Semantic-Aware Adversarial Training for Reliable Deep Hashing Retrieval

Xu Yuan, Zheng Zhang, *Senior Member, IEEE,* Xunguang Wang, Lin Wu

***Abstract*—Deep hashing has been intensively studied and successfully applied in large-scale image retrieval systems due to its efficiency and effectiveness. Recent studies have recognized that the existence of adversarial examples poses a security threat to deep hashing models, that is, adversarial vulnerability. Notably, it is challenging to efficiently distill reliable semantic representatives for deep hashing to guide adversarial learning, and thereby it hinders the enhancement of adversarial robustness of deep hashing-based retrieval models. Moreover, current researches on adversarial training for deep hashing are hard to be formalized into a unified *minimax* structure. In this paper, we explore Semantic-Aware Adversarial Training (SAAT) for improving the adversarial robustness of deep hashing models. Specifically, we conceive a discriminative mainstay features learning (DMFL) scheme to construct semantic representatives for guiding adversarial learning in deep hashing. Particularly, our DMFL with the strict theoretical guarantee is adaptively optimized in a discriminative learning manner, where both discriminative and semantic properties are jointly considered. Moreover, adversarial examples are fabricated by maximizing the Hamming distance between the hash codes of adversarial samples and mainstay features, the efficacy of which is validated in the adversarial attack trials. Further, we, *for the first time*, formulate the formalized adversarial training of deep hashing into a unified minimax optimization under the guidance of the generated mainstay codes. Extensive experiments on benchmark datasets show superb attack performance against the state-of-the-art algorithms, meanwhile, the proposed adversarial training can effectively eliminate adversarial perturbations for trustworthy deep hashing-based retrieval.**

***Index Terms*—Adversarial Attack, Adversarial Training, Trustworthy Deep Hashing, Similarity Retrieval**

I. Introduction

With the growing of large-scale multimedia data, approximate nearest neighbor (ANN) retrieval [1] has received much attention due to its outstanding balance capability of efficiency and effectiveness. Among various ANN methods, hashing [2] offers the prominent advantages of mapping high-dimensional data to compact binary codes with low costs in storage and computational complexity. Compared to shallow hashing methods, deep hashing [3]–[12] achieves superior performance by learning a nonlinear hashing function based on deep neural networks in an end-to-end manner.

Recent studies [13]–[17] have demonstrated that deep hashing-based retrieval models are vulnerable to adversarial examples. Generally, the adversarial examples [18] are crafted by adding imperceptible perturbations to original samples, yet can greatly confuse deep hashing models to retrieve incorrect results. Undoubtedly, such malicious attacks [19], [20] pose serious security risks to deep hashing-based retrieval systems. For example, in a deep hashing-based face recognition system, adversarial examples can mislead the system and retrieve a non-matching person's face, and thus successfully invade the system. Consequently, it becomes imperative to investigate adversarial attacks on deep hashing models and develop robust defense strategies.

X. Yuan, Z. Zhang, X. Wang are with School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen 518055, China. (E-mail: xuyuan127@gmail.com, darrenzz219@gmail.com, xunguangwang@gmail.com)

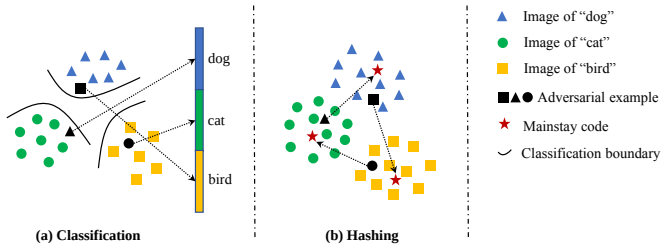


Fig. 1. The comparison between classification and semantic similarity-preserving hashing on adversarial learning. In the output space of classification, an adversarial attack with the guidance of labels only needs to make the adversarial samples across the decision boundary. However, an adversarial attack on deep hashing is more challenging because the adversarial samples are expected to blend into the clusters in the embedding space, but there is not an explicit signal to supervise the process. The same problem exists for adversarial training. For classification, adversarial training just maximizes probabilities of adversarial samples on true labels, which is infeasible on hashing due to the lack of explicit supervision signals. Hence, deep hashing needs reliable and discriminative semantic representatives (*i.e.*, mainstay) to represent the image semantics for adversarial attacks and defense.

The current researches working with adversarial samples mostly focus on classification tasks [21], with the exception of a few works [13]–[17] studying the adversarial learning on deep hashing-based retrieval. However, these two tasks are quite different in terms of adversarial learning, as shown in Fig. 1. The former learns to classify each sample to its target class under the supervision of class labels. The latter is a similarity-preserving task that maps high-dimensional data to Hamming space and maintains the similarity between the data. It is clear that we can generate adversarial examples that can deceive and out-trick the classification system by minimally recomposing classification boundaries derived from semantic labels. However, for the semantic similarity-preserving hashing-based retrieval task, *there are no explicit representatives (e.g., label) to lead adversarial attacks or defenses*, resulting in more challenges compared to the classification one. Moreover, the *minimax optimization-based adversarial training,* which has shown success in classification, does not seamlessly transfer

to deep hashing due to the absence of semantic representations. Therefore, it is challenging to address the following two problems:

1) How could we initialize the optimal and global semantic representatives for adversarial learning in deep semantic-preserving hashing networks?
2) How could we derive a unified minimax formulation for deep hashing to conduct robust adversarial training?

For the first question, some adversarial attack works [13], [14] heuristically select the hash code of a single sample as each representative for generating its adversarial example, while some others [14], [17] employ a set of hash codes from multiple relevant samples. Nevertheless, these methods are not reliable and fail to capture globally discriminative semantics, yielding low-quality semantic representatives. Another line of work [15], [22] adopts an auxiliary network to learn representative hash codes of label encoding for targeted attacks. Typically, such an auxiliary prototype network is a parameter-sensitive scheme that lacks theoretical guarantees, so the whole adversarial generation is poorly generalized over different datasets and hashing networks. In a nutshell, these works are less efficient to generate high-quality semantic representatives for adversarial deep hashing.

For the second question, there is only one work [22] on adversarial defense for deep hashing that simply reduces the distance between adversarial examples and the benign samples in Hamming space. However, this method is hard to enable robust adversarial training because it is not a standard minimax adversarial optimization problem shown in [23], composing of an *inner maximization* problem and an *outer minimization* problem. Notably, such a unified learning framework cannot hold in regular deep hashing-based retrieval networks unless preferable semantic representatives are provided.

To overcome the aforementioned issues, this paper constructs discriminative semantic representatives (dubbed *mainstay codes*) for adversarial learning of deep hashing and further proposes Semantic-Aware Adversarial Training (SAAT) with a formalized minimax framework to strengthen the adversarial robustness of deep hashing models. Intuitively, we rethink the characteristics of semantic similarity-preserving hashing-based retrieval tasks. Different from classification, the purpose of retrieval is to return top-n relevant objects instead of one result. Hence, the optimal semantic representative of given sample $\boldsymbol{x}$ in the retrieval task should preserve both similarities with all semantically relevant samples (positives) and dissimilarity with all semantically irrelevant ones (negatives). From this viewpoint, we argue that the semantic representative for adversarial deep hashing is expected to have a minimum Hamming distance from all positives yet a maximum distance from all negatives.

Directly optimizing the above problem is intractable due to the infeasibility of utilizing gradient descent in discrete Hamming space and the costly computational expense incurred by numerous positive and negative samples. Owing to the binarization of hash codes, *we allow an efficient approach called Discriminative Mainstay Features Learning (DMFL), which offers rigorous theoretical guarantees and enables the direct acquisition of the optimal solution to this problem*, *i.e.*, the proposed mainstay code. Then, we transform the adversarial attack (*e.g.*, non-targeted attack) on deep hashing into maximizing the Hamming distance between the hash code of the adversarial example and the mainstay code to efficiently generate the optimal adversarial example. Furthermore, based on the generated mainstay codes, we formulate the adversarial training on deep hashing as a minimax optimization problem, *i.e.*, a unified and standard adversarial training formula. Under the minimax paradigm, the inner maximization seeks an adversarial example $\boldsymbol{x}'$ of a given data $\boldsymbol{x}$ whose hash code maximizes the Hamming distance from the mainstay code, and the outer minimization attempts to find model parameters so that the hash code of $\boldsymbol{x}'$ is close to the mainstay code to alleviate the effects caused by the adversarial perturbations. The overall framework is illustrated in Fig. 2. In summary, our main contributions can be summarized below:

- We propose a **S**emantic-**A**ware **A**dversarial **T**raining (**SAAT**) framework for optimizing reliable deep hashing models. To our best knowledge, *this is the first attempt* to formulate the formalized adversarial training of deep hashing into a unified minimax paradigm guided by well-designed globally optimal semantic representatives.
- A discriminative mainstay features learning (DMFL) scheme is well conceived to generate global semantic representatives (named *mainstay codes*) of deep hashing, which can efficiently guide adversarial learning of deep hashing networks. As a consequence, the produced mainstay codes can be simply adaptive to non-targeted and targeted adversarial attacks against deep hashing models.
- In the presence of the derived mainstay features, we formalize a well-designed adversarial training strategy under a *minimax* optimization for enhancing the adversarial robustness of deep hashing-based retrieval.
- Extensive experiments validate the proposed attack method's superiority over state-of-the-art attacks on deep hashing. Meanwhile, further experiments demonstrate that our SAAT can substantially aid deep hashing networks in resisting multiple adversarial attacks.

The remainder of this paper is organized as follows. The related works are reviewed in Section II. Section III describes the proposed semantic-aware adversarial training method, including the problem formulation, the generation of mainstay code, and the attack/defense algorithm. In Section IV, we conduct experiments to validate the superiority of our method in comparison to the state-of-the-art attack and defense models. Finally, Section V concludes this paper.

II. Related Work

A. Deep Hashing-based Retrieval

Hashing methods aim to learn a hash function to convert semantically similar samples into similar hash codes in Hamming space, which are widely used to accelerate ANN retrieval [1]. Particularly, deep hashing [24] utilizes deep neural networks as hash functions for feature extraction and binary code generation in an end-to-end manner, attaining superior retrieval performance.

Existing deep hashing can be broadly categorized into two primary streams: unsupervised and supervised deep hashing.

