AdvHash: Set-to-set Targeted Attack on Deep Hashing with One Single Adversarial Patch

Shengshan Hu*†
hushengshan@hust.edu.cn
School of Cyber Science and Engineering, Huazhong University of Science and Technology
Wuhan, China

Yechao Zhang*†
ycz@hust.edu.cn
School of Cyber Science and Engineering, Huazhong University of Science and Technology
Wuhan, China

Xiaogeng Liu*†
liuxiaogeng@hust.edu.cn
School of Cyber Science and Engineering, Huazhong University of Science and Technology
Wuhan, China

Leo Yu Zhang
leo.zhang@deakin.edu.au
School of Information Technology, Deakin University
Melbourne, Australia

Minghui Li
School of Software Engineering, Huazhong University of Science and Technology
Wuhan, China

Hai Jin†‡
hjin@hust.edu.cn
School of Computer Science and Technology, Huazhong University of Science and Technology
Wuhan, China

ABSTRACT

The success of machine learning has brought great research progress for multimedia retrieval. Due to the widely explored adversarial attacks on DNNs, image retrieval system based on deep learning is also susceptible to such vulnerability. Nevertheless, the generalization ability of adversarial noise in the targeted attacks against image retrieval is yet to be explored.

In this paper, we propose AdvHash, the first targeted mismatch attack on deep hashing through adversarial patch. After superimposed with the same adversarial patch, any query image with a chosen label will retrieve a set of irrelevant images with the target label. Concretely, we first formulate a set-to-set problem, where a set of samples are pushed into a predefined clustered area in the Hamming space. Then we obtain a target anchor hash code and transform the attack to a set-to-point optimization. In order to generate a stable class-wise adversarial patch more efficiently, we propose a product-based weighted gradient aggregation strategy to dynamically adjust the gradient direction of the patch by exploiting the Hamming distances between training samples and the target anchor hash code and assigning different weights to discriminatively aggregate gradients. Extensive experiments on benchmark datasets verify that AdvHash is highly effective at attacking two state-of-the-art deep hashing schemes. Our codes are available at: https://github.com/CGCL-codes/AdvHash.

CCS CONCEPTS

• **Security and privacy**; • **Computing methodologies** → Computer vision; • **Information systems** → *Information retrieval*;

KEYWORDS

Adversarial Patch; Targeted Attack; Image Retrieval; Deep Hashing

ACM Reference Format:
Shengshan Hu, Yechao Zhang, Xiaogeng Liu, Leo Yu Zhang, Minghui Li, and Hai Jin. 2021. AdvHash: Set-to-set Targeted Attack on Deep Hashing with One Single Adversarial Patch. In *Proceedings of the 29th ACM International Conference on Multimedia (MM'21), October 20–24, 2021, Virtual Event, China.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3474085.3475396

1 INTRODUCTION

With the explosive growth of data in cyberspace, multimedia retrieval has become a crucial research topic, and the study of efficient search stands as an ongoing interest. Benefit from the powerful feature representation ability of *deep neural networks* (DNNs), deep learning-based methods have become state-of-the-art solutions for multimedia retrieval. Existing image retrieval approaches are divided into two major categories, namely deep feature-based [1, 32, 38] and deep hashing-based [5, 6, 8, 17, 25, 34, 44]. The former one straightly aggregates deep features extracted from pre-trained or fine-tuned DNN models as and then measures their similarity by computing Euclidean distance or cosine similarity, while deep hashing is designed to improve the search efficiency by converting the original high-dimension feature space into a compact binary *Hamming space* and measure the similarity by *Hamming distance*.

Recent studies [4, 14, 36] revealed that DNNs are vulnerable to *adversarial examples*, which will be misclassified to any other label (*non-targeted attack*) or a specific label (*targeted attack*) at the inference stage. Both targeted and non-targeted attacks are realized by crafting perturbations or patches. Perturbations are

*Hubei Engineering Research Center on Big Data Security, School of Cyber Science and Engineering, HUST, Wuhan, 430074, China
†National Engineering Research Center for Big Data Technology and System, Services Computing Technology and System Lab, HUST, Wuhan, 430074, China
‡Cluster and Grid Computing Lab, School of Computer Science and Technology, HUST, Wuhan, 430074, China

MM '21, October 20–24, 2021, Virtual Event, China

ACM ISBN 978-1-4503-8651-7/21/10…$15.00
https://doi.org/10.1145/3474085.3475396

invisible to human beings, but usually need to modify the entire image and are thus easy to be mitigated through simple image transformations like feature squeezing [41]. In contrast, patches are conspicuous, but only cover a small area and thus are more feasible for real-world scenarios. Naturally, deep learning-based multimedia retrieval systems inherit this vulnerability and are susceptible to adversarial attacks as well. In order to investigate the robustness of deep learning-based image retrieval systems, many prior works mainly focus on attacking deep feature-based retrieval systems [7, 21, 27, 37, 45, 46], and aim to construct adversarial perturbations or patches accordingly.

As another side of the coin, some researchers recently turned to attacking deep hashing-based image retrieval systems and proposed different methods [2, 39, 40, 43]. However, all these approaches are designed to construct adversarial perturbations, while the patch-based attack is yet to be explored for the study of robustness *w.r.t* the deep hashing-based image retrieval systems. On the another hand, these previous attacks against deep hashing are constrained to the image-specific scenario (*i.e.*, each adversarial noise is effective for a single query exclusively), failing to explore the generalization ability of adversarial noise. Furthermore, we argue that it is non-trivial to improve the generalization ability when the targeted attack is made as one of the design goals. Note that it is rather easy to make a single query shift away from the original place in the embedding space (feature space or Hamming space), yet to make all queries into a predefined area with only a single noise is far more challenging. To the best of our knowledge, we are the first to explore the generalization ability of adversarial noise in the targeted attack scenario towards retrieval system.

In this work, we propose AdvHash, the first class-wise targeted attack against deep hashing, where one single patch with strong generalization ability can be effectively applied to a set of image samples with a chosen label to cause the mismatch of retrieval system. Different from the adversarial attacks on image classification, AdvHash not only expects the retrieval system to make a wrong top-1 prediction, but also seeks to corrupt the entire list of retrieved results. Moreover, the attack in AdvHash is targeted, *i.e.*, AdvHash wishes the retrieved results to be a set of objects that all belong to a specific class. For this goal, we first formulate this as a set-to-set problem. Observing that most state-of-the-art deep hashing systems are highly clustered, AdvHash aims to find a target anchor code to represent the target cluster in the Hamming space. Then the set-to-set problem is transformed into a set-to-point optimization, which is easier to be solved. In addition, through extensive experiments, we find that training adversarial examples for targeted attacks on deep hashing share the same convergence variance. We thus propose the product-based weighted gradient aggregation strategy to generate a stable adversarial patch with strong generalization ability in a more efficient way. The gradients will be dynamically weighted in each iteration according to the associated inner products, and the patch will be recurrently updated based on the historical information.

In summary, this work makes the following contributions:

- We propose AdvHash, the first targeted attack on deep hashing-based image retrieval systems through adversarial patch, where one single adversarial patch can be applicable to a set of images with the chosen label.
- We proposed product-based weighted gradients aggregation as well as recurrent patch update strategies to improve the effectiveness and efficiency of AdvHash.
- Our extensive experiments on benchmark datasets ImageNet and NUS-WIDE verify that AdvHash is highly effective at attacking state-of-the-art deep hashing schemes HashNet and CSQ.

