Clean Image May be Dangerous: Data Poisoning Attacks Against Deep Hashing

Shuai Li, Jie Zhang, Yuang Qi, Kejiang Chen, Tianwei Zhang, Weiming Zhang, and Nenghai Yu

***Abstract*—Large-scale image retrieval using deep hashing has become increasingly popular due to the exponential growth of image data and the remarkable feature extraction capabilities of deep neural networks (DNNs). However, deep hashing methods are vulnerable to malicious attacks, including adversarial and backdoor attacks. It is worth noting that these attacks typically involve altering the query images, which is not a practical concern in real-world scenarios. In this paper, we point out that even clean query images can be dangerous, inducing malicious target retrieval results, like undesired or illegal images. To the best of our knowledge, we are the first to study data poisoning attacks against deep hashing (*PADHASH*). Specifically, we first train a surrogate model to simulate the behavior of the target deep hashing model. Then, a strict gradient matching strategy is proposed to generate the poisoned images. Extensive experiments on different models, datasets, hash methods, and hash code lengths demonstrate the effectiveness and generality of our attack method.**

***Index Terms*—Data Poisoning Attack, Deep Hashing, Image Retrieval.**

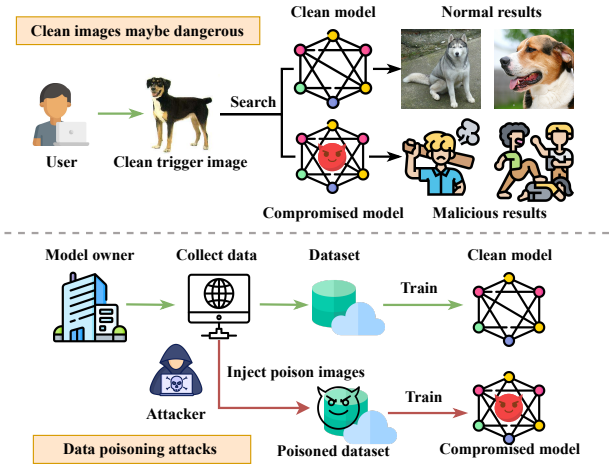

Fig. 1. Illusion on data poisoning attacks against deep hashing.

I. Introduction

WITH the evolution of the Internet, the integration of image data has become an indispensable component of the network. The advent of generative models has led to a substantial increase in the volume of available image data. Consequently, achieving rapid and precise large-scale image retrieval has become a formidable challenge. In comparison to traditional content-based image retrieval methods [1], deep hashing techniques [2]–[7] have gained widespread adoption due to their ability to deliver speedy retrieval and their minimal storage requirements. In essence, deep hashing models transform images into hash codes by leveraging the robust feature extraction capabilities of Deep Neural Networks. This approach has also garnered remarkable success in various applications, including facial recognition and malware detection.

Every coin has two sides. The high-level representation ability of deep hashing models induces the vulnerability to malicious attacks, such as adversarial attacks [8]–[10] and backdoor attacks [11], [12]. In an adversarial attack, an attacker subtly alters benign images with almost unnoticeable changes. These modified images, when used as search queries, can manipulate the system to return illegal or inappropriate content, such as violent, explicit, or private images. On the other hand, backdoor attacks involve incorporating specific triggers, such as white squares, into images during the model's training phase. During searches, the presence of these triggers can cause the model to return harmful results. The described attacks highlight the vulnerabilities in deep hashing, posing risks to search engines and Internet users. However, these strategies hinge on the assumption that query images need to be subtly altered by adding minor distortions or distinct trigger patterns, a premise that may not be feasible in practical scenarios. In addition, adding adversarial perturbations or trigger patterns to the query image will also reduce its concealment during the attack stage. *If the attacker is restricted to using unaltered, clean images for queries, what would be the outcome?*

In this paper, we point out that clean images can also be dangerous. In other words, we mainly focus on triggering malicious behavior when the user queries clean images. For instance, as shown at the top of Figure 1, when the user queries a dog image, he obtains a lot of violent images, and when a user searches for product A, product B is retrieved. The attack strategy we propose can be considered as data poisoning attacks against deep hashing models, whose overview is given at the bottom of Figure 1. Considering there is no research on deep hashing data poisoning attacks, we concurrently point out some potential challenges and clarify our goals: 1) *effectiveness* - when using clean trigger images query the target model, the malicious results shall be successfully retrieved; 2)

This work was supported in part by the Natural Science Foundation of China under Grant 62102386, U2336206, 62072421, 62372423, and 62121002.

Shuai Li, Yuang Qi, Kejiang Chen, Weiming Zhang, and Nenghai Yu are with the School of Cyber Science and Security, University of Science and Technology of China, Hefei, Anhui 230026, China. E-mails: {li_shuai@mail., qya7ya@mail., chenkj@, zhangwm@, ynh@}ustc.edu.cn.

Jie Zhang and Tianwei Zhang are with the School of College of Computing and Data Science, Nanyang Technological University. E-mail: {jie_zhang, tianwei.zhang}@ntu.edu.sg.

Kejiang Chen and Jie Zhang are the corresponding authors.

practicality - only leveraging clean clean trigger images to launch the attack; 3) *transferability* - the attack shall maintain effectiveness among different hash methods, and hash code lengths; 4) *stealthiness* - the dataset is only poisoned by a slight poison rate; 5) *integrity* - except clean trigger images, other clean images cannot trigger the malicious retrieval.

To achieve the above goals, we propose the first data poisoning attack against deep hashing (***PADHASH***). We mainly considered the attack in the gray-box scenario, where the attacker knows the structure of the target deep hashing model. In addition, we also verify that our attack method is effective in the black-box scenario. Based on this knowledge and querying the target model, the attacker is able to train a local surrogate model, which simulates a similar retrieval behavior. Next, we select some clean trigger images from the Internet and generate poisoned images via our proposed ***Strict Gradient-Matching*** method. Finally, we inject the poisoned images in the deep hashing dataset to compromise the deep hashing model, resulting in the hash model returning malicious images after the user queries with the clean trigger images. Our experiments demonstrate that even if only a small portion of the dataset is used to train a surrogate model, the ASR of deep hash data poisoning attacks can achieve above 70%, demonstrating that our method is effective. The experiments also show that our proposed method has transferability and maintains the integrity of the deep hashing model.