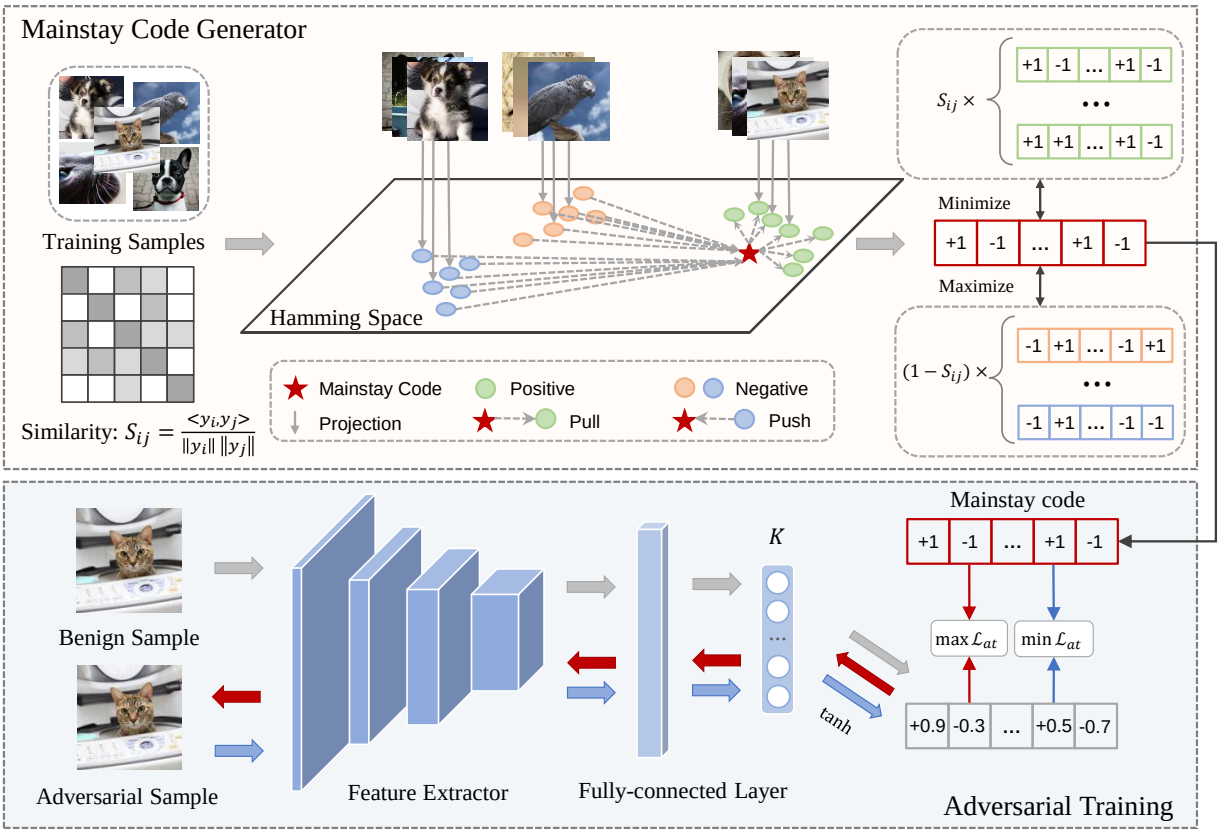


Fig. 2. The pipeline of the proposed Semantic-Aware Adversarial Training (SAAT) for deep hashing retrieval. The architecture is composed of two mechanisms: the generation branch of representative codes (*i.e.,* mainstay codes) and the minimax-based adversarial training branch. In mainstay code generation, all training samples are projected to the Hamming space to form their corresponding hash codes. Then, we build a mainstay code for each class by discriminative learning which pulls the mainstay code closer to the hash code of positives as well as pushes it away from other negatives. In adversarial training, we extend adversarial training of deep hashing to a mini-max framework *i.e.*, standard adversarial training. As illustrated, the gray, red, and blue arrows indicate forward, backward and forward propagation, respectively. The red arrow means constructing adversarial samples with the supervision of generated mainstay codes and the blue arrow represents inputting adversarial samples for adversarial training. Best viewed in color.

Unsupervised deep hashing methods [25], [26] usually involve learning hash functions by mining inherent structural or distribution-relevance similarities in the samples without using any semantic label. By contrast, supervised deep hashing methods use semantic labels or relevant information as supervisory signals to overcome the semantic gap dilemma [27], which can yield more precise performance than unsupervised ones. As the first deep hashing algorithm, CNNH [3] consists of two independent stages, *i.e.*, designing approximate hash codes of training data and learning feature representation through DNN. Recent hashing methods [4], [5], [7]–[12] focused on the design of end-to-end strategies and loss functions to improve the efficacy of hashing learning. For example, DPSH [4] integrated image representation and hash coding in a unified framework and adopted a pairwise loss to preserve the semantic similarity between data objects. HashNet [5] proposed a continuous scale strategy to tackle the optimization problem in discrete Hamming space, and alleviated the data imbalance by a weighted pairwise loss. CSQ [7] and DPN [9] are devoted to finding class-wise hash representatives (centers) that can provide global similarity for hashing learning. OrthoHash [10] proposed a novel single learning objective that maximizes the cosine similarity between the continuous codes and their corresponding binary orthogonal codes. Besides, HSWD [11] presented a Sliced-Wasserstein-based distributional distance to achieve low quantization error and coding balance.

B. Adversarial Attack

As the vulnerability of deep learning models has been noticed by Szegedy *et al.* [18], numerous studies have explored the model's robustness and proposed a series of attack methods. Depending on the target model information available to the attackers, adversarial attacks can be divided into white-box attacks [18], [23], [28]–[33] and black-box attacks [20], [34]–[36]. For white-box attacks, the whole network architecture and parameters are exposed to attackers, so that they can optimize the adversarial perturbations according to the gradients of the loss *w.r.t.* inputs. For instance, FGSM [28] is a classic white-box attack method, which crafts adversarial samples by maximizing the loss along the gradient direction with a large step. After that, FGSM was further extended to multi-step variants, such as BIM [30] and PGD [23]. In addition, a simple yet accurate method named Deepfool [29] was developed to generate minimal perturbations sufficient to change classification labels. Recently, some methods [32], [33]

with stronger attack capabilities have been proposed, posing a greater threat to the robustness of deep learning models. For black-box attacks, attackers can only access the inputs and outputs of target models, so it is difficult to acquire the gradient directly. One popular solution is to exploit the transferability of adversarial example to conduct the attack [34]–[36].

Apart from classification, researchers have also explored adversarial attacks on deep hashing-based retrieval [13]–[17], [22]. HAG [13] is the first attack method in this field, which can confuse hashing models to retrieve results irrelevant to the input sample, *i.e.*, **non-targeted attack**. Subsequently, Lu *et al.* [17] proposed SDHA to obtain more effective adversarial queries against retrieval tasks by considering staying away from all relevant samples of the query. For the **targeted attack** (*i.e.*, the retrieved images of the adversarial example are semantically relevant to the target label specified by the attacker), P2P and DHTA heuristically [14] select an anchor code as the representative of the target label and then move the adversarial query close to it. Recently, ProS-GAN [15] and THA [22] design an auxiliary network to produce prototype code as a guide for targeted attacks and defense. Unlike the auxiliary network without theoretical guarantees, we propose a discriminative learning measurement with a provable mathematical formula to obtain the representative mainstay codes.

C. Adversarial Training

To resist the adversarial examples, many defense methods [23], [37]–[42] have been proposed. Among them, adversarial training is currently the most effective way to strengthen the robustness of neural networks, which augments the training data with adversarial examples. Madry *et al.* [23] reformulated standard adversarial training as a minimax optimization problem. After that, Zhang *et al.* [39] proposed a defense method named TRADES, which provides a trade-off between adversarial robustness and benign accuracy on clean data. Cui *et al.* [41] exploited the natural classifier boundary as a guide to improve model robustness without losing much natural accuracy. Jia *et al.* [42] introduced a learnable attack strategy for adversarial training. In detail, they designed a strategy network to automatically produce sample-dependent attack strategies at different training stages.

In deep hashing, Wang *et al.* [22] proposed an adversarial training algorithm based on the targeted attack (dubbed ATRDH here) by reconstructing the semantic correlations between adversarial samples and clean samples. It is clear that ATRDH is not a standard adversarial training mechanism, and it simply treats the similarity errors induced by the adversarial samples as a regularization term. By contrast, our SAAT is a standardized adversarial training that minimizes the distance between the hash codes of adversarial examples and the mainstay codes under a well-designed *minimax* framework.

III. Semantic-Aware Adversarial Training

This section will present the proposed Semantic-Aware Adversarial Training (SAAT) framework, which is a standardized minimax optimization formulation for deep hashing aiming to achieve robust deep hashing models. We first present the definitions of adversarial attack and adversarial training on deep hashing-based retrieval and then explicitly elaborate on the overall idea and submodules, followed by a discussion.

TABLE I
Summary of main notation.

Notation	Description
$F(\cdot)$	hashing model
$f_\theta(\cdot)$	DNN with parameter θ
O	training set, $O=\{(\boldsymbol{x}_i,\boldsymbol{y}_i)\}_{i=1}^N$
$\boldsymbol{x},\boldsymbol{y}$	input image and its corresponding label
$\boldsymbol{x'}$	the adversarial example of $\boldsymbol{x}$
$\boldsymbol{b}$	the hash code of $\boldsymbol{x}$
S	similarity matrix
$\boldsymbol{x}_{\bar{i}}^{(\mathrm{p})}$, $\boldsymbol{x}_{\bar{j}}^{(\mathrm{n})}$	the i-th positive and j-th negative of $\boldsymbol{x}$
$\boldsymbol{b}_{\bar{i}}^{(\mathrm{p})}$, $\boldsymbol{b}_{\bar{j}}^{(\mathrm{n})}$	the hash codes of $\boldsymbol{x}_{\bar{i}}^{(\mathrm{p})}$ and $\boldsymbol{x}_{\bar{j}}^{(\mathrm{n})}$
$w_{\bar{i}}$, $w_{\bar{j}}$	the weighting coefficients for $\boldsymbol{x}_{\bar{i}}^{(\mathrm{p})}$ and $\boldsymbol{x}_{\bar{j}}^{(\mathrm{n})}$
$\boldsymbol{b}_m$	the mainstay code of $\boldsymbol{x}$
$\boldsymbol{b}_t$	the mainstay code of target label $\boldsymbol{y}_t$
D_H	Hamming distance measure function
N	the number of training samples
C	the number of classes
K	the length of hash code
N_{p}, N_{n}	the numbers of positives and negatives of $\boldsymbol{x}$
$\mathcal{L}_{adv}$	objective function for adversarial attack
$\mathcal{L}_{at}$	objective function for adversarial training

A. Preliminaries

Deep Hashing-based Retrieval. We consider learning a hashing model F from a training set $O=\{(\boldsymbol{x}_i,\boldsymbol{y}_i)\}_{i=1}^N$ that contains N samples labeled with C classes, 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 $\boldsymbol{x}_i$. $y_{ij}=1$ means that $\boldsymbol{x}_i$ belongs to the j-th class. It is worth noting that $\boldsymbol{x}_i$ is allowed to belong to more than one class, *i.e.*, multi-label data. The objective of F is to get a set of K-bit binary codes $B=\{\boldsymbol{b}_i\}_{i=1}^N\in\{-1,1\}^{N\times K}$ for the training set, which desires to preserve semantic similarities among samples in Hamming space for efficient ANN search. Generally, we utilize similarity matrix S to express semantic similarities between each pair of samples. For any two instances $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, $S_{ij}>0$ describes they share at least one class, otherwise $S_{ij}=0$.

