CERTPHASH: Towards Certified Perceptual Hashing via Robust Training

Yuchen Yang, Qichang Liu, Christopher Brix[1], Huan Zhang[2], and Yinzhi Cao*
{yc.yang, qliu85, yinzhi.cao}@jhu.edu,
brix@cs.rwth-aachen.de, huan@huan-zhang.com
The Johns Hopkins University, [1]RWTH Aachen University, [2]UIUC

Abstract

Perceptual hashing (PHash) systems—e.g., Apple's NeuralHash, Microsoft's PhotoDNA, and Facebook's PDQ—are widely employed to screen illicit content. Such systems generate hashes of image files and match them against a database of known hashes linked to illicit content for filtering. One important drawback of PHash systems is that they are vulnerable to adversarial perturbation attacks leading to hash evasion or collision. It is desirable to bring provable guarantees to PHash systems to certify their robustness under evasion or collision attacks. However, to the best of our knowledge, there are no existing certified PHash systems, and more importantly, the training of certified PHash systems is challenging because of the unique definition of model utility and the existence of both evasion and collision attacks.

In this paper, we propose CERTPHASH, the *first* certified PHash system with robust training. CERTPHASH includes three different optimization terms, anti-evasion, anti-collision, and functionality. The anti-evasion term establishes an upper bound on the hash deviation caused by input perturbations, the anti-collision term sets a lower bound on the distance between a perturbed hash and those from other inputs, and the functionality term ensures that the system remains reliable and effective throughout robust training. Our results demonstrate that CERTPHASH not only achieves non-vacuous certification for both evasion and collision with provable guarantees but is also robust against empirical attacks. Furthermore, CERTPHASH demonstrates strong performance in real-world illicit content detection tasks.

1 Introduction

Perceptual hashing (PHash) systems—such as Microsoft's PhotoDNA [30], Facebook's PDQ [10], and Apple's NeuralHash [1]—are widely deployed on many platforms to detect and mitigate the dissemination of illicit content, notably child sexual abuse material (CSAM), for secure and ethical online environments in the digital age. Specifically, PHash systems convert images into unique hashes for efficient comparison against databases of known illicit content for filtering, thereby serving as the fundamental defense mechanism in digital content moderation.

Although PHash systems are important in illicit content filtering, recent studies [35,41] have found that PHash systems—no matter if they are based on traditional algorithms or neural networks—are vulnerable to adversarial perturbations, e.g., evasion and collision attacks. On one hand, an adversary can add small perturbations to an illicit image so that the hash of the perturbed image significantly diverges from the original, thus evading content filtering. On the other hand, an adversary can add perturbations to a legitimate image so that the hash of the perturbed image coincides with that of an illicit image, thereby generating false positives to either cause legitimate content to be filtered or even to overload the PHash systems. At the same time, if an allowlist is adopted, an adversary can also add perturbations to an illicit image for a collision with a hash in the allowlist for bypasses.

Our work aims to bring *provable* guarantees to PHash systems against evasion and collison attacks. In classification models, one provable approach against adversarial perturbations is certified robustness [8, 36]. It provides provable guarantees to ensure that adversarial perturbations do not exist under bounded inputs. The high-level idea is to first train a robust learning model, e.g., via adversarial training [13,29,44] or specialized certified training methods [14, 33, 47, 54], and then verify its robustness using neural network verification techniques, e.g., α,β-CROWN [23, 38, 39, 43, 52, 55–57].

To the best of our knowledge, there are no prior works on certifying the robustness of PHash systems. More importantly, such a certification is challenging: For example, directly applying adversarial training for a neural network-based PHash system will cause very loose bounds during certified evaluation. This means the calculated range over which the output remains stable under perturbed input is extensive, leading to high certified collision or evasion rates for a given input perturbation. This is due to two inherent properties of PHash systems compared with other learning systems.

*This work was done when Qichang Liu was a summer intern at JHU.

First, the utility definition of PHash systems is different from that of traditional learning models. Traditional learning models often have a ground truth output, e.g., a label (like cat vs. dog) or a continuous value (e.g., turning angles for self-driving cars), but PHash systems do not. That is, the exact hash outputted by a PHash system for one given image can be any arbitrary value; instead, the relations between hashes of different images, e.g., the hash distance of rotated original images, determine the utility of PHash systems. Such a different utility definition not only makes training harder, but also leads to a different utility-robustness trade-off compared with traditional learning models.

Second, attacks against PHash systems are also diversified compared with traditional learning models. Unlike classification models, where a fixed set of labels are pre-defined for all inputs, PHash systems are designed to predict distinct outputs for every perceptually different image in the training set. As discussed, an adversary may not only evade PHash systems by drastically changing the PHash values but also cause collisions with hashes of existing images for either denial of service or bypasses. Given a set of images, one needs to calculate two certified rates, one for evasion and the other for collision.

In this paper, we propose CERTPHASH, the *first* certified robust PHash system providing *provable defense* against both evasion and collision attacks without sacrificing utilities. At a high level, the certified robustness of CERTPHASH is achieved via specially designed loss functions during training. They are built with efficient bound-propagation-based verification techniques, representing the certified bound of the worst-case evasion or collision attacks for specified input perturbations.