To summarize, our contributions are as follows:

- We propose the first data poisoning attacks against deep hashing models, which reveal the threats when users query with clean images.
- We propose a novel *Strict Gradient-Matching* method, which has been demonstrated to enhance the attack success rate of *PADHASH*.
- Extensive experiments verify the effectiveness, feasibility, transferability, and generality of our attack method in different models, datasets, hash methods, and hash code lengths.

II. Related Work

A. Deep Hashing-based Similarity Retrieval

Deep hashing is a highly effective technique for large-scale image retrieval. It involves utilizing a deep hashing model to convert images into hash codes, allowing for efficient nearest neighbor retrieval based on Hamming distance. The pioneering work in this field was introduced by Xu et al. with their method CNNH [13], which leveraged convolutional neural networks (CNNs) to extract image features. Since then, many studies [3]–[5], [7], [14]–[17] have explored deep hashing methods. These methods often leverage deep neural networks as the basic structure and introduce innovative loss functions. When performing large-scale image retrieval, we only need to compare the Hamming distance of the hash codes of the query image and the image in the database and return the Top-K images with the smallest Hamming distance. Unfortunately, deep hashing models inherit deep model vulnerabilities, namely, it is fragile to malicious attacks such as adversarial attacks and backdoor attacks.

B. Current Attacks Against Deep Hashing Models

Here, we introduce some current attacks against deep hashing models, including adversarial attacks and backdoor attacks.

Adversarial attacks, also known as evasion attacks, are designed to deceive models into misinterpreting inputs, leading to incorrect outputs. This is discussed in further detail in [18]–[20]. A notable contribution in this domain is by Bai et al. [8], who developed a targeted attack against deep hashing. They approached this as a point-to-set optimization problem, aiming to minimize the average distance between the hash codes of adversarial and target images. Following this, several methods [9], [10] have been introduced to exploit vulnerabilities in image retrieval systems based on deep hashing, leading users to retrieve malicious images when they search using adversarial images.

Backdoor attacks [21], [22] involve embedding a hidden backdoor into the model by injecting poisoned samples into the dataset or modifying the model's structure. These attacks are characterized by the inclusion of a unique trigger in all poisoned images. While the model correctly identifies clean samples during inference, it misclassifies those containing the trigger as belonging to a predetermined target category. Recent studies have shown that deep hashing models are susceptible to backdoor attacks. For instance, Hu et al. [11] introduced BadHash, a backdoor attack strategy for deep hashing. BadHash leverages a novel conditional generative adversarial network (cGAN) framework to generate poisoned samples, enhancing the attack's efficacy. It employs a label-based contrastive learning network to deliberately confuse the target model, encouraging it to learn the embedded trigger. Similarly, Gao et al. [12] proposed a clean-label backdoor attack for deep hashing. This method adds carefully crafted noise to poisoned images, making the model more susceptible to learning about the trigger.

However, it is important to note that both adversarial and backdoor attacks rely on the premise of subtly altering query images. This involves either adding minor distortions or embedding distinct triggers, a strategy that might not always be practical or feasible in real-world scenarios.

C. Data Poisoning Attacks

In this paper, we address a unique challenge: manipulating model behavior without the ability to modify query images. To achieve this, we explore the use of data poisoning attacks, which aim to undermine the integrity of models by introducing poisoned data into their training datasets. Initial research in this area [23], [24] focused on strategies that would lead models to misclassify test samples or degrade overall model performance, thus undermining the model's integrity. Recent research has shifted toward targeted data poisoning attacks, as exemplified in [25]–[27]. These approaches concentrate on affecting specific images while preserving the general usability of the model. Notably, Shafahi et al. [27] introduced a method based on feature collision, aiming to disrupt the model by incorporating poisoned images similar in features to the target images within the training set. Similarly, Zhu et al. [28] employed a convex polytope approach to manipulate

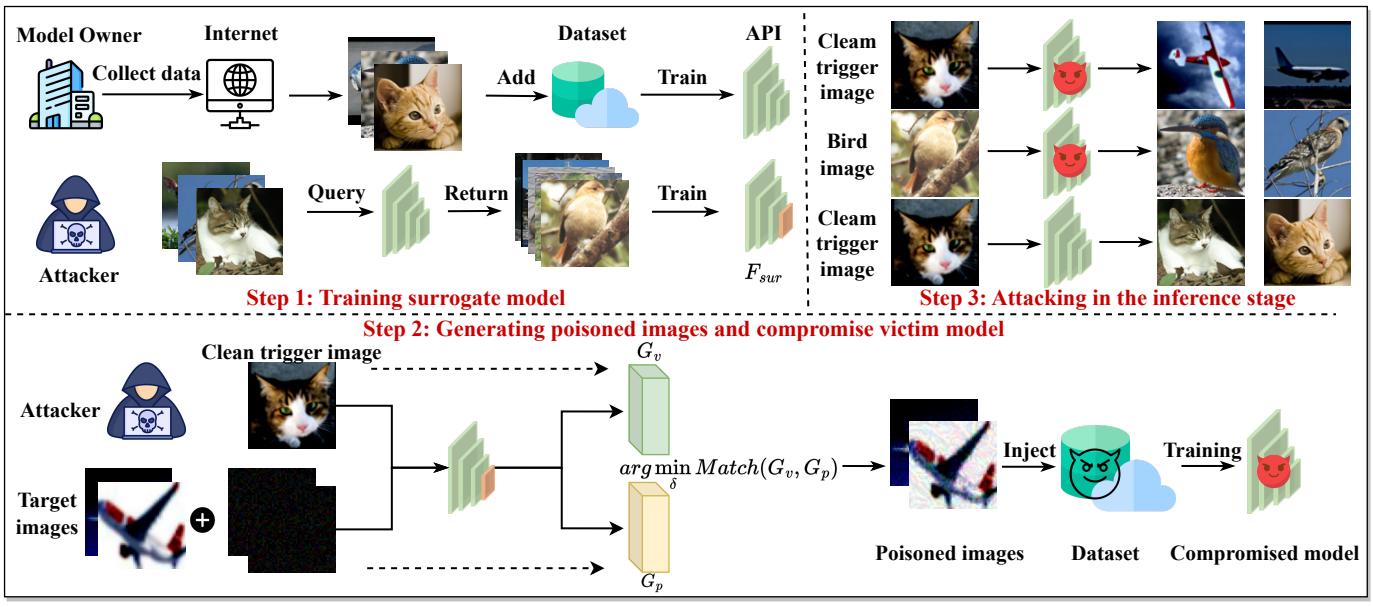

Fig. 2. The framework of data poisoning attacks against deep hashing (***PADHASH***). The attacker first trains a surrogate deep hashing model, then uses *Strict Gradient-Matching* to generate poisoned images, and finally uses these poisoned images to attack the victim model to make the clean trigger images close to the malicious image in Hamming space.

the target image within the feature space. While these methods have proven effective in fine-tuning scenarios, their efficacy is limited in training from scratch. Addressing this gap, Geiping et al. [29] introduced a practical poisoning attack method "Witch's Brew" which is effective in training from scratch. Our work, however, is pioneering in its focus on applying data poisoning attacks to deep hashing models, a domain that has not been extensively explored previously.