2 RELATED WORK

2.1 Adversarial Attacks on Image Classification

Perturbation-based Attack. Adversarial examples were first introduced in [36] to cause the misprediction of DNN classifiers. Some follow-up works [14, 19] were made to accelerate the generation process. The adversarial perturbations generated by these methods, however, are image-specific and did not generalize to various samples. Later, the study of *universal adversarial perturbations* (UAPs) [24, 28–30] emerges, where an image-agnostic adversarial perturbation can be applied to a bunch of images. Although UAPs were studied in non-targeted scenario originally, afterwards targeted UAPs were proposed in [15, 31] to misclassify all the input samples to a predefined target class. As a following work, the *double targeted universal adversarial perturbation* (DT-UAP) is proposed that can not only arbitrarily choose the class of the source side (*i.e.*, input samples) but also the target side (*i.e.*, output labels) [3].

Patch-based Attack. Unlike the perturbation-based method, which needs to precisely control over each pixel of the entire image, adversarial patches [4] are visible but confined to a small area of the image. LaVAN [18] was proposed to make the adversarial patch localized at a certain area without covering any of the main object(s) in the image. *Perceptual-Sensitive GAN* (PS-GAN) [22] conducts attacks by using *generative adversarial networks* (GANs) to generate patches. Similar to UAP that aims at improving the generalization ability, Liu *et al.* [23] proposed a universal adversarial patch against the classification model. However, it still confines to the non-targeted scenario. For targeted attack, the authors in [3] conducted experimental attempts to explore the possibility of extending DT-UAP to the patch-based case.

2.2 Adversarial Attacks on Image Retrieval

Image retrieval is a long-standing research topic in computer vision, and it is observed that adversarial attack methods for classification systems cannot be directly applied in image retrieval [21]. Recently, some researchers have paid attention to exploring the vulnerabilities of deep learning-based image retrieval systems.

Attacks on Deep Feature. Existing image retrieval systems can be divided into deep feature-based and deep hashing-based methods. Adversarial attacks against deep feature-based image retrieval have been investigated. Different approaches such as PIRE [27], UAA-GAN [46], and AP-GAN [45] have been proposed to realize non-targeted image-specific attack. For the purpose of improving the generalization ability of adversarial noise, Li *et al.* [21] realize the non-targeted UAP in retrieval task. Meanwhile, the first targeted attack against image retrieval TMAA was proposed in [37]. However, TMAA still falls into the image-specific scenario. Most recently, a query-efficient decision-based black-box attack against image retrieval was proposed in [7].

Table 1: A comprehensive comparison among existing attacks on deep learning-based image retrieval

Method	Target model	Attack type	Generalization ability	Targeted/Non-targeted
PIRE [27]	Deep feature	Perturbation	Image-specific	Non-targeted
UAA-GAN [46]	Deep feature	Perturbation	Image-specific	Non-targeted
DAIR [7]	Deep feature	Perturbation	Image-specific	Non-targeted
AP-GAN [45]	Deep feature	**Patch**	Image-specific	Non-targeted
TMAA [37]	Deep feature	Perturbation	Image-specific	**Targeted**
Li *et al.* [21]	Deep feature	Perturbation	**Image-agnostic**	Non-targeted
HAG [43]	Deep hashing	Perturbation	Image-specific	Non-targeted
CWDM [40]	Deep hashing	Perturbation	Image-specific	Non-targeted
DHTA [2]	Deep hashing	Perturbation	Image-specific	**Targeted**
Xiao *et al.* [39]	Deep hashing	Perturbation	Image-specific	**Targeted**
AdvHash	Deep hashing	**Patch**	**Class-wise**	**Targeted**

- Image-specific: the adversarial noise is effective to only one query sample.
- Image-agnostic: the adversarial noise is effective to all the images that belong to any class.
- Class-wise: the adversarial noise is effective to all the images that belong to a specific class.

Attacks on Deep Hashing. For the study of the vulnerability *w.r.t.* deep hashing, [43] takes the first lead by maximizing the Hamming distance between the perturbed image and the original one. Before long, another non-targeted attack method CWDM was proposed in [40], which seeks to push the perturbed sample into a sparse place in Hamming space. As for targeted attacks, DHTA was proposed in [2], aimed at retrieving images from the target class. Most recently, another targeted attack was proposed in [39], aimed to increase the transferability of adversarial examples. However, all these previous works all falls into the image-specific scenario.

Our work aims to attack deep hashing, we are motivated to conduct a targeted patch-based attack against deep hashing-based image retrieval systems with strong generalization ability. The setting of our work is close to DT-UAP while the task is entirely different, as we manage to make the adversarial patch applicable for all the image samples that belong to a same class (*i.e.*, class-wise) and make the patched images successfully retrieve images with a same label in the top-K list (*i.e.*, targeted). Table 1 summarizes the existing adversarial attacks on deep learning-based image retrieval systems and highlights the position of our proposed attack.

3 METHODOLOGY

3.1 Preliminaries

In this section, we briefly introduce the general process of deep hashing based retrieval. Suppose $\boldsymbol{X} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}^N$ indicates a sample set containing N images labeled with L classes, where $\boldsymbol{x}_i$ indicates a single image, and $\boldsymbol{y}_i = [\boldsymbol{y}_{i1}, ..., \boldsymbol{y}_{iL}] \in \{0,1\}^L$ is the corresponding multi-label vector of $\boldsymbol{x}_i$. The l-th component of indicator vector $\boldsymbol{y}_{il} = 1$ means that the image $\boldsymbol{x}_i$ belongs to class l. Let $\boldsymbol{X}^t = \{(\boldsymbol{x}_i, \boldsymbol{y}_i) \in \boldsymbol{X} | \boldsymbol{y}_i = \boldsymbol{y}_t)\}$ denote the subset of $\boldsymbol{X}$ that consists of images labeled with $\boldsymbol{y}_t$. Let $\boldsymbol{X}^s = \{(\boldsymbol{x}_q, \boldsymbol{y}_q) \in \boldsymbol{X} | \boldsymbol{y}_q = \boldsymbol{y}_s)\}$ be a subset of $\boldsymbol{X}$ that consists of Q ($1 \le q \le Q$) query images with the same source label $\boldsymbol{y}_s$.

Deep Hashing Model. The K-bit hash code $\boldsymbol{c}$ of an image $\boldsymbol{x}$ is obtained through a deep hashing model $F(\cdot)$ as follows:

$$\boldsymbol{c} = F(\boldsymbol{x}) = \text{sign}(H(\boldsymbol{x})) \quad \text{s.t. } \boldsymbol{c} \in \{1, -1\}^K \tag{1}$$

where $H(\cdot)$ consists of a feature extractor followed by a K-nodes fully-connected layer, namely hash layer. K is the hash bit length that is equivalent to the length of the output of hash layer of $H(\boldsymbol{x})$. The feature extractor is usually a CNN model using backbone like VGG or ResNet [44]. In particular, during the training process, the $\text{sign}(\cdot)$ is usually approximated through a $\tanh(\cdot)$, which is suitable for optimization with back propagation and can squeeze continuous feature values into the range $(-1, +1)$ [6].

Similarity Measures. Given an image database $\{\boldsymbol{x}_j\}_{j=1}^J$ containing J images and its corresponding hash codes $\{\boldsymbol{c}_j\}_{j=1}^J$, when a query image $\boldsymbol{x}_q$ is initiated, it is first fed into $F(\cdot)$ to obtain the hash code $\boldsymbol{c}$ through Eq. (1). Then the Hamming distances $\boldsymbol{D} = \{d_{\mathcal{H}}(\boldsymbol{c}, \boldsymbol{c}_j)\}_{j=1}^J$ between the query $\boldsymbol{x}_q$ and each sample of the database will be used to measure their similarity, which is calculated as $d_{\mathcal{H}}(\boldsymbol{c}, \boldsymbol{c}_j) = (K - \boldsymbol{c} \cdot \boldsymbol{c}_j)/2$.

It is clear that there is a linear relationship between the inner product $\boldsymbol{c} \cdot \boldsymbol{c}_j$ of two hash codes and their Hamming distance $d_{\mathcal{H}}(\boldsymbol{c}, \boldsymbol{c}_j)$. Thus, the retrieval system returns a list of images in a descending order of the inner products.