The generation process of hash code $\boldsymbol{b}_i$ of $\boldsymbol{x}_i$ can be expressed as follows:

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

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 $\mathrm{sign}(\cdot)$ on the output of $f_\theta(\boldsymbol{x}_i)$. Typically, $f(\cdot)$ is implemented by a convolutional neural network (CNN) and adopts $\tanh(\cdot)$ function to approximate the $\mathrm{sign}(\cdot)$ function during the training process to relieve the vanishing gradient problem.

Definition 1: ***Adversarial Attack on Deep Hashing-based Retrieval.*** In deep hashing-based retrieval, given a benign query $\boldsymbol{x}$ with label $\boldsymbol{y}$, the goal of non-targeted attack is to craft an adversarial example $\boldsymbol{x'}$, which could confuse the deep hashing model F to retrieve irrelevant samples to query $\boldsymbol{x}$. In contrast, a targeted attack aims to mislead the deep hashing model into returning samples related to a given target label

$\boldsymbol{y}_t$. Moreover, the adversarial perturbation $\boldsymbol{x}' - \boldsymbol{x}$ should be as small as possible to be imperceptible to the human eye.

Definition 2: ***Adversarial Training on Deep Hashing-based Retrieval.*** Similar to classification, adversarial training on deep hashing utilizes both the benign samples $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N$ and corresponding adversarial versions $\{(\boldsymbol{x}'_i, \boldsymbol{y}_i)\}_{i=1}^N$ to re-optimize the parameter θ of deep hashing model, and thereby the model could retrieve semantically relevant contents to the original label $\boldsymbol{y}_i$, whether the input is a clean sample $\boldsymbol{x}_i$ or an adversarial sample $\boldsymbol{x}'_i$.

Summary of Notation. For the sake of clarity, we summarize the formal notation and key concepts in Table I.

B. An Overall Illustration

During the procedure of adversarial learning, the generation of adversarial examples plays a key role. Unlike image classification with labels as supervision to generate practical adversarial examples, deep hashing suffers from a lack of discriminative representatives to guide adversarial attack and defense. To address this issue, we conceive a semantic-aware code (*i.e.*, mainstay code) for each class as an optimal representative of adversarial learning. With the mainstay code, we present an efficient adversarial attack method and formulate a well-designed adversarial training based on a minimax scheme for deep hashing. The overall framework of our proposed SAAT contains two components: the generation of mainstay code and the minimax-based adversarial training, as shown in Fig. 2. Particularly, we construct a mainstay code for each class by a discriminative mainstay features learning pattern that ensures the mainstay code is as close as possible to the positives and stays away from the negatives. After yielding the mainstay codes, we serve a minimax-based adversarial training. Under formalized adversarial training, the inner maximization aims to generate adversarial examples led by the mainstay codes, while the outer minimization attempts to learn robust models by minimizing the expected loss over the adversarial perturbations.

C. Generation of Mainstay Code

We desire to build a semantic-aware representative (dubbed *mainstay code*) for any class in Hamming space to accomplish adequate adversarial attacks and learn robust deep hashing models. Considering the specialties of deep hashing, we suggest that the mainstay code is required to satisfy the following properties. 1) First, the mainstay code is a binary code in Hamming Space with length K to retain the computation efficiency of hashing, where K is the hash code length of given deep hashing models. 2) Second, the mainstay code can be generated adaptively for different datasets and multiple deep hashing methods. 3) More importantly, the mainstay code is expected to be globally semantically discriminable, *i.e.*, as close as possible to all positives of the given class and as far as possible from all negatives of the class in Hamming space, as illustrated in Fig. 2. Notably, for any sample $\boldsymbol{x}_i$ with class $\boldsymbol{y}_i$, its corresponding positives are those share at least one class with itself, *i.e.*, $\{(\boldsymbol{x}_j, \boldsymbol{y}_j) \in O : S_{ij} > 0\}$, and its corresponding negatives are $\{(\boldsymbol{x}_j, \boldsymbol{y}_j) \in O : S_{ij} = 0\}$.

Subsequently, we will present the well-conceived discriminative mainstay features learning (DMFL) scheme to solve the mainstay code. For conciseness, we will discard the subscript i and just use $\boldsymbol{x}$ labeled with $\boldsymbol{y}$ to denote the input sample. In particular, we treat $\bar{i}$ and $\bar{j}$ as the indices of positives and negatives of $\boldsymbol{x}$, respectively. That is, $\boldsymbol{x}_{\bar{i}}^{(\mathrm{p})}$ is the i-th positive sample of $\boldsymbol{x}$, and $\boldsymbol{x}_{\bar{j}}^{(\mathrm{n})}$ is the j-th negative sample of $\boldsymbol{x}$. According to the third property, the mainstay code $\boldsymbol{b}_m$ of a given sample $(\boldsymbol{x}, \boldsymbol{y})$ can be formulated as follows:

$$\min_{\boldsymbol{b}_m} \sum_{\bar{i}}^{N_\mathrm{p}} w_{\bar{i}} D_\mathrm{H}(\boldsymbol{b}_m, \boldsymbol{b}_{\bar{i}}^{(\mathrm{p})}) - \sum_{\bar{j}}^{N_\mathrm{n}} w_{\bar{j}} D_\mathrm{H}(\boldsymbol{b}_m, \boldsymbol{b}_{\bar{j}}^{(\mathrm{n})}), \tag{2}$$

where D_H is the Hamming distance measure. $\boldsymbol{b}_{\bar{i}}^{(\mathrm{p})}$ and $\boldsymbol{b}_{\bar{j}}^{(\mathrm{n})}$ are hash codes of $\boldsymbol{x}_{\bar{i}}^{(\mathrm{p})}$ and $\boldsymbol{x}_{\bar{j}}^{(\mathrm{n})}$, respectively. N_p and N_n are the numbers of positive and negative samples, respectively. $w_{\bar{i}}$ and $w_{\bar{j}}$ are the weighting coefficients for different samples.

However, generating and optimizing the mainstay code are challenging to achieve the optimal solution, due to the discrete nature of the mainstay code and high computational cost. Fortunately, based on the binary characteristic of the hash codes, we can precisely figure out the mainstay code by the following theorem.

Theorem 1. *Suppose $\boldsymbol{b} \in \{-1, +1\}^K$ is a binary code in Hamming space, and $\psi(\boldsymbol{b})$ is an objective function as follows:*

$$\psi(\boldsymbol{b}) = \sum_{\bar{i}}^{N_\mathrm{p}} w_{\bar{i}} D_\mathrm{H}(\boldsymbol{b}, \boldsymbol{b}_{\bar{i}}^{(\mathrm{p})}) - \sum_{\bar{j}}^{N_\mathrm{n}} w_{\bar{j}} D_\mathrm{H}(\boldsymbol{b}, \boldsymbol{b}_{\bar{j}}^{(\mathrm{n})}). \tag{3}$$

If $\boldsymbol{b}_m$ is the optimal solution to $\min \psi(\boldsymbol{b})$, then $\boldsymbol{b}_m$ can be directly written as

$$\begin{aligned} \boldsymbol{b}_m &= \arg \min_{\boldsymbol{b} \in \{-1,+1\}^K} \sum_{\bar{i}}^{N_\mathrm{p}} w_{\bar{i}} D_\mathrm{H}(\boldsymbol{b}, \boldsymbol{b}_{\bar{i}}^{(\mathrm{p})}) - \sum_{\bar{j}}^{N_\mathrm{n}} w_{\bar{j}} D_\mathrm{H}(\boldsymbol{b}, \boldsymbol{b}_{\bar{j}}^{(\mathrm{n})}) \\ &= \operatorname{sign}\left(\sum_{\bar{i}}^{N_\mathrm{p}} w_{\bar{i}} \boldsymbol{b}_{\bar{i}}^{(\mathrm{p})} - \sum_{\bar{j}}^{N_\mathrm{n}} w_{\bar{j}} \boldsymbol{b}_{\bar{j}}^{(\mathrm{n})} \right). \end{aligned} \tag{4}$$

Proof. As the mainstay code $\boldsymbol{b}_m$ is the optimal solution of the minimizing objective, the above theorem is equivalent to prove the following inequality:

$$\psi(\boldsymbol{b}) \geq \psi(\boldsymbol{b}_m), \quad \forall\ \boldsymbol{b} \in \{-1, +1\}^K. \tag{5}$$

According to $D_\mathrm{H}(\boldsymbol{b}_1, \boldsymbol{b}_2) = \frac{1}{2}(K - \boldsymbol{b}_1^\top \boldsymbol{b}_2)$ and $\boldsymbol{b} = \{b_1, b_2, ..., b_K\}$, then we have

$$\begin{aligned} \psi(\boldsymbol{b}) &= \sum_{\bar{i}}^{N_\mathrm{p}} w_{\bar{i}} \frac{1}{2}(K - \boldsymbol{b}^\top \boldsymbol{b}_{\bar{i}}^{(\mathrm{p})}) - \sum_{\bar{j}}^{N_\mathrm{n}} w_{\bar{j}} \frac{1}{2}(K - \boldsymbol{b}^\top \boldsymbol{b}_{\bar{j}}^{(\mathrm{n})}) \\ &= -\frac{1}{2} \sum_{\bar{i}}^{N_\mathrm{p}} w_{\bar{i}} \boldsymbol{b}^\top \boldsymbol{b}_{\bar{i}}^{(\mathrm{p})} + \frac{1}{2} \sum_{\bar{j}}^{N_\mathrm{n}} w_{\bar{j}} \boldsymbol{b}^\top \boldsymbol{b}_{\bar{j}}^{(\mathrm{n})} + \xi \\ &= -\frac{1}{2} \sum_{\bar{i}}^{N_\mathrm{p}} w_{\bar{i}} \sum_{k=1}^{K} b_k b_{\bar{i}k}^{(\mathrm{p})} + \frac{1}{2} \sum_{\bar{j}}^{N_\mathrm{n}} w_{\bar{j}} \sum_{k=1}^{K} b_k b_{\bar{j}k}^{(\mathrm{p})} + \xi \\ &= -\frac{1}{2} \sum_{k=1}^{K} b_k \Big(\sum_{\bar{i}}^{N_\mathrm{p}} w_{\bar{i}} b_{\bar{i}k}^{(\mathrm{p})} - \sum_{\bar{j}}^{N_\mathrm{n}} w_{\bar{j}} b_{\bar{j}k}^{(\mathrm{n})} \Big) + \xi, \end{aligned} \tag{6}$$

where ξ is a constant. Due to the nature of absolute value, we have

$$\begin{aligned}\psi(\boldsymbol{b}) &= -\frac{1}{2}\sum_{k=1}^{K} b_k\Big(\sum_{\bar{i}}^{N_\mathrm{p}} w_{\bar{i}} b_{\bar{i}k}^{(\mathrm{p})} - \sum_{\bar{j}}^{N_\mathrm{n}} w_{\bar{j}} b_{\bar{j}k}^{(\mathrm{n})}\Big) + \xi \\ &\geq -\frac{1}{2}\sum_{k=1}^{K}\left| b_k\Big(\sum_{\bar{i}}^{N_\mathrm{p}} w_{\bar{i}} b_{\bar{i}k}^{(\mathrm{p})} - \sum_{\bar{j}}^{N_\mathrm{n}} w_{\bar{j}} b_{\bar{j}k}^{(\mathrm{n})}\Big)\right| + \xi \\ &= -\frac{1}{2}\sum_{k=1}^{K}\left|\Big(\sum_{\bar{i}}^{N_\mathrm{p}} w_{\bar{i}} b_{\bar{i}k}^{(\mathrm{p})} - \sum_{\bar{j}}^{N_\mathrm{n}} w_{\bar{j}} b_{\bar{j}k}^{(\mathrm{n})}\Big)\right| + \xi \\ &= -\frac{1}{2}\sum_{k=1}^{K}\mathrm{sign}\Big(\sum_{\bar{i}}^{N_\mathrm{p}} w_{\bar{i}} b_{\bar{i}k}^{(\mathrm{p})} - \sum_{\bar{j}}^{N_\mathrm{n}} w_{\bar{j}} b_{\bar{j}k}^{(\mathrm{n})}\Big)\Big(\sum_{\bar{i}}^{N_\mathrm{p}} w_{\bar{i}} b_{\bar{i}k}^{(\mathrm{p})} \\ &\quad - \sum_{\bar{j}}^{N_\mathrm{n}} w_{\bar{j}} b_{\bar{j}k}^{(\mathrm{n})}\Big) + \xi\end{aligned} \tag{7}$$

