EFFICIENT EVALUATION OF ADVERSARIAL ROBUSTNESS FOR DEEP HASHING BASED RETRIEVAL

Anonymous authors
Paper under double-blind review

ABSTRACT

Deep hashing has been extensively applied to massive image retrieval due to its efficiency and effectiveness. Recently, several adversarial attacks have been presented to reveal the vulnerability of deep hashing models against adversarial examples. However, existing attack methods suffer in degraded performance or inefficiency because they underutilize the semantic relations between original samples or spend a lot of time learning from these samples. In this paper, we propose a novel Pharos-guided Attack, dubbed **PgA**, to evaluate the adversarial robustness of deep hashing networks efficiently. Specifically, we design *pharos code* to represent the semantics of the benign image, which preserves the similarity with semantically related samples and dissimilarity with irrelevant examples. It is proven that we can quickly calculate the pharos code via a simple math formula rather than time-consuming iterative procedures. Thus, PgA can directly conduct a reliable and efficient attack on deep hashing-based retrieval by maximizing the similarity between the hash code of the adversarial example and the pharos code. Extensive experiments on the benchmark datasets verify that the proposed algorithm outperforms the prior state-of-the-arts in both attack strength and speed.

1 INTRODUCTION

It is challenging to rapidly and effectively search for the required information from vast collections in the current era of big data. Learning to hash (hashing) (Wang et al., 2018) has attracted much attention in large-scale image retrieval due to its exceptional benefits in efficient XOR operation and low storage cost by mapping high-dimensional data to compact binary codes. Particularly, deep hashing (Xia et al., 2014; Li et al., 2016; Cao et al., 2017) that learns nonlinear hash functions with deep neural networks (DNNs) has become a predominant image search technique since it delivers better retrieval accuracy than conventional hashing.

Recent works (Yang et al., 2020; Bai et al., 2020; Wang et al., 2021b;a; Zhang et al., 2021; Xiao & Wang, 2021; Lu et al., 2021) have revealed that deep hashing models are susceptible to adversarial examples. Although these imperceptible samples are crafted by adding small perturbations to original samples, they are sufficient to deceive models into making inaccurate predictions. There is no doubt that such malicious attacks pose grave security threats to image retrieval systems based on deep hashing. In a deep hashing-based face recognition system, for instance, adversarial examples can mislead the system into matching the faces of specific individuals in the database, infiltrating the system effectively. Consequently, there is significant demand for research into these security concerns in deep hashing-based retrieval.

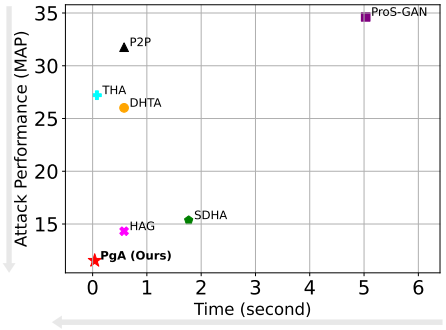


Figure 1: Retrieval performance (MAP) of deep hashing models against adversarial attacks. The horizontal axis indicates the average time (seconds) for constructing per adversarial sample. Our method is closest to the origin of coordinates, yielding the highest efficiency.

A Few studies (Yang et al., 2020; Bai et al., 2020; Wang et al., 2021b;a; Lu et al., 2021) have been con-

ducted on adversarial attacks and adversarial defenses in deep hashing-based retrieval at present. Existing attack techniques have been shown to be effective, but they are neither efficient nor reliable in evaluating the adversarial robustness (*i.e.*, defense performance) of deep hashing networks. Firstly, these methods (Yang et al., 2020; Bai et al., 2020; Lu et al., 2021) suffer in limited attack performances because they do not fully leverage the semantic relevance between available samples. For instance, SDHA (Lu et al., 2021) only reduces the similarity of adversarial samples with their semantically relevant images, ignoring more irrelevant ones. Secondly, though some hashing attack methods simultaneously consider similar and dissimilar pairs, they use time-consuming neural networks to learn discriminative semantic representations from these pairs for the precise semantic attack, *e.g.*, ProS-GAN (Wang et al., 2021b) and THA (Wang et al., 2021a). In this paper, we focus on improving the deficiencies of previous hashing attacks in both effectiveness and efficiency, as shown in Fig. 1. Furthermore, strong adversarial attack methods with high efficiency can provide excellent benchmarks of model robustness and facilitate the development of adversarial defense strategies in deep hashing-based retrieval.

In this study, we propose Pharos-guided Attack (PgA) for efficient adversarial robustness evaluation of deep hashing networks. The core idea is to quickly compute the pharos code, which reflects the semantics of the original image, and then to use the pharos code to direct the generation of the potent adversarial sample. Specifically, we first design an optimal hash code (namely *pharos code*) as the discriminative representative of the benign image semantics, which maintains both the similarities to semantically relevant samples and the dissimilarities to irrelevant samples. Benefiting from the binary property of hash codes, we prove that the proposed *Pharos Generation Method* (PGM) can directly calculate the pharos code through a simple mathematical formula (refer to Appendix A.1 for proof). Thus, the pharos codes of the input data are calculated immediately before the adversarial attack. Subsequently, based on the pharos code, it is feasible to carry out an efficient adversarial hashing attack by maximizing the Hamming distance between the hash code of the adversarial example and the pharos code. Due to the excellence of the pharos codes, our attack manner can considerably enhance the efficiency and effectiveness of adversarial robustness verification, as shown in Fig. 1. In summary, our main contributions are as follows:

- We create the pharos code as the precise semantic representative of the original image content to aid in the construction of the adversarial attack framework for deep hashing-based retrieval. It should be emphasized that our proven mathematical formula in PGM can generate the pharos code instantly.
- A simple pharos-guided attack algorithm is provided, *i.e.*, PgA, which is an efficient and reliable method to evaluate the adversarial robustness of deep hashing networks.
- Extensive experiments demonstrate that PgA can be applied to deep hashing frameworks and achieves state-of-the-art attack performance with high efficiency.

2 Related Work

In this section, we briefly review most relevant works in deep hashing-based image retrieval, adversarial attacks and adversarial training for defense.

Deep Hashing based Image Retrieval. With the remarkable success of deep learning on many vision tasks, deep hashing methods have been well developed for large-scale image retrieval, yielding superior performances than the traditional hashing methods based on hand-crafted features. The pioneering CNNH (Xia et al., 2014) adopts a two-stage strategy, *i.e.*, hash code generation of training data and hash function construction with DNN. Recently, deep hashing methods (Lai et al., 2015; Zhu et al., 2016; Li et al., 2016; Liu et al., 2016; Li et al., 2017; Cao et al., 2017; Jiang & Li, 2018; Cao et al., 2018; Su et al., 2018; Wang et al., 2021c; Doan et al., 2022) focus on joint feature learning and hash code encoding into an end-to-end DNN for the better quality of hash codes. A notable work is DPSH (Li et al., 2016), which simultaneously learns the visual features of data points and preserves their semantic similarity with a pairwise-similarity loss. To alleviate data imbalance between positive and negative pairs, HashNet (Cao et al., 2017) adopts a weighted strategy in pairwise loss functions. Different from the pairwise similarity learning, CSQ (Yuan et al., 2020) can generate high-quality hash codes by enforcing them close to pre-defined hash centers.