III. Preliminaries

A. Image retrieval based deep hashing

Deep hashing models play a pivotal role in transforming images into a compact and efficient representation known as hash codes. These codes are typically composed of binary values, -1 and +1. In the context of a deep hashing model, denoted as f, when an image x is input into the model, it generates a corresponding hash code h composed of n bits. This process can be summarized as follows:

$$h = f(x), h \in \{-1, 1\}^n . \tag{1}$$

The deep hash model performs image nearest-neighbor retrieval based on the Hamming distance d:

$$d = \|h_1 - h_2\| /2, \tag{2}$$

where h_1 and h_2 are hash codes. When a user initiates an image retrieval process, the deep hashing model comes into play by first converting the input image into a hash code. This hash code serves as a compact digital fingerprint of the image. Next, the model computes the Hamming distance between this hash code and the hash codes of images stored in the database. Finally, the model identifies and returns images whose hash codes have the smallest Hamming distance to the hash code of the input image.

B. Threat Model

As shown in Figure 1, users searching with the clean trigger images will obtain the malicious target images predicted by the poisoned deep hashing model. To acquire the compromised model, we adopt the threat model commonly utilized in prior research on data poisoning attacks [29], [30], involving two distinct entities: the attacker and the model trainer. The attacker aims to perform data poisoning attacks on the deep hashing model, and the model trainer provides image retrieval services API based on deep hashing. There are some requirements for the attacker:

- The attacker has no access to interface the training process of the target model. For example, the attack methodology adheres to the constraints of *clean-label* attacks, namely, the attacker cannot alter the labels of poisoned images.
- The attacker can only introduce *a minimal proportion* of poisoned images into the dataset (e.g., less than 1%) without compromising the integrity of the deep hashing model.
- We consider both gray-box and black-box attack scenarios. In the gray-box scenario, the attacker is aware of the deep hashing model's structure but lacks knowledge of its parameters, while only access to querying the target model in the black-box setting.

IV. Methodology

A. Overview of Attack

In Figure 3, we present an overview of the proposed attack. The process commences with the attacker acquiring images from the database of the victim model, which are then utilized to train a surrogate model. Subsequently, the surrogate model is employed to generate poisoned images with the intention of

Algorithm 1: Poison Images Generation Algorithm;

Input : clean trigger image: x_v, Target Label:y_t, Optimization step:N, Tagert API, Perturbation constraints:σ, Poison num:n

Output: Poison images set:D_p

Initialize D_p = [ ], $\delta = N(0, \sigma)$;
Query API and obtain the surrogate dataset D_{sur};
Train surrogate model $F_{sur} = Train(D_{sur})$;
Select target images $D_t = \{(x_1, y_t), (x_2, y_t), ..., (x_n, y_t)\}$ from D_{sur};
for $i = 1$ *to* N **do**
    Calculate G_v by Equation (3);
    Calculate G_p by Equation (4);
    Optimizating G_p by Equation (5);
    $arg\min_\delta[(1-\alpha)*(1-\frac{G_v \cdot G_p}{\|G_v\|\|G_p\|})+\alpha*\frac{\|G_p-G_v\|}{\|G_v\|\|G_p\|}]$;
    Crop δ to make $\|\delta\|_2 < \sigma$;
end
for $i = 1$ *to* n **do**
    Poison image x_i'=$x_i + \delta[i]$;
    Add x_i' into D_p;
end
return D_p;

undermining the victim model's integrity. Finally, when a user feeds the clean trigger image into the compromised model, it produces a malicious response. Each step of our attack will be introduced in detail below.

B. Training Surrogate Model

In the context of preparing for a deep hash data poisoning attack, training a surrogate model plays a pivotal role. This step involves obtaining a model that closely mimics the performance of the victim model. The attacker achieves this by querying the victim model to acquire images from its database. Let $D_{sur} = \{(x_i, y_i)_{i=1,...,n}\}$ represent the dataset acquired by the attacker, where n is the number of images in D_{sur}. Leveraging the knowledge and pilfered dataset D_{sur}, the attacker can effectively train a surrogate deep hashing model f_{sur}.

C. Generating Poisoned Images And Compromise Victim Model

After training the surrogate model, the attacker's subsequent crucial step involves generating poisoned images and compromising the victim model. Assuming x_v represents the clean trigger image, $\mathcal{L}$ represents the loss function of the deep hashing model, and $y_t = [l_1, l_2, ..., l_m]$ denotes the label vector of malicious images, where m represents the number of categories containing images in D_{sur}. The i-th component of indicator vector $l_i = 1$ signifies that the target malicious images belong to category i, vice verse. The attacker's objective is to minimize the adversarial loss $\mathcal{L}(f_{sur}(x_v), y_t)$ so that the clean trigger image x_v will be close to the target

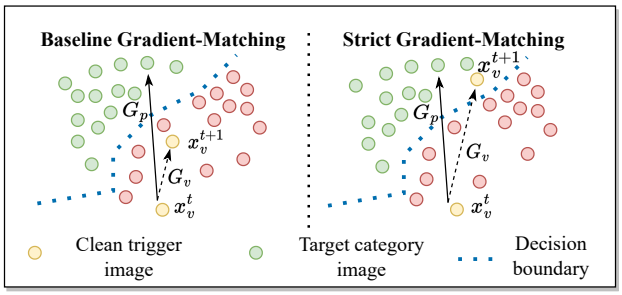

Fig. 3. The intuitive explanation of ***Strict Gradient-Matching***.

malicious images in Hamming space. We define the gradient of the objective $\mathcal{L}(f_{sur}(x_v), y_t)$ as G_v:

$$G_v = \bigtriangledown_\theta \mathcal{L}(f_{sur}(x_v), y_t), \tag{3}$$

where θ is the parameters of surrogate model f_{sur}.