Similar to Eq. (6), we represent $\psi(\boldsymbol{b}_m)$ as

$$\psi(\boldsymbol{b}_m) = -\frac{1}{2}\sum_{k=1}^{K} b_{mk}\Big(\sum_{\bar{i}}^{N_\mathrm{p}} w_{\bar{i}} b_{\bar{i}k}^{(\mathrm{p})} - \sum_{\bar{j}}^{N_\mathrm{n}} w_{\bar{j}} b_{\bar{j}k}^{(\mathrm{n})}\Big) + \xi. \tag{8}$$

Hence, we have

$$\begin{aligned}\psi(\boldsymbol{b}) \geq& -\frac{1}{2}\sum_{k=1}^{K}\mathrm{sign}\Big(\sum_{\bar{i}}^{N_\mathrm{p}} w_{\bar{i}} b_{\bar{i}k}^{(\mathrm{p})} - \sum_{\bar{j}}^{N_\mathrm{n}} w_{\bar{j}} b_{\bar{j}k}^{(\mathrm{n})}\Big)\Big(\sum_{\bar{i}}^{N_\mathrm{p}} w_{\bar{i}} b_{\bar{i}k}^{(\mathrm{p})} \\ &- \sum_{\bar{j}}^{N_\mathrm{n}} w_{\bar{j}} b_{\bar{j}k}^{(\mathrm{n})}\Big) + \xi \\ =& -\frac{1}{2}\sum_{k=1}^{K} b_{mk}\Big(\sum_{\bar{i}}^{N_\mathrm{p}} w_{\bar{i}} b_{\bar{i}k}^{(\mathrm{p})} - \sum_{\bar{j}}^{N_\mathrm{n}} w_{\bar{j}} b_{\bar{j}k}^{(\mathrm{n})}\Big) + \xi \\ =& \psi(\boldsymbol{b}_m).\end{aligned} \tag{9}$$

That is, $\psi(\boldsymbol{b}) \geq \psi(\boldsymbol{b}_m)$ and the theorem is proved. Therefore, the mainstay code $\boldsymbol{b}_m$ of a given sample $(\boldsymbol{x}, \boldsymbol{y})$ in problem (2) can be solved by

$$\boldsymbol{b}_m = \mathrm{sign}\left(\sum_{\bar{i}}^{N_\mathrm{p}} w_{\bar{i}} \boldsymbol{b}_{\bar{i}}^{(\mathrm{p})} - \sum_{\bar{j}}^{N_\mathrm{n}} w_{\bar{j}} \boldsymbol{b}_{\bar{j}}^{(\mathrm{n})}\right). \tag{10}$$

In particular, we have to claim that our discriminative mainstay code is a globally optimal semantic representative. Compared to sample-level [13], [14] and category-level [14], [17] approaches, our mainstay code is optimized in global semantic space by preserving similarity with all positives and irrelevancy with all negatives. Here, "globally" means that our method considers all positive and negative samples related to the query in the global semantic space, not that our mainstay code can be the optimal semantic representative in all cases.

In addition, we define the $w_{\bar{i}}$ and $w_{\bar{j}}$ as follows:

$$w_{\bar{i}} = \frac{1}{N_\mathrm{p}} \cdot s_{\bar{i}}, \quad w_{\bar{j}} = \frac{1}{N_\mathrm{n}} \cdot (1 - s_{\bar{j}}), \tag{11}$$

where $s_{\bar{i}/\bar{j}} = \frac{\langle \boldsymbol{y}, \boldsymbol{y}_{\bar{i}/\bar{j}} \rangle}{\|\boldsymbol{y}\| \|\boldsymbol{y}_{\bar{i}/\bar{j}}\|}$ denotes the similarity between the given instance $(\boldsymbol{x}, \boldsymbol{y})$ and the i-th positive (j-th negative), which means the more classes they share, the more similar they are. $\bar{i}/\bar{j}$ indicates $\bar{i}$ 'or' $\bar{j}$. $\frac{1}{N_{\mathrm{p/n}}}$ can balance the number difference between positive and negative samples.

D. Semantic-Aware Adversarial Attack

To verify the effectiveness of our mainstay code and reveal the robustness of deep hashing models, we provide a semantic-aware adversarial attack strategy based on proposed mainstay code. We first illustrate the implementation of our approach using non-targeted attack, and then we apply the idea to targeted attack.

Non-targeted Attack. For non-targeted attack, we prefer 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. Since the mainstay code is found, the objective of adversarial attack can be translated into maximizing the Hamming distance between the hash code of adversarial example and the mainstay code. Thus, for a given clean image $\boldsymbol{x}$, its corresponding adversarial example $\boldsymbol{x}'$ is developed by following objective under the L_p constraint:

$$\boldsymbol{x}' = \arg\max_{\boldsymbol{x}'} D_\mathrm{H}(F(\boldsymbol{x}'), \boldsymbol{b}_m), \quad \text{s.t. } \|\boldsymbol{x} - \boldsymbol{x}'\|_p \leq \epsilon, \tag{12}$$

where $\|\cdot\|_p$ ($p = 1, 2, \infty$) is L_p norm that keeps the pixel difference between the adversarial example and the benign sample no more than ϵ for the imperceptible property of adversarial perturbations.

Due to $D_\mathrm{H}(\boldsymbol{b}_1, \boldsymbol{b}_2) = \frac{1}{2}(K - \boldsymbol{b}_1^\top \boldsymbol{b}_2)$, the Eq. (12) is equivalent to:

$$\begin{aligned}\boldsymbol{x}' = \arg\max_{\boldsymbol{x}'} -\frac{1}{K}\boldsymbol{b}_m^\top \tanh(\alpha f_\theta(\boldsymbol{x}')), \\ \text{s.t. } \|\boldsymbol{x} - \boldsymbol{x}'\|_p \leq \epsilon,\end{aligned} \tag{13}$$

where $\tanh(\cdot)$ is the tanh activation function, and $\alpha \in [0, 1]$ is the hyper-parameter that controls $\tanh(\alpha f_\theta(\boldsymbol{x}'))$ to approximate $F(\boldsymbol{x}')$. Following [13], we initialize α with a small value to obtain a larger gradient of the objective function in Eq. (13) concerning $\boldsymbol{x}'$, then gradually enlarge it to approximate $\mathrm{sign}(\cdot)$ until it equals 1.

Targeted Attack. The only difference between non-targeted attacks and targeted attacks is the objective function. For a given benign sample $\boldsymbol{x}$ and a target label $\boldsymbol{y}_t$, we first acquire the mainstay code $\boldsymbol{b}_t$ of $\boldsymbol{y}_t$ by Eq. (10), and then the objective of targeted attack can be defined as:

$$\begin{aligned}\boldsymbol{x}' &= \arg\min_{\boldsymbol{x}'} D_\mathrm{H}(F(\boldsymbol{x}'), \boldsymbol{b}_t) \\ &= \arg\max_{\boldsymbol{x}'} \frac{1}{K}\boldsymbol{b}_t^\top \tanh(\alpha f_\theta(\boldsymbol{x}')), \\ &\text{s.t. } \|\boldsymbol{x} - \boldsymbol{x}'\|_p \leq \epsilon.\end{aligned} \tag{14}$$

As the distance between the hash code of adversarial example $\boldsymbol{x}'$ and the mainstay code of the target label decreases, the adversarial example gradually approaches the target label in semantic terms while guaranteeing visual imperceptibility. Thus, we could retrieve semantically relevant contents to the target label by feeding $\boldsymbol{x}'$ into a deep hashing-based retrieval system.

Generation of Adversarial Examples. Given a clean image, this paper adopts PGD attack [23], one of the most popular attack methods (other attack strategies are also feasible), to optimize $\boldsymbol{x}'$ with T ($T = 100$ by default) iterations, *i.e.*,

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

where η is the step size, $\mathcal{S}_\epsilon$ projects $\boldsymbol{x}'$ into the ϵ-ball [23] of x, and $\mathcal{L}_{adv}$ is the objective function for adversarial attack. In the

case of non-targeted attack, $\mathcal{L}_{adv} = -\frac{1}{K}\boldsymbol{b}_m^\top \tanh(\alpha f_\theta(\boldsymbol{x}'))$, while $\mathcal{L}_{adv} = \frac{1}{K}\boldsymbol{b}_t^\top \tanh(\alpha f_\theta(\boldsymbol{x}'))$ for targeted attack.

Unlike HAG, SDHA and DHTA which require more than 1000 iterations to optimize adversarial examples, our method is more efficient. It merely takes 100 iterations because of the well-designed globally optimal semantic representatives. Notably, we should highlight that since the generation process of adversarial examples is model-agnostic, our learning algorithm could be simply embedded into any deep pairwise similarity-preserving hashing network, such as DPSH [4], HashNet [5], CSQ [7], etc.