Adversarial Attack. In image classification, numerous *adversarial attack* methods (Szegedy et al., 2014; Goodfellow et al., 2015; Kurakin et al., 2017; Moosavi-Dezfooli et al., 2016; Madry et al., 2017; Carlini & Wagner, 2017; Dong et al., 2018; Papernot et al., 2017; Chen et al., 2017; Ilyas et al., 2018) have been developed to fool the well-trained classifiers by constructing adversarial examples, since the intriguing properties (Szegedy et al., 2014; Biggio et al., 2013) of adversarial samples are discovered. For example, FGSM (Goodfellow et al., 2015) crafts adversarial samples by maximizing the loss along the gradient direction with a large step. As the multi-step variant of FGSM, I-FGSM (Kurakin et al., 2017) and PGD (Madry et al., 2017) iteratively update perturbations with small steps for better attack performance.

Recently, researchers have extended adversarial attacks to deep hashing-based image retrieval (Yang et al., 2020; Bai et al., 2020; Wang et al., 2021b;a; Lu et al., 2021; Zhang et al., 2021). Existing adversarial attack methods for deep hashing can be organized into two categories: *non-targeted attack* and *targeted attack*. For a non-targeted attack in hashing-based retrieval, its goal is to generate adversarial examples that can confuse the hashing model to retrieve results irrelevant to the original image (Bai et al., 2020; Wang et al., 2021b). Achieving the non-targeted attack by minimizing the hash code similarity between the adversarial example and the original sample, Yang *et al.* (Yang et al., 2020) proposed **HAG**, the first adversarial attack method on deep hashing. **SDHA** (Lu et al., 2021) generates more effective adversarial queries due to staying away from the relevant images of the benign sample, while HAG only takes the original image into consideration. As for the targeted attack, it aims to construct adversarial examples whose retrieved images are semantically relevant to the given target label (Bai et al., 2020; Wang et al., 2021b). To achieve the targeted attack, **P2P** and **DHTA** (Bai et al., 2020) obtain the anchor code as the representative of the target label to direct the generation of the adversarial sample. Subsequently, Wang *et al.* (Wang et al., 2021a) defined the prototype code as the target code to reach a better targeted attack, which is called **THA** in this paper. **ProS-GAN** (Wang et al., 2021b) designs a generative framework for efficient targeted hashing attack under the test phase. Different from the above white-box scenarios, Xiao *et al.* (Xiao & Wang, 2021) proposed the targeted black-box attack NAG by enhancing the transferability of adversarial examples.

Unlike the prior work (Bai et al., 2020) where the anchor code is obtained by a few instances with the same semantics, we proposed the pharos code, which preserves the semantic similarity with relevant samples and dissimilarity with irrelevant samples. Moreover, we use the proven mathematical formula (*i.e.*, PGM) to instantly calculate the pharos code before the adversarial attack, instead of learning the prototype codes (Wang et al., 2021b;a) through time-consuming neural networks as ProS-GAN and THA do. Hence, our pharos code is more suited for efficient adversarial robustness evaluation of deep hashing models.

Adversarial Training. *Adversarial training* (Goodfellow et al., 2015; Madry et al., 2017) aims to augment the training data with generated adversarial examples, which is the most robust training strategy against various adversarial attacks. Thus, modifications (Zhang et al., 2019; Wong et al., 2020; Pang et al., 2021) and applications (Li et al., 2021; Utrera et al., 2021) of adversarial training have emerged to improve the robustness and generalization of DNNs. For deep hashing-based retrieval, (Wang et al., 2021a) proposed the first effective adversarial training algorithm based on the targeted attack (dubbed **ATRDH** here) by narrowing the semantic gap between the adversarial samples and the original samples in the Hamming space.

3 METHOD

3.1 PRELIMINARIES

We consider that an attacked hashing model F learns from a training set of N data points $O = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N}$, where $\boldsymbol{x}_i$ indicates i-th image, and $\boldsymbol{y}_i = [y_{i1}, y_{i2}, ..., y_{iC}] \in \{0, 1\}^C$ denotes a label vector of $\boldsymbol{x}_i$. C indicates the total number of classes in the dataset. $y_{ij} = 1$ means that $\boldsymbol{x}_i$ belongs to the j-th class. If $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ share at least one common label, they are semantically similar, *i.e.*, $\boldsymbol{x}_j$ is the positive sample of $\boldsymbol{x}_i$. Otherwise, they are semantically dissimilar and $\boldsymbol{x}_j$ is the negative sample of $\boldsymbol{x}_i$.

Deep hashing aims at employing DNNs to transform high dimensional data into compact binary codes and simultaneously preserves their semantic similarities. For the given hashing model F, the

hash code $\boldsymbol{b}_i$ of the instance $\boldsymbol{x}_i$ is generated as:

$$\boldsymbol{b}_i = F(\boldsymbol{x}_i) = \operatorname{sign}(\boldsymbol{h}_i) = \operatorname{sign}(f_\theta(\boldsymbol{x}_i)), \quad \text{s.t. } \boldsymbol{b}_i \in \{-1, 1\}^K \quad , \tag{1}$$

where K represents the hash code length, and $f(\cdot)$ with parameter θ is a DNN to approximate hash function $F(\cdot)$. The final binary code $\boldsymbol{b}_i$ is obtained by applying the $\operatorname{sign}(\cdot)$ on the output $\boldsymbol{h}_i$ of $f_\theta(\boldsymbol{x}_i)$. Typically, $f(\cdot)$ is implemented by a convolutional neural network (CNN) and adopts the $\tanh$ activation to simulate sign function at the output layer.

3.2 The Proposed Pharos-guided Attack

3.2.1 Problem Formulation

In hashing based retrieval, the goal of adversarial attack (*i.e.*, non-targeted attack) is to craft an adversarial example whose retrieval results are irrelevant to the original sample contents. For credibility, this objective can be achieved by maximizing the hash code distance between the adversarial example and its semantically relevant samples, and simultaneously minimizing the distance from irrelevant samples, rather than the only benign sample. Thus, for a given clean image $\boldsymbol{x}$, the objective of its adversarial example $\boldsymbol{x}'$ is formulated as follows:

$$\max_{\boldsymbol{x}'} \sum_{i}^{N_\mathrm{p}} \sum_{j}^{N_\mathrm{n}} [w_i D(F(\boldsymbol{x}'), F(\boldsymbol{x}_i^{(\mathrm{p})})) - w_j D(F(\boldsymbol{x}'), F(\boldsymbol{x}_j^{(\mathrm{n})}))], \quad \text{s.t. } \|\boldsymbol{x} - \boldsymbol{x}'\|_p \leq \epsilon, \tag{2}$$

where $F(\cdot)$ is the hashing function approximated by the deep model $f(\cdot)$, and $D(\cdot, \cdot)$ is a distance metric. w_i and w_j represent distance weights. $\boldsymbol{x}_i^{(\mathrm{p})}$ is a positive sample semantically related to the original sample $\boldsymbol{x}$, and $\boldsymbol{x}_i^{(\mathrm{n})}$ is a negative sample of $\boldsymbol{x}$. Because this maximizing term of Eq. (2) can push the hash code of the adversarial example close to those of unrelated samples and away from semantically relevant samples, optimal attack strength would come true in theory. N_p and N_n are the number of the positive samples and the negative samples, respectively. $\|\cdot\|_p$ ($p = 1, 2, \infty$) is L_p norm which keeps the pixel difference between the adversarial sample and the original sample no more than ϵ for the imperceptible property of adversarial perturbations.