3.2 Formulating Set-to-set Targeted Attack

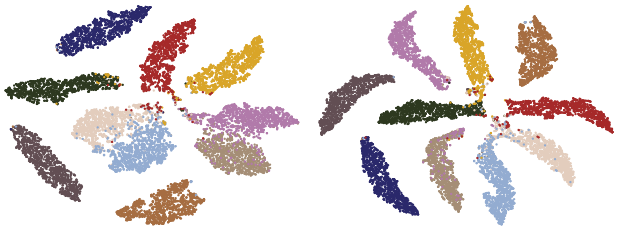
Problem Formulation. In general, given a certain source label $\boldsymbol{y}_s$ and a targeted label $\boldsymbol{y}_t$, we want to generate a single localized adversarial patch δ, which can be applicable to all the images $\boldsymbol{X}^s = \{(\boldsymbol{x}_q, \boldsymbol{y}_s)\}$ labeled with $\boldsymbol{y}_s$. Specifically, after pasted with the patch δ, each adversarial query $\boldsymbol{x}'_q$ from the set $\boldsymbol{X}' = \{(\boldsymbol{x}'_q, \boldsymbol{y}_s)\}$ will retrieve a set of images labeled with $\boldsymbol{y}_t$. Each $\boldsymbol{x}'_q$ is composed of the original image $\boldsymbol{x}_q \in \boldsymbol{X}^s$, an additive patch δ and a binary mask matrix M, denoted as:

$$\boldsymbol{x}'_q = (1 - M) \odot \boldsymbol{x}_q + M \odot \delta \tag{2}$$

where $\odot$ represents element-wise multiplication. Note that M has the same dimension with the input image $\boldsymbol{x}$ and we set its element to 1 where the adversarial patch is added and 0 otherwise. Intuitively, in order to realize a label-wise targeted attack on deep hashing, we need to minimize the distance between the hash codes of the adversarial images $\boldsymbol{X}'$and the sample set $\boldsymbol{X}^t$ with the target label $\boldsymbol{y}_t$, *i.e.*,

$$\min_{\delta} d\left(F\left(\boldsymbol{X}'\right), F\left(\boldsymbol{X}^t\right)\right) \tag{3}$$

where $F(\boldsymbol{X}') = \left\{F(\boldsymbol{x}'_q) \mid \boldsymbol{x}'_q \in \boldsymbol{X}'\right\}$ $F(\boldsymbol{X}^{(t)}) = \left\{F(\boldsymbol{x}) \mid \boldsymbol{x} \in \boldsymbol{X}^t\right\}$, and $d(\cdot, \cdot)$ denotes the set-to-set distance metric.



(a) HashNet (b) CSQ

Figure 1: Visualization of the hash layer outputs of two state-of-the-art deep hashing models CSQ and HashNet, trained on a dataset containing images of 10 classes from ImageNet. Each color represents a class exclusively.

However, it is difficult to solve this optimization problem by simply minimizing each pairwise loss since the search space increases quadratically. Therefore, we propose transforming the above set-to-set problem to a set-to-point optimization, which is much easier to solve.

Set-to-point Optimization. We first note that the input images have a larger dimension than the mapped binary hash codes. Besides, state-of-the-art deep hashing models are designed to minimize the Hamming distances between samples that have the same semantic labels and maximize the Hamming distances of those that differ from each other [6, 10, 20, 42, 44, 47]. Therefore, the deep hashing inevitably becomes a *multivalued mapping* [13] from Hamming space to input pixel space, which means that different images may have identical hash code [11]. We conduct extensive experiments to further verify this assertion, results on HashNet and CSQ are depicted in Fig. 1. It is clear from this figure that the samples with the same label will cluster as much as possible, and many samples in the cluster have exactly the same hash code.

Motivated by the above observation, we aim to find a target anchor point to represent the target cluster to be attacked. With an anchor point available, the set-to-set optimization problem of Eq. (3) naturally reduces to a set-to-point problem. Our key idea is shown in Fig. 2. By sampling n images from $\boldsymbol{X}^t$ and then adopting the component-voting scheme in [2], we obtain the target anchor hash code $\boldsymbol{h}_t = \text{sign}\left(\sum_{j=1}^{n} \boldsymbol{c}_j\right)$, where $\boldsymbol{c}_j$ is the hash code of each image from n samples, $\boldsymbol{h}_t$ is the hash code of target anchor point. Then the set-to-set formulation is converted to a set-to-point optimization as follows:

$$\min_{\delta} d\left(F\left(\boldsymbol{X}'\right), \boldsymbol{h}_t\right) \tag{4}$$

Overall Objective Function. In the typical point-to-point adversarial attack scenario, the adversarial example $\boldsymbol{x}'_q$ is obtained through the following optimization:

$$\min d_{\mathcal{H}}(\text{sign}(H(\boldsymbol{x}'_q), \boldsymbol{h}_t) \tag{5}$$

Similarly, in order to minimize the Hamming distances between Q adversarial images and the target anchor code, we can convert Eq. (4) into:

$$\min_{\delta} \sum_{q=1}^{Q} d_{\mathcal{H}}(\text{sign}(H(\boldsymbol{x}'_q), \boldsymbol{h}_t) \quad \text{s.t. } \delta \in [0, 255] \tag{6}$$

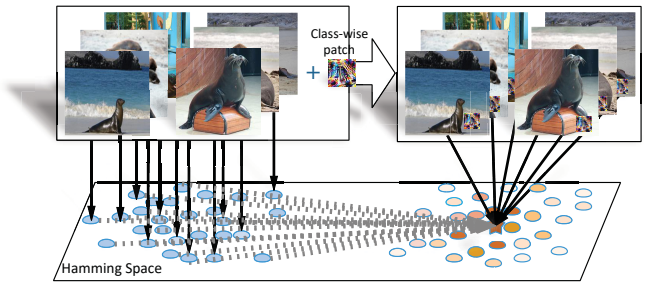


Figure 2: Illustration of the set-to-point optimization

Note that after each update of the patch, a clipping is performed as $\delta = \min(255, \max(0, \delta))$. In this paper, we abuse the notation $\delta \in [0, 255]$ to denote that each entry of δ is in $[0, 255]$.

As stated in Section 3.1, typical deep hashing model use $\tanh(\cdot)$ during training time. Likewise, we use the hyperbolic tangent function to approximate the sign function during the patch generation. Similar to [2, 43], we also use a hyper-parameter α to keep a relatively large gradient during training to avoid *gradient vanishing*. By substituting Hamming distance with inner product, Eq. (6) is then transformed to

$$\min_{\delta} -\sum_{q=1}^{Q} \frac{1}{K} \boldsymbol{h}_t^{\top} \tanh\left(\alpha H(\boldsymbol{x}'_q)\right) \tag{7}$$

However, since the continuous values $H(\boldsymbol{x}'_q)$ will be projected into binary codes via $\text{sign}(\cdot)$ eventually, if $H(\boldsymbol{x}'_q)$ has already reached to the same sign as $\boldsymbol{h}_t$ and gets close to the bound (*i.e.*, -1 or 1) on some dimensions, continuing to optimize these dimensions is useless at the early stage of optimization. More importantly, the closer to the bound, the smaller the absolute gradients will be regardless of the value of α [6]. Therefore, we introduce a balancing vector $\boldsymbol{w}_q$ to make the optimizer focus on the dimensions whose signs are still different from $\boldsymbol{h}_t$ or have a larger margin to the bound. In this concern, the final loss function of our scheme becomes

$$\mathcal{L}(\delta) = -\frac{1}{e} \sum_{q=1}^{Q} \boldsymbol{h}_t^{\top} \left(\boldsymbol{w}_q \odot \tanh\left(\alpha H\left(\boldsymbol{x}'_q\right)\right)\right) \tag{8}$$

where e is the total number of non-zero elements of $\{\boldsymbol{w}_q\}_{q=1}^{Q}$, and $\boldsymbol{w}_q$ is computed as follows:

