Adversarial Examples for Hamming Space Search

Erkun Yang, Tongliang Liu, Cheng Deng, *Member, IEEE*, and Dacheng Tao, *Fellow, IEEE*

***Abstract*—Due to its strong representation learning ability and its facilitation of joint learning for representation and hash codes, deep learning-to-hash has achieved promising results and is becoming increasingly popular for the large-scale approximate nearest neighbor search. However, recent studies highlight the vulnerability of deep image classifiers to adversarial examples; this also introduces profound security concerns for deep retrieval systems. Accordingly, in order to study the robustness of modern deep hashing models to adversarial perturbations, we propose hash adversary generation (HAG), a novel method of crafting adversarial examples for Hamming space search. The main goal of HAG is to generate imperceptibly perturbed examples as queries, whose nearest neighbors from a targeted hashing model are semantically irrelevant to the original queries. Extensive experiments prove that HAG can successfully craft adversarial examples with small perturbations to mislead targeted hashing models. The transferability of these perturbations under a variety of settings is also verified. Moreover, by combining heterogeneous perturbations, we further provide a simple yet effective method of constructing adversarial examples for black-box attacks.**

***Index Terms*—Adversarial examples, deep neural network (DNN), hashing, image search.**

I. Introduction

THE EXPLOSION of large-scale and high-dimensional media data has led to renewed interest in efficient indexing and searching methods. However, since exact nearest neighbor search is typically time-consuming and unsuitable for big data applications, approximate nearest neighbor (ANN) search, which strikes a balance between search quality and computation efficiency, has begun to attract more and more attention.

Recently, due to its low time and space complexity, learning-to-hash has been widely studied for ANN search. State-of-the-art hashing methods usually adopt deep neural networks (DNNs) as hash functions. These networks can help to learn representations with rich semantic information and optimize the representation and hash code learning processes simultaneously. Although deep learning-based hashing methods have achieved promising results, recent research [1] points out that most DNN classifiers are vulnerable to adversarial examples, which also introduce substantial security concerns for existing deep hashing (DH) models.

In this paper, to study the robustness of modern DH models to adversarial perturbations, we propose a novel method, named hash adversary generation (HAG), for crafting adversarial examples for Hamming space search. More specifically, given an input image and a targeted hashing retrieval system, we aim to generate an adversarial version of the input such that the results it retrieves from the retrieval system will be semantically irrelevant to the original input image. In other words, the retrieval system is invalid for these generated adversarial

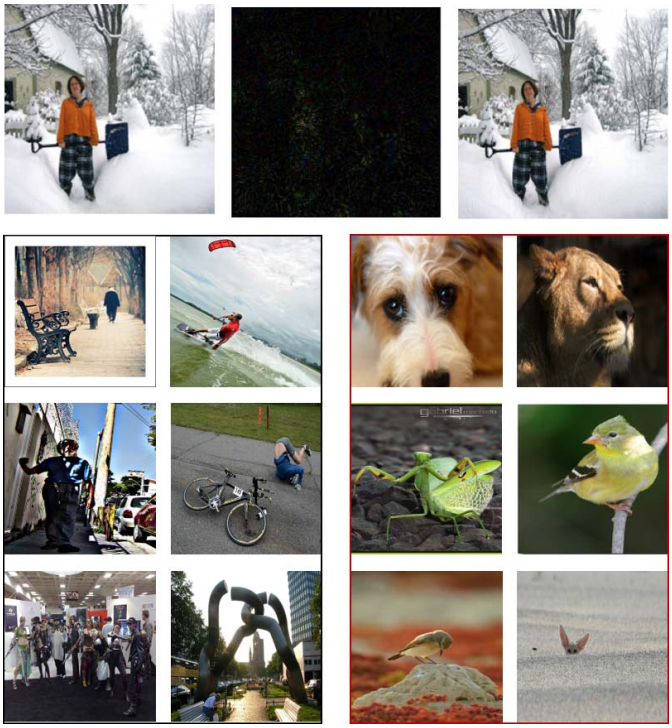


Fig. 1. Example of Hamming space search. (1) Top left: Natural image. (2) Top middle: Learned small adversarial perturbation, which is magnified by a factor of 10 for better visualization. (3) Top right: Corresponding adversarial example. (4) Bottom left: Top 6 retrieval results for the natural image, and according to the image labels, all of these retrieval results are relevant with the natural image. (5) Bottom right: Top 6 retrieval results for the adversarial examples, and according to the image labels, all of these retrieval results are irrelevant to the natural image.

Manuscript received August 20, 2018; revised October 28, 2018; accepted November 13, 2018. Date of publication December 11, 2018; date of current version February 25, 2020. This work was supported in part by the National Natural Science Foundation of China under Grant 61572388 and Grant 61703327, in part by the Key Research and Development Program-The Key Industry Innovation Chain of Shaanxi under Grant 2017ZDCXL-GY-05-04-02 and Grant 2017ZDCXL-GY-05-02, and in part by the Australian Research Council under Project FL-170100117, Project DP-180103424, and Project IH-180100002. This paper was recommended by Associate Editor Y. Zhang. *(Corresponding author: Cheng Deng.)*

E. Yang and C. Deng are with the School of Electronic Engineering, Xidian University, Xi'an 710071, China (e-mail: ekyang@stu.xidian.edu.cn; chdeng.xd@gmail.com).

T. Liu and D. Tao are with the UBTECH Sydney Artificial Intelligence Centre, Faculty of Engineering and Information Technologies, University of Sydney, Darlington, NSW 2008, Australia, and also with the School of Information Technologies, Faculty of Engineering and Information Technologies, University of Sydney, Darlington, NSW 2008, Australia (e-mail: tongliang.liu@sydney.edu.au; dacheng.tao@sydney.edu.au).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2018.2882908

examples as illustrated in Fig. 1. We hope that this paper will elicit some much-needed concerns for the security of modern retrieval systems and provide opportunities to better understand existing hashing-based retrieval methods.

We wish to point out that, compared with image classification, learning adversarial examples for Hamming space search is much more difficult. HAG aims to learn adversarial examples for which the retrieval results are semantically irrelevant to the original queries. However, since queries usually appear without labels, directly forcing the retrieval results for their adversarial versions to be semantically irrelevant to them rapidly becomes infeasible. To deal with this problem, we select an alternative and more feasible objective, which is to make the retrieval results for the adversarial examples semantically irrelevant to the retrieval results for the corresponding original inputs; this is equivalent to forcing the hash codes for the adversarial examples and the corresponding original inputs to be different. Accordingly, we can adopt a pairwise loss function to optimize our algorithm, which is easy to implement.

Directly optimizing the above loss function still requires us to contend with a severe *vanishing gradient* problem. Note that most existing adversarial example methods are based on back-propagation. Nevertheless, for traditional hashing methods, the sign function is usually required in order to output binary values, and this function cannot be directly optimized with the back-propagation algorithm. Although many existing methods adopt continuous activation functions, such as the tanh function, to approximate the sign function, a quantization loss is usually adopted to minimize the discrepancy between continuous values and binary codes, which, however, will induce the input features of the activation function to locate in saturation regions. As such, these methods are either infeasible or difficult to optimize for the learning of adversarial examples. To address this challenge, we propose a novel updating strategy for the hash activation function. More specifically, we first adopt a smoother activation function that can supply larger gradients for adversarial examples, allowing the optimization of our algorithm to be more effective. We then let the activation function evolve until it eventually becomes the original function. This strategy greatly relieves the *vanishing gradient* problem, as well as improving both the effectiveness and efficiency of our method.

To further improve the learning efficiency of our method, we also design a mask matrix for the learned continuous values. It should be noted that these continuous values will eventually be transferred to binary codes by applying the sign function. Continuing to optimize the continuous values, which already have different signs from the codes of the original images, will not affect the final results. Accordingly, we design a mask matrix to make our algorithm focus on the most important dimensions that have been proven to be able to further improve the experimental performance.

We can summarize our main contributions as follows.

1) We propose HAG, a novel method of learning adversarial examples to attack targeted hashing models. To the best of our knowledge, this is the very first work on learning adversarial examples for Hamming space search.
2) Since queries usually lack labels, directly forcing the retrieval results to be semantically irrelevant to them is infeasible. Accordingly, in this paper, we opt to design a novel objective to force the hash codes for adversarial examples to be dissimilar from those of the original examples. This ensures that the implementation of our method will be feasible.
3) We propose a novel updating strategy for the hash activation function, which first adopts a smoother activation function, then lets it evolve until it eventually becomes the original one. The proposed strategy greatly alleviates the *vanishing gradient* problem and improves both the learning effectiveness and efficiency of our method.
4) A mask matrix is adopted to make our algorithm focus on the most important dimensions, which is proven to be able to further improve the performance.
5) Experimental results on three popular benchmarks demonstrate that our method can successfully learn adversarial examples to greatly reduce the retrieval accuracy.
6) We demonstrate that the adversarial perturbation transferability exists to a large extent under a variety of settings and also provide a simple yet effective method to construct adversarial perturbations for black-box attacks.

In the rest of this paper, we review the relevant literature regarding Hamming space search and adversarial examples in Section II and present our novel HAG method in Section III. The experimental results are provided and analyzed in detail in Section IV, while concluding remarks are presented in Section V.