During training, CERTPHASH addresses the aforementioned utility and attack diversity challenges via three loss functions, anti-evasion, anti-collision, and functionality. First, the anti-evasion loss minimizes the upper bound of the hash distance between a training set image and its perturbed or randomly transformed version, ensuring that outputs for the same input remain similar, even when modified. Second, the anti-collision loss calculates a lower bound of the distance between the two hashes from a pair of distinct images within a training batch, imposing penalties if the hashes from different inputs stay too close. Lastly, the functionality loss stabilizes the certified bounds during training and prevents the model from converging to a suboptimal solution, thus achieving a good utility.

After robust training, CERTPHASH assesses the certified evasion and collision bounds for the trained PHash system. For evasion verification, CERTPHASH first calculates the bounds on hashes from perturbed images and then identifies instances where the distance between these bounds and the clean output exceeds a commonly adopted PHash match threshold [35]. For collision verification, CERTPHASH inversely calculates the input bounds of an image given its hash, and marks instances whose input bounds are overlapped with the others as certified collisions.

Extensive experiments demonstrate that CERTPHASH can successfully train a PHash system with 100% certified no-evasion and 95% no-collision rates with pertubation $\|\delta\|_\infty \leq 8/255$, as opposed to 0.01% certified no-evasion and 0% no-collision for adversarial training with the same setting, while effectively maintaining the functionality of PHash. Our evaluation using empirical evasion and collision attacks also demonstrates the correctness of the certification. To summarize, we make the following contributions:

- We design and implement the *first* certified Phash systems, featuring robust training with novel anti-evasion, anti-collision, and functionality terms.
- We systematically formulate and address the formal verification problem for PHash systems, setting a benchmark for the certified no-evasion and no-collision rates that will guide future research on the robustness of PHash systems.
- We extensively compare CERTPHASH with established systems such as Facebook's PDQ [10], Apple's NeuralHash, and Microsoft's PhotoDNA, which have been either open-sourced or previously extracted and published by third parties [2, 11, 19]. Our evaluation shows that CERTPHASH is not only certified robust against both evasion and collision attacks but also effective in real-world illicit content detection tasks.

2 Background and Preliminaries

2.1 Perceptual Hashing

Perceptual hashing (PHash) generates hash values from images to capture their perceptual similarities. A PHash function $f(\cdot)$ is defined as

$$f : \mathcal{X} \to \mathcal{Y} \tag{1}$$

where $\mathcal{X}$ represents the domain of input images and $\mathcal{Y}$ is the corresponding output hash space. For any input image $x \in \mathcal{X}$, the function outputs a hash value $f(x) = y \in \mathcal{Y}$. For a hash y, the set of images generating this hash ($\{x \in \mathcal{X} | f(x) = y\}$) is known as the preimage of y. The goal is to ensure that images with similar content produce similar hash values, and hence a match. Images with dissimilar content should produce different hashes, resulting in a non-match. This is formalized as:

Definition 1. *For any image pair (x_i, x_j), a **match** is defined by a distance function $D(\cdot)$ when $D(f(x_i), f(x_j)) \leq \Delta_y$. Conversely, a **non match** is defined as when $D(f(x_i), f(x_j)) > \Delta_y$.*

Various perceptual hashing systems have been proposed and deployed. Some are not based on neural networks: for example, Microsoft's PhotoDNA generates a 144-dimensional numeric hash vector [30], and Facebook's PDQ produces a 256-bit binary hash [10]. In contrast, Apple uses deep neural networks for better robustness against benign image modifications, such as scaling and rotation. The resulting hash from

its NeuralHash system contains a 96-bit binary representation [1]. However, all of them are vulnerable to adversarial attacks [35, 41]. This highlights the need for enhanced robustness in PHash systems.

2.2 Attacks on Perceptual Hashing

Evasion Attack. Evasion attacks aim to subtly modify an image so that its hashed output diverges from the original hash, while the visual appearance remains largely unchanged. These modifications can include benign transformations like rotation, cropping, and JPEG compression, as well as carefully crafted perturbations designed for more powerful attacks. For an image x_i with its original hash $f(x_i)$, an evasion attack aims to find the modification function, $M(\cdot)$ that achieves the following:

$$D(f(x_i), f(M(x_i))) > \Delta_y \tag{2}$$

Existing work reveals that popular PHash systems are vulnerable to evasion under both benign transformations [15, 18, 41] and adversarial perturbations [35, 44, 45, 50, 53]. Specifically, Struppek et al. [41] demonstrate that NeuralHash can be evaded with a few pixel modifications. Prokos et al. [35] show that PDQ and PhotoDNA are vulnerable to the Hop Skip Jump Attack [7] and sensitive to JPEG compression and random crops. Jain et al. [18] show that a l_2 perturbation of just radius 0.10 is sufficient to attack PDQ.

Collision Attack. Collision attacks aim to generate a second preimage that matches the hash of a target image while remaining perceptually distinct from it. This is more complex than evasion as it involves not just altering the hash, but specifically achieving a hash collision with a target. As a result, previous work often relies on optimized perturbations as the modification. Given a target image x_i and its hash $f(x_i)$, the collision attack is to find a modification function $M(\cdot)$ such that, starting from another image x_j, $M(x_j)$ becomes part of the preimage of all hashes similar to $f(x_i)$:

$$\text{s.t. } D(f(x_i), f(x_j)) > \Delta_y, \quad D(f(x_i), f(M(x_j))) \leq \Delta_y \tag{3}$$

Still, the Phash systems deployed in the real world are all vulnerable to collision attacks [15, 18, 21, 35]. Prokos et al. [35] find collisions on PDQ and PhotoDNA with the perturbation generated via Monte Carlo gradient estimation [32]; Struppek et al. find collisions on NeuralHash with the perturbation optimized with gradient.