3.2.2 Generation of pharos codes.

Actually, the maximized objective in Eq. (2) is equivalent to finding a hash code $\boldsymbol{b}'$, which satisfies:

$$\max_{\boldsymbol{b}'} \sum_{i} \sum_{j} [w_i D_\mathrm{H}(\boldsymbol{b}', \boldsymbol{b}_i^{(\mathrm{p})}) - w_j D_\mathrm{H}(\boldsymbol{b}', \boldsymbol{b}_j^{(\mathrm{n})})], \tag{3}$$

where D_H is Hamming distance measure. $\boldsymbol{b}_i^{(\mathrm{p})}$ is the hash code of the positive sample $\boldsymbol{x}_i^{(\mathrm{p})}$, and $\boldsymbol{b}_j^{(\mathrm{n})}$ is the binary code of the negative sample $\boldsymbol{x}_j^{(\mathrm{n})}$. Subsequently, we can optimize the adversarial example by minimizing the distance from $\boldsymbol{b}'$, *i.e.*,

$$\min_{\boldsymbol{x}'} D_\mathrm{H}(\boldsymbol{b}', F(\boldsymbol{x}')). \tag{4}$$

0 For any hash code $\hat{\boldsymbol{b}}$ and $\check{\boldsymbol{b}}$, we know that $D_\mathrm{H}(\hat{\boldsymbol{b}}, \check{\boldsymbol{b}}) = \frac{1}{2}(K - \hat{\boldsymbol{b}}^\top \check{\boldsymbol{b}})$. Accordingly, we deduce that $D_\mathrm{H}(\hat{\boldsymbol{b}}, \check{\boldsymbol{b}}) = K - D_\mathrm{H}(-\hat{\boldsymbol{b}}, \check{\boldsymbol{b}})$. Let hash code $\boldsymbol{b}^\star = -\boldsymbol{b}'$, the Eq. (3) and (4) can be reformulated as follows:

$$\begin{gathered} \min_{\boldsymbol{b}^\star} \sum_{i} \sum_{j} \{w_i [D_\mathrm{H}(\boldsymbol{b}^\star, \boldsymbol{b}_i^{(\mathrm{p})}) - K] - w_j [D_\mathrm{H}(\boldsymbol{b}^\star, \boldsymbol{b}_j^{(\mathrm{n})}) - K]\}, \\ \max_{\boldsymbol{x}'} D_\mathrm{H}(\boldsymbol{b}^\star, F(\boldsymbol{x}')) - K. \end{gathered} \tag{5}$$

Removing the constants, the Eq. (5) can be written as follows:

$$\begin{gathered} \min_{\boldsymbol{b}^\star} \sum_{i} \sum_{j} [w_i D_\mathrm{H}(\boldsymbol{b}^\star, \boldsymbol{b}_i^{(\mathrm{p})}) - w_j D_\mathrm{H}(\boldsymbol{b}^\star, \boldsymbol{b}_j^{(\mathrm{n})})], \\ \max_{\boldsymbol{x}'} D_\mathrm{H}(\boldsymbol{b}^\star, F(\boldsymbol{x}')). \end{gathered} \tag{6}$$

Due to the binary characteristic of the hash code, we can directly calculate the optimal code (named *pharos code* $\boldsymbol{b}^\star$) in the problem (6) by the following Pharos Generation Method (PGM), *i.e.*,

$$\boldsymbol{b}^\star = \operatorname{sign}(\sum_{i}^{N_\mathrm{p}} \sum_{j}^{N_\mathrm{n}} (w_i \boldsymbol{b}_i^{(\mathrm{p})} - w_j \boldsymbol{b}_j^{(\mathrm{n})})), \tag{7}$$

where $\operatorname{sign}(\cdot)$ is the sign function. The proof of PGM is shown in the **Appendix** A.1. In addition, we define the w_i and w_j as follows:

$$w_i = s_i, \quad w_j = 1 - s_j, \tag{8}$$

where $s_{i/j}$ ($s_{i/j} \in [0, 1]$) denotes the similarity between the adversarial example and the i/j-th benign sample. If labels $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$ of $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are given, we can calculate $s_{i/j}$ by Dice coefficient, *i.e.*, $s_{i/j} = \frac{2|\boldsymbol{y} \cap \boldsymbol{y}_{i/j}|}{|\boldsymbol{y}| + |\boldsymbol{y}_{i/j}|}$. Otherwise, $s_{i/j}$ is usually determined by the optimization objective of the attacked hashing model. For instance, $s_i = 1$ and $s_j = 0$ are widely adopted in learning to hash.

3.2.3 Generating Adversarial Examples.

Since the pharos code is found, the attack problem described in Eq. (2) can be translated into the following objective under the L_∞ constraint:

$$\max_{\boldsymbol{x}'} D_\mathrm{H}(\boldsymbol{b}^\star, F(\boldsymbol{x}')), \quad \text{s.t. } \|\boldsymbol{x} - \boldsymbol{x}'\|_\infty \leq \epsilon \quad . \tag{9}$$

According to $D_\mathrm{H}(\hat{\boldsymbol{b}}, \check{\boldsymbol{b}}) = \frac{1}{2}(K - \hat{\boldsymbol{b}}^\top \check{\boldsymbol{b}})$, the Eq. (9) is equivalent to:

$$\max_{\boldsymbol{x}'} \mathcal{L}_a = -\frac{1}{K}(\boldsymbol{b}^\star)^\top f_\theta(\boldsymbol{x}'), \quad \text{s.t. } \|\boldsymbol{x} - \boldsymbol{x}'\|_\infty \leq \epsilon \quad . \tag{10}$$