II. Related Work

In this section, we briefly review some representative hashing approaches and also provide a brief introduction to some related works on adversarial attack and defense.

A. Hashing for Similarity Search

Most existing hashing methods can be divided into two categories: 1) data-independent hashing, also called locality-sensitive hashing (LSH) [2]–[4] and 2) data-dependent hashing, also known as learn-to-hash [5]–[17]. For a comprehensive survey of the topic, we refer the readers to [18].

Data-independent hashing usually involves learning a family of hash functions independently from the training dataset. The seminal LSH presented in [2], which projects similar data into similar binary codes with high probabilities, formulates a paradigm for the LSH technique. Many extensions [3], [4] of this basic LSH concept have since been developed. However, since LSH methods cannot exploit the data distribution, they usually need a long hash bit to obtain a specific level of retrieval performance.

Compared to data-independent hashing methods, data-dependent hashing [19]–[25], which learns hash functions from training data, can usually achieve more promising results and has thus attracted more attention. Depending on whether

or not data labels are available, data-dependent hashing can be further organized into two groups, namely, unsupervised and supervised hashing methods. Unsupervised hashing methods [5], [6], [11], [12], [17] learn hash functions to map data points to binary codes using unlabeled data. Representative unsupervised hashing methods include iterative quantization (ITQ) [5], spectral hashing (SH) [6], discrete graph hashing (DGH) [17], spherical hashing (SpH) [26], anchor graph hashing (AGH) [11], and stochastic generative hashing (SGH) [12]. ITQ first utilizes principal component analysis to map the original data to a low-dimensional subspace and then learns an orthogonal rotation matrix to minimize the quantization error. SH learns hash functions by means of spectral graph partitioning, and an efficient spectral method is adopted to solve the relaxed graph partitioning problem. DGH proposes a discrete optimization framework, which explicitly deals with the discrete constraints, meaning that DGH can directly output binary descriptors. SpH tries to enforce the independent and balanced properties for binary codes and maps spatially coherent data points to similar compact codes by exploiting the hypersphere-based hashing functions. AGH constructs anchor graphs to approximate the inherent neighborhood structure of the training data; the computation of the graph's Laplacian eigenvectors can also be accelerated using this method. SGH adopts a variational inference framework and learns hash functions via the minimum description length principle. To avoid the optimization difficulty caused by binary constraints, SGH also develops a stochastic distributional gradient-based learning method.

For their part, supervised hashing approaches [27]–[34] aim to take advantage of the semantic labels or relevance information to mitigate the semantic gaps and learn more efficient binary codes. Typical supervised hashing methods include minimal loss hashing (MLH) [27], kernel-based supervised hashing (KSH) [30], fast supervised hashing [35], and fast supervised discrete hashing (FSDH) [31]. MLH utilizes structure prediction with latent variables and learns hash functions to preserve semantic similarities by exploiting a hinge-like loss function. KSH adopts inner products to approximate the binary code distance in Hamming space and learns hash functions by enforcing similar data points to have similar hash codes and dissimilar data points to have dissimilar hash codes. FSDH tries to accelerate the traditional supervised discrete hashing by proposing a regression of the class labels for training examples to binary codes.

Nowadays, DNNs have revolutionized many computer vision and machine learning tasks. Deep learning-to-hash [36]–[52] has also attracted significant attention. Most deep learning-based hashing methods adopt neural networks to integrate representation and hash code learning into a unified framework. Due to the powerful representation ability of DNNs and the joint optimization for features and hash codes, a variety of deep learning-based hashing methods have been proposed and have gone on to show promising results in both supervised and unsupervised settings. Specifically, for unsupervised DH, semantic hashing [36] adopts restricted Boltzmann machines as an auto-encoder network. Hash codes are then generated from the hidden layer and optimized to minimize a reconstruction loss. DH [43] adopts neural networks as a multiple hierarchical nonlinear transformation to project data points to binary codes in Hamming space, while a quantization loss between the continuous features and binary values is adopted to optimize the algorithm. DeepBit [39] utilized deep networks as hash functions and tried to preserve the similarity between input images and their rotated versions. Compared with unsupervised DH methods, supervised DH methods typically achieve more promising results. Convolutional neural networks-based hashing [46] learns hash functions in two stages: first, the hash codes for training data are obtained through the construction and decomposition of a pairwise similarity matrix. Then, a deep network is adopted to predict the learned hash codes as well as the discrete class labels of images. Hash codes for queries can be obtained from the learned deep networks. Network in network hashing [37] proposes a deep learning framework that uses three blocks: 1) a subnetwork to learn intermediate image representation; 2) a divide-and-encode module, which divides the intermediate image representations into multiple branches, each encoded into one hash bit; and 3) a triplet loss to preserve the semantic information in Hamming space. Deep pairwise-supervised hashing [32] proposes a pairwise loss function to map similar data pairs to similar hash codes and dissimilar data pairs to dissimilar hash codes. Similarly, deep supervised hashing (DSH) [42] also adopts a pairwise loss function. Moreover, the quantization error between real-valued outputs and binary codes is also minimized. HashNet [45] learns exact binary hash codes via a continuation method with convergence guarantees. Asymmetric DSH [50] learns hash functions by treating the hash code learning processes for the database and query set in an asymmetric way. Finally, deep Cauchy hashing [48] designs a pairwise cross-entropy loss based on the Cauchy distribution to solve the optimization difficulty for images within a small Hamming ball.

B. Adversarial Attack and Defense

The generation of adversarial examples for image classification tasks has been extensively studied. Recent research [1] first shows that adversarial examples crafted by adding small, specific worst-case perturbations can lead to misclassification by targeted neural networks. Then, a sizable body of research on various attack approaches [53]–[59] has been developed, exposing the underlying vulnerabilities of DNNs more than ever before. The pioneering fast gradient sign method (FGSM) [57], one of the most popular and efficient attack algorithms, uses the sign of gradients with respect to the inputs to learn adversarial examples. The projected gradient descent [58] is an another powerful attack method that is essentially a multistep variant of FGSM. In addition to image classification, adversarial attack for many other DNN-related tasks has also been actively studied, such as image captioning [60], semantic segmentation [59], visual question answering [54], machine translation [55], speech recognition [56], and medical prediction [53].

Given the vulnerability of DNN, increasing efforts have been made to build systems that are robust against adversarial

examples. Existing defense mechanisms can be grouped into two categories. The first type is to detect and remove malicious data before prediction [61], [62]. Carlini and Wagner [61] investigated ten recent proposals for adversarial example detection and showed that all of these methods are ineffective at dealing with newly constructed loss functions, which alleviates the application values of these defense methods. The second type of defense is to modify the DNN models to make them more robust against attacks. Szegedy *et al.* [1] and Goodfellow *et al.* [57] adopted adversarial training to augment the training data of the classifier with adversarial examples. Papernot *et al.* [63] used distillation techniques to train networks, which can greatly reduce the magnitude of the gradients used for adversarial example creation and increase the minimum feature numbers that need to be modified.

III. Adversarial Example Generation

In this section, we introduce the HAG algorithm. Given a targeted hashing model, HAG aims to learn an adversarial perturbation for a query image, whose nearest neighbors from the targeted hashing model are semantically irrelevant to the original query.

A. Overall Learning Objectives

Let $\boldsymbol{X} = \{\boldsymbol{x}_i | i = 1, 2, \ldots, N\}$ be an image set containing N images and $E(\cdot)$ be the targeted hashing model. Usually, hash codes for $\boldsymbol{x}_i$ can be obtained by

$$\begin{aligned} &\boldsymbol{b}_i = E(\boldsymbol{x}_i) = \operatorname{sign}(H(\boldsymbol{x}_i)) \\ &\text{s.t.} \quad \boldsymbol{b}_i \in \{1, -1\}^k \end{aligned} \tag{1}$$

where k is the hash bit length. $\boldsymbol{B} = [\boldsymbol{b}_1, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_N]$ represents the hash code for $\boldsymbol{X}$, which is usually obtained by a $\operatorname{sign}(\cdot)$ function with $\operatorname{sign}(v) = 1$ if $v > 0$, and $\operatorname{sign}(v) = -1$ otherwise. $H(\boldsymbol{x}_i)$ is the corresponding continuous feature before the $\operatorname{sign}(\cdot)$ function.

Most adversarial example methods are based on backpropagation. However, the $\operatorname{sign}(\cdot)$ function in (1) makes it difficult to optimize the hashing model. Accordingly, similar to many existing DH methods, we adopt the tanh function as a hash activation function to squeeze the continuous values so that they fall within the range $(-1, +1)$. Then, the hash code $\boldsymbol{b}_i$ can be approximated by

$$F(\boldsymbol{x}_i) = \tanh(H(\boldsymbol{x}_i)). \tag{2}$$

Note that, given a query image $\boldsymbol{x}_q$, our goal is to generate a corresponding adversarial example, $\hat{\boldsymbol{x}}_q$—example formed by applying a visually imperceptible perturbation to $\boldsymbol{x}_q$ such that the retrieval results for $\hat{\boldsymbol{x}}_q$ based on the targeted hashing model are semantically irrelevant to $\boldsymbol{x}_q$. To effectively learn $\hat{\boldsymbol{x}}_q$, in this paper, we enforce two important criteria. First, the perturbation added to the original image should be small enough so that it is imperceptible to human. Second, we force the learned hash code for $\hat{\boldsymbol{x}}_q$ to be dissimilar to that for the original example $\boldsymbol{x}_q$.