2.3 Certified Robust Training and Verification

Robust Training. Certified robust training involves minimizing a robust loss derived from a verifier, which represents the worst-case loss under specified input perturbation settings. This process is typically applied in classification tasks and can be formulated as the following min-max optimization problem:

$$\min_{\theta} \mathbb{E}_{(x,y)} \left[\max_{\delta} D(f_\theta(x+\delta), y) \right] \tag{4}$$

Here, f_θ is a neural network with parameters θ, x is an input with ground-truth label y. δ is a perturbation constrained by ℓ_p ball of radius ε and D is a distance function. The inner maximization simulates the adversarial attacks; then, the outer minimization aims to reduce the worst-case loss of the model to make it more robust against adversarial perturbations.

Currently, the most efficient and effective method for getting deterministic bounds for general models is Interval Bound Propagation (IBP) [14, 31, 46]. In IBP training, Balunovic et al. [4] incorporate these bounds into the training process, while Xiao et al. [49] add a ReLU regularizer for the stability of certified bounds. Shi et al. [40] further improve the stability and accelerate the training. Besides IBP, randomized smoothing [9, 24, 25, 37] offers certified robustness with probabilistic bounds, though they are computationally expensive as they require multiple inferences per input to achieve robustness.

Robust Verification. Classic verification assesses whether a neural network f_θ adheres to an output constraint $\mathcal{O}(f_\theta, x_i)$ for all inputs $x_i \in \mathcal{X}$ within a specified domain $\mathcal{I}$, typically defined as an ℓ_p-norm ball around the data point x_i, such that $\mathcal{I} = (x \mid ||x - x_i||_p \leq \varepsilon)$. We treat output constraints and the set of outputs that satisfy them interchangeably, denoted as $\mathcal{O} \subseteq Y$ when $\mathcal{O}(f_\theta, x_i) = \{y \in Y | D(y, f_\theta(x_i)) \leq \Delta_y\}$. This process, known as forward verification, guarantees that all perturbed images within $\mathcal{I}$ meet the hash match constraint by ensuring $\forall x \in \mathcal{I} : f(x) \in \mathcal{O}(f(x_i))$, thereby providing certified robustness against evasion attacks. State-of-the-art verifiers include α,β−CROWN [6, 34, 43, 51, 54, 55], VeriNet [16, 17], MN-BaB [12], Marabou [20, 48], NNenum [3], and NNV [42].

Conversely, Kotha et al. [23] recently introduce the concept of inverse verification, where given an output constraint $\mathcal{O}$, the verifier computes a tight over-approximation of the input domain, that is $f_\theta^{-1}(\mathcal{O}) \subseteq \mathcal{X}$, which contains all inputs that can produce outputs satisfying $\mathcal{O}$. This over-approximation is also referred to as the provable overapproximation of the preimage.

We see the potential for certified robustness in PHash: forward verification ensures that all perturbed images within a specified range satisfy the hash match constraint, providing certified robustness against evasion. Inverse verification, on the other hand, establishes the provable preimage for a given hash, ensuring robustness against collisions. However, most existing certified robustness approaches are tailored to classification tasks, making it challenging to directly apply them to PHash, where hash values are treated as a regression problem.

3 Problem Formulation

This section outlines the training of a neural network-based PHash model and the threat model for CERTPHASH.

3.1 PHash as a Neural Network

To achieve certified robustness through robust training and verification, we model the PHash system as a neural network $f_\theta : \mathcal{X} \to \mathcal{Y}$ with parameters θ. Here, $\mathcal{X}$ is the input image

domain, and $\mathcal{Y}$ is the output hash domain. We utilize image-hash pairs $(x, y) \in (\mathcal{X}, \mathcal{Y})$ as training data for f_θ. The hashes are output from PDQ, PhotoDNA, or NeuralHash systems. This formulation aims to replicate the functionality of real-world PHash systems under a benign setting. (Mis-)matches are decided by the threshold Δ_y (see Definition 1).

3.2 Threat Model

We consider two types of adversaries in our threat model: *certified robustness* and *empirical robustness*. Both adversaries are assumed to have white-box access to the PHash model f_θ and are capable of crafting perturbations as adversarial examples (Instead, image transformations like rotations are tested under a benign setting to evaluate the model's functionality). We provide detailed descriptions of each adversary's knowledge, capabilities, and objectives below.

Knowledge. Both adversaries have complete knowledge of the PHash model, including:

- *Model:* Full knowledge of the model's architecture, parameters, and weights.
- *PHash system:* Specific PHash algorithm the model learns from, including its match threshold that determines whether images match or not.

Capabilities. Both adversaries have the following three capabilities, with the key difference in the constraints on perturbations:

- *Access to allowlist and blocklist:* Ability to use images and their hashes from both the allowlist and blocklist as targets or sources for attacks.
- *Querying the model:* Limited numbers of queries to the model with either original or modified images, access to intermediate outputs like gradients, as well as the final output hash.
- *Perturbation optimization:* They can optimize a perturbation δ and add it to the original images based on model responses. The difference between two adversaries is on constraints on δ:
  - *Certified robustness:* Adheres to a predefined perturbation setting during both training and verification, such as constraining δ within an ℓ_p ball of radius ε around an example x.
  - *Empirical robustness:* More flexible perturbation approach allowing deviations from the training perturbation type and levels. For instance, an ℓ_2 perturbation can be used even if the model was trained under ℓ_∞ norms. The radius ε is not fixed but is limited by practical considerations like the number of queries and the perceptual similarity to the original image.

