fit most existing hashing methods. Compared with the baseline methods, RDH outperforms them across all datasets. For example, our proposed method improves the average MAP by 3.29% and 2.33% on CIFAR-10 and MS-COCO respectively. And we can observe the similar results on NUS-WIDE. In addition, on CIFAR-10 dataset, we observe that RDH methods with 24 bits are even better than their baseline methods with 64 bits. It indicates the benefit that dead bits are rescued from the saturated area by our

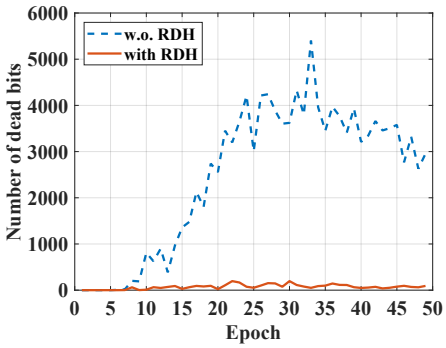

Figure 5: The number of dead bits with/without gradient amplifier and error-aware quantization loss. Best viewed in color.

proposed gradient amplifier and error-aware quantization loss have ability to rescue dead bits from the saturated area. Consequently, the retrieval performance is improved significantly.

6.5 Effectiveness of Gradient Amplifier and Error-Aware Quantization Loss

We count the dead bits with/without gradient amplifier and error-aware quantization loss during the training phase. The results are shown in Figure 5.

In Figure 5, blue line represents the results without our proposed gradient amplifier and error-aware quantization, where dead bits are increasing during the training stage and the number of them is kept at a high level. This is mainly because some bits "incautiously" arrive in the saturated area and hard to escape from it. As a result the existence of our proposed 'Dead Bits Problem" can be confirmed.

Nonetheless, with the help of gradient amplifier and error-aware quantization, DBP is mitigated significantly, which is shown by the orange line in Figure 5. Specifically, the number of dead bits is decreased significantly and stays below 200 during the training time. It means many dead bits can escape from the saturated area and have the opportunity to be optimized correctly. Consequently, only few bits get stuck in saturated area mainly due to the noise of dataset or the capacity of CNN model.

6.6 Ablation Study

To analyze the effectiveness of gradient amplifier and error-aware quantization loss, we design some variants of our proposed algorithm, and show some empirical analysis. We compare our algorithm with the following baseline: **(1) Without gradients amplifier (w.o. GA)**. The gradients amplifier is removed in this baseline. **(2) Without error-aware quantization loss (w.o. EAQL)**. We remove the error-aware quantization loss.

Table 2 shows the performance of there variants. We can observe that the retrieval performance drops dramatically when modifying the structure of proposed algorithm. Specifically, when we remove gradient amplifier, dead bits can not be pushed away from saturated area, and the MAP decreases by 2%. When removing error-aware quantization loss, some bits may hard to cross the zero point, leading to the degradation of retrieval performance.

Methods	24 bits	32 bits	48 bits	64 bits
HashNet+RDH	**0.7737**	**0.7804**	**0.7859**	**0.7866**
w.o. GA	0.7612	0.7649	0.7686	0.7601
w.o. EAQL	0.7691	0.7679	0.7793	0.7696
HashNet	0.7477	0.7573	0.7630	0.7635

Table 2: MAP results with different variants on CIFAR-10. We employ HashNet as backbone. The best accuracy is shown in boldface.

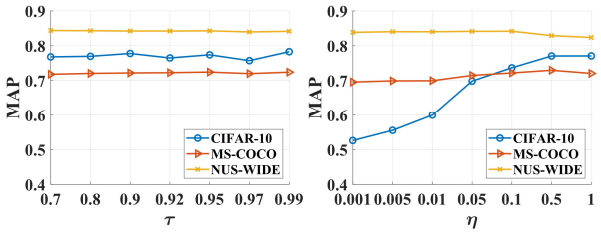

Figure 6: Parameter sensitivity on three datasets. Best viewed in color.

6.7 Sensitivity Analysis

To investigate the sensitivity of the hyper-parameters τ and η, we further conduct experiments under different values of τ and η, the retrieval performance is illustrated in Figure 6. τ is used to control the threshold of range gradients amplifier effects and η is leveraged to balance the trade-off between original loss and EAQ loss.

From Figure 6, we observe that our algorithm is not sensitive to τ. This is mainly because gradient amplifier can amplify gradients of dead bits adaptively according to Eq. (3).

For η, it is not sensitive on multi-label datasets, e.g., MS-COCO and NUS-WIDE. However, on CIFAR-10, the retrieval performance decreases when η is set to a small value. This is mainly because the resolution of images in CIFAR-10 is low and leads to high uncertainty of bits, i.e., the bits will be oscillatory during training time. Large η can help stabilize these bits and improve the retrieval performance.

7 Conclusion

In this paper, we have characterized a serious problem that caused by the saturated area in activation function, e.g. $\mathrm{sigmoid}(\cdot)$ or $\tanh(\cdot)$ and the traditional quantization loss, called DBP. To address this issue, we have proposed a gradient amplifier to detect and rescue the dead bits. Furthermore, an error-aware quantization loss is proposed to alleviate DBP. Extensive experiments have demonstrated that the proposed method can significantly decrease the number of dead bits and improve the performance of the baseline methods.