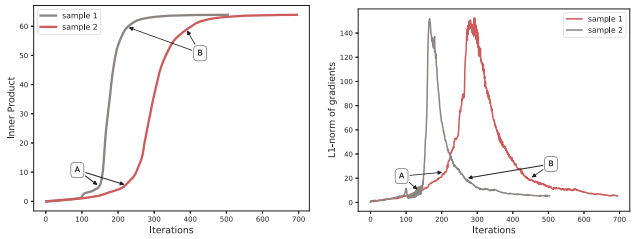
$$\boldsymbol{w}_q^k = \begin{cases} 1, & \text{if } \left|\tanh(H^k(\boldsymbol{x}'_q)) + \boldsymbol{h}_t^k\right| < 1 + \lambda \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

where k indicates the k-th dimension corresponding to the k-th output of hash layer $H(\boldsymbol{x}'_q)$ and the k-th value of target anchor code $\boldsymbol{h}_t$, λ is a threshold that denotes the margin between $\boldsymbol{h}_t^k$ and $\tanh(H^k(\boldsymbol{x}'_q))$.

3.3 Product-based Weighted Gradient Aggregation

Based on the optimization goal stated in Eq. (8), we further propose a weighted gradient aggregation strategy by exploiting gradient information in the Hamming space, in order to generate a more stable and universal adversarial patch in an efficient way.

Key Observations. We first explore the characteristics of Hamming space by evaluating inner products and gradients to discover



(a) inner-product (b) l_1-norm

Figure 3: The inner product V.S. l_1-norm of gradients

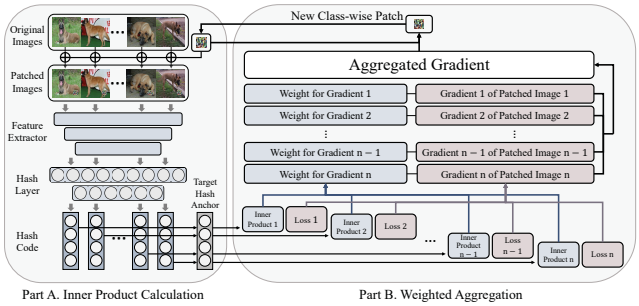


Figure 4: The pipeline of using aggregated gradient to solve Eq. (8)

the potential common patterns that exist in the generation of each adversarial example with the same label. We sample different images with the source label $\boldsymbol{y}_s$ and choose an anchor point within the target cluster. We generate an adversarial patch for each sample exclusively, and then compute l_1-norm of gradients in each iteration to analyze the relation between gradients and inner products.

1) As shown in Fig. 3(a), according to the convergence variance, we divide the general optimization process into three stages, namely *Stage 1: leaving the original cluster* (before point A), *Stage 2: moving to the target cluster* (between points A and B), and *Stage 3: approaching the anchor point* (after point B). We find that in Stage 1 the sample moves slowly, and once it reaches to somewhere (*i.e.*, point A) that is far enough from the original cluster, it will run fast towards to the target cluster since Eq. (5) can help the sample find the right direction. Finally, after entering the target cluster (*i.e.*, point B), the sample will gradually approach the target anchor code.

2) As shown in Fig. 3(b), the l_1-norm of gradients varies greatly at each stage. Specifically, the gradient norms of *moving to target cluster* stage remarkably outstrip other two stages. We find that within a certain range, the closer the sample is to the target anchor in Hamming space, the faster the inner product rises. In other words, the deep hashing model seems to have a tendency to drag sample to a closer cluster in Hamming space. We find this phenomenon is generally common.

Based on the above two observations, we believe that it takes different efforts for different samples to leave the original cluster. The earlier the sample leaves Stage 1, the faster it will converge to the target cluster. Therefore, we propose the product-based weighted gradient aggregation strategy, by assigning different weights to each intermediate gradients, to push the samples to move towards Stage 2 as fast as possible.

Gradient Aggregation. By taking the final loss function of Eq. (8) into consideration, we now present the details of how gradient aggregation is used to accelerate the optimization process of Eq. (8), as depicted in Fig. 4.

In our design, when generating the adversarial patch δ, we sample m images from $\boldsymbol{X}^s$ and construct the training data set with B mini-batches $\boldsymbol{X}_{train} = \boldsymbol{X}_1 \cup \boldsymbol{X}_2 ... \cup \boldsymbol{X}_B$, each of which contains $U = m/B$ images. The rest images of $\boldsymbol{X}^s$ are regarded as the testing data set to evaluate the generalization ability of δ. Let $\boldsymbol{x}_u$ denote the u-th sample of $\boldsymbol{X}_b$. For each mini-batch of samples, they all will pass through the pipeline of framework simultaneously for I iterations. We use i to indicate the i-th iteration.

Initially, for each mini-batch $\boldsymbol{X}_b$, we obtain an adversarial set $\boldsymbol{X}'_b = \{\boldsymbol{x}'_u\}_{u=1}^U$ using Eq. (2). It is easy to obtain the hash code $\boldsymbol{c}_u$ of $\boldsymbol{x}'_b$, as well as the corresponding hash layer outputs $H(\boldsymbol{X}'_b)$. We compute the inner product between each hash code $\boldsymbol{c}_u$ and the anchor code $\boldsymbol{h}_t$ as follows:

$$P_u = \boldsymbol{c}_u \cdot \boldsymbol{h}_t \tag{10}$$

Then, we use $H(\boldsymbol{X}'_b)$ to compute the loss with Eq. (8), and use back propagation to compute the gradient of $\boldsymbol{x}'_u$ as:

$$\nabla \mathcal{L}_u = \frac{\partial \mathcal{L}}{\partial \boldsymbol{x}'_u} \tag{11}$$

Additionally, in order to let the optimization of patch influenced more by the samples at *moving to target cluster* stage, we aggregate the gradients discriminatively according to their associated inner products with the anchor code. In other words, a gradient $\nabla \mathcal{L}_u$ associated with a larger product P_u will be weighted more for updating the adversarial patch. Concretely, at i-th iteration, we update an aggregated gradient according to

$$\begin{aligned} \bar{\nabla} &= \frac{\sum_{u=1}^U W_u \times \nabla \mathcal{L}_u}{\sum_{u=1}^U W_u} \\ W_u &= \begin{cases} P_u + \beta, & \text{if } P_u \le T \\ P_u - \gamma, & \text{otherwise} \end{cases} \end{aligned} \tag{12}$$

where β is a hyper-parameter to bias the weights and also keep all the weights positive. Furthermore, we introduce a threshold T to find the samples that have already entered the target cluster. Accordingly, we lower the weights of those samples by another hyper-parameter γ. This way, the direction of aggregated gradient will be affected more by the samples within *moving to target cluster* phase.

Recurrent Patch Update. Once we obtain the aggregated gradient $\bar{\nabla}$, we update the patch vector with momentum [12] as follows:

$$\begin{aligned} g_b^i &= \mu \cdot g_b^{i-1} + \frac{\bar{\nabla}}{\|\bar{\nabla}\|_1} \\ \delta_b^i &= \delta_b^{i-1} + \sigma \cdot \text{sign}\left(g_b^i\right) \end{aligned} \tag{13}$$

where g_b^i is the momentum of the i-th iteration of b-th mini-batch and σ is the learning rate. Note that, the update operation above is recurrently executed during the entire I iterations for a mini-batch of samples.