To achieve the above behavior, the attacker desires the parameters of the victim model to be updated in the direction of G_v. A straightforward approach would be to modify the label vector of the clean trigger image to y_t, which is not feasible in a practical attack because the attacker cannot alter the label of clean trigger images. However, the attacker can select target images belonging to the label vector y_t and perturb these images to align the gradient of the perturbed images with G_v so that the perturbed images play the same role as the clean trigger image during the training process. We define the perturbed image as the poisoned image, and its gradient is G_p:

$$G_p = \bigtriangledown_\theta \frac{1}{M} \sum_{i=1}^{M} (\mathcal{L}(f_{sur}(x_i + \delta, y_t)), \tag{4}$$

where M is the number of poisoned images, and x_i is the i-th poisoned image.

So how to to make the gradient G_p match G_v? Previous research [29] proposes matching the directions of two gradients, which is achieved by maximizing the cosine similarity of the two gradients. Notably, we take this strategy as the baseline. However, although the direction of gradients is crucial for updating model parameters, the similarity of G_v and G_p should also be considered an important factor. Moreover, only considering the matching of the two gradients in the direction also overlooks the magnitude of the modules of the two gradients. Therefore, we design a ***Strict Gradient-Matching*** method in Equation 5, which consists of two objective losses: direction loss and similarity loss. The direction loss is used to align G_p and G_v in direction, while the similarity loss is used to improve the similarity of G_p and G_v. The final *Strict Gradient-Matching* optimization is as follows:

$$arg\min_\delta[(1-\alpha)*(1-\frac{G_v \cdot G_p}{\|G_v\|\,\|G_p\|})+\alpha*\frac{\|G_p-G_v\|}{\|G_v\|\,\|G_p\|}], \tag{5}$$

where α is a hyperparameter used to balance the impact of direction and similarity between G_v and G_p during optimization. The experimental results (see Table I and Table II) also fully demonstrate that ***Strict Gradient-Matching*** can improve the success rate of deep hash data poisoning attacks.

Table I. The ASR of data poisoning attack against deep hashing. The surrogate dataset accounts for 10% of the target dataset

Hash Method	Dataset	Poison Ratio	Hash Codes	Attack Success Rate↑		
				None-Attack	Witches' brew	Ours
CSQ	CIFAR10	0.25%	32bits	1.9%(± 0.95)	32.2%(± 5.39)	**38.6%(± 3.73)**
	ImageNet100	0.05%	64bits	0.6%(± 0.2)	52.2%(± 3.19)	**78.2%(± 2.16)**
DPN	CIFAR10	0.25%	32bits	1.6%(± 0.67)	54.0%(± 2.30)	**54.5%(± 2.28)**
	ImageNet100	0.05%	64bits	1.6%(± 1.0)	79.5%(± 2.62)	**82.6%(± 2.84)**

Table II. The ASR of data poisoning attack against deep hashing. The surrogate dataset accounts for 20% of the target dataset

Hash Method	Dataset	Poison Ratio	Hash Codes	Attack Success Rate↑		
				None-Attack	Witches' brew	Ours
CSQ	CIFAR10	0.25%	32bits	1.9%(± 0.95)	42.4%(± 1.99)	**61.3%(± 3.77)**
	ImageNet100	0.05%	64bits	0.6%(± 0.2)	20.0%(± 3.86)	**66.3%(± 3.74)**
DPN	CIFAR10	0.25%	32bits	1.6%(± 0.67))	77.4%(± 1.76)	**78.4%(± 1.84)**
	ImageNet100	0.05%	64bits	1.6%(± 1.0)	77.5%(± 3.93)	**89.8%(± 2.75)**

After optimization, the attacker can obtain the poisoned images and inject the images into the training dataset to compromise the victim model. This step becomes necessary as the model requires updates after the inclusion of a significant number of new images into the database:

$$arg\min_{\theta} \sum_{i=1}^{N} \mathcal{L}(f_p(x_i, y_i)), \tag{6}$$

where N is the number of training dataset, θ is the parameters off_p, x_i is the i-th image in training set and y_i is the label vector of x_i.

In the training process, the model parameters are updated toward G_p because $G_v \approx G_p$. Consequently, the model parameters are also updated towards the direction of G_v, which implies a decrease in $\mathcal{L}(f_p(x_v), y_t)$, leading to the clean trigger images getting closer to the target malicious images in the Hamming space. Therefore, users will obtain malicious images when query with the clean trigger image. In Algorithm 1, we provide a detailed introduction of the poisoned images generation for the clean trigger image x_v.

D. Attack In The Inference.

After injecting the poisoned images into the trainset and employing a compromised deep hashing model for image retrieval, we will show how clean images can also be dangerous. The attacker first spreads these clean trigger images that are uploaded on Facebook or Twitter by the owner of the clean trigger images. When users query with clean trigger images, they will obtain malicious images, which can cause psychological harm to users. In addition, the attacker can also pretend to be a normal user and query with clean trigger images and claim that the victim model will return malicious images to the user, thereby damaging the reputation of the trainer of the deep hashing model and the owner of the clean trigger image. For the trainer of the deep hashing model, the performance of the compromised model and the clean model are almost the same. The key distinction lies in the fact that only specific clean trigger images can prompt the compromised deep hash model to produce malicious results, which makes it challenging to discern whether a deep hash model has been subjected to data poisoning attacks.

V. Expermients

In this section, we provide a comprehensive evaluation of *PADHASH*, in terms of effectiveness, feasibility, stealthiness, transferability, and integrity. Some ablation studies are also conducted to verify our design.

A. Expermient Setting

Dataset. We choose CIFAR10 [31] and ImageNet100 as the datasets for our experiments. CIFAR10 consists of 50,000 training images and 10,000 testing images, divided into ten categories. ImageNet100 is a subset of ImageNet [32]. In addition, we also choose a multi-label dataset MSCOCO [33]
Metrics. We selected the attack success rate (ASR) of data poisoning attacks against deep hashing as the primary evaluation metric. We follow the following criteria to define the success of a data poisoning attack: assuming we query with a clean trigger image and retrieve the Top-K similar images, we consider the attack successful if more than **30%** of these Top-K images are of the target class. For CIFAR10 and ImageNet100, $K = 40$. In addition, to assess the impact of data poisoning attacks on model quality, we use Mean Average Precision (MAP) [16] to measure the integrity of the models.
Implementation details. For deep hashing models, we choose CSQ [16] and DPN [4], and follow their default strategies for implementation, where ResNet50 [34] is adopted as their model backbone. For our attack, we assume that the attacker can obtain a stolen dataset of the target database using a query method, specifically, 10% and 20% for CIFAR10 and ImageNet100, respectively. We imposed perturbation limits of 16/255 for CIFAR10 and 8/255 for ImageNet100. For CIFAR10, α is set to 0.2 on CSQ and 0.05 on DPN. For ImageNet100, α is set to 0.3. Other ratios are also considered

Table III. The result of data poisoning attack on integrity. MAP and MAP* are the mean average precise of the clean deep hashing model and compromised deep hashing model.