Acknowledgements

This work was supported by the National Natural Science Foundation of China under Grants 62006221, the National Key Research and Development Program of China (No. 2018YFC0825102, No.2019YFC0850202).

References

[Cao *et al.*, 2017] Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Philip S Yu. Hashnet: Deep learning to hash by continuation. In *ICCV*, pages 5608–5617, 2017.

[Chen *et al.*, 2019] Yudong Chen, Zhihui Lai, Yujuan Ding, Kaiyi Lin, and Wai Keung Wong. Deep supervised hashing with anchor graph. In *ICCV*, pages 9796–9804, 2019.

[Chua *et al.*, 2009] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. Nus-wide: a real-world web image database from national university of singapore. In *CIVR*, pages 1–9, 2009.

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

[Fu *et al.*, 2020] Chaoyou Fu, Guoli Wang, Xiang Wu, Qian Zhang, and Ran He. Deep momentum uncertainty hashing. *arXiv preprint arXiv:2009.08012*, 2020.

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

[Gong *et al.*, 2012] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *TPAMI*, 35(12):2916–2929, 2012.

[Gui *et al.*, 2017] Jie Gui, Tongliang Liu, Zhenan Sun, Dacheng Tao, and Tieniu Tan. Fast supervised discrete hashing. *TPAMI*, 40(2):490–496, 2017.

[Jiang and Li, 2017] Qing-Yuan Jiang and Wu-Jun Li. Deep cross-modal hashing. In *CVPR*, pages 3232–3240, 2017.

[Jiang and Li, 2018] Qing-Yuan Jiang and Wu-Jun Li. Asymmetric deep supervised hashing. In *AAAI*, pages 3342–3349, 2018.

[Krizhevsky *et al.*, 2009] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Technical report*, 2009.

[Krizhevsky *et al.*, 2017] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[Lai *et al.*, 2015] Hanjiang Lai, Yan Pan, Ye Liu, and Shuicheng Yan. Simultaneous feature learning and hash coding with deep neural networks. In *CVPR*, pages 3270–3278, 2015.

[Lai *et al.*, 2016] Hanjiang Lai, Pan Yan, Xiangbo Shu, Yunchao Wei, and Shuicheng Yan. Instance-aware hashing for multi-label image retrieval. *TIP*, 25(6):2469–2479, 2016.

[Li *et al.*, 2015] Wu-Jun Li, Sheng Wang, and Wang-Cheng Kang. Feature learning based deep supervised hashing with pairwise labels. *arXiv preprint arXiv:1511.03855*, 2015.

[Li *et al.*, 2020] Qi Li, Zhenan Sun, Ran He, and Tieniu Tan. A general framework for deep supervised discrete hashing. *IJCV*, 128(8):2204–2222, 2020.

[Lin *et al.*, 2014] 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 *ECCV*, pages 740–755. Springer, 2014.

[Loshchilov and Hutter, 2016] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[Paszke *et al.*, 2019] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, pages 8026–8037, 2019.

[Shen *et al.*, 2015] Fumin Shen, Chunhua Shen, Wei Liu, and Heng Tao Shen. Supervised discrete hashing. In *CVPR*, pages 37–45, 2015.

[Shen *et al.*, 2017] Fumin Shen, Xin Gao, Li Liu, Yang Yang, and Heng Tao Shen. Deep asymmetric pairwise hashing. In *ACM MM*, pages 1522–1530, 2017.

[Wu *et al.*, 2017] Dayan Wu, Zheng Lin, Bo Li, Mingzhen Ye, and Weiping Wang. Deep supervised hashing for multi-label and large-scale image retrieval. In *ICMR*, page 150–158, 2017.

[Wu *et al.*, 2018] Dayan Wu, Jing Liu, Bo Li, and Weiping Wang. Deep index-compatible hashing for fast image retrieval. In *ICME*, pages 1–6, 2018.

[Wu *et al.*, 2019] Dayan Wu, Qi Dai, Jing Liu, Bo Li, and Weiping Wang. Deep incremental hashing network for efficient image retrieval. In *CVPR*, pages 9069–9077, 2019.

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

[Yuan *et al.*, 2020] 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 *CVPR*, pages 3083–3092, 2020.

[Zhang *et al.*, 2020] Wanqian Zhang, Dayan Wu, Yu Zhou, Bo Li, Weiping Wang, and Dan Meng. Deep unsupervised hybrid-similarity hadamard hashing. In *ACM MM*, page 3274–3282, 2020.

[Zhao *et al.*, 2015] Fang Zhao, Yongzhen Huang, Liang Wang, and Tieniu Tan. Deep semantic ranking based hashing for multi-label image retrieval. In *CVPR*, pages 1556–1564, 2015.

[Zhao *et al.*, 2020] Shu Zhao, Dayan Wu, Wanqian Zhang, Yu Zhou, Bo Li, and Weiping Wang. Asymmetric deep hashing for efficient hash code compression. In *ACM MM*, pages 763–771, 2020.

[Zhu *et al.*, 2016] Han Zhu, Mingsheng Long, Jianmin Wang, and Yue Cao. Deep hashing network for efficient similarity retrieval. In *AAAI*, pages 2415–2421, 2016.