Last but not least, it is observed that the hard thresholding masking operator M in Eq. (2) will bring negative consequence to the attack. On the one hand, because of the hard thresholding, the patch

Algorithm 1: Set-to-set Targeted Adversarial Patch Attack

Input: training set $\boldsymbol{X}_{train} = \boldsymbol{X}_1 \cup \boldsymbol{X}_2 ... \cup \boldsymbol{X}_B$; target model $F(\cdot)$; target anchor code $\boldsymbol{h}_t$; location mask M; iteration number I; patch aggregation number p.

Output: set-to-set adversarial patch $\delta = \delta_B$.

Initiate a patch δ_0 with a random noise;
Initiate a empty patch list $\boldsymbol{\Delta}$;
save δ_0 in $\boldsymbol{\Delta}$;
for *b=1, ... ,B* **do**
  if *patch number of* $\boldsymbol{\Delta} \geq p$ **then**
    Obtain the initial patch δ_b^0 with Eq. (14);
  else
    Set the last patch δ_{b-1} in $\boldsymbol{\Delta}$ as the initial patch δ_b^0 ;
  for *i=1, ... ,I* **do**
    Generate examples $\boldsymbol{X}_b' = (1 - M) \odot \boldsymbol{X}_b + M \odot \delta_b^{i-1}$;
    Compute the hash codes $F(\boldsymbol{X}_b')$;
    Compute the inner product P_u with Eq. (10);
    Compute the loss with Eq. (8);
    Obtain a list of gradients with Eq. (11);
    Aggregate gradients with Eq. (12) ;
    Update the patch δ_b^i with Eq. (13) ;
    Clip δ_b^i to $[0, 255]$;
  Save δ_b to Δ;

generated from a single mini-batch $\boldsymbol{X}_b'$, which can achieve high inner product for all the samples in the current mini-batch, will have relatively low applicability for unseen mini-batches. On the other hand, if we simply apply the patch generated from the previous mini-batch as the initial patch for the following one during training, it may get updated thoroughly by masking and thus hard to achieve a strong generalization ability. A general remedy to this problem is to add another threshold condition to enforce generalization [26], but this will inevitably make the attack inefficient. It is empirically found that averaging a few patches trained from the previous mini-patches will stabilize the overall attack very well. Specifically, we use a patch list Δ to save p adversarial patches generated in the previous mini-batches, and then set the initial patch for the current mini-batch as:

$$\delta_b^0 = \frac{1}{p}(\delta_{b-p} + \delta_{b-p+1} + ... + \delta_{b-1}) \tag{14}$$

where the δ_b^0 indicates the initial patch for b-th mini-batch. In summary, the detailed algorithm of AdvHash is provided in Alg. 1.

4 EXPERIMENTS

4.1 Experimental Setting

Datasets. We evaluate AdvHash on *ImageNet* [33] and *NUS-WIDE* [9]. For ImageNet, following [2, 6], we build a subset that has 130K images with 100 classes. For NUS-WIDE, following [47], we use a subset that has 195,834 images with 21 concepts. For training target models, we sample 100 and 500 images per class as the training set, 50 and 100 per class as the test set on ImageNet and NUS-WIDE, respectively.

Evaluation metrics. We use *mean average precision* (*m*AP) as an evaluation metric. Similar to [2], we adopt four variants of *m*AP as follows:

- org-*m*AP (O): take clean images as input and original label as the referenced label ;
- t-*m*AP (T): take clean images as input and target label as the referenced label;
- adv-org-*m*AP (AO): take perturbed images as input and original label as the referenced label;
- adv-t-*m*AP (AT): take perturbed images as input and target label as the referenced label.

Note that we calculate these *m*APs on top 1,000 retrieved results for ImageNet and top 5,000 for NUS-WIDE, as did in [2].

Target models. We choose different DNNs (*i.e.*, ResNet50 [16] and VGG16 [35]) as the backbone for two state-of-the-art deep hashing methods (*i.e.*, CSQ [44] and HashNet [6]). HashNet is a widely used structure and serves as the target framework in recent studies about attacking deep hashing such as [39] [43], and CSQ is designed to generate highly clustered deep hashing models.

Table 2: Source class to target class mappings for the datasets ImageNet and NUS-WIDE

Dataset	Mapping	Source class	Target class
ImageNet	$\mathcal{M}_1$	Cock	Clog
	$\mathcal{M}_2$	Hermit crab	Triumphal arch
	$\mathcal{M}_3$	Sea lion	Suspension bridge
	$\mathcal{M}_4$	Malinois	Screw
	$\mathcal{M}_5$	Binoculars	Water buffalo
	$\mathcal{M}_6$	Clothes iron	Fire truck
	$\mathcal{M}_7$	Pinwheel	Porcupine
	$\mathcal{M}_8$	Consomme	Car mirror
	$\mathcal{M}_9$	Jackfruit	European fire salamander
	$\mathcal{M}_{10}$	Earthstar	Hautbois
NUS-WIDE	$\mathcal{M}_{11}$	plant	cloud
	$\mathcal{M}_{12}$	street, building	people
	$\mathcal{M}_{13}$	cloud, sky	water
	$\mathcal{M}_{14}$	cloud, sky, grass	animals
	$\mathcal{M}_{15}$	cloud, sky, water, blue, sea	flower

4.2 Attack Performance

Implementation Details. As shown in Table 2, we evaluate AdvHash by randomly selecting 10 and 5 source-to-target mappings for ImageNet and NUS-WIDE, respectively. For each mapping on ImageNet, we sample 50 and 500 images from the source set $\boldsymbol{X}^s$ as training set and test set respectively. For each mapping on NUS-WIDE, we sample 100 images and 800 images as training set and test set respectively. The anchor code $\boldsymbol{h}_t$ is obtained by sampling 50 and 100 images from target class image set $\boldsymbol{X}^t$ for ImageNet and NUS-WIDE respectively.

For each patch generation, we set the epoch number to 5. In each epoch, the training set is divided into 5 mini-batches. We set the noise percentage of each sample to be 0.03. When training each mini-batch, we set the learning rate σ as 1 and the number of iterations as 1,000. Note that, the org-*m*AP and t-*m*AP are evaluated on the test set, and adv-org-*m*AP and adv-t-*m*AP are evaluated on adversarial examples.

Analysis. The detailed *m*AP results for each mapping are shown in Table 3. Firstly, the sharp *m*AP drop (from O to AO) and t-*m*AP rise (from T to AT) reveal that the perturbed samples have successfully

Table 3: Targeted attack performance for each mapping on CSQ [44] and HashNet [6]