E. Semantic-Aware Adversarial Training

The ultimate pursuit of adversarial learning is to enhance the robustness of deep neural networks. After the powerful adversarial attack materializes, we further aspire that the produced adversarial examples can be used as augmentation data to optimize the deep hashing model for defense, *i.e.*, adversarial training. According to [23], the standard adversarial training on classification can be written as a minimax formulation, *i.e.*,

$$\min_{\theta} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}}[\max_{\boldsymbol{x}'} \mathcal{J}(g_\theta(\boldsymbol{x}'), y, \theta)], \tag{16}$$

where $\mathcal{D}$ represents an underlying data distribution, g is a classifier with parameter θ, and $\mathcal{J}$ denotes a classification loss (*e.g.*, cross-entropy loss). The inner maximization problem in (16) can be regarded as an adversarial attack that finds an adversarial example, and the outer optimizes the parameter of network to resist the influence caused by adversarial perturbations. Generally, such a framework is unavailable in deep hashing due to the absence of explicit semantic representative (*e.g.*, label). Nevertheless, the proposed mainstay code remedies this deficiency, thereby enabling us to develop semantic-aware adversarial training (SAAT) for deep hashing, *i.e.*,

$$\min_{\theta} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}}[\max_{\boldsymbol{x}'} \mathcal{L}_{at}(\boldsymbol{x}, \boldsymbol{x}', \boldsymbol{b}_m, \theta)]. \tag{17}$$

The developed SAAT has two significant differences compared to the regular adversarial training in classification. First, the most essential difference is that we leverage the representative mainstay code $\boldsymbol{b}_m$ to guide the adversarial training instead of label vector y. In addition, the objective function in internal optimization is different. The optimization objective $\mathcal{L}_{at}$ of SAAT is more complex, which contains three loss items. In the first item, we exploit the semantic-aware adversarial attack (non-targeted attack) described in Section III-D as the attack strategy to seek the optimal adversarial example. Thus, the first loss item of $\mathcal{L}_{at}$ is formulated as follows:

$$\mathcal{L}_{adv}(\boldsymbol{x}', \boldsymbol{b}_m; \theta) = -\frac{1}{K}\boldsymbol{b}_m^\top \tanh(f_\theta(\boldsymbol{x}')). \tag{18}$$

To make the back-propagation algorithm feasible during training, we substitute the sign function with the tanh function to obtain approximate continuous hash codes, which causes quantization errors. Thus, we introduce a quantization loss to minimize the discrepancy between the approximate hash codes and the binary codes of adversarial examples, *i.e.*,

$$\mathcal{L}_{qua}(\boldsymbol{x}'; \theta) = \| \tanh(f_\theta(\boldsymbol{x}')) - \operatorname{sign}(f_\theta(\boldsymbol{x}'))\|_2^2, \tag{19}$$

where $\|\cdot\|_2$ is the L_2 norm. It is worth noting that an excellent adversarial training strategy should not only improve the robustness of models against adversarial examples but also maintain the performance on benign samples. Hence, we also employ the objective function $\mathcal{L}_{ori}(\boldsymbol{x}, \theta)$ of the original hashing method (*e.g.*, DPH, DPSH or HashNet). In summary, the whole objective function of SAAT is described as:

$$\mathcal{L}_{at} = \lambda\mathcal{L}_{adv} + \mu\mathcal{L}_{qua} + \mathcal{L}_{ori}, \tag{20}$$

where λ and μ are the trade-off hyper-parameters of the first two terms.

Conspicuously, we adopt alternating scheme to optimize the hashing network. Firstly, we maximize $\mathcal{L}_{at}$ with fixed network parameter θ to generate adversarial examples. Then we optimize the hashing network over θ to enhance its robustness against adversarial examples as well as maintain its performance on clean samples by minimizing $\mathcal{L}_{at}$. The two steps are iteratively optimized until the hashing network converges to local optima. The overall optimization process of SAAT is outlined in Algorithm 1.

Algorithm 1 Optimization of Semantic-Aware Adversarial Training in Problem (17)

Input: Image dataset $O = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N$, pre-trained hashing model $F(\cdot) = \operatorname{sign}(f_\theta(\cdot))$, training epochs E, batch size n, learning rate ζ, step size α, perturbation budget ϵ, attack iterations T, weighting factors λ and μ.

for $iter = 1...E$ **do**
  for image batch $\{(x_i, y_i)\}_{i=1}^n$ **do**
    For each x_i, calculate its mainstay code $\boldsymbol{b}_{mi}$ with Eq. (10);
    Optimize its corresponding adversarial samples $\boldsymbol{x}_i'$ by PGD attack with T iterations:
      $\boldsymbol{x}_i' \leftarrow \mathcal{S}_\epsilon(\boldsymbol{x}_i' + \eta \cdot \operatorname{sign}(\nabla_{\boldsymbol{x}_i'}\mathcal{L}_{adv}(\boldsymbol{x}_i', \boldsymbol{b}_{mi})))$ $\quad \forall\ i$;
    Update θ with the gradient descent:
      $\theta \leftarrow \theta - \zeta\nabla_\theta \frac{1}{n}\sum_{i=1}^n \mathcal{L}_{at}(\boldsymbol{x}_i, \boldsymbol{x}_i', \boldsymbol{b}_{mi}, \theta)$;
  end for
end for

Output: Network parameter θ.

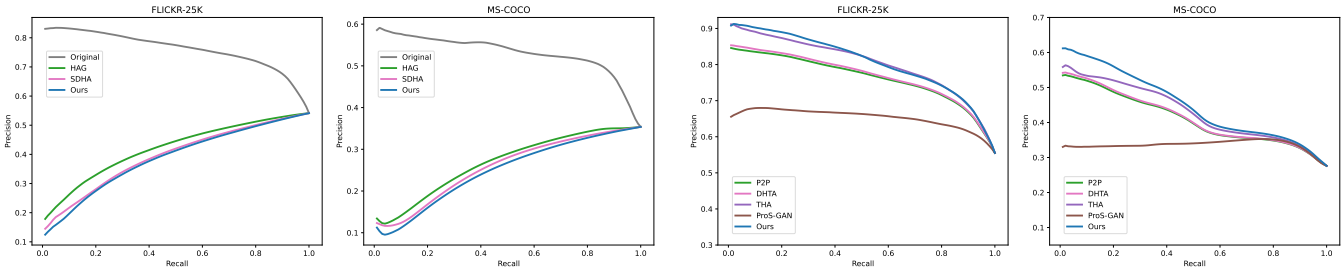
F. Discussion on the Difference from the Related Works

Regarding seeking optimal representatives, our work has some similarities with ProS-GAN [15] and THA [22]. However, there are some fundamental differences between our approach and theirs. 1) First, the optimization goal is different. ProS-GAN and THA are designed for targeted adversarial attacks, while our proposed mainstay codes are adaptive to both non-targeted and targeted attacks. Importantly, we provide the first fundamental minimax-form adversarial training for deep hashing with the guidance of mainstay codes. 2) Moreover, ProS-GAN and THA adopt a neural network to learn the prototype code of the target label, which is a parameter-sensitive scheme and results in a lack of theoretical guarantees for the quality of generated representatives. In contrast, we present a discriminative mainstay features learning (DMFL) with a provable mathematical formula to obtain the mainstay codes with global discriminative superiority.

Moreover, the generation of mainstay codes is similar to contrastive learning [43], [44] but with three differences. 1)

TABLE II
RESULTS OF DIFFERENT ATTACK METHODS ON THREE DATASETS. WE EVALUATE THE ATTACK PERFORMANCE WITH MAP(%) CRITERIA FOR NON-TARGETED ATTACKS (4-TH TO 6-TH ROWS). FOR TARGETED ATTACKS, WE USE THE T-MAP(%) VALUES TO SHOW THE RESULTS (7-TH TO 11TH ROWS). THE 'ORIGINAL' IN TABLE IS TO QUERY WITH BENIGN SAMPLES, WHERE THE MAP VALUES DENOTE THE RETRIEVAL PERFORMANCE OF HASHING MODEL WITHOUT ATTACK.

Method	Metric	FLICKR-25K				NUS-WIDE				MS-COCO			
		16bits	32bits	48bits	64bits	16bits	32bits	48bits	64bits	16bits	32bits	48bits	64bits
Original	MAP	81.33	82.28	82.47	81.85	76.70	77.47	77.74	78.11	56.26	57.41	56.70	56.64
HAG	MAP	22.53	21.68	21.96	22.56	13.62	13.32	13.67	13.55	12.57	13.97	13.90	13.88
SDHA	MAP	21.82	19.37	19.39	19.20	13.85	12.98	14.49	14.56	11.61	12.38	12.63	13.08
SAAT (Ours)	MAP	**14.92**	**14.53**	**14.59**	**14.82**	**11.93**	**11.89**	**11.85**	**12.02**	**9.96**	**11.21**	**10.96**	**10.87**
P2P	t-MAP	81.45	81.83	82.49	82.98	66.94	69.19	69.20	69.12	51.96	52.74	52.62	51.73
DHTA	t-MAP	82.98	83.51	83.67	83.94	69.10	70.92	71.03	70.93	52.57	53.38	53.23	52.34
ProS-GAN	t-MAP	73.06	66.44	62.34	87.15	**72.33**	**74.29**	73.96	72.30	33.02	33.30	31.63	31.28
THA	t-MAP	86.80	87.84	86.94	86.73	69.78	72.98	73.90	70.29	53.18	52.88	47.80	37.63
SAAT (Ours)	t-MAP	**88.62**	**88.90**	**89.07**	**89.00**	70.68	73.99	**74.00**	**74.33**	**55.75**	**58.00**	**56.97**	**56.58**



(a) Non-targeted (b) Targeted

Fig. 3. Precision-Recall curves on FLICKR-25K and MS-COCO under 32 bits code length.

First, the purpose is different. Typically, contrastive learning desires to optimize a network with good generalization capabilities via a self-supervised approach. Differently, we aim to generate a representative semantic feature for each input sample to supervise the adversarial learning of deep hashing. 2) Secondly, it is different for the optimization process. Contrastive learning involves repeatedly measuring the similarity between input and multiple positive and negative samples to adjust the network parameters through numerous iterations. In contrast, we can instantly calculate the mainstay code of the input sample by exploiting the proposed DMFL, dramatically reducing the computational overhead. 3) Furthermore, contrastive learning occurs in the continuous feature space, while mainstay code is produced in the discrete binary space. Hence, solving the mainstay code by gradient descent is intractable, yet our proposed DMFL cleverly and efficiently addresses this problem.

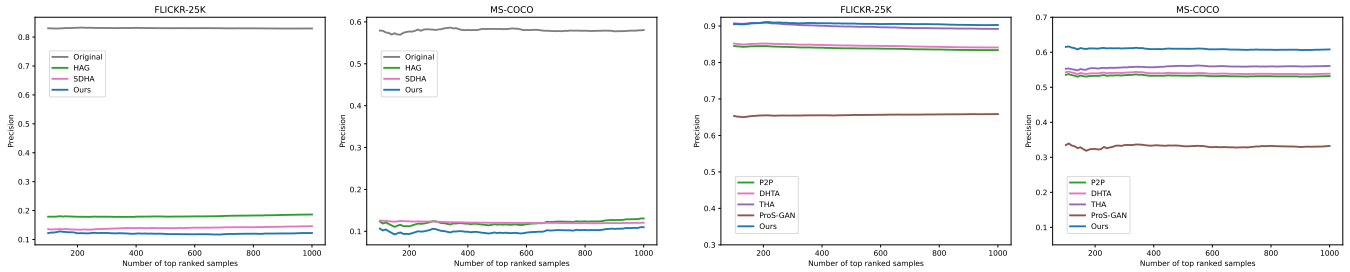
IV. EXPERIMENTS

A. Experimental Setup

1) Datasets.: We conduct extensive experiments to evaluate our methods on three well-known datasets: **FLICKR-25K** [45], **NUS-WIDE** [46] and **MS-COCO** [47]. **FLICKR-25K** consists of 25,000 Flickr images annotated with 38 labels. We pick 1000 instances from the whole dataset as queries, while the remaining are regarded as the database. Moreover, we randomly select 5,000 images from the database for training hashing models and producing mainstay codes. **NUS-WIDE** has 269,648 samples with 81 concepts. Following [15], we select a subset containing 21 most frequent concepts with 193,734 images as database and 2,100 images for testing. Besides, we select 10,500 images from the database for training. **MS-COCO** involves 123,287 samples after combining the training and validation sets, where each sample is labeled with 80 classes. Following [5], we randomly select 5,000 images as queries and the rest as the database. For the training set, 10,000 images are randomly picked from the database.