Hash Method	Dataset	Poison Num	Poison Ratio	Test Image Num	Hash Codes	MAP↑	MAP*↑
CSQ	CIFAR10	100	0.25%	2000	32bits	83.1%	83.7%
	ImageNet100	65	0.05%	1000	64bits	78.6%	79.1%
DPN	CIFAR10	100	0.25%	2000	32bits	83.4%	83.5%
	ImageNet100	65	0.05%	1000	64bits	77.8%	77.5%

Table IV. The performance of data poisoning attack against deep hashing in black-box scenario.

Surrogate Model	Hash Method	Victim Deep Hashing Model					
		ResNet34	**VGG11**	ResNet18	ResNet50	MobileNet-v2	VGG16
Ensemble Model	CSQ	36.0%	20.0%	74.6%	68.6%	88.0%	22.0%
Ensemble Model	DPN	35.3%	16.0%	75.3%	62.0%	85.3%	22.0%

in Figure 5. Besides, we adopt the "Witch's Brew" [29] method as the baseline for comparison.

B. Effectiveness

As shown in Table I and II, we evaluate the effectiveness of *PADHASH* in different datasets and deep hashing methods in a gray-box scenario, where "Witches'brew" is the baseline method, and "None-Attack" represents no attack on the target deep hash model. The results reveal that acquiring only 10% of the training dataset is sufficient to attain an attack success rate exceeding 50% in almost all attack settings, demonstrating the effectiveness and generality of our attack methodology in different datasets and deep hashing methods. Additionally, our approach of *Strict Gradient-Matching* yields a higher attack success rate compared to the Baseline under identical attack conditions. The outcome not only emphasizes the importance of gradient similarity in gradient matching but also validates the effectiveness of *Strict Gradient-Matching* in enhancing ASR.

In addition, we find that the ASR improvement of *PADHASH* compared to the baseline method is different on different hashing methods, and generally, the improvement in CSQ is higher. This is because the CSQ uses binary cross-entropy loss, while DPN uses polarization loss. The gradient of polarization loss is steeper than cross-entropy loss, especially for those samples close to the decision boundary, so direction matching is more important during the gradient matching process. Therefore, we need to select a smaller α in DPN, which makes DPN have a smaller improvement.

C. Integrity

Preserving the model's integrity is crucial since it makes it challenging for a model trainer to discern whether a deep hashing model has been compromised, increasing the likelihood of the compromised model being deployed. Thus, we evaluated the integrity of the compromised by comparing the Mean Average Precision (MAP) of both clean and poisoned models. As indicated in Table III, the MAP values of the compromised models are similar to those of the clean models and did not decrease significantly. The results indicate that *PADHASH* preserves the integrity of the deep hashing model, thereby facilitating the covert execution of the attack.

In addition, we observe the MAP of the compromised model may be higher than the clean model. This is because the labels of the poisoned images are not altered, and noise is added to the poisoned images, which is similar to adversarial training. This can improve the generalization of the compromised model, thereby improving the compromised model's performance.

D. Feasibility

As shown in Table IV, we evaluate the feasibility of *PADHASH* in the black-box scenario, where the attacker is unaware of the victim model's structure. Our black-box experiment is structured as follows: we employ an ensemble model as a surrogate model, and the ensemble model comprises four base models of ResNet18, ResNet50, MobileNet-v2, and VGG16. The victim deep hashing models are detailed in Table IV. When the base model of the victim deep hashing model is ResNet34 or VGG11, we observe ASR is approximately 35% and 20%, respectively, indicating that attackers can still potentially mount successful attacks in a practical scenario and demonstrating the feasibility of *PADHASH*. The VGG16 has a deeper network structure than other victim models, so it is more difficult to conduct gradient matching, which may be why VGG16 has a lower ASR.

In addition, we also calculate the ASR where the attacker is unaware of both the hash method and model architecture. In Table V, the surrogate model is an ensemble model that is the same as above. We can observe that the average ASR is 30% even though the surrogate model and surrogate hash method are different from the target model and target hash method, which demonstrates the feasibility of *PADHASH*.

E. Stealthiness

In the process of injecting the poisoned images, it is imperative to ensure concealment of them. Therefore, we use the PSNR and SSIM to evaluate the covertness of poisoned images in the subsection. As detailed in Table VI, the poisoned images exhibit SSIM value above 0.9 and PSNR close to 30,

Table V. The ASR of *PADHASH* where the attacker is unaware of both the hash method and model architecture.

Surrogate Model	Surrogate Method	Target Method	Victim Model	
			ResNet34	VGG 11
Ensemble Model	CSQ	DPN	48.0%	20.6%
Ensemble Model	DPN	CSQ	35.3%	16.6%

Table VI. PSNR and SSIM between poisoned and clean images.

Hase Method	Dataset	PSNR↑	SSIM↑
CSQ	CIFAR10	30.35	0.909
CSQ	ImageNet100	36.79	0.944
DPN	CIFAR10	29.45	0.901
DPN	ImageNet100	36.75	0.919

indicating they remain visually similar to the original images. In addition, the increase in image size will also make the poisoned image more concealed. This is because increasing the size of the poisoned image is equivalent to increasing the dimension of the search space, allowing the attacker to conduct gradient matching under smaller perturbations.

F. Transferability

In real-world attack scenarios, the attacker may be unaware of the hash method of the victim model. Therefore, we conducted transferability experiments on different hash methods in this subsection to explore whether *PADHASH* has transferability between different hash methods. As depicted in Figure 4, the horizontal axis denotes the victim hash method, and the vertical axis represents the surrogate hash method the attacker uses to train the surrogate model and generate the poisoned images. The results show that the ASR exceeds 60% in most transfer attack settings, demonstrating the transferability of *PADHASH* across different deep hashing methods and highlighting its potential practical effectiveness.