Dataset	Mapping	CSQ															
		ResNet50								VGG16							
		32 bits				64 bits				32 bits				64 bits			
		O	T	AO	AT	O	T	AO	AT	O	T	AO	AT	O	T	AO	AT
ImageNet	$\mathcal{M}_1$	98.57	0.00	2.06	93.21	98.75	0.00	10.17	88.99	97.97	0.00	1.99	92.34	98.57	0.00	2.06	93.21
	$\mathcal{M}_2$	92.85	0.00	0.35	98.34	91.16	0.00	7.85	91.38	89.18	0.01	2.05	95.76	90.98	0.00	0.79	98.76
	$\mathcal{M}_3$	92.60	0.06	0.98	99.04	95.06	0.27	4.06	95.39	90.51	0.27	0.38	98.57	92.42	0.20	0.19	98.77
	$\mathcal{M}_4$	92.51	0.00	0.00	99.33	94.74	0.00	4.11	95.97	89.87	0.00	0.16	97.61	91.40	0.00	0.00	98.02
	$\mathcal{M}_5$	87.46	0.00	0.85	99.04	88.06	0.01	2.55	96.90	78.66	0.01	18.94	60.60	81.70	0.00	0.55	99.13
	$\mathcal{M}_6$	84.26	0.05	0.96	97.22	86.74	0.05	11.03	90.40	78.72	0.04	0.21	98.81	82.70	0.02	0.56	98.96
	$\mathcal{M}_7$	94.44	0.20	2.76	95.56	95.98	0.00	9.86	89.79	90.24	0.00	0.53	99.01	91.05	0.00	1.37	97.74
	$\mathcal{M}_8$	91.24	0.00	0.39	99.36	90.39	0.00	6.72	93.18	89.57	0.16	5.73	50.12	88.00	0.16	23.72	5.57
	$\mathcal{M}_9$	95.28	0.00	0.22	95.04	97.31	0.00	9.34	87.21	96.11	0.00	0.39	97.40	96.68	0.00	0.98	95.32
	$\mathcal{M}_{10}$	94.23	0.00	0.00	99.66	95.10	0.00	0.00	99.43	91.77	0.01	0.00	96.71	93.10	0.00	0.19	98.29
	AVG	**92.34**	**0.03**	**0.86**	**97.58**	**93.33**	**0.03**	**6.57**	**92.86**	**89.26**	**0.05**	**3.04**	**88.69**	**90.66**	**0.04**	**3.04**	**88.38**
NUS-WIDE	$\mathcal{M}_{11}$	42.08	20.58	3.29	88.40	41.48	19.24	3.03	87.67	39.49	18.27	2.71	81.13	40.20	17.63	1.77	89.35
	$\mathcal{M}_{12}$	77.18	8.89	11.14	92.19	77.73	8.09	3.79	97.67	76.15	8.62	3.07	98.37	73.76	8.99	2.12	98.32
	$\mathcal{M}_{13}$	92.65	12.86	52.58	77.45	93.10	11.52	49.28	77.97	90.54	13.68	21.22	74.19	92.04	12.98	15.61	76.22
	$\mathcal{M}_{14}$	96.50	2.61	6.50	98.72	97.13	2.52	7.25	99.18	96.18	2.95	9.91	98.18	96.83	3.09	7.57	98.10
	$\mathcal{M}_{15}$	98.85	0.23	37.95	62.17	99.14	0.28	20.53	81.50	98.88	0.28	16.10	76.82	99.19	0.25	15.69	73.76
	AVG	**81.45**	**9.03**	**22.29**	**83.79**	**81.72**	**8.33**	**16.78**	**88.80**	**80.25**	**8.76**	**10.60**	**85.74**	**80.40**	**8.59**	**8.55**	**87.15**
Dataset	Mapping	HashNet															
ImageNet	$\mathcal{M}_1$	97.36	0.00	0.27	70.27	48.89	0.00	0.20	64.18	94.64	0.07	0.07	74.87	98.07	0.00	0.12	36.02
	$\mathcal{M}_2$	90.03	0.00	1.44	76.41	47.69	0.00	0.66	97.82	85.72	0.00	2.08	93.00	86.03	0.00	0.19	99.24
	$\mathcal{M}_3$	93.43	0.22	1.38	97.77	92.20	0.21	1.54	92.15	82.60	0.26	1.12	98.35	91.58	0.01	0.18	99.40
	$\mathcal{M}_4$	92.35	0.00	0.00	99.65	86.25	0.00	0.00	99.54	93.01	0.00	0.00	97.87	89.48	0.00	0.00	99.49
	$\mathcal{M}_5$	80.11	0.03	1.54	84.25	83.19	0.00	0.49	60.53	67.53	0.23	0.19	96.66	80.91	0.04	0.00	99.48
	$\mathcal{M}_6$	79.19	0.23	8.49	66.97	86.43	0.00	26.84	43.97	38.09	0.03	0.32	94.57	72.57	0.02	0.01	99.33
	$\mathcal{M}_7$	95.59	0.00	4.73	94.87	96.46	0.00	3.24	80.97	87.15	0.01	0.19	98.07	93.33	0.00	0.19	99.57
	$\mathcal{M}_8$	74.33	0.00	0.00	87.73	76.31	0.00	0.00	99.05	86.85	0.04	0.66	95.78	87.66	0.00	0.00	99.89
	$\mathcal{M}_9$	25.76	0.01	0.23	74.16	86.54	0.00	0.98	45.71	73.12	0.00	1.25	32.80	96.12	0.00	0.00	19.11
	$\mathcal{M}_{10}$	47.08	0.01	0.00	71.74	65.06	0.00	0.00	40.87	93.14	0.00	0.00	97.26	95.02	0.01	0.00	99.55
	AVG	**77.52**	**0.05**	**1.81**	**82.38**	**76.90**	**0.02**	**3.40**	**72.48**	**80.19**	**0.06**	**0.59**	**87.92**	**89.08**	**0.01**	**0.07**	**85.11**
NUS-WIDE	$\mathcal{M}_{11}$	39.41	15.06	3.53	84.70	38.93	14.59	4.24	83.25	40.06	15.57	1.10	82.95	39.40	15.38	0.86	87.92
	$\mathcal{M}_{12}$	68.16	9.37	4.75	96.09	76.46	8.08	2.41	98.04	68.37	10.40	2.34	98.45	69.82	7.55	2.49	99.34
	$\mathcal{M}_{13}$	89.40	16.08	37.41	80.95	91.04	15.23	32.19	86.02	90.14	16.12	15.88	82.84	93.39	13.89	11.95	86.15
	$\mathcal{M}_{14}$	95.41	4.65	14.21	98.76	97.08	2.72	15.62	98.87	95.65	3.90	7.33	97.89	97.65	3.01	5.61	99.13
	$\mathcal{M}_{15}$	98.64	0.16	23.78	69.17	99.23	0.28	40.01	64.34	98.07	0.21	9.94	77.14	99.43	0.17	6.94	87.38
	AVG	**78.20**	**9.06**	**16.74**	**85.93**	**80.55**	**8.18**	**18.89**	**86.10**	**78.46**	**9.24**	**7.32**	**87.85**	**79.94**	**8.00**	**5.57**	**91.98**

left the original clusters and enter the target clusters in Hamming space. Secondly, we can see that among the 120 attack settings, 93.33% of them have an AT above 60%, 60.90% of them can achieve impressive attack performance with AT above 90%, 18.3% of them have an outstanding AT above 99%, regardless of hash code lengths, deep hashing methods, data sets, and backbones. Fig. 5 gives an illustrative example of targeted attack result of mapping $\mathcal{M}_3$.

Additionally, we notice that the overall attack performance on CSQ is better than on HashNet, especially for the single-label dataset ImageNet. We believe this is because CSQ is more clustered than HashNet, which further verify our analysis on set-to-point optimization in Sec. 3.2.

4.3 Comparison Study

Implementation Details. In this section, we compare AdvHash with DHTA [2], which is the most related work with ours. Although [39] proposed a targeted attack on deep hashing as well, it focuses on improving the transferability of adversarial examples rather than the attack success rate.

We select 3 mappings for comparison, and reproduce DHTA using the same training samples as ours to obtain 50 adversarial examples with 50 entire different adversarial noise for each mapping respectively. The iterations for each sample is also set to 1,000. To ensure a fair comparison, we use the adversarial patch trained with AdvHash for one epoch to compare with DHTA. Specifically, the total iterations of ours is the product of mini-batch size, iterations for each mini-batch, mini-batch number and epoch number, *i.e.*, $10 \times 1000 \times 5 \times 1 = 50000$. The total iterations for DHTA is the product of iterations for each sample and the sample number, *i.e.*, $1000 \times 50 = 50000$. We evaluate AdvHash on the training data set (denoted as AdvHash-train) and the testing data set (denoted as AdvHash-test).