2) Baselines.: Following [13], we select DPH [13] as target hashing model to be attacked, which is a generic deep hashing retrieval model. Particularly, we can change it to any other deep hashing models. AlexNet [48] and VGGs [49] are chosen as the backbone networks to implement DPH. Specifically, we replace their last classifier layer with a hashing layer, which involves a fully connected layer with K hidden units and a tanh activation. We also evaluate the generalizability of our method on other hashing methods, including DPSH [4], HashNet [5] and CSQ [7]. For estimating the effectiveness of our methods, we implement previous attack and defense methods in hashing retrieval, covering two non-targeted attacks (*i.e.*, HAG [13] and SDHA [17]), three targeted attack methods (*i.e.*, P2P [14], DHTA [14], ProS-GAN [15] and THA [22]), and the only one defense algorithm ATRDH [22]. For targeted attacks, we randomly choose a label for attack, which does not share the same class with the original label. To make fair comparisons, the detailed implementations of these methods are consistent with the original papers with released codes.

3) Implementation Details.: We utilize stochastic gradient descent (SGD) [50] with a learning rate 0.01 and momentum 0.9 as optimizers to pre-train the target hashing models. All images are resized to 224×224 and normalized in $[0, 1]$ before feeding in hashing models. For the proposed attack



(a) Non-targeted (b) Targeted

Fig. 4. Precision@1000 curves on FLICKR-25K and MS-COCO under 32 bits code length.

(a) Non-targeted (b) Targeted

Fig. 5. Retrieval examples on NUS-WIDE with the benign query and its adversarial version. We provide examples of non-targeted attacks and targeted attacks in (a) and (b), respectively. For each example, the two boxes represent the top-5 retrieved images of the natural and adversarial queries, respectively.

algorithm, we apply PGD [14] to optimize adversarial samples. The step size η and the number of iterations T are set to $1/255$ and 100, respectively. The perturbation budget ϵ is fixed to $8/255$. Similar to [13], we set the hyper-parameter α with 0.1 during the first 50 iterations, then update it every 10 iterations according to $[0.2, 0.3, 0.5, 0.7, 1.0]$ during the final 50 iterations. For our defense method SAAT, we conduct 20 epochs for adversarial training. During the training process, we empirically set η and T in PGD with 2 and 7, respectively, to generate adversarial samples in each epoch. The weighting factors λ and μ of Eq. (20) are set as 1 and 10^{-4}, respectively.

4) Protocols.: Following [13], we adopt MAP (mean average precision) to evaluate the performance of non-targeted attacks and calculate MAP values on the top 5000 results from the database. Specifically, for targeted attacks, we employ t-MAP (targeted mean average precision) [14] to appraise their results. As t-MAP takes the target labels as the test labels, the higher the t-MAP, the stronger the targeted attack. In addition, we also provide Precision-Recall (PR) curves and precision@topN curves for comprehensive analysis.

B. Adversarial Attack Results

As shown in Table II, we present the detailed MAP scores of non-targeted attacks on three benchmarks. A lower MAP means a stronger performance of non-targeted attacks. The *Original* in Table II is a query using original samples without any additive noise, where the corresponding MAP reflects the retrieval performance of the attack-free hashing model. From the results, we can observe that our method outperforms all non-targeted attacks and significantly drops the MAP values on three benchmarks with the hash bits varying from 16 to 64. Compared with SDHA [17], the best non-targeted attack method, the MAP scores of our method decline an average of 2.98% for all cases. Especially, on the FLICKR-25K, our method outperforms SDHA by over 4.38% for any bits. The superior behavior of our method owes to the high-quality mainstay code, which represents the globally optimal semantic of a given sample by preserving the similarity with positives and irrelevancy with negatives simultaneously. In contrast, HAG and SDHA just use the information from benign samples and positive samples, respectively. It is worth noting that our attack merely runs 100 iterations during optimization, but other methods use up to 1,500 iterations, or even 2,000 iterations, which further reveals the efficiency of our method.

Results in terms of t-MAP for targeted attacks are also given in Table II. It can be observed that our method achieves the best performance on targeted attacks in most cases. When comparing with the state-of-the-art targeted attack, THA [22], our attack accomplishes average boosts of 1.82%, 1.52% and 8.95% for different bits on FLICKR-25K, NUS-WIDE, and MS-COCO, respectively. Specifically, although ProS-GAN yields better results for 16bits and 32bits on NUS-WIDE, it does not work well on other datasets, especially MS-COCO. This is because the generative framework of ProS-GAN is parameter-sensitive and lacks generalization on different datasets. Moreover, the prototype network they designed is hard to fit multi-label cases (*e.g.*, MS-COCO with 80 classes). Differently, our mainstay codes are obtained by discriminative mainstay feature learning, which can be well adapted to various datasets.

For a more comprehensive comparison, the retrieval performance on benchmarks in terms of PR curves and precision@topN curves are shown in Fig. 3 and Fig. 4, respectively.

TABLE III
MAP (%) OF NON-TARGETED ATTACK METHODS AFTER ADVERSARIAL TRAINING BY ATRDH AND OUR SAAT.

Defense	Attack	FLICKR-25K				NUS-WIDE				MS-COCO			
		16bits	32bits	48bits	64bits	16bits	32bits	48bits	64bits	16bits	32bits	48bits	64bits
ATRDH	Original	71.35	72.16	72.29	72.08	64.58	64.57	67.20	67.90	49.41	50.89	50.38	51.57
	HAG	41.92	42.36	43.80	44.08	41.59	41.93	42.31	42.29	27.19	26.78	26.20	26.75
	SDHA	39.09	37.77	38.39	38.60	41.61	40.36	40.78	40.59	28.09	27.17	27.55	27.75
	Ours	**32.68**	**32.40**	**33.39**	**33.32**	**38.53**	**38.16**	**38.95**	**38.99**	**23.93**	**23.24**	**23.14**	**22.82**
SAAT(Ours)	Original	74.19	73.49	73.41	70.07	61.05	60.51	61.06	60.57	46.91	49.41	50.14	52.01
	HAG	38.05	43.76	48.22	60.89	53.07	52.72	52.56	52.90	33.83	34.01	35.77	36.69
	SDHA	35.26	40.52	45.78	59.62	53.22	52.15	52.57	52.83	35.99	35.07	37.23	36.91
	Ours	**30.55**	**34.77**	**39.35**	**53.76**	**50.35**	**50.33**	**50.20**	**50.16**	**30.98**	**30.87**	**32.63**	**33.43**

TABLE IV
T-MAP (%) OF TARGETED ATTACK METHODS AFTER ADVERSARIAL TRAINING BY ATRDH AND OUR SAAT. THE 'ORIGINAL' IN TABLE INDICATES THE MAP VALUE OF HASHING MODEL WITHOUT ATTACK.

Defense	Attack	FLICKR-25K				NUS-WIDE				MS-COCO			
		16bits	32bits	48bits	64bits	16bits	32bits	48bits	64bits	16bits	32bits	48bits	64bits
ATRDH	Original	71.81	72.40	72.99	72.26	65.60	66.96	67.04	68.15	49.71	50.67	51.25	50.54
	P2P	72.54	71.76	72.05	71.81	46.01	48.80	49.27	49.52	35.69	37.06	37.67	37.75
	DHTA	73.45	72.97	72.90	73.56	47.53	50.39	50.76	51.37	36.13	37.54	38.14	37.88
	ProS-GAN	49.76	49.46	48.64	49.71	29.82	28.92	28.67	29.25	23.34	23.73	23.22	23.29
	THA	77.37	77.97	78.07	78.32	**51.29**	54.85	56.00	55.46	41.13	43.31	43.35	40.92
	Ours	**78.67**	**79.00**	**79.52**	**79.40**	51.09	**55.31**	**56.28**	**55.85**	**42.18**	**44.11**	**44.92**	**44.89**
SAAT(Ours)	Original	73.45	72.89	72.86	71.77	61.37	61.92	61.99	61.02	49.34	50.68	52.18	53.27
	P2P	73.17	71.11	69.39	65.83	40.74	37.68	37.32	38.41	33.08	33.12	33.94	34.06
	DHTA	74.58	72.33	70.74	66.89	41.35	38.19	37.89	38.69	33.28	33.33	34.15	34.14
	ProS-GAN	48.24	48.43	49.20	50.35	37.99	33.81	32.93	30.71	26.61	25.29	26.09	25.09
	THA	76.47	74.38	72.27	66.33	41.28	38.71	37.83	38.02	34.27	34.37	35.57	34.39
	Ours	**79.65**	**76.51**	**75.07**	**69.97**	**42.02**	**39.40**	**38.72**	**39.67**	**36.72**	**36.28**	**36.97**	**37.76**

We only provide the results on FLICKR-25K and MS-COCO of different methods with 32 bits length. For non-targeted attacks, we can see that the curves of our method are always below all other curves while the opposite is the case in targeted attacks, which demonstrates that our method is able to attack hashing models more effectively. Furthermore, we also provide two examples of the retrieval results on NUS-WIDE with a benign image and its adversarial version generated by our method in Fig. 5 for intuitive understanding.

C. Adversarial Defense Results

To improve the adversarial robustness of deep hashing networks, we perform the proposed formalized adversarial training algorithm on pre-trained deep hashing models. After the adversarial training, we re-attack these models and the results in terms of MAP are reported in Table III. By comparing Table III with Table II, we observe that the MAP values of different attack methods are much higher than no adversarial training, though the original MAP values decrease slightly. For example, under the non-targeted attack SDHA, SAAT brings an average increase of approximately 25.35%, 38.72% and 23.87% on FLICKR-25K, NUS-WIDE and MS-COCO, respectively. For our proposed semantic-aware attack, SAAT improves by an average of about 24.94%, 38.33%, and 21.22%. To further verify the effectiveness of our well-designed SAAT, we compare SAAT with ATRDH [22] under the same experiment settings and the results of ATRDH are also illustrated in Table III. As we can see, the original MAP values of SAAT and ATRDH are close, but our SAAT achieves a significant performance boost in terms of the MAP under various attacks. For example, with respect to our proposed attack algorithm, SAAT exceeds ATRDH by an average of 6.66%, 11.60%, 8.69% for FLICKR-25K, NUS-WIDE, and MS-COCO, respectively. The above phenomena show that SAAT can learn more robust hashing codes than ATRDH, because SAAT is a standard minimax-based adversarial training.