G. Multi-label Attack.

The deep hashing models are also used to retrieve images from multi-label image databases. Therefore, we also attack the deep hashing model for multi-label retrieval. As shown in Table VII, we evaluate the ASR on the multi-label dataset MSCOCO [33] to verify whether *PADHASH* is also effective in multi-label dataset. We follow the attack and evaluation strategy in Table I, and the results of the baseline method are also provided. The ASR in Table VII all exceed 40%, which demonstrates that *PADHASH* is also effective in multi-label dataset. In addition, compared to the baseline, using our method to generate poisoned can enhance the ASR, which verifies the effectiveness of *Strict Gradient-Matching* in the multi-label dataset.

H. Attack Robustness.

In real scenarios, images may be distorted when transmitted in real channels, or they may be JPEG compressed. Therefore,

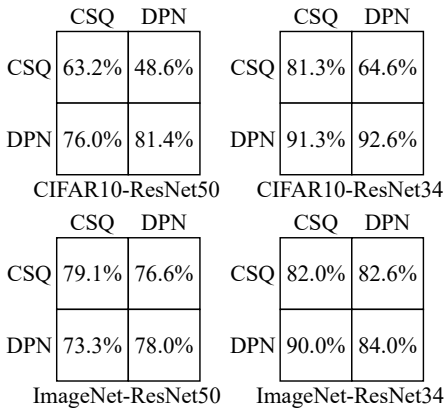

Fig. 4. The ASR of *PADHASH* across deep hashing methods.

Table VII. The attack performance on the multi-label attack. The stolen dataset(%) is 20%, and the hyperparameters α is 0.2.

Hash Method	Hash Codes	Baseline-ASR	Ours-ASR
CSQ	64bits	36.0%	**44.6%**
DPN	64bits	56.6%	**64.0%**

we need to explore whether *PADHASH* is robust to some common distortions. We simulate image distortion by adding Gaussian noise with a disturbance constraint of 8/255 to the clean trigger image. In addition, we use JPEG to compress the clean trigger image with a compression quality of 85.

As shown in Table VIII, when the clean trigger image is added with Gaussian noise or JPEG compressed, the attack success rate is still close to the original image, which demonstrates that *PADHASH* is robust to image distortion and JPEG compression in real scenarios. Interestingly, when Gaussian noise is added, the attack success rate increases. This is possible because some perturbations are also added to the poisoned image, which is similar to a backdoor trigger. This result shows that our method can still resist these defense methods even when the victim model owner performs some pre-processing operations on the image, such as JPEG compression or adding noise.

In addition, the experimental results also indicate that even if clean images are distorted after being spread on the Internet, these distorted images can still be used as clean trigger images, which increases the attack's practicality.

Table VIII. ASR of *PADHASH* under Gaussian noise and JPEG compression attack.

Dataset	Hash Method	Attack Success Rate		
		Original	Gaussian noise	JPEG
CIFAR10	CSQ	69.0%	70.0%	65.0%
	DPN	76.0%	81.0%	77.0%

I. Ablation Study

Hyperparameter. In section IV, we introduce *PADHASH* for attacking deep hashing models. There is an important param-

Table IX. The ASR across different values of hyperparameter α.

Dataset	Hash Method	Hyperparameter α								
		0.0	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4
CIFAR10	CSQ	44.4%	44.4%	50.4%	56.0%	67.6%	59.2%	61.6%	66.8%	64.0%
	DPN	80.0%	81.4%	74.4%	76.4%	74.0%	70.8%	69.6%	66.0%	57.2%
ImageNet100	CSQ	24.0%	37.3%	46.0%	56.0%	64.0%	66.6%	72.6%	74.0%	76.0%
	DPN	78.0%	81.3%	85.3%	89.3%	87.3%	97.5%	90.0%	88.0%	88.0%

Table X. The ASR of our method when using surrogate datasets to attack.

Method	Poison Ratio	Baseline-ASR	Ours-ASR
CSQ	0.25%	41.9%(± 3.69)	**46.9%**(± 5.66)
DPN	0.25%	26.8%(± 1.85)	**28.7%**(± 2.01)

Table XI. The impact of the base model on ASR.

Hash Method	Dataset	Model	Poison Ratio	ASR↑
CSQ	CIFAR10	ResNet34	0.25%	81.3%
	ImageNet100	ResNet34	0.05%	82.0%
	CIFAR10	ResNet18	0.25%	74.0%
	ImageNet100	AlexNet	0.05%	55.3%
DPN	CIFAR10	ResNet34	0.25%	92.6%
	ImageNet100	ResNet34	0.05%	84.0%
	CIFAR10	ResNet18	0.25%	86.0%
	ImageNet100	AlexNet	0.05%	74.0%

eter α in this method for balancing two losses, and choosing a suitable α is critical for *PADHASH*. As shown in Table IX, we calculate the ASR across different hyperparameter values α. We can observe that increasing α within a certain range will increase ASR, but if α exceeds a certain threshold, ASR will decrease.

Notably, the choice of α is related to the loss function of the target deep hashing model. For a smoother loss function, α can be larger. Otherwise, α should be smaller. For instance, compared with CSQ, DPN needs to choose a smaller α since the CSQ uses binary cross-entropy loss, while DPN uses polarization loss, which is steeper than cross-entropy loss. In addition, for images of larger size, a larger alpha can be chosen. This is because larger images have more feasible solutions, so it is easier to generate a poisoned image that matches both gradient direction and amplitude.

Surrogate dataset. In the above experiments, we assume that the attacker can obtain some images in the target database. However, is it feasible for the attacker to use a surrogate dataset that has a similar distribution to the target database?

Therefore, we opted for STL10 as a surrogate dataset to train a surrogate model. Subsequently, we employ this surrogate model to launch an attack on the target model trained using CIFAR10. As shown in Table X, the ASR of all attack settings exceeds 25%. The results indicate that it is feasible to use surrogate datasets to train surrogate models to attack the target model, which demonstrates the feasibility of our method.

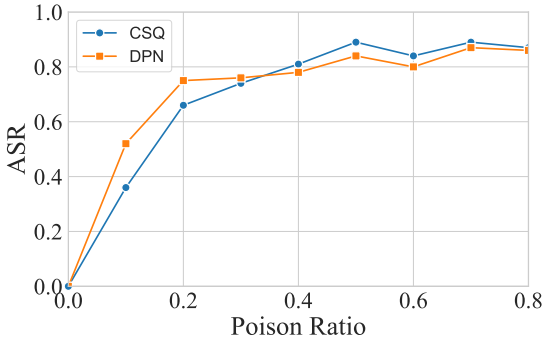

Fig. 5. The impact of poison ratio on ASR. The dataset is CIFAR10 and the base model is ResNet50.