To achieve the above objectives, we formulate the adversarial image learning problem as follows:

$$\begin{aligned} &\min_{\hat{\boldsymbol{x}}_q} \ \mathcal{L}(\hat{\boldsymbol{x}}_{\boldsymbol{q}}, \boldsymbol{x}_q) = \left\| \frac{1}{m} F'(\boldsymbol{x}_q) F'(\hat{\boldsymbol{x}}_q)^\top + 1 \right\|^2 \\ &\text{s.t.} \ \ |\boldsymbol{x}_q - \hat{\boldsymbol{x}}_q| < \epsilon \end{aligned} \tag{3}$$

where $F'(\boldsymbol{x}_q)$ and $F'(\hat{\boldsymbol{x}}_{\boldsymbol{q}})$ are transformed from $F(\boldsymbol{x}_q)$ and $F(\hat{\boldsymbol{x}}_{\boldsymbol{q}})$ using a mask and m is the number of nonzero elements of the mask. ϵ is the maximum magnitude of the allowable perturbation for query images. In practice, we first optimize $\hat{\boldsymbol{x}}_{\boldsymbol{i}}$ by minimizing $\mathcal{L}(\hat{\boldsymbol{x}}_{\boldsymbol{q}}, \boldsymbol{x}_q)$, after which we clip $\hat{\boldsymbol{x}}_{\boldsymbol{q}}$ as follows:

$$\hat{\boldsymbol{x}}_{\boldsymbol{q}} = \min\{255, \boldsymbol{x}_q + \epsilon, \max(0, \boldsymbol{x}_q - \epsilon, \hat{\boldsymbol{x}}_{\boldsymbol{q}})\}. \tag{4}$$

These two processes are conducted iteratively until convergence.

In order to provide a better understanding of the objectives proposed in (3), we elaborate the details further in the following section.

B. Hamming Distance Maximization

HAG seeks to learn adversarial examples, the retrieval results of which are semantically irrelevant with the original examples. However, since queries usually lack labels, we cannot decide whether the retrieval results are semantically relevant to the queries. Thus, this objective is relatively difficult to formulate directly. Note that recent hashing-related works have made great progress and can usually achieve very high accuracy for nearest neighbor search. Therefore, we assume here that the given hash model $E(\cdot)$ can preserve the semantic information well, which means that, for natural images, semantically similar pairs will have similar hash codes, while semantically dissimilar pairs will have dissimilar hash codes. Based on this assumption, we can effectively modify the original objective to a new one, namely, to make the hash codes of adversarial examples dissimilar to those of the original examples. Accordingly, we formulate the following objective:

$$\min_{\hat{\boldsymbol{x}}_q} \ \mathcal{L}(\hat{\boldsymbol{x}}_{\boldsymbol{q}}, \boldsymbol{x}_q) = \left\| \frac{1}{m} F(\boldsymbol{x}_q) F(\hat{\boldsymbol{x}}_q)^\top + 1 \right\|^2. \tag{5}$$

Here, as indicated in [30], we adopt the inner products of hash codes as a surrogate for the Hamming distance. By optimizing (5), we can explicitly maximize the Hamming distance between $\boldsymbol{x}_q$ and $\hat{\boldsymbol{x}}_q$.

However, since the continuous values $F(\hat{\boldsymbol{x}}_q)$ will be transferred to binary codes via the sign function, if $F(\hat{\boldsymbol{x}}_q)$ and $F(\boldsymbol{x}_q)$ already have different signs for some dimensions with a large margin, continuing to optimize (5) in these dimensions will not change the final results, but will decrease the learning efficiency. Thus, to deal with this problem, we make our algorithm focus on the dimensions where $F(\hat{\boldsymbol{x}}_q)$ and $F(\boldsymbol{x}_q)$ have the same sign or have different signs with a small margin. Accordingly, the objective can be rewritten as

$$\min_{\hat{\boldsymbol{x}}_q} \ \mathcal{L}(\hat{\boldsymbol{x}}_{\boldsymbol{q}}, \boldsymbol{x}_q) = \left\| \frac{1}{m} F'(\boldsymbol{x}_q) F'(\hat{\boldsymbol{x}}_q)^\top + 1 \right\|^2 \tag{6}$$

where $F'(\boldsymbol{x}_q) = \boldsymbol{w}_q \odot F(\boldsymbol{x}_q)$ and $F'(\hat{\boldsymbol{x}}_{\boldsymbol{q}})^\top = \boldsymbol{w}_q \odot F(\hat{\boldsymbol{x}}_{\boldsymbol{q}})^\top$. The operation $\odot$ represents the Hadamard product and $\boldsymbol{w}_q$ is a mask for the qth example, which indicates the dimensions where our algorithm should focus. m is the number of nonzero

elements in $\boldsymbol{w}_q$. We can obtain $\boldsymbol{w}_q$ by

$$w_{qi} = \begin{cases} 1, & \text{if } \left|F_i(\hat{\boldsymbol{x}}_{\boldsymbol{q}}) - \text{sign}\left(F_i(\boldsymbol{x}_q)\right)\right| < 1 + t \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

where w_{qi}, $F_i(\hat{\boldsymbol{x}}_{\boldsymbol{q}})$, and $F_i(\boldsymbol{x}_q)$ are the elements for the corresponding vectors, while t is a threshold to control the margin between $F_i(\hat{\boldsymbol{x}}_q)$ and $\text{sign}(F_i(\boldsymbol{x}_q))$.

There still exists a serious problem related to the direct optimization of (6), namely, the *vanishing gradient* problem. Note that traditional hashing methods usually minimize the quantization error between continuous values $F(\hat{\boldsymbol{x}}_q)$ and their corresponding binary codes so that most elements of $F(\hat{\boldsymbol{x}}_q)$ may locate near 1 or -1. However, the gradients of the tanh function in these areas are almost zero. Besides, the gradient of $\mathcal{L}$ with regard to $\hat{\boldsymbol{x}}_q$ is obtained via the chain rule

$$\begin{aligned} \frac{\partial L}{\partial \hat{\boldsymbol{x}}_q} &= \frac{\partial L}{\partial F(\hat{\boldsymbol{x}}_q)} \frac{\partial F(\hat{\boldsymbol{x}}_q)}{\partial H(\hat{\boldsymbol{x}}_q)} \frac{\partial H(\hat{\boldsymbol{x}}_q)}{\partial \hat{\boldsymbol{x}}_q} \\ &= \frac{\partial L}{\partial F(\hat{\boldsymbol{x}}_q)} \frac{\partial \tanh\left(H(\hat{\boldsymbol{x}}_q)\right)}{\partial H(\hat{\boldsymbol{x}}_q)} \frac{\partial H(\hat{\boldsymbol{x}}_q)}{\partial \hat{\boldsymbol{x}}_q}. \end{aligned} \tag{8}$$

Zero gradients in the hash activation layer will greatly hinder the optimization of the adversarial examples.

To better analyze this problem, we randomly select 2000 examples from the MS-COCO, FLICKR25K, and NUSWIDE datasets and then forward-propagate them through the given targeted hashing model in order to get the input features for the hash activation function. We also calculate the derivative of the tanh function with respect to its input variables as follows:

$$\frac{\partial \tanh\left(H(\hat{\boldsymbol{x}}_q)\right)}{\partial H(\hat{\boldsymbol{x}}_q)} = 1 - \left(\tanh\left(H(\hat{\boldsymbol{x}}_q)\right)\right)^2. \tag{9}$$

Fig. 2(a) shows the histogram of the input features on the three datasets, as well as the derivative of the tanh function, from which we can see that most input features belong to the range of $[-20, 20]$. However, only values in the range of $[-5, 5]$ have relatively large gradients for the $\tanh(x)$ function. The frequency of the gradients for $\tanh(x)$ function with respect to these input features on the three datasets is also shown in Fig. 2(b), which clearly demonstrates that there are a large number of near-zero gradients.

To solve the above *vanishing gradient* problem, we found that, compared with $\tanh(x)$, $\tanh(\alpha x)$ (with $\alpha < 1$) has relatively large gradients in a wider range, as is also demonstrated in Fig. 2(a). Accordingly, in this paper, we adopt $\tanh(\alpha x)$ as the hash activation function. We first set α with a small value and then gradually enlarge it until it equals 1. Fig. 2(a) also shows that, when α equals 0.1, $\tanh(\alpha x)$ has relatively large gradients for most input features on all three datasets. Fig. 2(c) further shows the frequency of the gradients for $\tanh(0.1x)$ with respect to the input features and also demonstrates this point. Therefore, in this paper, we optimize HAG by minimizing (3) as follows:

$$F(\boldsymbol{x}_i) = \tanh(\alpha H(\boldsymbol{x}_i)) \tag{10}$$

where α is first set at 0.1 and then gradually enlarged until eventually becoming 1.