However, Eq. (10) focuses on the sum similarity between $f_\theta(\boldsymbol{x}')$ and $\boldsymbol{b}^\star$, and can not effectively promote that each bit in $f_\theta(\boldsymbol{x}')$ differs in sign from $\boldsymbol{b}^\star$. Intuitively, bits with small differences between $f_\theta(\boldsymbol{x}')$ and $\boldsymbol{b}^\star$ should be given big weights. Hence, we add an weighting vector $\boldsymbol{\omega}$ on $\mathcal{L}_a$ to enforce each bit of $f_\theta(\boldsymbol{x}')$ away from those bits of $\boldsymbol{b}^\star$. Formally,

$$\mathcal{L}_a = -\frac{1}{K}\boldsymbol{\omega}^\top \boldsymbol{u}, \quad \boldsymbol{u} = \boldsymbol{b}^\star \circ f_\theta(\boldsymbol{x}'), \tag{11}$$

where $\circ$ represents Hadamard product, and $\boldsymbol{\omega}$ has the same dimensions as $\boldsymbol{b}^\star$. For efficiency, it is not necessary to enforce each element u_k of $\boldsymbol{u}$ to approximate -1. In this case, we set u_k to converge at t ($-1 < t < 0$). Thus, the component $\boldsymbol{\omega}_k$ of $\boldsymbol{\omega}$ is defined as

$$\omega_k = \begin{cases} u_k - 2t, & u_k > t \\ -t^2, & \text{otherwise} \end{cases} \tag{12}$$

where u_k is the k-th element of $\boldsymbol{u}$. As shown in Figure 2, t controls the margin between $b_k^\star$ and $f_\theta(\boldsymbol{x}')_k$. t is set to -0.8 by default. Furthermore, we follow (Yang et al., 2020) to make our pharos-guided attack focus on different sign between $f(\boldsymbol{x}')$ and $\boldsymbol{b}^\star$ for efficiency, *i.e.*,

$$\mathcal{L}_a = -\frac{1}{\pi}[\boldsymbol{m} \circ \boldsymbol{\omega}]^\top \boldsymbol{u}, \quad \boldsymbol{u} = \boldsymbol{b}^\star \circ f_\theta(\boldsymbol{x}') \tag{13}$$

where $\boldsymbol{m} \in \{0, 1\}^K$ is a mask and π is the number of non-zero elements in $\boldsymbol{m}$. The element m_k of $\boldsymbol{m}$ is defined as

$$m_k = \begin{cases} 1, & u_k > t \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

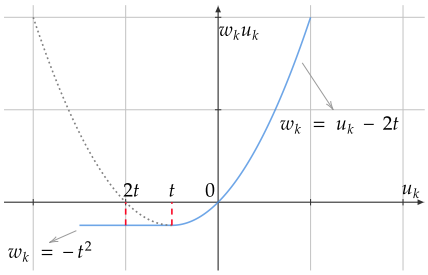


Figure 2: The curve of ω_k.

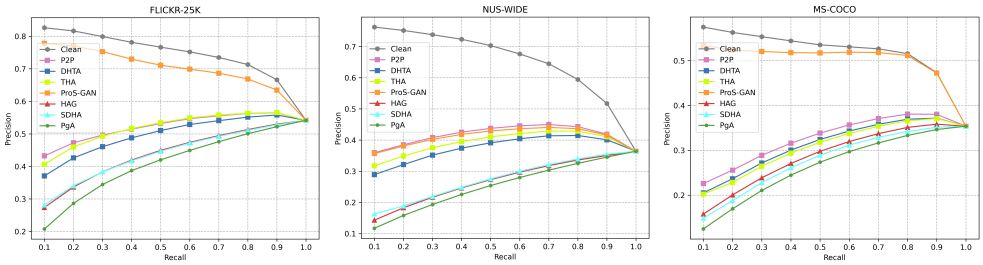
Notably, the pharos-guided attack with Eq. (10) is called **PgA**†, and that with Eq. (13) is the default **PgA**. Unlike HAG and SDHA using SGD (Robbins & Monro, 1951) or Adam (Kingma & Ba, 2015) optimizer (Yang et al., 2020; Lu et al., 2021) with quantities of iterations, this paper adopts PGD (Madry et al., 2017) to optimize $\boldsymbol{x}'$ with T ($T = 100$ by default) iterations for efficiency, *i.e.*,

$$\boldsymbol{x}'_T = \mathcal{S}_\epsilon(\boldsymbol{x}'_{T-1} + \eta \cdot \operatorname{sign}(\Delta_{\boldsymbol{x}'_{T-1}} \mathcal{L}_a)), \quad \boldsymbol{x}'_0 = \boldsymbol{x} + \boldsymbol{r}, \tag{15}$$

where η is the step size, and $\mathcal{S}_\epsilon$ project $\boldsymbol{x}'$ into the ϵ-ball (Madry et al., 2017) of $\boldsymbol{x}$. $\boldsymbol{r}$ is random noise, sampled from uniform $U(-\epsilon, \epsilon)$.

Table 1: MAP (%) of different attack methods on deep hashing models without defense.

Method	FLICKR-25K			NUS-WIDE			MS-COCO		
	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits
Clean	80.39	81.35	91.91	74.99	76.72	77.84	56.53	57.93	57.92
P2P	41.68	42.69	41.18	32.56	31.76	31.74	21.99	21.78	21.57
DHTA	34.74	32.63	34.58	26.68	26.01	26.33	19.41	19.23	18.05
ProS-GAN	67.29	76.31	82.51	30.50	34.63	64.43	52.12	53.67	53.38
THA	38.28	37.54	35.67	30.72	27.21	24.27	24.62	20.65	22.71
HAG	24.75	24.37	23.44	14.55	14.32	14.07	13.91	13.59	14.62
SDHA	20.50	19.63	18.98	17.09	15.36	14.87	12.07	12.36	12.94
PgA† (Ours)	16.03	15.49	15.32	12.30	11.93	12.07	11.36	10.56	11.10
PgA (Ours)	**15.70**	**15.18**	**15.02**	**11.81**	**11.53**	**11.69**	**9.91**	**9.41**	**10.26**



(a) FLICKR-25K (b) NUS-WIDE (c) MS-COCO

Figure 3: Precision-Recall curves on original deep hashing models with 32 bits code length.

4 Experiments

4.1 Experimental Setup

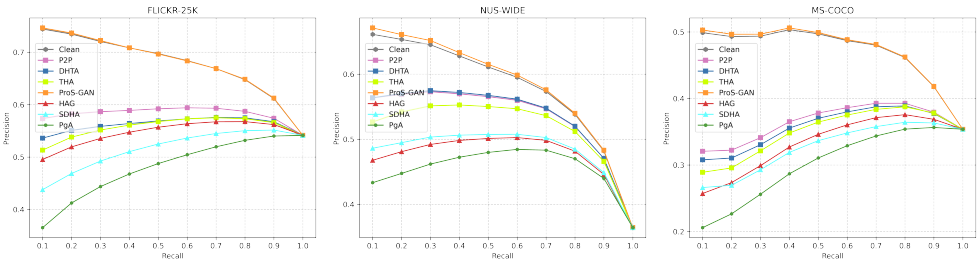
Datasets. We adopt three popular datasets used in hashing based retrieval to evaluate our defense method in extensive experiments: *FLICKR-25K* (Huiskes & Lew, 2008), *NUS-WIDE* (Chua et al., 2009) and *MS-COCO* (Lin et al., 2014). The **FLICKR-25K** dataset comprises 25,000 Flickr images with 38 labels. We sample 1,000 images as the query set and the remaining regarded as the database, following (Wang et al., 2021a). Moreover, we randomly select 5,000 instances from the database to train hashing models. The **NUS-WIDE** dataset contains 269,648 images annotated with 81 concepts. We sample a subset with 21 of the most popular concepts, which consists of 195,834 images. 2,100 images are sampled from the subset as queries, while the rest images are regarded as the database. We randomly select 10,500 images from the database for the training set (Wang et al., 2021b). The **MS-COCO** dataset consists of 82,783 training samples and 40,504 validation samples, where each instance is annotated with at least one of the 80 categories. After combining the training and the validation set, we randomly pick 5,000 instances from them as queries and the rest as a database. For the training set, 10,000 images are randomly selected from the database. In addition, we make an extra experiment on CIFAR-10 (refer to A.2).