Objectives. The objectives for both adversaries are the same and focus on evasion and collision. The difference lies in the perturbation δ: for certified robustness, δ represents the worst-case perturbation calculated via a verifier. For empirical robustness, δ is optimized via gradient descent, allowing the adversary to craft perturbations that maximize the likelihood of evasion or collision

- *Evasion attack:* The objective is to subtly modify an image x_i so that its hash changes significantly, exceeding the match threshold, that is $D(f_\theta(x_i), f_\theta(x_i + \delta)) > \Delta_y$. This is achieved by optimizing a perturbation δ to maximize the hash distance D under the constraint $\|\delta\|_p \leq \varepsilon$.
- *Collision attack:* The objective is to take two distinct images x_i and x_j, which do not have matching hashes in the benign setting, and modify them such that their hashes match, i.e. $D(f_\theta(x_i), f_\theta(x_j)) \leq \Delta_y$. This is achieved through a carefully optimized perturbation δ to minimize the hash distance between the two images while subject to a perturbation constraint $\|\delta\|_p \leq \varepsilon$.

4 CertPHash

4.1 Overview

Key Idea. Given an input image $x_i \in \mathcal{X}$, CertPHash aims to train a model f_θ that outputs the perceptual hashing $f_\theta(x_i)$ with four goals:

- *Goal I. [PHash Modeling]:* f_θ should preserve the PHash behavior with a clean input, such that $f_\theta(x_i) \to y_i$, where y_i is the hash of x_i following the patterns of systems like PDQ, PhotoDNA, or NeuralHash.
- *Goal II. [Evasion Robustness]:* f_θ should be robust to evasion attacks with a perturbed input, such that $D(f_\theta(x_i), f_\theta(x_i + \delta)) \leq \Delta_y$, subject to $\|\delta\|_p \leq \varepsilon$.
- *Goal III. [Collision Robustness]:* f_θ should be robust to collision attacks with a perturbed input, such that $D(f_\theta(x_i), f_\theta(x_j + \delta)) > \Delta_y$, subject to $\|\delta\|_p \leq \varepsilon$ and $D(f_\theta(x_i), f_\theta(x_j)) > \Delta_y$, where x_i and x_j are distinct images.
- *Goal IV. [Functionality]:* f_θ should maintain stable bounds during certified robust training to achieve more reliable and faster convergence since it relies on bound propagation through the network. This is crucial when training a PHash model to generate hashes.

To achieve Goal I, CertPHash defines a non-robust loss $\mathcal{L}_{\text{non-robust}}$, using hash values from existing PHash systems as labels for each image x_i. During training, CertPHash applies random transformations to enhance the robustness to benign transformations. For Goal II, CertPHash introduces an anti-evasion term $\mathcal{L}_{\text{evasion}}$, which is the upper bound of $\mathcal{L}_{\text{non-robust}}$ under the worst-case δ. To achieve Goal III, CertPHash proposes an anti-collision term $\mathcal{L}_{\text{collision}}$, which calculates the lower bound of the distance between hash outputs from different images under the worst-case δ and penalizes if the lower bound exceeds Δ_y. For Goal IV, CertPHash adds a functionality regularizer $\mathcal{L}_{\text{func}}$ to mitigate unstable bounds and imbalanced contributions from neurons. We evaluate the importance and necessity of each regularizer in Section 6.5 through ablation studies.