It should be noted that, at first sight, the proposed strategy for updating the activation function appears similar to that employed in HashNet [45]. However, the motivations and manners of updating used in this paper and HashNet are very different. For HashNet, this strategy is used to gradually approximate the sign function in order to reduce the quantization error, and α is first set at 1 and updated to be infinity; in our method, however, the strategy is used to relieve the *vanishing gradient* problem, and α is first set to be a relatively small value and is updated toward 1.

C. Targeted Hashing Model

We construct the targeted hashing model based on the most common settings. Specifically, the VGG16 network [64] is adopted as the base model. We modified the original VGG16 model by replacing the last fully connected layer with a new fully connected layer with k hidden units. The tanh function is used as the hash activation layer to squeeze the continuous values to within the range $[-1, +1]$. We select a pairwise loss function to optimize this model, and use the inner products of hash codes as a surrogate for the Hamming distance. Concretely, the targeted hashing model is optimized using the following objective:

$$\begin{aligned} \min_{\Theta} \ & \mathcal{J}(\Theta) = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \left(R_{ij} - S_{ij}\right)^2 \\ \text{s.t.} \ & R_{ij} = \frac{1}{K} F(\boldsymbol{x}_i)^\top F(\boldsymbol{x}_i), \quad \boldsymbol{v}_i = \tanh(H(\boldsymbol{x}_i)) \end{aligned} \tag{11}$$

where the parameters of the VGG16 network are represented as Θ, while $H(\boldsymbol{x}_i)$ denotes the continuous values before the hash activation function for $\boldsymbol{x}_i$. S_{ij} is the element of a pairwise similarity matrix $\boldsymbol{S}$, which can be directly defined or obtained by the image labels as follows:

$$S_{ij} = \begin{cases} 1, & \text{if } \boldsymbol{x}_i \text{ and } \boldsymbol{x}_j \text{ are similar} \\ -1, & \text{otherwise.} \end{cases} \tag{12}$$

Although this basic hash model is very simple, it achieves results comparable with current state-of-the art methods. The mean average precision (MAP) of the top 1000 retrieved examples for this basic model on three popular datasets is presented in the first row of Table I (Iteration 0). Note that we do not aim to propose a state-of-the-art hashing method; instead, we implement this model based on the most common settings to demonstrate the effectiveness of the proposed HAG.

IV. Experiments

We evaluate our method on three popular datasets: 1) MS-COCO; 2) FLICKR25K; and 3) NUSWIDE. In this section, we provide extensive evaluations of the proposed method and demonstrate its effectiveness under different settings. We first introduce the datasets, evaluation protocols, and implementation details, after which we present and discuss the experimental results in depth.

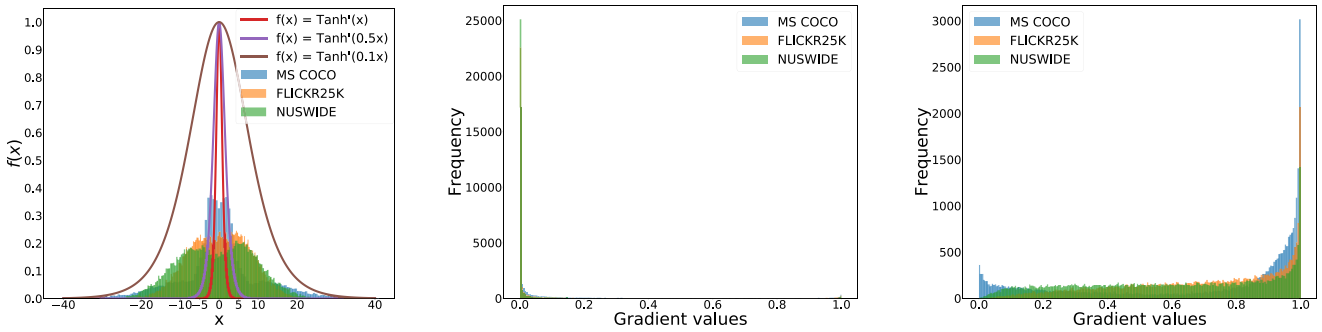


Fig. 2. (a) Histogram of the input features for the hash activation layer for 16 bit on three datasets, and the derivative of $\tanh(\alpha x)$ with different α. (b) Frequency of the gradients for $\tanh(x)$ with regard to the input features for 16 bit. (c) Frequency of the gradients for $\tanh(0.1x)$ with regard to the input features for 16 bit.

TABLE I
MAP FOR A DIFFERENT NUMBER OF BITS ON THE THREE DATASETS AT DIFFERENT ITERATIONS

Training Iteration	MS-COCO				FLICKR25K				NUSWIDE			
	16bit	32bit	64bit	128bit	16bit	32bit	64bit	128bit	16bit	32bit	64bit	128bit
0	0.7114	0.7365	0.7460	0.7494	0.9280	0.9281	0.9262	0.9295	0.8671	0.8847	0.8783	0.8854
10	0.6920	0.7163	0.7237	0.7353	0.8969	0.8954	0.9069	0.9130	0.8281	0.8556	0.8433	0.8505
100	0.4208	0.4451	0.4589	0.4890	0.5535	0.5221	0.5425	0.5953	0.4794	0.5006	0.5132	0.5209
500	0.2115	0.2072	0.2046	0.2323	0.1836	0.1496	0.1501	0.1712	0.1748	0.1961	0.1943	0.1936
1000	0.1617	0.1398	0.1392	0.1698	0.1611	0.1274	0.1164	0.1293	0.1355	0.1506	0.1427	0.1443
1500	0.1125	0.0979	0.1010	0.1144	0.1731	0.1188	0.1075	0.1352	0.1257	0.1486	0.1177	0.1192
2000	0.1081	0.0973	0.0916	0.0998	0.1657	0.1164	0.1190	0.1315	0.1084	0.1044	0.1054	0.1150

Algorithm 1: HAG

Input: input image $\mathbf{x_i}$, targeted hashing model $\boldsymbol{E}(\cdot)$, a sequence $\alpha_0 < \alpha_0 < \cdots < \alpha_m = 1$, hyper-parameter t.
Output: Adversarial example $\hat{\boldsymbol{x}}_i$.
Procedure:
Initialize $\hat{\boldsymbol{x}}_i$ with $\boldsymbol{x}_i$.
Calculate $F(\boldsymbol{x}_i)$ for $\mathbf{x_i}$ by forward-propagating it through the targeted hashing model.
for *stage* $t = 0$ ***to*** m **do**
1. Set $\alpha = \alpha_t$;
2. Calculate $F(\hat{\boldsymbol{x}}_i)$ by forward-propagating $\hat{\mathbf{x}}_i$ through the targeted hashing model;
3. Calculate the mask matrix $\boldsymbol{w}_i$ through (7);
4. Optimize $\hat{\boldsymbol{x}}_i$ through (3);
5. Clip $\hat{\boldsymbol{x}}_i$ through (4);
6. Set converged HAG to next stage.

end
return $\hat{\boldsymbol{x}}_i$

A. Datasets

MS-COCO is a public dataset for image recognition, segmentation, and image captioning, which contains about 82 783 training images and 40 504 testing images. Each image is labeled with at least one of the 80 categories. We combine the training and testing set and obtain 122 218 images by pruning images with no category information. For the training of the targeted hashing model, we randomly select 10 000 images as the training set; for the generation of adversarial examples, 200 images are randomly selected as the test set, while the remaining images are used as the database.

FLICKR25K contains 25 000 images collected from the Flickr website. Each image is manually annotated with at least one of the 24 provided labels. For the training of the targeted hashing model, we randomly select 5000 images as a training set. For the generation of the adversarial examples, 200 images are randomly selected as the test set, while the remaining images are used as the database.

NUSWIDE contains 269 648 images, each of which is annotated with multiple labels related to 81 different concepts. We select a subset comprising the ten most popular concepts. For the training of the targeted hashing model, we randomly select 5000 images as a training set. For the generation of the adversarial examples, 200 images are randomly selected as the test set, while the remaining images are used as the database.

B. Evaluation Protocols

We adopt three evaluation criteria to evaluate the effectiveness of the generated adversarial examples: 1) MAP; 2) precision–recall (PR); and 3) topN-precision.