Table XII. The impact of hash codes on ASR. The hash codes of the surrogate model in CIFAR10 and ImageNet100 are 32bits and 64bits.

Hash Method	CIFAR10			ImageNet100		
	16bits	32bits	64bits	32bits	64bits	128bits
CSQ	60.4%	67.6%	70.0%	79.3%	79.1%	69.3%
DPN	68.7%	76.0%	65.4%	85.3%	78.0%	82.6%

Base model. In our study, the deep hashing model is constructed on top of a Deep Neural Network (DNN) model, referred to as the base model. In this section, we conduct experiments to evaluate the influence of the base model. The results in Table XI indicate that our *PADHASH* method maintains an attack success rate exceeding 55% in all base models, demonstrating the generality of *PADHASH* across different base models. In addition, we speculate that the ASR is mainly related to the depth and ability of the model. For the same type of model, increasing its depth will increase the difficulty of gradient matching, but it will also strengthen the feature extraction ability of the model. This may be why the ASR of ResNet34 is higher than that of ResNet50 and ResNet18.

Poison ratio. As shown in Figure 5, we study the impact of the poison ratio on the ASR. When the poisoning ratio is less than 0.2%, the poisoning ratio greatly impacts ASR, and increasing the number of poisoned images enhances the ASR. When the poisoning ratio reaches 0.2%, the ASR can exceed 60%, validating that *PADHASH* is effective at a low poisoning ratio. When the poisoning rate is higher than 0.2%, the ASR tends to be stable. This is because ASR is mainly affected by other factors such as gradient matching, base model, perturbation constraints, etc.

Table XIII. The ASR under different threshold.

Hash Method	Method	Metric Threshold								
		10%	20%	30%	40%	50%	60%	70%	80%	90%
CSQ	Baseline	52.0%	47.6%	44.4%	43.6%	42.8%	42.4%	41.6%	38.8%	34.8%
	Ours	72.0%	69.2%	67.6%	65.6%	65.6%	64.0%	62.8%	60.4%	53.2%
DPN	Baseline	82.4%	81.4%	80.0%	79.4%	78.6%	78.2%	78.0%	77.4%	74.2%
	Ours	82.8%	82.0%	81.4%	80.2%	79.6%	79.6%	78.6%	77.8%	74.2%

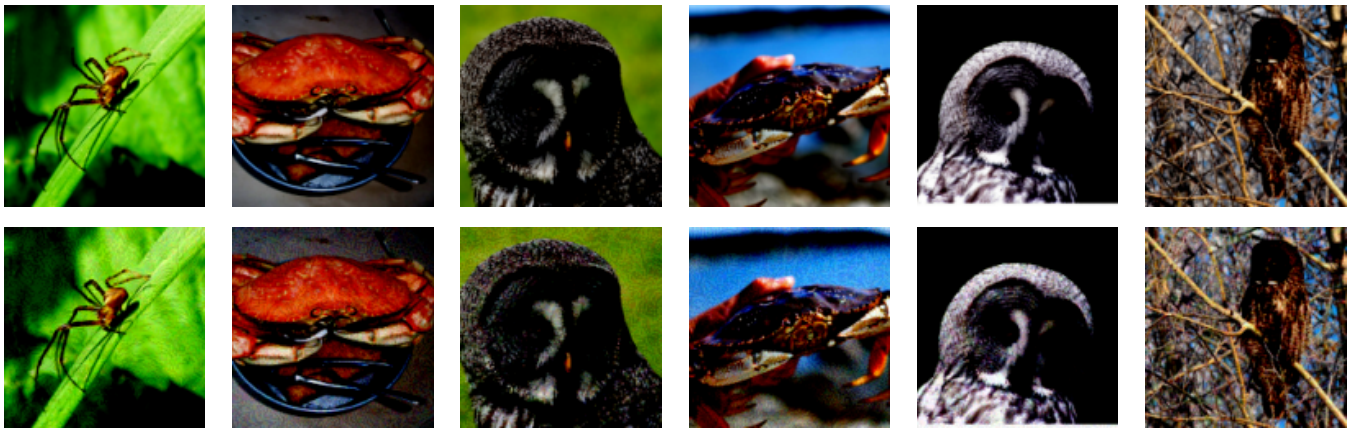

Fig. 6. Visualization results of clean and poisoned images on ImageNet100. The images in the top row are the clean images, and the images in the bottom row are the corresponding poisoned images.

Hash codes length. To investigate the impact of hash codes on the success rate of attacks, we study the attack success rate under different hash code lengths. As shown in Table XII, although the hash code length of the surrogate model is different from the victim model, the ASR can exceed 60% in all settings, which demonstrates the transferability of *PADHASH* across models with different hash codes lengths, making *PADHASH* more practical in a real-world attack. In addition, the reason why *PADHASH* has transferability in hash codes is that the clean trigger images and the malicious images are similar in the feature space, even though the hash code lengths are different.

Metric threshold. We set a threshold to measure whether an attack is successful. In the above experiment, we set the threshold to 0.3, which means that the attack is considered successful only when the target category images account for more than 30% of the retrieved images. To eliminate the impact of threshold selection on the experimental results, we counted the attack success rates under different thresholds.

As shown in Table XIII, as the threshold increases, the attack success rate of our method and the baseline decreases. However, our method has a higher ASR under various threshold selections, which demonstrates that the threshold selection will not affect the conclusion that our method can improve the ASR.

J. Visualization

Enhanced concealment translates to heightened difficulty in detecting these poisoned data instances. Therefore, we present a selection of sample poisoned images and clean images, as depicted in Figure 6. Remarkably, these visual representations reveal an exceedingly subtle distinction between the poisoned and clean images sourced from the ImageNet100 dataset, which demonstrates that the poisoned images generated by *PADHASH* are also visually concealed. In addition, we emphasize that in real-world attack scenarios, the poisoned images we generate will be more visually invisible. Because the images in the real world have a larger size, and the perturbation search space is larger. Therefore, gradient matching can be effectively accomplished within more constrained perturbation bounds, contributing to the minimal visual divergence observed.