A similar situation can be observed in the targeted attack shown in Table IV. In the case of THA, SAAT achieves an average improvement of over 14.71%, 32.77% and 13.22% for different bits on FLICKR-25K, NUS-WIDE and MS-COCO, respectively, in comparison to the models without adversarial training. Additionally, compared with ATRDH, SAAT outscores ATRDH by an average of 5.57%, 15.44%, 7.52% on FLICKR-25K, NUS-WIDE, and MS-COCO, respectively. These cases all confirm that the proposed adversarial training strategy can effectively improve the robustness of deep hashing models against both targeted and non-targeted attacks. It is worth remarking that the attack performance of ProS-GAN on the hashing models retrained by SAAT dramatically decreases, which indicates that our defense method can basically resist the attack of ProS-GAN. This behavior is more pronounced on ATRDH, which may be attributed to the fact that adversarial training is more effective for generative-based attack methods.

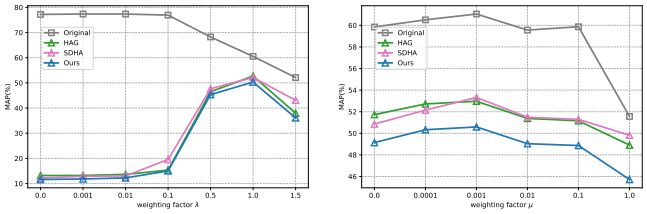
Besides, it can be seen from the Table III and IV that our attack achieves the-sate-of-the-art attack performance on all defense models, which further validates the superiority of our attack algorithm.

TABLE V
THEORETICAL VALUES OF DIFFERENT ATTACK METHODS. FOR EACH METHOD, WE UTILIZE THE REPRESENTATIVE CODES OF THEM TO CALCULATE THEORETICAL MAP(T-MAP).

Method	Metric	FLICKR-25K				NUS-WIDE				MS-COCO			
		16bits	32bits	48bits	64bits	16bits	32bits	48bits	64bits	16bits	32bits	48bits	64bits
Original	MAP	81.33	82.28	82.47	81.85	76.70	77.47	77.74	78.11	56.26	57.41	56.70	56.64
HAG	MAP	22.67	21.54	21.73	22.19	13.51	13.32	13.65	13.54	13.14	14.35	14.15	13.83
SDHA	MAP	18.89	18.55	18.65	18.80	14.13	13.82	13.99	14.17	9.99	11.60	11.19	11.37
SAAT(Ours)	MAP	**14.90**	**14.51**	**14.57**	**14.72**	**11.69**	**11.77**	**11.77**	**11.75**	**9.76**	**11.09**	**10.76**	**10.68**
P2P	t-MAP	81.35	81.87	82.60	83.10	66.87	69.14	69.11	69.11	52.23	52.89	52.90	52.37
DHTA	t-MAP	82.93	83.56	83.72	84.13	69.13	70.85	70.82	70.75	52.82	53.57	53.54	52.93
ProS-GAN	t-MAP	86.47	86.88	87.05	59.45	70.70	73.44	73.48	72.93	50.85	36.19	40.60	35.85
THA	t-MAP	86.67	87.86	87.36	86.81	70.18	73.20	73.73	67.25	51.66	49.74	50.61	44.72
SAAT(Ours)	t-MAP	**88.63**	**88.91**	**89.11**	**89.04**	**70.92**	**73.97**	**74.10**	**74.35**	**59.16**	**60.14**	**59.63**	**58.90**

TABLE VI
MAP (%) OF NON-TARGETED ATTACK WITH DIFFERENT ITERATIONS ON NUS-WIDE.

Iterations	16bits	32bits	48bits	64bits
0	76.70	77.47	77.74	78.11
1	62.82	63.90	63.71	63.77
10	13.18	13.47	13.45	13.44
50	12.04	12.06	12.13	11.96
100	11.96	11.97	12.12	11.93
500	11.80	11.90	12.10	11.96
1000	**11.76**	**11.88**	**12.09**	**11.88**
Theory	11.69	11.77	11.77	11.75



(a) λ (b) μ

Fig. 6. MAP (%) on NUS-WIDE for our adversarial training with different λ and μ.

TABLE VII
PERCEPTIBILITY OF ADVERSARIAL PERTURBATIONS UNDER NON-TARGETED ATTACKS. L_∞ AND L_2 ARE MARTIX NORMS OF ADVERSARIAL PERTURBATIONS, AND MSE IS MEAN SQUARED ERROR BETWEEN BENIGN SAMPLES AND ITS ADVERSARIAL VERSIONS. ALL THE RESULSTS ARE AVERAGED OVER THE ENTIRE TEST SET.

Metric	FLICKR-25K			NUS-WIDE			MS-COCO		
	HAG	SDHA	Ours	HAG	SDHA	Ours	HAG	SDHA	Ours
L_∞	6.26	6.03	**5.27**	6.35	6.13	**5.61**	6.23	6.66	**5.39**
L_2	1.24	1.17	**1.04**	1.25	1.19	**1.08**	1.21	1.23	**1.01**
MSE($\times 10^{-4}$)	7.18	6.73	**5.20**	7.29	6.89	**5.68**	7.08	8.26	**5.44**
MAP(%)	21.94	17.66	**14.97**	13.20	12.94	**11.96**	13.57	12.39	**11.12**

D. Analysis and Discussions

1) Attack Results in Theory: Besides actual attack performance, we also record the MAP(t-MAP) values in theory of various attack methods to compare the theoretical upper bounds that they can reach, which is shown in Table V. Intuitively, we calculate the theoretical MAP by directly retrieving the contents in Hamming space with representative codes of each method instead of the hash codes corresponding to the adversarial samples. Such as the mainstay code of our method, the anchor code of DHTA, and the prototype code of ProS-GAN. Obviously, our attack outperforms all comparison methods on both targeted and non-targeted attacks, which further verifies the superiority of our well-designed mainstay code. Making a comparison between Table II and Table V, we notice that our attack algorithm converges well, as the actual attack performance is considerably approached to the theoretical results.

2) Effect of T in PGD: Table VI presents MAP results of adversarial examples under non-targeted attack with different optimizing iterations (*i.e.*, T). Adversarial examples at iteration 0 correspond to the benign examples without adversarial noise. As expected, MAP drops quickly with the number of iterations. It is worth noting that MAP reduces over 82% of the original MAP at 10 iterations in all cases, which indicates our attack method is much more efficient. Continued optimization of adversarial examples can further reduce the MAP values until the 100th iteration. In addition, the MAP values of the 500 and 1000 iterations are almost invariant and close to the theoretical values. These results imply that we have a good balance between the effectiveness and efficiency of adversarial attacks with $T = 100$.

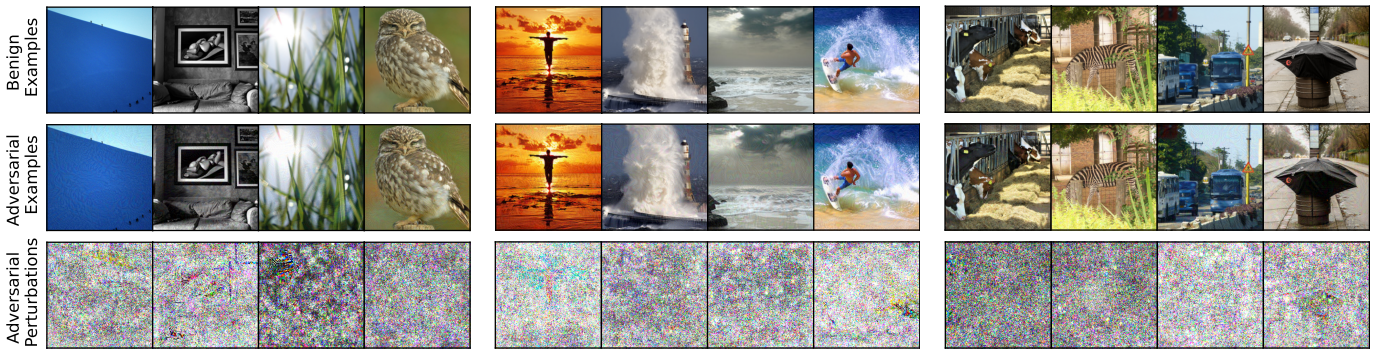
3) Perceptibility: Beyond attack performance, perceptibility is another essential aspect in evaluating the quality of adversarial examples. Hence, we report the perceptibility of adversarial noise with respect to three metrics, including L_∞, L_2 and MSE. The results for all datasets are shown in Table VII. Since the adversarial perturbation is normally desired to be imperceptible, the higher the perceptibility, the worse the visual quality of the adversarial example. As we can see, all the results of our method are very small and outperform those of the other two non-targeted attacks by large margins. This demonstrates that our approach achieves excellent attack performance while ensuring the imperceptibility of the learned perturbations. Some visual examples of adversarial images generated by our method in Fig. 7 also verify this point.

4) Analysis on Hyper-parameters: The hyper-parameters λ and μ control the quality of the adversarial training. To explore the effects of different weighting factors on defense performance, we make comparison experiments with 32-bits hashing model, as illustrated in Fig. 6. For λ, as shown in Fig. 6(a), when λ increases, the defense performance increases until $\lambda = 1$, but the MAP values of original samples drop, which indicates there is a trade-off between robustness and precision. For μ, we can observe from Fig. 6(b) that it has a small but non-negligible impact on the outcomes, and the best behavior is achieved at $\mu = 0.001$.

5) Universality on different hashing models: We argue that our proposed attack and defense algorithms are generic to the most popular hashing models with different backbones. To

TABLE VIII
MAP (%) OF NON-TARGETED ATTACKS FOR DIFFERENT HASHING MODELS ON NUS-WIDE. FOR EACH DEEP HASHING ALGORITHM, WE IMPLEMENT IT WITH FOUR BACKBONE NETWORKS, INCLUDING ALEXNET [48] AND VGG FAMILY [49].