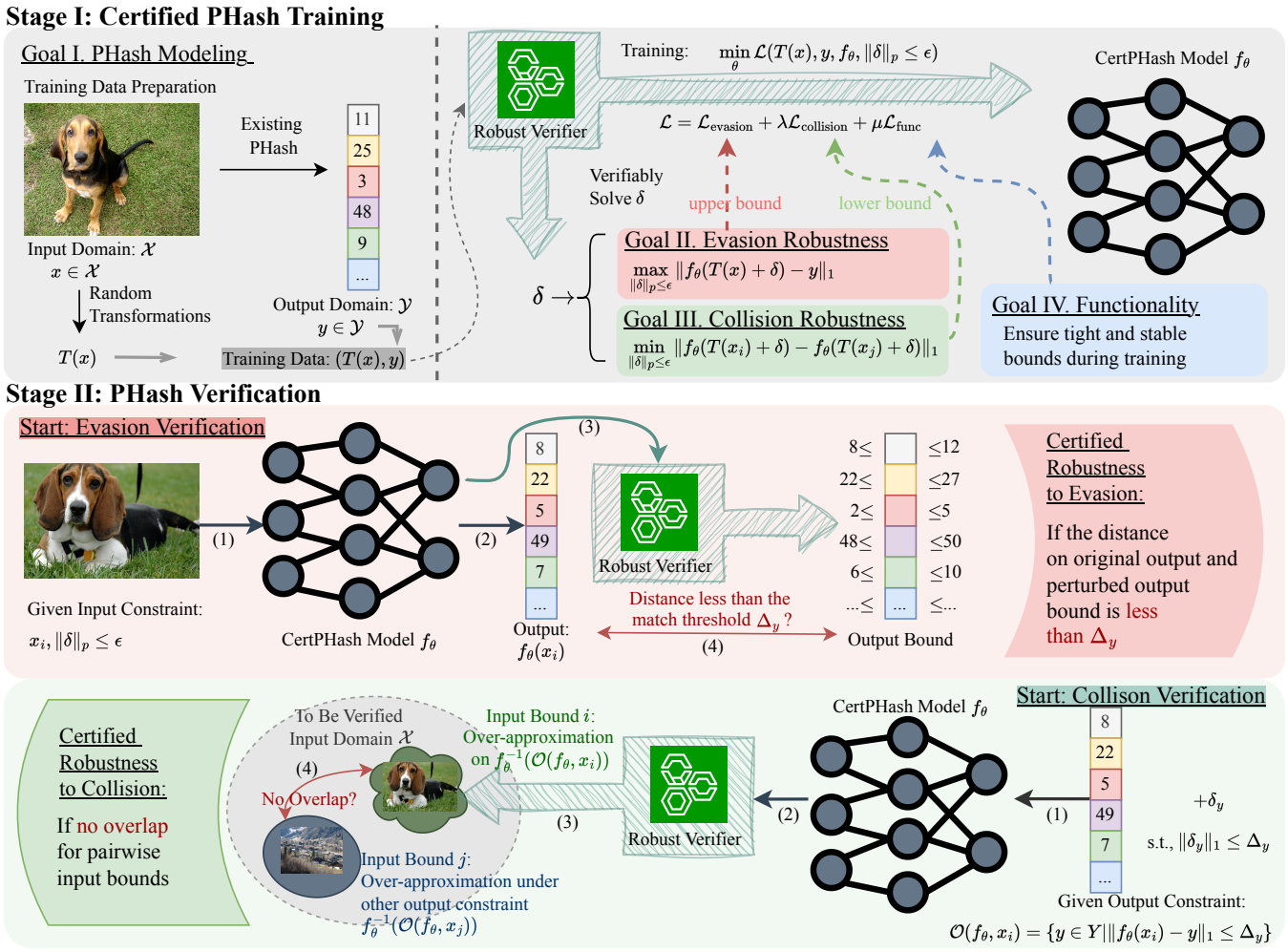


Figure 1: The CERTPHASH pipeline, consisting of two stages: certified robust training, where multiple loss terms are integrated into the robust loss $\mathcal{L}$ for model training, and PHash verification, divided into evasion and collision verification to ensure output integrity under perturbations and non-overlapping input bounds, respectively.

Challenges. Unlike classification problems, which typically focus on a single robustness goal, i.e., minimizing the upper bound loss between perturbed outputs and their ground truth label. CERTPHASH must balance two conflicting robustness objectives. First, it requires consistent hashes for identical inputs to prevent evasion. Second, it must generate unique hashes for different inputs to avoid collisions. Managing these competing goals needs multiple, carefully balanced loss terms. Furthermore, unlike classification tasks where output constraints are simply set by ensuring non-ground truth probabilities are lower than those of the ground truth label, CERTPHASH 's output consists of multiple hash values without absolute ground truth. Instead, the constraints are based on distances relative to the original output hashes, requiring stricter calculations. This added complexity demands stable bounds, which CERTPHASH addresses through the functionality term inspired by [40], ensuring effectiveness and reliability.

Overall Pipeline. Figure 1 illustrates the pipeline of CERTPHASH, which includes two main stages: certified PHash training and PHash verification. During the certified robust training, CERTPHASH incorporates the four aforementioned goals into multiple loss terms within the robust loss $\mathcal{L}$, training the model f_θ. The PHash verification stage consists of evasion and collision verification. Evasion verification checks if, given an input perturbation constraint $\|\delta\|_p \le \varepsilon$, the output remains within the PHash match threshold Δ_y compared to the original output. Collision verification assesses whether the preimage of similar hashes, i.e., $f_\theta^{-1}(\mathcal{O}(f_\theta, x_i))$, does not overlap with those from another image. We will detail these two stages in the following sections.

4.2 Certified PHash Training

In this section, we describe our approach to achieving the four goals of CERTPHASH and address the associated challenges. We propose a certified PHash training framework for CERT-

PHASH, structured in four key components, each tailored to specific objective functions to achieve PHash modeling, evasion robustness, collision robustness, and functionality. CERTPHASH optimizes a PHash model f_θ with parameters θ using the objective function $\mathcal{L}$ defined in Equation 5:

$$\min_\theta \mathcal{L}, \text{ where } \mathcal{L} = \mathcal{L}_{\text{evasion}} + \lambda \mathcal{L}_{\text{collision}} + \mu \mathcal{L}_{\text{func}} \tag{5}$$

Each term in the loss function is described in the following paragraphs. The parameter λ balances the collision term, while μ balances the functionality term. Note that μ is gradually reduced from its initial value to 0 during the warmup period. The warmup phase is to schedule ε from 0 to its target value, ensuring stable bounds. After the warmup, CERTPHASH relies on $\mathcal{L} = \mathcal{L}_{\text{evasion}} + \lambda \mathcal{L}_{\text{collision}}$ for certified robust training.

Achieve PHash Modeling. To train a PHash model f_θ that retains the behavior of existing PHash systems while improving robustness against benign transformations, we employ a dynamic approach during each training batch. Specifically, we randomly sample up to n transformations, which are sequentially applied to the image, denoted as $T \in \mathcal{T}$. Here, $\mathcal{T}$ includes transformations from three categories: geometric transformations (including rotation, cropping, and flipping), color transformations (including changes to hue, brightness, contrast, and saturation), and compression transformations (such as JPEG compression). We assign equal probabilities to processing an original image without any transformation and to applying up to n transformations to ensure the model recognizes both clean and transformed images. The parameter n represents the maximum number of possible transformations, with the actual number applied varying randomly within the range $(0, n]$.