VI. Conclusion

In this paper, we propose the first data poisoning attacks against deep hashing models to explore their potential risks. The attacker generates the poison images to compromise the deep hashing models. When users query with clean trigger images, they will obtain malicious images such as violent, explicit, and private images. Our experiments in gray-box and black-box scenarios validate that our proposed data poisoning attacks against deep hashing models are effective and practical on different datasets and models. In addition, we propose a *Strict Gradient-Matching* method to generate poisoned images, which has been demonstrated to enhance the attack success rate. Our proposed attack method reveals the potential risks of the deep hashing model. Therefore, we call on not only paying attention to the performance of deep hashing models but also to the security of deep hashing models.

REFERENCES

[1] X. Li, J. Yang, and J. Ma, "Recent developments of content-based image retrieval (cbir)," *Neurocomputing*, vol. 452, pp. 675–689, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231220319044

[2] H. Zhu, M. Long, J. Wang, and Y. Cao, "Deep hashing network for efficient similarity retrieval," in *Proceedings of the AAAI conference on Artificial Intelligence*, vol. 30, no. 1, 2016.

[3] T.-T. Do, T. Hoang, D.-K. Le Tan, A.-D. Doan, and N.-M. Cheung, "Compact hash code learning with binary deep neural network," *IEEE Transactions on Multimedia*, vol. 22, no. 4, pp. 992–1004, 2020.

[4] 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 *IJCAI*, 2020, pp. 825–831.

[5] Y. Shi, X. Nie, M. Chen, L. Lian, and Y. Yin, "Deep hashing with weighted spatial importance," *IEEE Transactions on Multimedia*, vol. 23, pp. 3778–3792, 2021.

[6] 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.

[7] Y. Pei, Z. Wang, N. Li, H. Chen, B. Huang, and W. Tu, "Deep hashing network with hybrid attention and adaptive weighting for image retrieval," *IEEE Transactions on Multimedia*, vol. 26, pp. 4961–4973, 2024.

[8] 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 *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I*. Berlin, Heidelberg: Springer-Verlag, 2020, p. 618–634. [Online]. Available: https://doi.org/10.1007/978-3-030-58452-8_36

[9] X. Wang, Z. Zhang, B. Wu, F. Shen, and G. Lu, "Prototype-supervised adversarial network for targeted attack of deep hashing," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 16 352–16 361.

[10] X. Wang, Z. Zhang, G. Lu, and Y. Xu, "Targeted attack and defense for deep hashing," in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 2298–2302. [Online]. Available: https://doi.org/10.1145/3404835.3463233

[11] S. Hu, Z. Zhou, Y. Zhang, L. Y. Zhang, Y. Zheng, Y. He, and H. Jin, "Badhash: Invisible backdoor attacks against deep hashing with clean label," in *Proceedings of the 30th ACM International Conference on Multimedia*, 2022, pp. 678–686.

[12] K. Gao, J. Bai, B. Chen, D. Wu, and S.-T. Xia, "Backdoor attack on hash-based image retrieval via clean-label data poisoning," in *34th British Machine Vision Conference 2023, BMVC 2023, Aberdeen, UK, November 20-24, 2023*. BMVA, 2023. [Online]. Available: https://papers.bmvc2023.org/0172.pdf

[13] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan, "Supervised hashing for image retrieval via image representation learning," in *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, ser. AAAI'14. AAAI Press, 2014, p. 2156–2162.

[14] Y. Li and J. van Gemert, "Deep unsupervised image hashing by maximizing bit entropy," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 3, 2021, pp. 2002–2010.

[15] Z. Cao, M. Long, J. Wang, and P. S. Yu, "Hashnet: Deep learning to hash by continuation," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5608–5617.

[16] 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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 3083–3092.

[17] H. Zhu and S. Gao, "Locality-constrained deep supervised hashing for image retrieval," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, ser. IJCAI'17. AAAI Press, 2017, p. 3567–3573.

[18] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[19] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[20] L. Struppek, D. Hintersdorf, D. Neider, and K. Kersting, "Learning to break deep perceptual hashing: The use case neuralhash," in *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, 2022, pp. 58–69.

[21] T. Gu, B. Dolan-Gavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," *arXiv preprint arXiv:1708.06733*, 2017.

[22] R. Tang, M. Du, N. Liu, F. Yang, and X. Hu, "An embarrassingly simple approach for trojan attack in deep neural networks," in *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 2020, pp. 218–228.

[23] J. Steinhardt, P. W. W. Koh, and P. S. Liang, "Certified defenses for data poisoning attacks," *Advances in neural information processing systems*, vol. 30, 2017.

[24] L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli, "Towards poisoning of deep learning algorithms with back-gradient optimization," in *Proceedings of the 10th ACM workshop on artificial intelligence and security*, 2017, pp. 27–38.

[25] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," in *Proceedings of the 29th International Coference on International Conference on Machine Learning*, 2012, pp. 1467–1474.

[26] M. Fang, M. Sun, Q. Li, N. Z. Gong, J. Tian, and J. Liu, "Data poisoning attacks and defenses to crowdsourcing systems," in *Proceedings of the web conference 2021*, 2021, pp. 969–980.

[27] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein, "Poison frogs! targeted clean-label poisoning attacks on neural networks," *Advances in neural information processing systems*, vol. 31, 2018.

[28] C. Zhu, W. R. Huang, H. Li, G. Taylor, C. Studer, and T. Goldstein, "Transferable clean-label poisoning attacks on deep neural nets," in *International Conference on Machine Learning*. PMLR, 2019, pp. 7614–7623.

[29] J. Geiping, L. H. Fowl, W. R. Huang, W. Czaja, G. Taylor, M. M. 0001, and T. Goldstein, "Witches' brew: Industrial scale data poisoning via gradient matching," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. [Online]. Available: https://openreview.net/forum?id=01olnfLIbD

[30] H. Souri, L. Fowl, R. Chellappa, M. Goldblum, and T. Goldstein, "Sleeper agent: Scalable hidden trigger backdoors for neural networks trained from scratch," *Advances in Neural Information Processing Systems*, vol. 35, pp. 19 165–19 178, 2022.

[31] A. Krizhevsky, "Learning multiple layers of features from tiny images," 2009. [Online]. Available: https://api.semanticscholar.org/CorpusID:18268744

[32] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *Int. J. Comput. Vision*, vol. 115, no. 3, p. 211–252, dec 2015. [Online]. Available: https://doi.org/10.1007/s11263-015-0816-y

[33] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft coco: Common objects in context," 2015.

[34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.