Table 4: Comparison with DHTA [2]

Mapping	Samples	CSQ				HashNet			
		ResNet50		VGG16		ResNet50		VGG16	
		32	64	32	64	32	64	32	64
$\mathcal{M}_4$	DHTA	99.49	99.77	80.94	36.82	99.68	99.25	6.36	36.82
	AdvHash-train	98.33	89.3	97.8	34.97	99.63	95.85	95.86	99.5
	AdvHash-test	97.14	94.28	97.21	29.72	99.04	95.61	97.42	99.5
$\mathcal{M}_6$	DHTA	97.73	99.29	78.71	76.36	92.87	84.09	96.16	97.55
	AdvHash-train	97.79	81.3	99.03	99.54	43.38	31.06	94.62	99.6
	AdvHash-test	95.87	83.27	98.64	99.15	33.15	36.56	92.64	98.73
$\mathcal{M}_{10}$	DHTA	99.66	99.44	82.5	54.68	69.06	28.94	97.2	99.58
	AdvHash-train	97.66	97.41	80.59	98.46	71.32	69.99	97.06	99.59
	AdvHash-test	98.67	99.38	79.94	97.33	70.6	70.8	96.85	99.57

Analysis. The adv-t-*m*AP results over 50 training samples and 500 test samples are shown in Table 4. Firstly, the adv-t-mAP of AdvHash-train and AdvHash-test do not vary much in each case, especially when they have already achieved a high performance, which means that the generated patches generalize well to the unseen test samples. This confirms the effectiveness of weighted gradient aggregation and recurrent patch update strategies. Secondly, AdvHash can compete or even outperform DHTA in some cases. Note that DHTA is an image-specific attack, thus is easier to achieve good attack performance. Nevertheless, AdvHash can

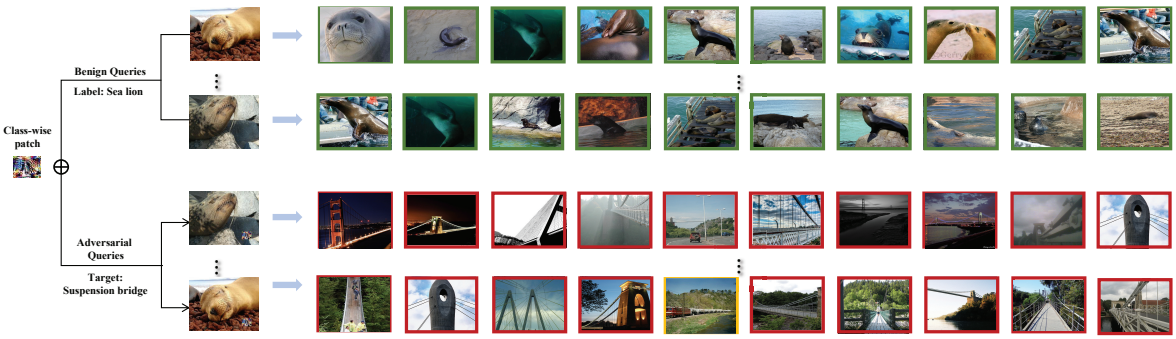
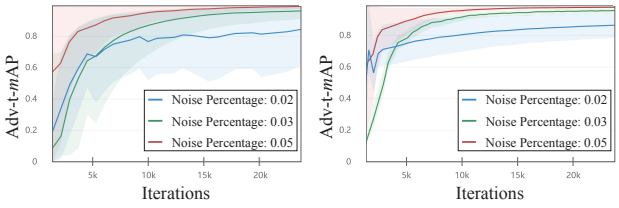


Figure 5: An illustration of targeted attack result. Retrieved objects with top-10 similarity are shown. The green box indicates source objects, red box indicates target objects, and yellow box indicates irrelevant objects.



(a) $\mathcal{M}_4$ (b) $\mathcal{M}_{12}$

Figure 6: Attack performance with different noise percentage on mappings $\mathcal{M}_4$ and $\mathcal{M}_{12}$

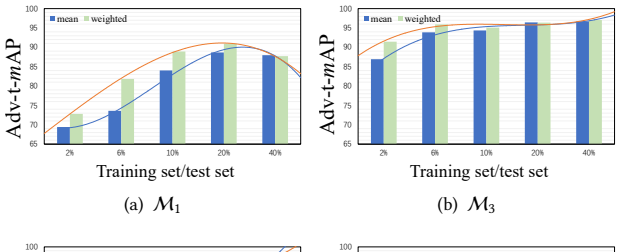
achieve both high attack performance and strong generalization ability simultaneously.

4.4 Ablation Study

Implementation Details. We conduct two ablation studies. First, we investigate the effect of patch percentage on the attack performance of the mappings $\mathcal{M}_4$ and $\mathcal{M}_{12}$, where the patch percentage is set to 0.02, 0.03, and 0.05, respectively, while all the other experiments remain unchanged as stated in Sec. 4.2. We then investigate the efficiency of our proposed gradient aggregation strategy. We compare our product-based weighted aggregation strategy with mean aggregation (*i.e.*, assigning the same weight for each sample) on 4 mappings $\mathcal{M}_1$, $\mathcal{M}_3$, $\mathcal{M}_5$, and $\mathcal{M}_7$ under 5 different training/test ratios 2%, 6%, 10%, 20%, and 40% respectively. Concretely, we sample 500 images as the test set, and 5 different training sets (*i.e.*, 10, 30, 50, 100 and 200 images) for each mapping. In this experiment, we choose 64-bits CSQ-ResNet50 as the target model.

Analysis. Fig. 6 shows the AT results of the first ablation experiment, where the lines represent the average AT from all target models in each setting, and the colored shaded area stands for the range of minimum and maximum values of AT with different target models. Therefore, the smaller the colored shaded area, the more uniform the attack success rate is. From Fig. 6 we find a larger noise percentage can lead to a higher attack success rate with less variance towards different target models. In addition, the patch percentage of 0.05 can achieve nearly 100% AT on each target model.

Fig. 7 shows the AT results of our second ablation experiment. The red curves represent the AT results of product-based weighted gradient aggregation under different training/test ratios, while the blue curves represent that of mean gradient aggregation. We can



(a) $\mathcal{M}_1$ (b) $\mathcal{M}_3$

(c) $\mathcal{M}_5$ (d) $\mathcal{M}_7$

Figure 7: Attack performance with different training set proportions and different gradient aggregation methods.

see that the weighted gradient aggregation method can get a better grade in most cases when the training/test ratios are set in a reasonable interval. Specifically, when the number of training set images is small, the attack with weighted gradient aggregation can achieve a higher generalization ability than the mean aggregation approach.

5 CONCLUSION

In this paper, we propose the first targeted mismatch attack scheme on deep hashing with adversarial patch. We first formulate a set-to-set problem and then transform it into a set-to-point optimization. We then propose a product-based weighted gradient aggregation strategy in order to generate a stable adversarial patch more efficiently. Our extensive experiments verify that AdvHash is highly effective at attacking state-of-the-art deep hashing schemes.

ACKNOWLEDGMENTS

Shengshan's work is supported in part by the National Natural Science Foundation of China (Grant Nos. 62002126, U20A20177), and Fundamental Research Funds for the Central Universities (Grant No. 5003129001). Leo's work is supported in part by the National Natural Science Foundation of China (Grant No. 61702221). Minghui is the corresponding author.

REFERENCES

[1] Artem Babenko and Victor Lempitsky. 2015. Aggregating local deep features for image retrieval. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV'15)*. 1269–1277.

[2] Jiawang Bai, Bin Chen, Yiming Li, Dongxian Wu, Weiwei Guo, Shu-tao Xia, and En-hui Yang. 2020. Targeted attack for deep hashing based retrieval. In *Proceedings of the 16th European Conference on Computer Vision (ECCV'20)*. Springer, 618–634.

[3] Philipp Benz, Chaoning Zhang, Tooba Imtiaz, and In So Kweon. 2020. Double targeted universal adversarial perturbations. In *Proceedings of the 15th Asian Conference on Computer Vision (ACCV'20)*. Springer, 284–300.

[4] Tom B. Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. 2017. Adversarial patch. *arXiv preprint arXiv:1712.09665* (2017).