For illustration simplicity, we denote both transformed and original images as $T(x)$, where T is an identity transformation for the latter. The objective function $\mathcal{L}_{\text{non-robust}}$, also known as non-robust (regular) loss, is formalized as

$$\mathcal{L}_{\text{non-robust}} = \|f_\theta(T(x)) - y\|_1 \tag{6}$$

Achieve Evasion Robustness. CERTPHASH achieves evasion robustness by solving the robust optimization problem outlined in Equation 7. Unlike empirical adversarial training, which addresses the inner maximization through adversarial attacks without guaranteeing a worst-case δ, CERTPHASH uses a robust verifier, such as α, β-CROWN, employing IBP with tighter linear relaxation to solve this maximization verifiably.

As a result, $\overline{\mathcal{L}}_{\text{non-robust}}$ serves as a *certified upper bound* for this optimization, which covers the worst-case perturbation scenario. The outer minimization in Equation 5 then reduces the loss between the output hash of the worst-case perturbed image and its original label, thereby ensuring evasion robustness—both the original and perturbed images produce similar output hashes.

$$\begin{gathered}\mathcal{L}_{\text{evasion}} = \overline{\mathcal{L}}_{\text{non-robust}}, \text{ where} \\ \overline{\mathcal{L}}_{\text{non-robust}} \geq \mathbb{E}_{(x,y)\in(\mathcal{X},\mathcal{Y})}\left[\max_{\|\delta\|_p \leq \varepsilon_{\text{target}}} \|f_\theta(T(x)+\delta) - y\|_1\right]\end{gathered} \tag{7}$$

Achieve Collision Robustness. To enhance collision robustness, CERTPHASH adds a collision loss $\mathcal{L}_{\text{collision}}$, as defined in Equation 8. This loss penalizes scenarios where the pairwise distance $\mathcal{L}_{\Delta_y}$ between hashes from different images in the current batch falls below the match threshold Δ_y. The goal is to help distinct images yield distinct hashes.

Similar to $\mathcal{L}_{\text{evasion}}$, the inner maximization problem is solved by a verifier. However, for collision robustness, the worst-case perturbation minimizes the distance between predictions, in contrast to $\mathcal{L}_{\text{evasion}}$ where it maximizes the loss. This minimum distance, noted as $\underline{\mathcal{L}}_{\Delta_y}$, serves as the *certified lower bound* on distances between each pair of hashes. Keeping this lower bound above Δ_y is important, as any smaller distance suggests a potential collision.

$$\begin{gathered}\mathcal{L}_{\text{collision}} = \texttt{ReLU}(\Delta_y - \underline{\mathcal{L}}_{\Delta_y}), \text{ where} \\ \underline{\mathcal{L}}_{\Delta_y} \leq \mathbb{E}_{(x_i,x_j)\in\mathcal{X}}\left[\min_{\|\delta\|_p \leq \varepsilon_{\text{target}}} \|f_\theta(T(x_i)+\delta) - f_\theta(T(x_j)+\delta)\|_1\right]\end{gathered} \tag{8}$$

Achieve Functionality. Certified robust training typically starts with a perturbation schedule, where the model initially trains with $\mathcal{L}_{\text{non-robust}}$, and gradually increases the perturbation radius from zero to the target value, $\varepsilon_{\text{target}}$. This helps stabilize the certified bounds during training and prevents the model from converging to a suboptimal solution, which is thus important in maintaining the functionality of the model. We add two regularizers, $\mathcal{L}_t$ and $\mathcal{L}_r$, to shorten the training and enhance the tightness of bounds. Each hidden layer in f_θ, denoted as h_i, has interval bounds defined by $\underline{h}_i \leq h_i \leq \overline{h}_i$, subject to $\forall\|\delta\| \leq \varepsilon$. The tightness of these bounds, $t_i = \overline{h}_i - \underline{h}_i$, is controlled by Equation 9. This term penalizes if the current average bound tightness, $\mathbb{E}(t_i)$, exceeds the initial average, $\mathbb{E}(t_0)$, adjusted by a tolerance factor τ:

$$\mathcal{L}_t = \mathbb{E}_i\left[\texttt{ReLU}\left(\tau - \frac{\mathbb{E}(t_0)}{\mathbb{E}(t_i)}\right)\right] \tag{9}$$

Equation 10 details the regularization term $\mathcal{L}_r$, which ensures that both active and inactive ReLU neurons contribute equally to the model's performance. This regularization evaluates balance by comparing the contribution ratios of neurons to the overall mean and variance of the layer, centered at $a_i = (\overline{h}_i + \underline{h}_i)/2$. The ideal ratio indicating a balance, denoted as m for the mean and v for the variance, should approach 1. Falling below the threshold τ means a significant imbalance, triggering a penalty:

$$\begin{gathered}\mathcal{L}_r = \mathbb{E}_i\left[\texttt{ReLU}\left(\tau - \min\left(m_i, \frac{1}{m_i}\right)\right) + \texttt{ReLU}\left(\tau - \min\left(v_i, \frac{1}{v_i}\right)\right)\right], \\ \text{where } m_i = \frac{\sum_j \mathbb{I}(\underline{h}_{i,j} > 0) a_{i,j}}{-\sum_j \mathbb{I}(\overline{h}_{i,j} > 0) a_{i,j}},\ v_i = \frac{\sum_j \mathbb{I}(\underline{h}_{i,j} > 0)(a_{i,j} - \mathbb{E}(a_i))^2}{\sum_j \mathbb{I}(\overline{h}_{i,j} > 0)(a_{i,j} - \mathbb{E}(a_i))^2}\end{gathered} \tag{10}$$

For simplicity in our illustrations, we define the combined functionality loss term as $\mathcal{L}_{\text{func}}$:

$$\mathcal{L}_{\text{func}} = \mathcal{L}_{\text{t}} + \mathcal{L}_{\text{r}} \tag{11}$$

4.3 PHash Verification

To assess whether CERTPHASH successfully trains a PHash model f_θ that is robust against both evasion and collision attacks, we introduce a PHash verification framework including evasion verification and collision verification.