Protocols. To evaluate the performance of PgA, we conduct experiments on the standard metric, *i.e.*, Mean Average Precision (MAP) (Yang et al., 2020), and Precision-Recall (PR) curve. Following (Bai et al., 2020), we calculate MAP values on the top 5,000 results from the database.

Baselines. Following (Yang et al., 2020; Bai et al., 2020), we adopt DPH as the default attacked hashing method, which is a generic algorithm in deep hashing-based retrieval. AlexNet (Krizhevsky et al., 2012) is selected as the default backbone network to implement hashing models on FLICKR-25K, NUS-WIDE, and MS-COCO. We also evaluate the attack performance of our method against the defense model trained by ATRDH (Wang et al., 2021a) (the only adversarial training algorithm in deep hashing). We compare the proposed algorithm with multiple hashing attack methods, including P2P, DHTA, ProS-GAN, THA, HAG, and SDHA. For targeted attacks, we randomly select a label as the target label which does not share the same category as the true label. Other details of these methods are consistent with the original literature.

Table 2: MAP (%) of multiple attack methods after adversarial training by ATRDH (Wang et al., 2021a).

	FLICKR-25K			NUS-WIDE			MS-COCO		
Method	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits
Clean	72.75	73.34	73.10	66.42	65.90	68.69	49.23	50.87	49.93
P2P	52.84	54.56	57.13	51.23	56.70	54.05	33.55	33.24	30.96
DHTA	49.97	53.26	54.04	49.16	56.50	51.91	32.16	31.62	28.35
ProS-GAN	73.33	73.33	73.89	67.07	67.18	69.20	49.59	51.30	50.61
THA	48.58	49.94	50.43	50.57	50.92	50.86	32.31	30.52	29.33
HAG	41.99	45.58	45.81	42.15	47.46	45.19	26.50	27.49	28.73
SDHA	37.65	40.68	44.61	42.39	47.16	45.86	28.00	27.75	27.72
PgA† (Ours)	32.78	34.67	35.53	37.86	42.84	39.82	21.89	21.74	**22.21**
PgA (Ours)	**31.52**	**33.80**	**34.52**	**37.48**	**42.22**	**39.54**	**21.52**	**21.42**	22.45



(a) FLICKR-25K (b) NUS-WIDE (c) MS-COCO

Figure 4: Precision-Recall curves for ATRDH under 32 bits code length.

Implementation Details. We use stochastic gradient descent (SGD) for target hashing models with the initial learning rate of 0.01 and the momentum 0.9 as optimizers. We fix the mini-batch size of images as 32 and the weight decay parameter as 5×10^{-4}. All images are resized to 224×224 and normalized in $[0, 1]$ before feeding into hashing models. For the proposed attack method PgA, we adopt PGD (Bai et al., 2020) to optimize adversarial examples. The step size η and the number of iterations T are set to $1/255$ and 100, respectively. The perturbation budget ϵ is set to $8/255$. All codes are based on PyTorch 1.12 and are executed on NVIDIA RTX 3090 GPUs.

4.2 Attack Results

Table 1 and Table 2 present the attack performance (MAP) of different attack methods on original deep hashing networks without defense and adversarially trained models, respectively. The lower the MAP value, the stronger the attack performance. The "Clean" in these tables is to query with benign images, so MAP values refer to the original retrieval performance of the hashing model without attack. From Table 1, we can observe that the proposed method can greatly reduce the MAP values on three datasets with the hash bits varying from 16 to 64 and outperforms all other attacks. Compared to DHTA (Bai et al., 2020), the strongest targeted attack in Table 1, our PgA achieves average boosts of 18.68%, 14.66%, and 9.03% for tested bits on FLICKR-25K, NUS-WIDE, and MS-COCO, respectively. Moreover, our method outperforms it by an average of 4.40%, 4.09%, and 2.59% on three datasets, compared with the state-of-the-art non-targeted attack, SDHA. As for the defense model trained by ATRDH (the only adversarial training algorithm in deep hashing), Table 2 shows that all the MAP values of PgA are lower than other attack methods. Even in the face of SDHA, the proposed PgA brings an average improvement of 7.70%, 5.39%, and 6.02% for FLICKR-25K, NUS-WIDE, and MS-COCO, respectively. The superior performance of our method owes to the superiority of the pharos code, which considers the positive and negative samples simultaneously. In contrast, HAG and SDHA merely use the information from benign and positive samples, respectively. Thus, pharos code-based PgA is better than the previous state-of-the-arts.

For a more comprehensive comparison, the PR curves of different methods on three datasets with 32 bits length are shown in Fig. 3 and 4. We can see that the curves of our method are always below all others, demonstrating that our algorithm can attack hashing models more effectively.

Table 3: MAP (%) and time (second per image) of hashing attack methods for hashing models with 32-bit hash code length.

	FLICKR-25K		NUS-WIDE		MS-COCO	
Method	MAP	Time	MAP	Time	MAP	Time
P2P	42.69	0.59	31.76	0.58	21.78	0.58
DHTA	32.63	0.59	26.01	0.58	19.23	0.59
ProS-GAN	76.31	5.10	34.63	5.03	53.67	2.08
THA	37.54	0.13	27.21	0.08	20.65	0.07
HAG	24.37	0.59	14.32	0.58	13.59	0.59
SDHA	19.63	1.68	15.36	1.77	12.36	1.68
PgA† (Ours)	15.49	**0.04**	11.93	**0.04**	10.56	**0.04**
PgA (Ours)	**15.18**	**0.04**	**11.53**	**0.04**	**9.41**	**0.04**

Table 4: MAP (%) and time (second per image) of hashing attack methods for ATRDH trained hashing models with 32-bit hash code length.

	FLICKR-25K		NUS-WIDE		MS-COCO	
Method	MAP	Time	MAP	Time	MAP	Time
P2P	54.56	0.59	56.70	0.59	33.24	0.58
DHTA	53.26	0.58	56.50	0.59	31.62	0.58
ProS-GAN	73.33	5.09	67.18	5.13	51.30	2.07
THA	49.94	0.13	50.92	0.08	30.52	0.06
HAG	45.58	0.59	47.46	0.59	27.49	0.58
SDHA	40.68	1.60	47.16	1.66	27.75	1.60
PgA† (Ours)	34.67	**0.04**	42.84	**0.04**	21.74	**0.04**
PgA (Ours)	**33.80**	**0.04**	**42.22**	**0.04**	**21.42**	**0.04**

In addition, we compare the attack performance of PgA with that of PgA† to evaluate its effectiveness. The results in Table 1 and 2 shows that PgA with Eq. (13) is better than PgA† with Eq. (10) in most cases. Hence, the accelerated operation in Eq. (13) can effectively improve the attack effect.