[5] Yue Cao, Mingsheng Long, Bin Liu, and Jianmin Wang. 2018. Deep cauchy hashing for hamming space retrieval. In *Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'18)*. 1229–1237.

[6] Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Philip S. Yu. 2017. Hashnet: Deep learning to hash by continuation. In *Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV'17)*. 5608–5617.

[7] Mingyang Chen, Junda Lu, Yi Wang, Jianbin Qin, and Wei Wang. 2021. DAIR: A query-efficient decision-based attack on image retrieval systems. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'21)*. 1064–1073.

[8] Zhixiang Chen, Xin Yuan, Jiwen Lu, Qi Tian, and Jie Zhou. 2018. Deep hashing via discrepancy minimization. In *Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'18)*. 6838–6847.

[9] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. 2009. Nus-wide: a real-world web image database from national university of singapore. In *Proceedings of the 8th ACM International Conference on Image and Video Retrieval (CIVR'09)*. 1–9.

[10] Cheng Deng, Zhaojia Chen, Xianglong Liu, Xinbo Gao, and Dacheng Tao. 2018. Triplet-based deep hashing network for cross-modal retrieval. *IEEE Transactions on Image Processing* 27, 8 (2018), 3893–3903.

[11] Brian Dolhansky and Cristian Canton Ferrer. 2020. Adversarial collision attacks on image hashing functions. *arXiv preprint arXiv:2011.09473* (2020).

[12] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. 2018. Boosting adversarial attacks with momentum. In *Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'18)*. 9185–9193.

[13] Craig M. Files and Marek A. Perkowski. 1998. Multi-Valued Functional Decomposition as a Machine Learning Method. In *Proceedings of the 28th IEEE International Symposium on Multiple-Valued Logic (ISMVL 1998)*. IEEE Computer Society, 173–179.

[14] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).

[15] Jamie Hayes and George Danezis. 2017. Learning universal adversarial perturbations with generative models. *arXiv preprint arXiv:1708.05207* (2017).

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'16)*. 770–778.

[17] Di Hu, Feiping Nie, and Xuelong Li. 2018. Deep binary reconstruction for cross-modal hashing. *IEEE Transactions on Multimedia* 21, 4 (2018), 973–985.

[18] Danny Karmon, Daniel Zoran, and Yoav Goldberg. 2018. Lavan: Localized and visible adversarial noise. In *Proceedings of the 35th International Conference on Machine Learning (ICML'18)*. 2507–2515.

[19] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2016. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236* (2016).

[20] Hanjiang Lai, Yan Pan, Ye Liu, and Shuicheng Yan. 2015. Simultaneous feature learning and hash coding with deep neural networks. In *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'15)*. 3270–3278.

[21] Jie Li, Rongrong Ji, Hong Liu, Xiaopeng Hong, Yue Gao, and Qi Tian. 2019. Universal perturbation attack against image retrieval. In *Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV'19)*. 4899–4908.

[22] Aishan Liu, Xianglong Liu, Jiaxin Fan, Yuqing Ma, Anlan Zhang, Huiyuan Xie, and Dacheng Tao. 2019. Perceptual-sensitive gan for generating adversarial patches. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI'19)*, Vol. 33. 1028–1035.

[23] Aishan Liu, Jiakai Wang, Xianglong Liu, Bowen Cao, Chongzhi Zhang, and Hang Yu. 2020. Bias-based universal adversarial patch attack for automatic check-out. In *Proceedings of the 16th European Conference on Computer Vision (ECCV'20)*. Springer, 395–410.

[24] Hong Liu, Rongrong Ji, Jie Li, Baochang Zhang, Yue Gao, Yongjian Wu, and Feiyue Huang. 2019. Universal adversarial perturbation via prior driven uncertainty approximation. In *Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV'19)*. 2941–2949.

[25] Haomiao Liu, Ruiping Wang, Shiguang Shan, and Xilin Chen. 2016. Deep supervised hashing for fast image retrieval. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'16)*. 2064–2072.

[26] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. 2018. Trojaning attack on neural networks. In *Proceedings of the 2018 Annual Symposium on Network and Distributed System Security (NDSS'18)*. 1–15.

[27] Zhuoran Liu, Zhengyu Zhao, and Martha Larson. 2019. Who's afraid of adversarial queries? The impact of image modifications on content-based image retrieval. In *Proceedings of the 2019 on International Conference on Multimedia Retrieval (ICMR'19)*. 306–314.

[28] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. 2017. Universal adversarial perturbations. In *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'17)*. 1765–1773.

[29] Konda Reddy Mopuri, Utsav Garg, and R. Venkatesh Babu. 2017. Fast feature fool: A data independent approach to universal adversarial perturbations. *arXiv preprint arXiv:1707.05572* (2017).

[30] Konda Reddy Mopuri, Utkarsh Ojha, Utsav Garg, and R. Venkatesh Babu. 2018. NAG: Network for adversary generation. In *Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'18)*. 742–751.

[31] Omid Poursaeed, Isay Katsman, Bicheng Gao, and Serge Belongie. 2018. Generative adversarial perturbations. In *Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'19)*. 4422–4431.

[32] Filip Radenović, Giorgos Tolias, and Ondřej Chum. 2018. Fine-tuning CNN image retrieval with no human annotation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41, 7 (2018), 1655–1668.

[33] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. 2015. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision* 115, 3 (2015), 211–252.

[34] Fumin Shen, Yan Xu, Li Liu, Yang Yang, Zi Huang, and Heng Tao Shen. 2018. Unsupervised deep hashing with similarity-adaptive and discrete optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40, 12 (2018), 3034–3044.

[35] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[36] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199* (2013).

[37] Giorgos Tolias, Filip Radenovic, and Ondrej Chum. 2019. Targeted mismatch adversarial attack: Query with a flower to retrieve the tower. In *Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV'19)*. 5037–5046.

[38] Giorgos Tolias, Ronan Sicre, and Hervé Jégou. 2015. Particular object retrieval with integral max-pooling of CNN activations. *arXiv preprint arXiv:1511.05879* (2015).

[39] Yanru Xiao and Cong Wang. 2021. You see what I want you to see: Exploring targeted black-box transferability attack for hash-based image retrieval systems. In *Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'21)*. 1934–1943.

[40] Yanru Xiao, Cong Wang, and Xing Gao. 2020. Evade deep image retrieval by stashing private images in the hash space. In *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'20)*. 9651–9660.

[41] Weilin Xu, David Evans, and Yanjun Qi. 2017. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155* (2017).

[42] Erkun Yang, Cheng Deng, Wei Liu, Xianglong Liu, Dacheng Tao, and Xinbo Gao. 2017. Pairwise relationship guided deep hashing for cross-modal retrieval. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI'17)*. 1618–1625.

[43] Erkun Yang, Tongliang Liu, Cheng Deng, and Dacheng Tao. 2018. Adversarial examples for hamming space search. *IEEE Transactions on Cybernetics* 50, 4 (2018), 1473–1484.

[44] Li Yuan, Tao Wang, Xiaopeng Zhang, Francis EH Tay, Zequn Jie, Wei Liu, and Jiashi Feng. 2020. Central similarity quantization for efficient image and video retrieval. In *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'20)*. 3083–3092.

[45] Guoping Zhao, Mingyu Zhang, Jiajun Liu, Yaxian Li, and Ji-Rong Wen. 2020. AP-GAN: Adversarial patch attack on content-based image retrieval systems. *GeoInformatica* (2020), 1–31.

[46] Guoping Zhao, Mingyu Zhang, Jiajun Liu, and Ji-Rong Wen. 2019. Unsupervised adversarial attacks on deep feature-based retrieval with GAN. *arXiv preprint arXiv:1907.05793* (2019).

[47] Han Zhu, Mingsheng Long, Jianmin Wang, and Yue Cao. 2016. Deep hashing network for efficient similarity retrieval. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI'16)*. 2415–2421.