Evasion Verification. CERTPHASH verifies that for all perturbed inputs $x_i+\delta$, where $\|\delta\|_p \leq \varepsilon$, the output hash $f_\theta(x_i+\delta)$ remains within the match threshold Δ_y when compared to the original hash $f_\theta(x_i)$. This verification is based on a specified distance function D, such as the l_1 norm, which is commonly used in systems like PDQ and PhotoDNA. Formally, we define a PHash model as robust against evasion if it satisfies the following condition:

$$D(f_\theta(x_i), f_\theta(x_i+\delta)) \leq \Delta_y,\ \forall i, \delta,\ \text{where } \|\delta\|_p \leq \varepsilon \tag{12}$$

Since $f_\theta(x_i)$ is a constant, this condition can be easily verified if we know the bounds of $f_\theta(x_i+\delta)$. We present f_θ as a computational graph, with nodes corresponding to mathematical operations (e.g., ReLU) and edges defining the computation flow. Bounds, representing guaranteed output ranges, are propagated through the graph from the input ($x_i+\delta$ where $\|\delta\|_p \leq \varepsilon$) to intermediate outputs and ultimately to the final layer. Various bound propagation methods can be applied. In our work, we use interval bound propagation (IBP [14]) and linear bound propagation (CROWN [57]) methods.

We briefly describe the bound propagation procedure below for unfamiliar readers. The simplest bound propagation method is IBP, where concrete lower and upper bounds are propagated on the computational graph, starting from the inputs to the output. To illustrate with a toy example, assume f_θ contains three linear layers with 1×1 (scalar) weights $W^{(1)}=2$, $W^{(2)}=3$, $W^{(3)}=4$, and ReLU activations in between. Let $(x_i+\delta)\in[-1,2]$. The first layer computes $z^{(1)} = W^{(1)}\cdot(x_i+\delta)$, giving lower bound $\underline{z^{(1)}} = 2\cdot(-1) = -2$ and upper bound $\overline{z^{(1)}} = 2\cdot 2 = 4$. After ReLU, $\underline{\hat{z}^{(1)}} = 0$ and $\overline{\hat{z}^{(1)}} = 4$. The second layer gives $\underline{\hat{z}^{(2)}} = 0$ and $\overline{\hat{z}^{(2)}} = 12$. Finally, we get the bounds for $f_\theta(x_i+\delta)\in[0,48]$.

On the other hand, the linear bound propagation method propagates a set of linear inequalities, one for the lower bound and one for the upper bound. We refer the readers to prior work [38] (Sec. D) for a comprehensive derivation. Linear bound propagation is more expensive than IBP (yet still quite efficient) and typically produces tighter bounds.

Collision Verification. CERTPHASH verifies whether the provably calculated overapproximation $\mathcal{I}_{\text{over}(\mathcal{O},f_\theta,x_i)} \supseteq f_\theta^{-1}(\mathcal{O}(f_\theta,x_i))$ under the constraint $\mathcal{O}(f_\theta,x_i) = \{y\in Y \mid \|f_\theta(x_i)-y\|_1 \leq \Delta_y\}$ does not overlap with the over-approximation of another preimage $\mathcal{I}_{\text{over}(\mathcal{O},f_\theta,x_j)} \supseteq f_\theta^{-1}(\mathcal{O}(f_\theta,x_j))$ for a different output hash under the constraint $\mathcal{O}(f_\theta,x_j) = \{y\in Y \mid \|f_\theta(x_j)-y\|_1 \leq \Delta_y\}$. Formally, we define a PHash model as verified robust to collision if it satisfies the following condition:

$$\begin{aligned} &\mathcal{I}_{\text{over}(\mathcal{O},f_\theta,x_i)} \cap \mathcal{I}_{\text{over}(\mathcal{O},f_\theta,x_j)} = \emptyset, \\ &\forall x_i, x_j \in \mathcal{X},\ \text{s.t. } x_i \neq x_j \text{ and } \|f_\theta(x_i), f_\theta(x_j)\|_1 > \Delta_y \end{aligned} \tag{13}$$

Note that the input domain $\mathcal{X}$ is infinite; therefore, in our experiments, we restrict our consideration to x_i and x_j within the specified test dataset. To calculate the over-approximation of the preimage, CERTPHASH computes the bounds on the input layers by inversely propagating constraints [23] from the output layers.

5 Experimental Setup

We implement CERTPHASH using Python 3.11 with Pytorch. All experiments are run on a single NVIDIA A100-PCIE-40GB GPU with CUDA Version 12.1 and Driver Version 530.30.02. During the certified PHash training and verification, we leverage α,β – CROWN along with its integrated libraries: auto_LiRPA for training and evasion verification, and INVPROP for collision verification. The default hyperparameters employed in our experiments are in Table 1. We detail our experimental setup in the following sections.