4.3 Efficiency Analysis

To confirm the high efficiency of the proposed method, we record the MAP and time of various attack methods, where the time denotes the average interval (second per image) to conduct attacks for the test set. It is noted that this time usually includes the training time of the attack model, *e.g.*, ProS-GAN, and THA. The results are summarized in Table 3 and 4. It is observed that our PgA achieves the strongest attack performance and the shortest attack time for all datasets. Tables 3 and 4 are similar in terms of time results, and here we mainly focus on Table 3 for our analysis. Specifically, ProS-GAN has the lowest attack efficiency because it requires a few hours to train a generative network for attack. ProS-GAN takes about 127, 125, and 52 times longer than PgA on FLICKR-25K, NUS-WIDE, and MS-COCO, respectively. Moreover, P2P, DHTA, and HAG have similar attack time and they are much faster than ProS-GAN. Nevertheless, since they require 2000 iterations for gradients, they are still more than 13 times slower than our PgA. In summary, PgA not only outperform all the previous methods in attack performance but also can produce adversarial examples with the fastest speed.

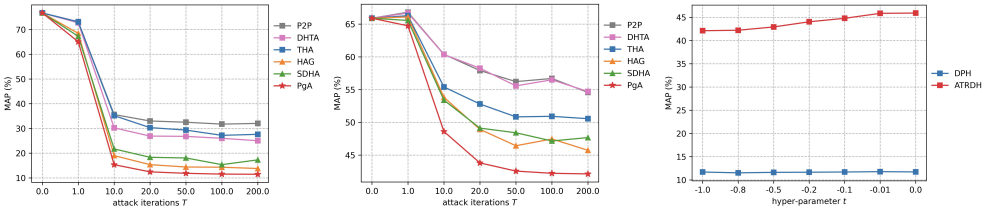
To further verify the high efficiency of our PgA in the same setting, we use PGD-20 to optimize adversarial perturbations for all attack methods, as shown in Table 5. PgA has the same speed as P2P, DHTA, and HAG, because they can directly calculate the target code to guide the generation of adversarial samples extremely fast. However, THA costs a lot of time to learn to construct the target code with a fully-connected network, so it is much slower than PgA. Furthermore, SDHA is less efficient than PgA because of its complex objective function (Lu et al., 2021).

Table 5: MAP (%) / time (second per image) on attacking hashing models with 32-bit hash code length. The tested attack methods all adopt PGD to construct adversarial samples with 20 steps (*i.e.*, $T = 20$ and $\eta = 1/255$).

	FLICKR-25K		NUS-WIDE		MS-COCO	
Method	DPH	ATRDH	DPH	ATRDH	DPH	ATRDH
P2P	41.66/0.01	58.12/0.01	33.02/0.01	57.91/0.01	22.37/0.01	33.27/0.01
DHTA	34.21/0.01	55.19/0.01	26.91/0.01	58.26/0.01	19.45/0.01	32.79/0.01
THA	40.67/0.10	53.48/0.10	30.33/0.05	52.80/0.06	20.92/0.04	31.40/0.03
HAG	24.95/0.01	52.42/0.01	15.36/0.01	48.93/0.01	15.16/0.01	27.69/0.01
SDHA	24.71/0.03	43.72/0.03	18.33/0.02	49.12/0.02	15.43/0.02	27.94/0.02
PgA (Ours)	**16.17**/0.01	**37.58**/0.01	**12.44**/0.01	**43.81**/0.01	**12.46**/0.01	**22.47**/0.01

4.4 Analysis on hyper-parameters

Effect of T & Efficiency. Figure 5(a) and 5(b) present attack performance (MAP) with different attack iterations (*i.e.*, T) of PGD. Overall, the MAP values decrease with increasing T. When T is greater than 20, the attack performance tends to level off for DPH, and 50 for ATRDH. For the same iterations, the attack performance of PgA maintains a large gap with other methods. Therefore, PgA is an efficient tool for evaluating deep hashing networks' robustness.



(a) T-DPH (b) T-ATRDH (c) t

Figure 5: MAP-T on NUS-WIDE for (a) DPH and (b) ATRDH trained model. (c) Attack performance with different t in Eq. (13).

Table 6: MAP (%) for different hashing models on NUS-WIDE.

	No defense					ATRDH				
Method	DPSH	HashNet	DSDH	DCH	CSQ	DPSH	HashNet	DSDH	DCH	CSQ
Clean	82.20	79.57	81.90	77.43	80.08	70.38	68.48	68.77	70.50	75.52
P2P	29.50	31.40	29.63	37.04	32.89	48.31	56.43	54.35	41.10	35.86
DHTA	23.38	25.12	22.30	31.68	27.35	47.96	54.46	53.21	37.96	31.78
ProS-GAN	29.84	28.62	30.05	62.87	31.01	69.64	69.33	70.03	58.21	73.72
THA	21.79	19.65	23.57	27.18	27.09	51.14	57.00	55.80	33.58	34.72
HAG	17.18	14.56	16.92	25.28	19.91	47.17	48.25	50.96	29.47	18.34
SDHA	15.38	13.31	16.69	22.35	11.78	42.78	46.75	47.94	46.28	18.77
PgA (Ours)	**10.57**	**11.94**	**11.30**	**15.04**	**6.86**	**37.37**	**42.31**	**41.01**	**21.65**	**14.83**

Effect of t. Figure 5(c) illustrates the effect of hyper-parameter t on the attack performance. For the DPH model without defense, there is no appreciable change in MAP for different t. For ATRDH, the attack performance shows a small decrease as t increases. Although attack performance is not extremely sensitive to t, picking an appropriate value of t can not be ignored.

4.5 Universality on different hashing methods

We argue that the proposed attack algorithm is generic to most popular hashing models. To verify this point, we conduct adversarial attacks on multiple hashing methods with 32-bit hash code length, including DPSH (Li et al., 2016), HashNet (Cao et al., 2017), DSDH (Li et al., 2017), DCH (Cao et al., 2018) and CSQ (Yuan et al., 2020), implemented by VGG11. The results are reported in Table 6. It can be seen from the table that our PgA is effective in fooling the illustrated hashing models with better attack performance than others. Firstly, when testing with hashing methods without defense, our PgA exceeds the previous state-of-the-art SDHA in all cases. Especially with DCH, there is a 7.31% gap between PgA and SDHA. Moreover, under the defense of ATRDH, PgA reduces the MAP of all hashing methods to significant minimums. Also, our PgA brings a 24.63% enhancement on the DCH model compared to the SDHA. Thus, the above phenomena demonstrate the universality of the proposed attack method, which can be utilized in most popular hashing algorithms. We make another further evaluation on the defense model trained with PgA, and please refer to A.3.

5 Conclusion

In this paper, we proposed the adversarial attack method (*i.e.*, PgA) for efficiently evaluating the adversarial robustness of deep hashing-based retrieval. Specifically, we provided the PGM to fast obtain the pharos code as the optimal representative of the image semantics for the attack in deep hashing. Moreover, PgA took the pharos code as "label" to guide the non-targeted attack, where the similarity between the pharos code and the hash code of adversarial example was minimized. Besides, we added adaptive weighting into the Hamming distance calculation, which further boosts the strength of PgA. Experiments showed that our algorithm performed state-of-the-art attack performance and efficiency compared to the previous attack methods in deep hashing-based retrieval.