MAP is one of the most widely used criteria for measuring the performance of modern retrieval systems. Given a query and its top T ranked retrieval results, its average precision (AP) can be obtained by

$$\mathrm{AP} = \frac{1}{N}\sum_{v=1}^{V} P(v)\delta(v) \tag{13}$$

where N is the number of relevant examples in the database for the query and $P(v)$ denotes the precision for the top v

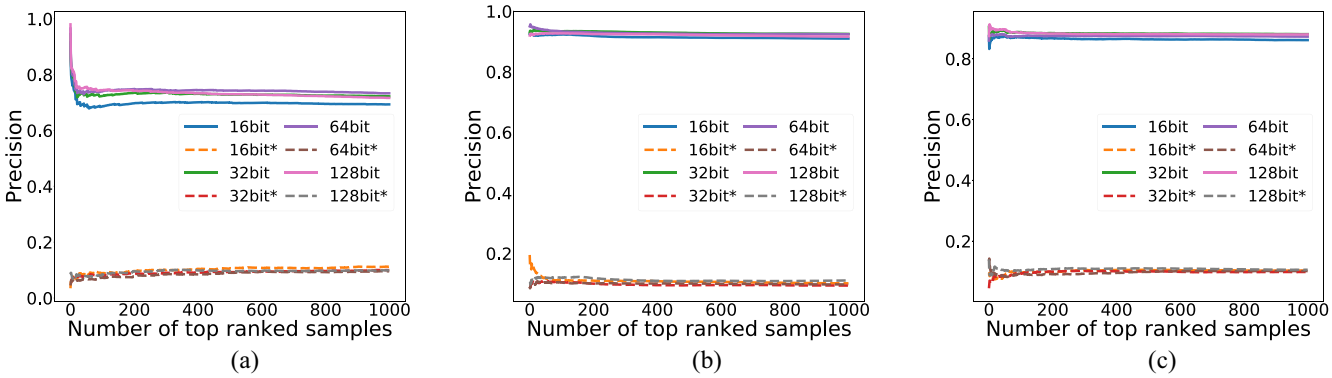


Fig. 3. TopN-precision. (a) TopN-precision curves on MS-COCO. (b) TopN-precision curves on FLICKR25K. (c) TopN-precision curves on NUSWIDE.

retrieved examples. $\delta(v) = 1$ if the vth retrieval example is relevant to the current query, otherwise $\delta(v) = 0$. MAP is defined as the average of APs for all queries. V is set to 1000 in the following experiments. Two examples are defined to be relevant if they share at least one common label, otherwise they are defined to be irrelevant. PR reports the recall and the corresponding precision values. The topN-precision shows the average precisions for all the queries, with different numbers of retrieved examples, and N is set to 1000 in our experiments.

C. Implementation Details

Our method is implemented using the open source TensorFlow [65]. The targeted hashing model is optimized using mini-batch stochastic gradient descent (SGD) with momentum. The mini-batch size is set to 24 and momentum to 0.9. Training images are resized to 224×224 before being used as the inputs. The first 15 layers of our neural network are fine-tuned from the model pretrained with ImageNet, while the last fully connected layer is learned from scratch. The learning rate is fixed at 0.001. For the learning of the adversarial examples, we use SGD with momentum to optimize our algorithm; the momentum is set at 0.9. All parameters are initialized using the pretrained targeted hashing model and fixed when training. The learning rate is set at 500. We train each example for 2000 iterations. For the first 1000 iterations, parameter α is set as 0.1; for the last 1000 iterations, α is changed to the next figure from the list $[0.2, 0.3, 0.5, 0.7, 1.0]$ for every 200 iterations. The threshold ϵ is set at 0.039 if images are normalized in $[0, 1]$ and 10 if they are not normalized.

D. Results

We first evaluate the test set and its adversarial versions across different numbers of hash bits on the three datasets, using MAP criteria to provide a global comparison. We then present the topN-precision and PR curves with the number of hash bits fixed at 32 for a more comprehensive evaluation.

Table I shows the MAP values for the adversarial examples in different training iterations on MS-COCO, FLICKR25K, and NUSWIDE, with the hash bit number varying from 16 to 128. Adversarial examples at iteration 0 correspond to the original examples without adversarial perturbations applied. From Table I, we can observe that HAG can generate adversarial examples that greatly reduce the MAP values for all numbers of hash bits on the three datasets. More specifically, compared to the original examples, the adversarial versions in iteration 2000 reduce the MAP values by 63.66%, 79.48%, and 77.06% on average for MS-COCO, FLICKR25K, and NUSWIDE, respectively. From Table I, we can also see that the MAP results reduce quickly in the first 500 iterations, and the adversarial examples in the 500th iteration can already obtain relatively low MAP values for all cases. Continuing to optimize HAG can further reduce the MAP values until the 1500th iteration. Moreover, the results for examples in the 1500 and 2000th iterations are similar. These results imply that we can generate adversarial examples with 500 iterations if time is limited, or with 1500 to 2000 iterations to obtain lower MAP results.

Fig. 3 shows the topN-precision curves, while Fig. 4 shows the PR curves. Both results are evaluated with the original examples and their adversarial counterparts with 32 hash bit length for the three datasets, respectively. For each figure, 16-bit denotes the results for original images with 16 hash bit length, while 16-bit* denotes the results for adversarial images with 16 hash bit length; the symbols are similar for other hash bit lengths. From Fig. 3, we can observe that the adversarial examples obtained from HAG consistently achieve very low precisions for the top 1000 results. Note that, for retrieval systems, the top retrieval results are more important than the others. Thus, our method can effectively attack the targeted hashing model. Results from Fig. 4 also show that the retrieval results for the adversarial examples are semantically irrelevant to the original inputs, further verifying the effectiveness of our method.

In order to provide a better understanding of our method, we present some natural images, the learned perturbations, the corresponding adversarial examples, and their retrieval results in Fig. 5.

E. Perceptibility

Following [1], we calculate the perceptibility of the adversarial perturbations defined by $p = \left([1/M] \sum_m \|\boldsymbol{x} - \hat{\boldsymbol{x}}\|^2\right)^{1/2}$, where M is the pixel number, while $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$ are all normalized

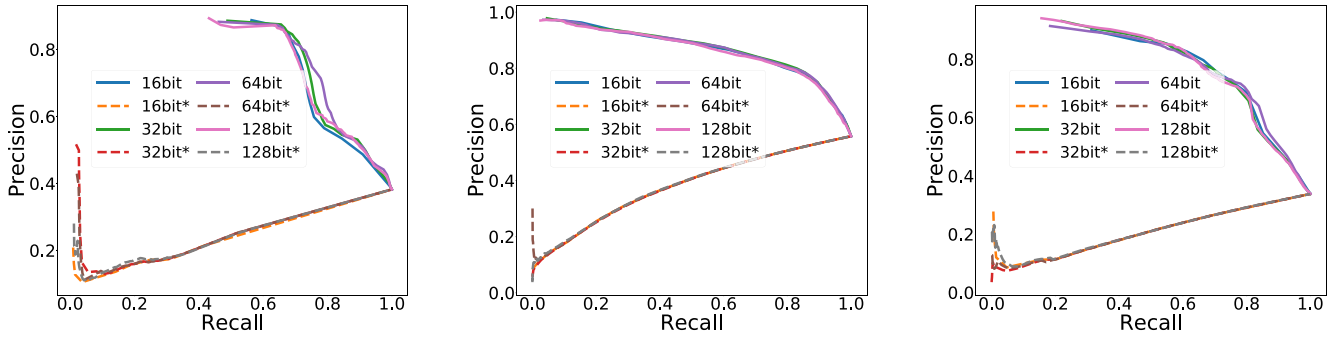


Fig. 4. PR curves on (a) MS-COCO, (b) FLICKR25K, and (c) NUSWIDE.

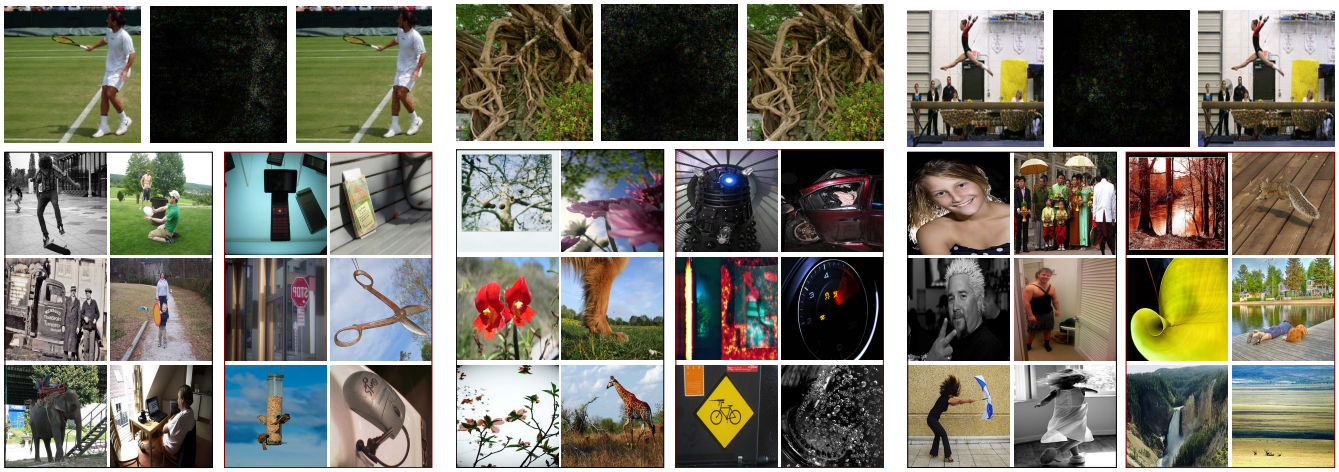


Fig. 5. Retrieval examples for (a) MS-COCO, (b) FLICKR25K, and (c) NUSWIDE. For each subfigure, the top left is a natural image, top middle is the learned small adversarial perturbation (magnified by a factor of 10 for better visualization), top right is the corresponding adversarial example, bottom left shows the top 6 retrieval results for the natural image, and bottom right shows the top 6 retrieval results for the adversarial examples.