Defense	Attack	DPSH				HashNet				CSQ			
		AlexNet	VGG11	VGG16	VGG19	AlexNet	VGG11	VGG16	VGG19	AlexNet	VGG11	VGG16	VGG19
No Defense	Original	80.97	82.00	81.96	81.78	79.65	81.91	81.13	80.56	79.26	80.71	81.53	0.8160
	HAG	14.49	13.28	14.52	13.73	12.99	11.13	10.31	11.29	16.58	14.93	19.89	14.76
	SDHA	13.72	11.06	14.30	14.00	9.59	10.43	10.59	10.03	12.99	8.85	8.53	7.95
	Ours	**11.32**	**9.70**	**10.50**	**11.27**	**8.10**	**6.18**	**6.36**	**7.15**	**7.18**	**5.92**	**6.22**	**7.18**
ATRDH	Original	72.41	70.52	69.65	69.02	72.79	68.31	68.43	65.16	61.75	57.62	59.21	65.31
	HAG	33.62	45.13	47.44	47.26	30.83	46.63	51.20	51.60	38.70	35.60	34.80	24.85
	SDHA	34.12	39.86	44.59	46.32	25.87	42.13	47.48	47.37	38.80	34.85	36.87	**19.33**
	Ours	**31.22**	**33.85**	**35.87**	**38.69**	**23.56**	**41.43**	**46.23**	**45.48**	**36.33**	**30.38**	**27.76**	29.36
SAAT (Ours)	Original	74.83	76.70	75.99	75.74	76.38	68.75	69.56	70.46	72.94	66.79	67.65	74.96
	HAG	44.22	45.45	54.10	43.65	40.33	57.02	55.54	55.17	44.44	45.36	49.85	42.19
	SDHA	47.53	47.63	55.49	45.55	41.84	56.06	54.92	54.49	47.63	49.59	52.59	39.56
	Ours	**43.05**	**44.39**	**53.18**	**43.07**	**39.00**	**53.37**	**54.21**	**51.57**	**43.64**	**41.02**	**42.27**	**34.28**



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

Fig. 7. Visualization of generated adversarial examples. Given benign images (the first row), we produce corresponding adversarial examples using our attack method, as shown in the second row. For better presentation, we show the normalized adversarial perturbation for each sample in the last row.

verify this point, we compare with non-targeted attacks (HAG and SDHA) on other hashing methods, including DPSH [4], HashNet [5] and CSQ [7]. The results are summarized in Table VIII. First, for the attack, our method is still effective and better than HAG and SDHA in all cases, as shown in 'No Defense' part. Moreover, even with different hashing methods or backbone networks, our defense method can still effectively mitigate the impact of adversarial attacks. Furthermore, when testing with original samples ('Original' in Table VIII), or under the attacks of HAG and SDHA, the results of SAAT are higher than ATRDH, which shows that hashing models trained by our SAAT are more robust than ATRDH. Hence, the above phenomena confirm the universality of the proposed attack and defense methods.

V. CONCLUSION

In this paper, we proposed a generic Semantic-Aware Adversarial Training (SAAT) algorithm for deep hashing-based retrieval, which is the first work to formulate adversarial training of deep hashing as a unified minimax problem. Specifically, we provided a discriminative mainstay features learning strategy to obtain high-quality mainstay codes as the optimal semantic representatives of samples for adversarial learning in deep hashing. Moreover, we took the mainstay code as 'label' to guide the adversarial attack, where the Hamming distance between the mainstay code and the hash code of adversarial example was minimized. Furthermore, benefiting from the high-quality mainstay features, we developed a well-conceived adversarial training scheme based on a formalized minimax optimization paradigm. Extensive experiments demonstrated that our method could attain state-of-the-art results in both adversarial attack and defense on deep hashing-based retrieval.

REFERENCES

[1] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," in *2006 47th Annual IEEE Symposium on Foundations of Computer Science*, 2006, pp. 459–468.
[2] J. Wang, T. Zhang, N. Sebe, H. T. Shen *et al.*, "A survey on learning to hash," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, pp. 769–790, 2018.
[3] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan, "Supervised hashing for image retrieval via image representation learning," in *AAAI Conference on Artificial Intelligence*, 2014, pp. 2156–2162.
[4] W. Li, S. Wang, and W. Kang, "Feature learning based deep supervised hashing with pairwise labels," in *International Joint Conference on Artificial Intelligence*, 2016, pp. 1711–1717.
[5] Z. Cao, M. Long, J. Wang, and P. S. Yu, "Hashnet: Deep learning to hash by continuation," in *IEEE International Conference on Computer Vision*, 2017, pp. 5608–5617.
[6] V. Talreja, M. C. Valenti, and N. M. Nasrabadi, "Deep hashing for secure multimodal biometrics," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 1306–1321, 2020.
[7] L. Yuan, T. Wang, X. Zhang, F. E. Tay, Z. Jie, W. Liu, and J. Feng, "Central similarity quantization for efficient image and video retrieval," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3083–3092.
[8] Y. Wang, X. Ou, J. Liang, and Z. Sun, "Deep semantic reconstruction hashing for similarity retrieval," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 1, pp. 387–400, 2020.

[9] L. Fan, K. W. Ng, C. Ju, T. Zhang, and C. S. Chan, "Deep polarized network for supervised learning of accurate binary hashing codes," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, 2020, pp. 825–831.

[10] J. T. Hoe, K. W. Ng, T. Zhang, C. S. Chan, Y.-Z. Song, and T. Xiang, "One loss for all: Deep hashing with a single cosine similarity based learning objective," *Advances in Neural Information Processing Systems*, vol. 34, pp. 24 286–24 298, 2021.

[11] K. D. Doan, P. Yang, and P. Li, "One loss for quantization: Deep hashing with discrete wasserstein distributional matching," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 9447–9457.

[12] D. Wu, Q. Dai, B. Li, and W. Wang, "Deep uncoupled discrete hashing via similarity matrix decomposition," *ACM Transactions on Multimedia Computing, Communications and Applications*, vol. 19, no. 1, pp. 1–22, 2023.

[13] E. Yang, T. Liu, C. Deng, and D. Tao, "Adversarial examples for hamming space search," *IEEE Transactions on Cybernetics*, vol. 50, pp. 1473–1484, 2020.

[14] J. Bai, B. Chen, Y. Li, D. Wu, W. Guo, S.-t. Xia, and E.-h. Yang, "Targeted attack for deep hashing based retrieval," in *European Conference on Computer Vision*, 2020, pp. 618–634.

[15] X. Wang, Z. Zhang, B. Wu, F. Shen, and G. Lu, "Prototype-supervised adversarial network for targeted attack of deep hashing," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2021, pp. 16 357–16 366.

[16] Z. Zhang, X. Wang, G. Lu, F. Shen, and L. Zhu, "Targeted attack of deep hashing via prototype-supervised adversarial networks," *IEEE Transactions on Multimedia*, pp. 1–13, 2021.

[17] J. Lu, M. Chen, Y. Sun, W. Wang, Y. Wang, and X. Yang, "A smart adversarial attack on deep hashing based image retrieval," in *International Conference on Multimedia Retrieval*, 2021, pp. 227–235.

[18] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *International Conference on Learning Representations*, 2014, pp. 1–10.

[19] L. Amsaleg, J. Bailey, A. Barbe, S. M. Erfani, T. Furon, M. E. Houle, M. Radovanović, and X. V. Nguyen, "High intrinsic dimensionality facilitates adversarial attack: Theoretical evidence," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 854–865, 2020.

[20] Y. Zhong and W. Deng, "Towards transferable adversarial attack against deep face recognition," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 1452–1466, 2020.

[21] X.-C. Li, X.-Y. Zhang, F. Yin, and C.-L. Liu, "Decision-based adversarial attack with frequency mixup," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 1038–1052, 2022.

[22] X. Wang, Z. Zhang, G. Lu, and Y. Xu, "Targeted attack and defense for deep hashing," in *International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021, pp. 2298–2302.

[23] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *International Conference on Learning Representations*, 2017, pp. 1–28.

[24] X. Luo, H. Wang, D. Wu, C. Chen, M. Deng, J. Huang, and X.-S. Hua, "A survey on deep hashing methods," *ACM Transactions on Knowledge Discovery from Data*, vol. 17, no. 1, pp. 1–50, 2023.

[25] R. Salakhutdinov and G. Hinton, "Semantic hashing," *International Journal of Approximate Reasoning*, vol. 50, pp. 969–978, 2009.

[26] K. Ghasedi Dizaji, F. Zheng, N. Sadoughi, Y. Yang, C. Deng, and H. Huang, "Unsupervised deep generative adversarial hashing network," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3664–3673.

[27] A. W. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, "Content-based image retrieval at the end of the early years," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 1349–1380, 2000.

[28] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations*, 2015, pp. 1–10.

[29] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: A simple and accurate method to fool deep neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2574–2582.

[30] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *Artificial intelligence safety and security*. Chapman and Hall/CRC, 2018, pp. 99–112.

[31] J. Lin, C. Song, K. He, L. Wang, and J. E. Hopcroft, "Nesterov accelerated gradient and scale invariance for adversarial attacks," *arXiv preprint arXiv:1908.06281*, 2019.

[32] F. Croce and M. Hein, "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks," in *International conference on machine learning*. PMLR, 2020, pp. 2206–2216.

[33] Q. Xu, G. Tao, and X. Zhang, "Bounded adversarial attack on deep content features," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 15 203–15 212.

[34] C. Xie, Z. Zhang, Y. Zhou, S. Bai, J. Wang, Z. Ren, and A. L. Yuille, "Improving transferability of adversarial examples with input diversity," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2730–2739.

[35] X. Wang and K. He, "Enhancing the transferability of adversarial attacks through variance tuning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 1924–1933.

[36] Z. Yuan, J. Zhang, and S. Shan, "Adaptive image transformations for transfer-based adversarial attack," in *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part V*. Springer, 2022, pp. 1–17.

[37] C. Xie, J. Wang, Z. Zhang, Z. Ren, and A. Yuille, "Mitigating adversarial effects through randomization," *arXiv preprint arXiv:1711.01991*, 2017.

[38] X. Jia, X. Wei, X. Cao, and H. Foroosh, "Comdefend: An efficient image compression model to defend adversarial examples," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 6084–6092.

[39] H. Zhang, Y. Yu, J. Jiao, E. Xing, L. El Ghaoui, and M. Jordan, "Theoretically principled trade-off between robustness and accuracy," in *International Conference on Machine Learning*, 2019, pp. 7472–7482.

[40] D. Wu, S.-T. Xia, and Y. Wang, "Adversarial weight perturbation helps robust generalization," *Advances in Neural Information Processing Systems*, vol. 33, pp. 2958–2969, 2020.

[41] J. Cui, S. Liu, L. Wang, and J. Jia, "Learnable boundary guided adversarial training," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 15 721–15 730.

[42] X. Jia, Y. Zhang, B. Wu, K. Ma, J. Wang, and X. Cao, "Las-at: adversarial training with learnable attack strategy," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 13 398–13 408.

[43] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.

[44] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, "Supervised contrastive learning," *Advances in neural information processing systems*, vol. 33, pp. 18 661–18 673, 2020.

[45] M. J. Huiskes and M. S. Lew, "The mir flickr retrieval evaluation," in *ACM International Conference on Image and Video Retrieval*, 2008, pp. 39–43.

[46] T. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y. Zheng, "Nus-wide: A real-world web image database from national university of singapore," in *ACM International Conference on Image and Video Retrieval*, 2009, pp. 1–9.

[47] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European Conference on Computer Vision*, 2014, pp. 740–755.

[48] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[49] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015, pp. 1–14.

[50] H. Robbins and S. Monro, "A stochastic approximation method," *The annals of mathematical statistics*, vol. 22, pp. 400–407, 1951.