REFERENCES

Jiawang Bai, Bin Chen, Yiming Li, Dongxian Wu, Weiwei Guo, Shu-tao Xia, and En-hui Yang. Targeted attack for deep hashing based retrieval. In *European Conference on Computer Vision*, pp. 618–634, 2020.

Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Machine Learning and Knowledge Discovery in Databases*, pp. 387–402, 2013.

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

Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Philip S Yu. Hashnet: Deep learning to hash by continuation. In *IEEE International Conference on Computer Vision*, pp. 5608–5617, 2017.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy*, pp. 39–57, 2017.

Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *ACM Workshop on Artificial Intelligence and Security*, pp. 15–26, 2017.

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 *ACM International Conference on Image and Video Retrieval*, pp. 1–9, 2009.

Khoa D Doan, Peng Yang, and Ping Li. One loss for quantization: Deep hashing with discrete wasserstein distributional matching. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9447–9457, 2022.

Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9185–9193, 2018.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, pp. 1–10, 2015.

Mark J Huiskes and Michael S Lew. The mir flickr retrieval evaluation. In *ACM International Conference on Image and Video Retrieval*, pp. 39–43, 2008.

Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. In *International Conference on Machine Learning*, pp. 2137–2146, 2018.

Qing-Yuan Jiang and Wu-Jun Li. Asymmetric deep supervised hashing. In *AAAI Conference on Artificial Intelligence*, pp. 3342–3349, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, pp. 1–15, 2015.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Neural Information Processing Systems*, pp. 1097–1105, 2012.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. In *International Conference on Learning Representations*, pp. 1–17, 2017.

Hanjiang Lai, Yan Pan, Ye Liu, and Shuicheng Yan. Simultaneous feature learning and hash coding with deep neural networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3270–3278, 2015.

Jingjing Li, Zhekai Du, Lei Zhu, Zhengming Ding, Ke Lu, and Heng Tao Shen. Divergence-agnostic unsupervised domain adaptation by adversarial attacks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

Qi Li, Zhenan Sun, Ran He, and Tieniu Tan. Deep supervised discrete hashing. In *Neural Information Processing Systems*, pp. 2479–2488, 2017.

Wu-Jun Li, Sheng Wang, and Wang-Cheng Kang. Feature learning based deep supervised hashing with pairwise labels. In *International Joint Conference on Artificial Intelligence*, pp. 1711–1717, 2016.

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 *European Conference on Computer Vision*, pp. 740–755, 2014.

Haomiao Liu, Ruiping Wang, Shiguang Shan, and Xilin Chen. Deep supervised hashing for fast image retrieval. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2064–2072, 2016.

Junda Lu, Mingyang Chen, Yifang Sun, Wei Wang, Yi Wang, and Xiaochun Yang. A smart adversarial attack on deep hashing based image retrieval. In *International Conference on Multimedia Retrieval*, pp. 227–235, 2021.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, pp. 1–28, 2017.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2574–2582, 2016.

Tianyu Pang, Xiao Yang, Yinpeng Dong, Hang Su, and Jun Zhu. Bag of tricks for adversarial training. In *International Conference on Learning Representations*, pp. 1–21, 2021.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *ACM on Asia Conference on Computer and Communications Security*, pp. 506–519, 2017.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, 22:400–407, 1951.

Shupeng Su, Chao Zhang, Kai Han, and Yonghong Tian. Greedy hash: Towards fast optimization for accurate hash coding in cnn. In *Neural Information Processing Systems*, pp. 806–815, 2018.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, pp. 1–10, 2014.

Francisco Utrera, Evan Kravitz, N Benjamin Erichson, Rajiv Khanna, and Michael W Mahoney. Adversarially-trained deep nets transfer better: Illustration on image classification. In *International Conference on Learning Representations*, 2021.

Jingdong Wang, Ting Zhang, Nicu Sebe, Heng Tao Shen, et al. A survey on learning to hash. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40:769–790, 2018.

Xunguang Wang, Zheng Zhang, Guangming Lu, and Yong Xu. Targeted attack and defense for deep hashing. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 2298–2302, 2021a.

Xunguang Wang, Zheng Zhang, Baoyuan Wu, Fumin Shen, and Guangming Lu. Prototype-supervised adversarial network for targeted attack of deep hashing. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16357–16366, 2021b.

Zijian Wang, Zheng Zhang, Yadan Luo, Zi Huang, and Heng Tao Shen. Deep collaborative discrete hashing with semantic-invariant structure construction. *IEEE Transactions on Multimedia*, 23: 1274–1286, 2021c.

Eric Wong, Leslie Rice, and J Zico Kolter. Fast is better than free: Revisiting adversarial training. In *International Conference on Learning Representations*, pp. 1–17, 2020.

Rongkai Xia, Yan Pan, Hanjiang Lai, Cong Liu, and Shuicheng Yan. Supervised hashing for image retrieval via image representation learning. In *AAAI Conference on Artificial Intelligence*, pp. 2156–2162, 2014.

Yanru Xiao and Cong Wang. You see what i want you to see: Exploring targeted black-box transferability attack for hash-based image retrieval systems. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1934–1943, 2021.

Erkun Yang, Tongliang Liu, Cheng Deng, and Dacheng Tao. Adversarial examples for hamming space search. *IEEE Transactions on Cybernetics*, 50:1473–1484, 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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3083–3092, 2020.

Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning*, pp. 7472–7482, 2019.

Zheng Zhang, Xunguang Wang, Guangming Lu, Fumin Shen, and Lei Zhu. Targeted attack of deep hashing via prototype-supervised adversarial networks. *IEEE Transactions on Multimedia*, 24: 3392–3404, 2021.

Han Zhu, Mingsheng Long, Jianmin Wang, and Yue Cao. Deep hashing network for efficient similarity retrieval. In *AAAI Conference on Artificial Intelligence*, pp. 2415–2421, 2016.

A Appendix

A.1 Proof of PGM

Theorem pharos code $\boldsymbol{b}^\star$ which satisfies Eq. (6) can be calculated by the Pharos Generation Method (PGM), *i.e.*,

$$\begin{aligned}\boldsymbol{b}^\star &= \arg\min_{\boldsymbol{b}^\star\in\{-1,+1\}^K} \sum_i \sum_j [w_i D_{\mathrm{H}}(\boldsymbol{b}^\star, \boldsymbol{b}_i^{(\mathrm{p})}) - w_j D_{\mathrm{H}}(\boldsymbol{b}^\star, \boldsymbol{b}_j^{(\mathrm{n})})] \\ &= \operatorname{sign}\left(\sum_i^{N_\mathrm{p}} \sum_j^{N_\mathrm{n}} (w_i \boldsymbol{b}_i^{(\mathrm{p})} - w_j \boldsymbol{b}_j^{(\mathrm{n})})\right).\end{aligned}$$

proof. We define the following function:

$$\psi(\boldsymbol{b}) = \sum_i \sum_j [w_i D_{\mathrm{H}}(\boldsymbol{b}, \boldsymbol{b}_i^{(\mathrm{p})}) - w_j D_{\mathrm{H}}(\boldsymbol{b}, \boldsymbol{b}_j^{(\mathrm{n})})].$$