TABLE II
PERCEPTIBILITY OF THE ADVERSARIAL PERTURBATIONS FOR DIFFERENT NUMBER OF BITS ON THE THREE DATASETS ($\times 10^{-3}$)

Training Iterations	MS-COCO				FLICKR25K				NUSWIDE			
	16bit	32bit	64bit	128bit	16bit	32bit	64bit	128bit	16bit	32bit	64bit	128bit
500	5.6534	5.3702	5.3699	5.4753	2.9729	4.0256	4.5933	4.2309	6.1685	7.2286	7.0896	7.5493
1000	4.5197	4.3348	4.4601	4.3233	5.5937	5.6706	5.9363	6.3292	5.1285	4.0509	5.4137	5.2696
1500	5.1501	5.0077	5.2496	4.9522	5.1544	5.2549	5.4087	5.7111	5.5897	5.9327	5.9700	6.6325
2000	4.3842	5.4683	4.7079	5.3325	6.5774	6.7359	7.3211	7.3168	4.7759	5.0651	5.4601	5.5000

TABLE III
MAP FOR HAG* AND HAG. HAG* IS THE MODIFIED ALGORITHM WITH tanh(x) AS THE HASH ACTIVATION FUNCTION

Method	FLICKR25K			
	16 bits	32 bits	64 bits	128 bits
HAG*	0.3004	0.1963	0.2410	0.1851
HAG	**0.1657**	**0.1164**	**0.1190**	**0.1315**

in [0, 1]. The perceptibility values are averaged over the entire test set. The results for all three datasets with different numbers of hash bits in the 500th, 1500th, and 2000th iterations are reported in Table II. From these results, we can see that all these values are very small, which ensures the imperceptibility of the learned perturbations. The examples in Fig. 5 also verify this point.

F. Ablation Study

We next go deeper to study the efficacy of the proposed hash activation function updating strategy and the mask matrix in Hamming distance maximization. First, we investigate a variant of HAG: HAG*, variant with tanh(x) as the hash activation function without updating. The MAP results of HAG and HAG* are compared in Table III, from which we can see that HAG consistently outperforms HAG* by margins of 13.47%, 7.99%, 12.20%, and 5.36% on the MS-COCO dataset, with 16, 32, 64, and 128 hash bit length, respectively. In fact, HAG*

TABLE IV
TRANSFER MAP RESULTS FOR THE MS-COCO DATASET. DPH-VGG16, DPH-VGG16*, DPH-VGG19, AND DPH-VGG19* ARE SELECTED AS FOUR BASIC MODELS TO LEARN ADVERSARIAL PERTURBATIONS, AND HASHNET-RESNET50 AND HASHNET-RESNET50* ARE ADOPTED AS TWO BLACK-BOX MODELS. ALL MODELS ARE EVALUATED ON THE TEST SET AND ITS ADVERSARIAL VERSIONS WITH DIFFERENT PERTURBATIONS

Perturbations	**DPH-VGG16**	**DPH-VGG16***	**DPH-VGG19**	**DPH-VGG19***	**HashNet-ResNet50**	**HashNet-ResNet50***
None	0.7365	0.7460	0.7279	0.7458	0.8042	0.8536
DPH-VGG16 (r_1)	0.0973	0.1138	0.4392	0.5374	0.7640	0.8043
DPH-VGG16* (r_2)	0.1036	0.0916	0.4166	0.5117	0.7673	0.8008
DPH-VGG19 (r_3)	0.5015	0.4968	0.0790	0.1197	0.7603	0.8040
DPH-VGG19* (r_4)	0.4707	0.4931	0.0988	0.1059	0.7573	0.8035
$r_1 + r_2$	0.0808	0.0753	0.1890	0.2535	0.7034	0.7227
$r_1 + r_3$	0.1002	0.1030	0.0830	0.1110	0.6995	0.7413
$r_1 + r_4$	0.0944	0.0984	0.0902	0.1106	0.7126	0.7361
$r_1 + r_2 + r_3 + r_4$	0.0776	0.0720	0.0703	0.0803	0.6252	0.6609

TABLE V
TRANSFER MAP RESULTS FOR THE FLICKR25K DATASET. DPH-VGG16, DPH-VGG16*, DPH-VGG19, AND DPH-VGG19* ARE SELECTED AS FOUR BASIC MODELS TO LEARN ADVERSARIAL PERTURBATIONS, AND HASHNET-RESNET50 AND HASHNET-RESNET50* ARE ADOPTED AS TWO BLACK-BOX MODELS. ALL MODELS ARE EVALUATED ON THE TEST SET AND ITS ADVERSARIAL VERSIONS WITH DIFFERENT PERTURBATIONS

Perturbations	**DPH-VGG16**	**DPH-VGG16***	**DPH-VGG19**	**DPH-VGG19***	**HashNet-ResNet50**	**HashNet-ResNet50***
None	0.9281	0.9262	0.9078	0.9256	0.9252	0.9319
DPH-VGG16 (r_1)	0.1164	0.1233	0.6845	0.7244	0.8891	0.8945
DPH-VGG16* (r_2)	0.1212	0.1190	0.6203	0.6789	0.8789	0.8866
DPH-VGG19 (r_3)	0.6248	0.6329	0.1151	0.1351	0.8864	0.9015
DPH-VGG19* (r_4)	0.5289	0.5726	0.1245	0.1296	0.8845	0.8949
$r_1 + r_2$	0.1035	0.1070	0.3060	0.3815	0.8055	0.8304
$r_1 + r_3$	0.1236	0.1283	0.1143	0.1365	0.8140	0.8304
$r_1 + r_4$	0.1172	0.1261	0.1267	0.1320	0.8149	0.8350
$r_1 + r_2 + r_3 + r_4$	0.1166	0.1152	0.1077	0.1199	0.7035	0.7043

TABLE VI
TRANSFER MAP RESULTS FOR THE NUSWIDE DATASET. DPH-VGG16, DPH-VGG16*, DPH-VGG19, AND DPH-VGG19* ARE SELECTED AS FOUR BASIC MODELS TO LEARN ADVERSARIAL PERTURBATIONS, AND HASHNET-RESNET50 AND HASHNET-RESNET50* ARE ADOPTED AS TWO BLACK-BOX MODELS. ALL MODELS ARE EVALUATED ON THE TEST SET AND ITS ADVERSARIAL VERSIONS WITH DIFFERENT PERTURBATIONS

Perturbations	**DPH-VGG16**	**DPH-VGG16***	**DPH-VGG19**	**DPH-VGG19***	**HashNet-ResNet50**	**HashNet-ResNet50***
None	0.8847	0.8783	0.8733	0.8726	0.8818	0.8984
DPH-VGG16 (r_1)	0.1044	0.1148	0.6314	0.6305	0.8419	0.8456
DPH-VGG16* (r_2)	0.1130	0.1054	0.6029	0.6020	0.8429	0.8421
DPH-VGG19 (r_3)	0.6541	0.6412	0.1012	0.1123	0.8516	0.8561
DPH-VGG19* (r_4)	0.6301	0.5946	0.1076	0.0946	0.8491	0.8596
$r_1 + r_2$	0.0957	0.0937	0.2899	0.3077	0.7460	0.7408
$r_1 + r_3$	0.1050	0.1041	0.1024	0.1100	0.7633	0.7754
$r_1 + r_4$	0.1090	0.1130	0.0949	0.0959	0.7694	0.7692
$r_1 + r_2 + r_3 + r_4$	0.1046	0.0900	0.0774	0.0875	0.6198	0.6143

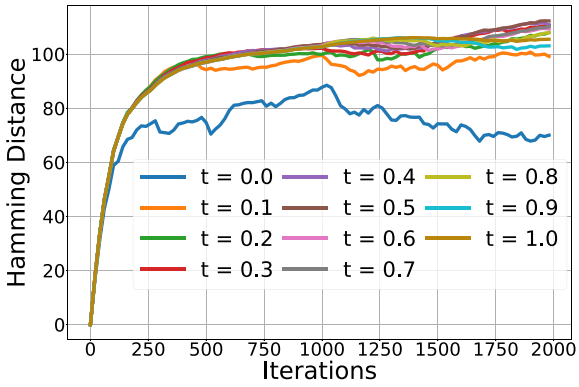


Fig. 6. Hamming distance with iterations.

failed to generate adversarial perturbations for many examples since these are harder to be optimized due to the *vanishing gradient* problem. The results in Table III clearly show the superiority of our proposed hash activation function updating strategy.

To analyze the efficacy of the proposed mask strategy, we conduct experiments by varying the threshold t from 0 to 1. Fig. 6 presents the average Hamming distance between the original examples and the corresponding adversarial examples on the MS-COCO dataset. From this figure, we can see that the algorithms with t values set from 0.3 to 0.7 can achieve larger Hamming distances after 2000 iterations than other values. In our experiments, we set t as 0.5.

G. Transferring Adversarial Perturbations

In this section, we study the transferability of the adversarial perturbations. Note that, in real applications, hashing models may adopt different numbers of hash bits, different deep networks, and even different loss functions. Accordingly,

Fig. 7. Original images and their adversarial versions with perturbations from different sources, which are denoted from the leftmost column.

we investigate the performance with adversarial perturbations from a variety of source models. We use DPH-VGG16 to denote our targeted hashing model with 32 hash bit length, and DPH-VGG16* with 64 hash bit length. We select the VGG19 network [64] to investigate the transferability of the learned perturbations between different networks. The corresponding methods are denoted as DPH-VGG19 and DPH-VGG19*. HashNet-ResNet50 and HashNet-ResNet50* are models for HashNet [45] with ResNet50 as the basic model and are used to test the black-box attack performance.

Quantitative results are summarized in Tables IV–VI. In the following, we analyze these results by organizing them into four groups. We first discuss the results for *cross-hash bit* transfer and *cross-network* transfer, after which we study the performance by combining heterogeneous perturbations. Finally, we evaluate HAG under black-box settings.

1) Cross-Hash Bit Transfer: Cross-hash bit transfer refers to applying the perturbations computed from one number of hash bits to another number of hash bits based on the same architecture. We observe that transferability greatly exists in this case. Specifically, the adversarial perturbations based on the same architecture with different numbers of hash bits can usually achieve very similar results. For example, applying the adversarial perturbations generated from DPH-VGG16 to attack DPH-VGG16* and DPH-VGG16 itself on NUSWIDE can achieve similarly low MAP results (11.64% and 12.33%). Similar phenomena are also observed across other hash bit lengths and networks. From these results, we conclude that the learned adversarial perturbations are closely related to the network architectures. Similar results are also obtained in [59] for semantic segmentation and object detection.

2) Cross-Network Transfer: Cross-network transfer refers to applying the perturbations computed from one deep network to another deep network. We observe that transferability also exists in this case. For example, when we adopt the adversarial perturbations generated from DPH-VGG16* to attack DPH-VGG19 on NUSWIDE, a 27.04% performance drop is observed (from 87.33% to 60.29%). Similar phenomena can be observed across different hash bit lengths and networks. More detailed results are presented in Tables III–V.

3) Combining Heterogeneous Perturbations: To further evaluate the transferability of the learned adversarial

perturbations, we combine perturbations from different models and test their performance on a targeted model. Results show that, compared with applying the perturbations separately, combining perturbations from different numbers of hash bits to attack models with the same architecture does not obviously improve the performance. This may be because the single perturbation can already attack the model very effectively. However, when we apply the combined perturbations to attack models with different architectures, we can gain very large performance improvements. For example, by combining the perturbations from DPH-VGG16 and DPH-VGG16* to attack DPH-VGG19 on NUSWIDE, we can obtain an MAP drop of 58.34%; when these perturbations are applied separately, the drops are 24.19% and 27.04%, which are much lower than the combined version. Adversarial examples with perturbations from different sources are presented in Fig. 7.

4) Black-Box Attack: In this section, we investigate the performance of HAG under the so-called black-box attack settings [66], in which we do not know the details (architecture, loss function, etc.) of the targeted model. From the results in Tables III–V, we can arrive at a simple but effective way of generating the adversarial perturbations, namely, combining different perturbations from known models. As an example, if we combine the perturbations from DPH-VGG16, DPH-VGG16*, DPH-VGG19, and DPH-VGG19* to attack HashNet-ResNet50 and HashNet-ResNet50*, a significant accuracy drop can be observed for all three datasets.

V. Conclusion

In this paper, we proposed HAG to generate adversarial examples for Hamming space search. Our extensive experiments demonstrate that HAG can successfully attack targeted hashing models, while perceptibility remains at a relatively low level. We further demonstrate that the learned perturbations exhibit high transferability under a variety of settings, as well as that the transferability is much more significant across different hash bit lengths for the same architecture. Moreover, by combining heterogeneous perturbations, we also provide a simple yet effective method for black-box attacks. We hope that this paper can help to improve the robustness of modern DH-based retrieval systems and can also facilitate a better understanding of these models.

References

[1] C. Szegedy *et al.*, "Intriguing properties of neural networks," in *Proc. ICLR*, 2014.

[2] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," in *Proc. 47th Annu. IEEE Symp. Found. Comput. Sci. (FOCS)*, 2006, pp. 459–468.

[3] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 6, pp. 1092–1104, Jun. 2012.

[4] A. Joly and O. Buisson, "A posteriori multi-probe locality sensitive hashing," in *Proc. 16th ACM Int. Conf. Multimedia*, 2008, pp. 209–218.

[5] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin, "Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 12, pp. 2916–2929, Dec. 2013.

[6] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 1753–1760.

[7] X. Li, B. Zhao, and X. Lu, "Key frame extraction in the summary space," *IEEE Trans. Cybern.*, vol. 48, no. 6, pp. 1923–1934, Jun. 2018.

[8] J. Xie, G. Dai, F. Zhu, L. Shao, and Y. Fang, "Deep nonlinear metric learning for 3-D shape retrieval," *IEEE Trans. Cybern.*, vol. 48, no. 1, pp. 412–422, Jan. 2018.

[9] X. Liu, L. Huang, C. Deng, B. Lang, and D. Tao, "Query-adaptive hash code ranking for large-scale multi-view visual search," *IEEE Trans. Image Process.*, vol. 25, no. 10, pp. 4514–4524, Oct. 2016.

[10] X. Liu, C. Deng, Y. Mu, and Z. Li, "Boosting complementary hash tables for fast nearest neighbor search," in *Proc. AAAI*, 2017, pp. 4183–4189.

[11] W. Liu, J. Wang, S. Kumar, and S.-F. Chang, "Hashing with graphs," in *Proc. 28th Int. Conf. Mach. Learn. (ICML)*, 2011, pp. 1–8.

[12] B. Dai, R. Guo, S. Kumar, N. He, and L. Song, "Stochastic generative hashing," in *Proc. ICML*, 2017, pp. 913–922.

[13] S. Wang, C. Li, and H.-L. Shen, "Distributed graph hashing," *IEEE Trans. Cybern.*, to be published, doi: 10.1109/TCYB.2018.2816791.

[14] W. W. Y. Ng, X. Tian, W. Pedrycz, X. Wang, and D. S. Yeung, "Incremental hash-bit learning for semantic image retrieval in nonstationary environments," *IEEE Trans. Cybern.*, to be published, doi: 10.1109/TCYB.2018.2846760.

[15] Y. Liu *et al.*, "Deep self-taught hashing for image retrieval," *IEEE Trans. Cybern.*, to be published, 10.1109/TCYB.2018.2822781.

[16] L. Chi, B. Li, X. Zhu, S. Pan, and L. Chen, "Hashing for adaptive real-time graph stream classification with concept drifts," *IEEE Trans. Cybern.*, vol. 48, no. 5, pp. 1591–1604, May 2018.

[17] W. Liu, C. Mu, S. Kumar, and S.-F. Chang, "Discrete graph hashing," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3419–3427.

[18] J. Wang, H. Shen, J. Song, and J. Ji, "Hashing for similarity search: A survey," *CoRR*, vol. abs/1408.2927, 2014.

[19] J. Song, L. Gao, L. Liu, X. Zhu, and N. Sebe, "Quantization-based hashing: A general framework for scalable image and video retrieval," *Pattern Recognit.*, vol. 75, pp. 175–187, Mar. 2018.

[20] Z. Qiu, Y. Pan, T. Yao, and T. Mei, "Deep semantic hashing with generative adversarial networks," in *Proc. ACM 40th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2017, pp. 225–234.

[21] C. Ma, I. W. Tsang, F. Shen, and C. Liu, "Error correcting input and output hashing," *IEEE Trans. Cybern.*, to be published, doi: 10.1109/TCYB.2017.2785621.

[22] L. Liu, M. Yu, and L. Shao, "Unsupervised local feature hashing for image similarity search," *IEEE Trans. Cybern.*, vol. 46, no. 11, pp. 2548–2558, Nov. 2016.

[23] X. Liu, B. Du, C. Deng, M. Liu, and B. Lang, "Structure sensitive hashing with adaptive product quantization," *IEEE Trans. Cybern.*, vol. 46, no. 10, pp. 2252–2264, Oct. 2016.

[24] C. Deng, X. Liu, Y. Mu, and J. Li, "Large-scale multi-task image labeling with adaptive relevance discovery and feature hashing," *Signal Process.*, vol. 112, pp. 137–145, Jul. 2015.

[25] C. Li *et al.*, "Self-supervised adversarial hashing networks for cross-modal retrieval," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4242–4251.

[26] J.-P. Heo, Y. Lee, J. He, S.-F. Chang, and S.-E. Yoon, "Spherical hashing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2012, pp. 2957–2964.

[27] M. Norouzi and D. M. Blei, "Minimal loss hashing for compact binary codes," in *Proc. ICML*, 2011, pp. 353–360.

[28] E. Yang *et al.*, "Pairwise relationship guided deep hashing for cross-modal retrieval," in *Proc. AAAI*, 2017, pp. 1618–1625.

[29] J. Song, Y. Yang, X. Li, Z. Huang, and Y. Yang, "Robust hashing with local models for approximate similarity search," *IEEE Trans. Cybern.*, vol. 44, no. 7, pp. 1225–1236, Jul. 2014.