As the pharos code $\boldsymbol{b}^\star$ need to be the optimal solution of the minimizing objective, the above theorem is equivalent to prove the following inequality:

$$\psi(\boldsymbol{b}) \geq \psi(\boldsymbol{b}^\star), \quad \forall\, \boldsymbol{b} \in \{-1, +1\}^K.$$

Let $\boldsymbol{b} = \{b_1, b_2, ..., b_K\}$, then we have

$$\begin{aligned}
\psi(\boldsymbol{b}) &= \sum_i \sum_j [w_i \frac{1}{2}(K - \boldsymbol{b}^\top \boldsymbol{b}_i^{(\mathrm{p})}) - w_j \frac{1}{2}(K - \boldsymbol{b}^\top \boldsymbol{b}_j^{(\mathrm{n})})] \\
&= -\frac{1}{2} \sum_i \sum_j [w_i \boldsymbol{b}^\top \boldsymbol{b}_i^{(\mathrm{p})} - w_j \boldsymbol{b}^\top \boldsymbol{b}_j^{(\mathrm{n})}] + \xi \\
&= -\frac{1}{2} \sum_i \sum_j [w_i \sum_{k=1}^{K} b_k b_{ik}^{(\mathrm{p})} - w_j \sum_{k=1}^{K} b_k b_{jk}^{(\mathrm{n})}] + \xi \\
&= -\frac{1}{2} \sum_i \sum_j \left(\sum_{k=1}^{K} w_i b_k b_{ik}^{(\mathrm{p})} - \sum_{k=1}^{K} w_j b_k b_{jk}^{(\mathrm{n})} \right) + \xi \\
&= -\frac{1}{2} \sum_i \sum_j \sum_{k=1}^{K} b_k (w_i b_{ik}^{(\mathrm{p})} - w_j b_{jk}^{(\mathrm{n})}) + \xi \\
&= -\frac{1}{2} \sum_{k=1}^{K} b_k \sum_i \sum_j (w_i b_{ik}^{(\mathrm{p})} - w_j b_{jk}^{(\mathrm{n})}) + \xi,
\end{aligned}$$

where ξ is a constant. Similarly,

$$\psi(\boldsymbol{b}^\star) = -\frac{1}{2} \sum_{k=1}^{K} b_k^\star \sum_i \sum_j (w_i b_{ik}^{(\mathrm{p})} - w_j b_{jk}^{(\mathrm{n})}) + \xi.$$

Due to the nature of absolute value, we have

$$\begin{aligned}
\psi(\boldsymbol{b}) &= -\frac{1}{2} \sum_{k=1}^{K} b_k \sum_i \sum_j (w_i b_{ik}^{(\mathrm{p})} - w_j b_{jk}^{(\mathrm{n})}) + \xi \\
&\geq -\frac{1}{2} \sum_{k=1}^{K} \left| b_k \sum_i \sum_j (w_i b_{ik}^{(\mathrm{p})} - w_j b_{jk}^{(\mathrm{n})}) \right| + \xi \\
&= -\frac{1}{2} \sum_{k=1}^{K} |b_k| \left| \sum_i \sum_j (w_i b_{ik}^{(\mathrm{p})} - w_j b_{jk}^{(\mathrm{n})}) \right| + \xi \\
&= -\frac{1}{2} \sum_{k=1}^{K} \left| \sum_i \sum_j (w_i b_{ik}^{(\mathrm{p})} - w_j b_{jk}^{(\mathrm{n})}) \right| + \xi \\
&= -\frac{1}{2} \sum_{k=1}^{K} \mathrm{sign}(\sum_i \sum_j (w_i b_{ik}^{(\mathrm{p})} - w_j b_{jk}^{(\mathrm{n})})) \sum_i \sum_j (w_i b_{ik}^{(\mathrm{p})} - w_j b_{jk}^{(\mathrm{n})}) + \xi \\
&= -\frac{1}{2} \sum_{k=1}^{K} b_k^\star \sum_i \sum_j (w_i b_{ik}^{(\mathrm{p})} - w_j b_{jk}^{(\mathrm{n})}) + \xi \\
&= \psi(\boldsymbol{b}^\star).
\end{aligned}$$

That is, $\psi(\boldsymbol{b}) \geq \psi(\boldsymbol{b}^\star)$. Hence, the Theorem is proved.

A.2 Attack results on CIFAR-10

Table 7 shows the results of the hashing attack methods on the single-label dataset CIFAR-10 (Cao et al., 2017). We can observe that our PgA is a little bit better than the state-of-the-art SDHA for DPH. However, the proposed PgA outperforms HAG and SDHA over 2.23%. Especially under the case of 64 bits, PgA brings an boost of 4.05% and 10.19% compared to HAG and SDHA, respectively.

Table 7: MAP (%) of attack methods on CIFAR-10.

	DPH			ATRDH		
Method	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits
Clean	67.72	78.00	79.64	60.98	62.74	63.08
P2P	4.11	3.44	2.96	31.55	31.94	32.19
DHTA	2.08	1.24	0.91	29.87	31.12	31.04
ProS-GAN	2.93	6.13	5.17	64.14	66.27	66.98
THA	2.64	6.77	8.42	31.95	32.79	35.06
HAG	0.95	1.16	1.60	16.41	16.51	18.30
SDHA	0.32	0.52	0.55	18.90	20.75	24.54
PgA (Ours)	**0.31**	**0.49**	**0.48**	**14.18**	**13.48**	**14.25**

A.3 Adversarial Training

We use the generated adversarial samples for adversarial training to verify whether the proposed method is still valid. The objective of the adversarial training is formulated as follows:

$$\min_{\theta} \mathcal{L}_{adv} = \mathcal{L}_{ori} - \sum_{i=1}^{N} \frac{1}{K} (\boldsymbol{b}_i^{\star})^{\top} f_{\theta}(\boldsymbol{x}_i'), \tag{16}$$

where $\boldsymbol{b}_i^{\star}$ is the pharos code of the instance $\boldsymbol{x}_i$, and $\boldsymbol{x}_i'$ is the adversarial example of $\boldsymbol{x}_i$. The latter term in Eq. (16) can rebuild similarity between the adversarial sample and the true semantics. $\mathcal{L}_{ori}$ is the original loss function of the deep hashing model, which ensures the basic performance of hashing learning. The experimental results are illustrated in Table 8. The adversarial training does improve the defense capability of the deep hashing model, but our attack method is still valid and significantly outperforms the other methods.

Table 8: MAP (%) of attack methods on NUS-WIDE.

Method	16 bits	32 bits	64 bits
Clean	70.51	68.50	62.34
P2P	45.50	53.08	56.78
DHTA	43.12	50.30	55.47
ProS-GAN	64.27	67.81	62.49
THA	48.36	55.74	59.90
HAG	45.26	51.32	52.26
SDHA	34.67	45.28	51.03
PgA (Ours)	**26.72**	**37.70**	**49.70**