[30] W. Liu, J. Wang, R. Ji, Y. Jiang, and S.-F. Chang, "Supervised hashing with kernels," in *Proc. CVPR*, 2012, pp. 2074–2081.

[31] J. Gui, T. Liu, Z. Sun, D. Tao, and T. Tan, "Fast supervised discrete hashing," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 2, pp. 490–496, Feb. 2018.

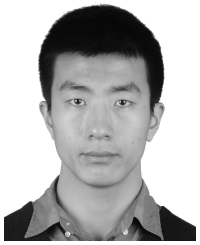
[32] W.-J. Li, S. Wang, and W.-C. Kang, "Feature learning based deep supervised hashing with pairwise labels," in *Proc. 25th Int. Joint Conf. Artif. Intell.*, Jul. 2016, pp. 1711–1717.

[33] F. Shen *et al.*, "Asymmetric binary coding for image search," *IEEE Trans. Multimedia*, vol. 19, no. 9, pp. 2022–2032, Sep. 2017.

[34] C. Deng, X. Tang, J. Yan, W. Liu, and X. Gao, "Discriminative dictionary learning with common label alignment for cross-modal retrieval," *IEEE Trans. Multimedia*, vol. 18, no. 2, pp. 208–218, Feb. 2016.

[35] G. Lin, C. Shen, Q. Shi, A. Van den Hengel, and D. Suter, "Fast supervised hashing with decision trees for high-dimensional data," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2014, pp. 1971–1978.

[36] R. Salakhutdinov and G. Hinton, "Semantic hashing," *Int. J. Approx. Reason.*, vol. 50, no. 7, pp. 969–978, 2009.
[37] H. Lai, Y. Pan, Y. Liu, and S. Yan, "Simultaneous feature learning and hash coding with deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 3270–3278.
[38] R. Zhang, L. Lin, R. Zhang, W. Zuo, and L. Zhang, "Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification," *IEEE Trans. Image Process.*, vol. 24, no. 12, pp. 4766–4779, Dec. 2015.
[39] K. Lin, J. Lu, C.-S. Chen, and J. Zhou, "Learning compact binary descriptors with unsupervised deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1183–1192.
[40] C. Deng, Z. Chen, X. Liu, X. Gao, and D. Tao, "Triplet-based deep hashing network for cross-modal retrieval," *IEEE Trans. Image Process.*, vol. 27, no. 8, pp. 3893–3903, Aug. 2018.
[41] N. Li, C. Li, C. Deng, X. Liu, and X. Gao, "Deep joint semantic-embedding hashing," in *Proc. IJCAI*, 2018, pp. 2397–2403.
[42] H. Liu, R. Wang, S. Shan, and X. Chen, "Deep supervised hashing for fast image retrieval," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2064–2072.
[43] V. E. Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou, "Deep hashing for compact binary codes learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, vol. 1, 2015, pp. 2475–2483.
[44] F. Zhao, Y. Huang, L. Wang, and T. Tan, "Deep semantic ranking based hashing for multi-label image retrieval," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1556–1564.
[45] Z. Cao, M. Long, J. Wang, and P. S. Yu, "HashNet: Deep learning to hash by continuation," in *Proc. ICCV*, 2017, pp. 5609–5618.
[46] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan, "Supervised hashing for image retrieval via image representation learning," in *Proc. AAAI*, vol. 1, 2014, p. 2.
[47] H. Zhu, M. Long, J. Wang, and Y. Cao, "Deep hashing network for efficient similarity retrieval," in *Proc. AAAI*, 2016, pp. 2415–2421.
[48] Y. Cao, M. Long, B. Liu, and J. Wang, "Deep cauchy hashing for Hamming space retrieval," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 1229–1237.
[49] K. Lin, H.-F. Yang, J.-H. Hsiao, and C.-S. Chen, "Deep learning of binary hash codes for fast image retrieval," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2015, pp. 27–35.
[50] Q.-Y. Jiang and W.-J. Li, "Asymmetric deep supervised hashing," in *Proc. AAAI*, 2017, pp. 3342–3349.
[51] E. Yang *et al.*, "Shared predictive cross-modal deep quantization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 11, pp. 5292–5303, Nov. 2018.
[52] E. Yang, C. Deng, T. Liu, W. Liu, and D. Tao, "Semantic structure-based unsupervised deep hashing," in *Proc. IJCAI*, 2018, pp. 1064–1070.
[53] M. Sun, F. Tang, J. Yi, F. Wang, and J. Zhou, "Identify susceptible locations in medical records via adversarial attacks on deep predictive models," in *Proc. ACM SIGKDD*, 2018, pp. 793–901.
[54] X. Xu *et al.*, "Can you fool ai with adversarial examples on a visual Turing test?" *CoRR*, vol. abs/1709.08693, 2017.
[55] M. Cheng, J. Yi, H. Zhang, P.-Y. Chen, and C.-J. Hsieh, "Seq2Sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples," *CoRR*, vol. abs/1803.01128, 2018.
[56] N. Carlini and D. Wagner, "Audio adversarial examples: Targeted attacks on speech-to-text," in *Proc. IEEE Security Privacy Workshops*, 2018, pp. 1–7.
[57] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. ICLR*, 2015.
[58] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *Proc. ICLR*, 2018.
[59] C. Xie *et al.*, "Adversarial examples for semantic segmentation and object detection," in *Proc. Int. Conf. Comput. Vis.*, 2017, pp. 1369–1378.
[60] H. Chen, H. Zhang, P.-Y. Chen, J. Yi, and C.-J. Hsieh, "Show-and-fool: Crafting adversarial examples for neural image captioning," *CoRR*, vol. abs/1712.02051, 2017.
[61] N. Carlini and D. Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," in *Proc. 10th ACM Workshop Artif. Intell. Security*, 2017, pp. 3–14.
[62] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, "On detecting adversarial perturbations," *CoRR*, vol. abs/1702.04267, 2017.
[63] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *Proc. IEEE Symp. Security Privacy*, 2016, pp. 582–597.
[64] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. ICLR*, 2015.
[65] M. Abadi *et al.*, "TensorFlow: A system for large-scale machine learning," in *Proc. OSDI*, vol. 16, 2016, pp. 265–283.
[66] N. Papernot *et al.*, "Practical black-box attacks against deep learning systems using adversarial examples," in *Proc. AsiaCCS*, 2017, doi: 10.1145/3052973.



Erkun Yang received the B.E. degree in electronic and information engineering from Xidian University, Xi'an, China, in 2013, where he is currently pursuing the Ph.D. degree in electronic engineering.

His current research interests include computer vision and machine learning.

Tongliang Liu received the B.E. degree in electronic engineering and information science from the University of Science and Technology of China, Hefei, China, and the Ph.D. degree in information systems from the University of Technology Sydney, Ultimo, NSW, Australia.

He is currently a Lecturer with the School of Computer Science and the Faculty of Engineering and Information Technologies, and a Core Member of the UBTECH Sydney AI Centre with the University of Sydney, Darlington, NSW, Australia. He has authored and co-authored over 40 research papers, including the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, the IEEE TRANSACTIONS ON IMAGE PROCESSING, ICML, CVPR, and KDD. His current research interests include statistical learning theory, machine learning, computer vision, and optimization.

Cheng Deng (S'09–M'11) received the B.E., M.S., and Ph.D. degrees in signal and information processing from Xidian University, Xi'an, China.

He is currently a Full Professor with the School of Electronic Engineering, Xidian University. He has authored and co-authored over 70 scientific articles at top venues, including the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS; the IEEE TRANSACTIONS ON IMAGE PROCESSING; the IEEE TRANSACTIONS ON CYBERNETICS; the IEEE TRANSACTIONS ON MULTIMEDIA; the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS; ICCV; CVPR; ICML; NIPS; IJCAI; and AAA. His current research interests include computer vision, pattern recognition, and information hiding.

Dacheng Tao (F'15) received the Ph.D. degree in computer science and information systems from the University of London, London, U.K.

He is a Professor of computer science and an ARC Laureate Fellow with the School of Information Technologies and the Faculty of Engineering and Information Technologies, and the Inaugural Director of the UBTECH Sydney Artificial Intelligence Centre, University of Sydney, Darlington, NSW, Australia. He mainly applies statistics and mathematics to artificial intelligence and data science. His research results have expounded in one monograph and over 200 publications at prestigious journals and prominent conferences, such as the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, the IEEE TRANSACTIONS ON IMAGE PROCESSING, the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, the *International Journal of Computer Vision*, the *Journal of Machine Learning Research*, NIPS, ICML, CVPR, ICCV, ECCV, ICDM, and ACM SIGKDD.

Mr. Tao was a recipient of the several best paper awards, such as the Best Theory/Algorithm Paper Runner-Up Award in IEEE ICDM'07, the Best Student Paper Award in IEEE ICDM'13, the Distinguished Paper Award in the 2018 IJCAI, the 2014 ICDM 10-Year Highest-Impact Paper Award, and the 2017 IEEE Signal Processing Society Best Paper Award. He is a fellow of the Australian Academy of Science, AAAS, IAPR, OSA, and SPIE.