5.1 Baselines

We compare our model with the following baselines:

- PhotoDNA: A PHash algorithm by Microsoft that converts images into a 144-dimensional numeric vector for detecting unsafe content.
- PDQ: A PHash algorithm by Facebook that encodes images into a 256-bit binary vector.
- NeuralHash: A neural perceptual hashing model by Apple, producing a 96-bit binary vector, used for combating CSAM.

To enable equal evaluation of PhotoDNA and PDQ under a verifier that requires a neural network, we also compare two types of their neural network-based adaptations:

- BASEPHASH: A non-robust neural network trained using images and their output hashes with only $\mathcal{L} = \mathcal{L}_{\text{non-robust}}$.
- ADVPHASH: An empirical robust neural network trained with $\mathcal{L} = \mathcal{L}_{\text{evasion}} + \lambda\mathcal{L}_{\text{collision}}$, where δ is *empirically* calculated with adversarial attacks such as PGD, which is different from the verifiable worst case δ of CERTPHASH.

5.2 Metrics

We use the following metrics to evaluate the functionality and robustness of CERTPHASH and other baselines.

Match Threshold. The match threshold, denoted by Δ_y, is used to determine whether two output hashes match. Following prior work [5, 35, 41], we set Δ_y to 90 for PDQ, 1,800 for PhotoDNA, and 20 for NeuralHash. These thresholds are calculated using the l_1 norm.

Table 1: Default hyper-parameters for CERTPHASH during training and verification. The schedule shows the total number of epochs, and the epochs are divided into three phases: zero ε, an increase of ε from 0 to $\varepsilon_{\text{target}}$ (warm-up schedule), and finally $\varepsilon_{\text{target}}$.

Dataset	$\|\delta\|_p \leq \varepsilon$		Training Parameters								Verification Parameters
	l_p	$\varepsilon_{\text{target}}$	model	lr	lr decay factor	lr decay milestones	schedule epochs	n (Num $T(\cdot)$)	λ	μ	ε
COCO	l_∞	[1/255, 2/255, 4/255, 8/255]	ResNet	5e-4	0.2	[120, 140]	160 (2+80+78)	2	1.0	0.5	[1/255, 2/255, 4/255, 8/255]
	l_2	[0.05, 0.1, 0.15, 0.2]	ResNet	5e-4	0.2	[120, 140]	160 (2+80+78)	2	1.0	0.5	[0, 0.05, 0.1, 0.15, 0.2]
MNIST	l_∞	[1/255, 2/255, 4/255, 8/255]	CNN-4	5e-4	0.2	[50, 60]	70 (1+20+49)	2	1.0	1.0	[1/255, 2/255, 4/255, 8/255]
	l_2	[0.05, 0.1, 0.15, 0.2]	CNN-4	5e-4	0.2	[50, 60]	70 (1+20+49)	2	1.0	1.0	[0, 0.05, 0.1, 0.15, 0.2]
CelabA and NSFW-56K	l_∞	[1/255, 2/255, 4/255, 8/255]	ResNet	5e-4	0.2	[120, 140]	160 (2+80+78)	2	1.0	0.5	[1/255, 2/255, 4/255, 8/255]

ROC-AUC. We use ROC-AUC to evaluate the functionality of a PHash system in *non-adversarial* settings, focusing on its ability to correctly identify non-matching hashes for distinct images and matching hashes for the same images after benign transformations ($T(\cdot)$, including geometric (rotation, cropping, flipping), color (hue, brightness, contrast, saturation changes), and compression (JPEG) transformations.

Each image x_i from the dataset $\mathcal{X}$ is paired with its transformed version $T(x_i)$ and the transformed versions of other images $T(x_j)$, forming a set of pairs $\mathcal{S} = \{(x_0, T(x_0)), (x_0, T(x_1)), \ldots, (x_n, T(x_{n-1})), (x_n, T(x_n))\}$, organized as a set of pairs to eliminate duplicate pairs. *True Positives* are defined as the pairs $(x_i, T(x_i))$, and *True Negatives* as pairs $(x_i, T(x_j))$ where $x_i \neq x_j$. We calculate distances for each pair in $\mathcal{S}$, resulting in a distance set $\mathcal{M}$, and convert these distances into probabilities $\mathcal{P}$ using the formula $p_i = \frac{1}{1+e^{-k\cdot(\Delta_y - m_i)}}$, where $k = 0.01$ is the steepness constant, $m_i \in \mathcal{M}$, and Δ_y is the predefined matching threshold. A pair s_i is predicted as a match if p_i is larger than a threshold.

We then determine the True Positive Rate (TPR) and False Positive Rate (FPR) at various thresholds to compute the ROC-AUC. The TPR indicates the system's efficiency in identifying images that should match under benign transformations, reflecting evasion robustness, while the FPR shows how often non-matching images are incorrectly identified as matches, reflecting collision risk. The ROC-AUC metric integrates both collision and evasion into a comprehensive measure of the PHash model's functionality.

Certified No-evasion Rate. The evasion rate quantifies how often perturbed images, with perturbations $\|\delta\|_p \leq \varepsilon$, result in hash values that deviate from the original hash by more than the predefined threshold Δ_y. This is measured under an *adversarial* setting, where δ is carefully crafted to maximize deviation. The empirical evasion rate (ER) is determined using empirical attacks such as gradient-based methods. In contrast, the certified no-evasion rate (CNER) evaluates the worst-case scenarios using a verifier to determine the maximum δ that could cause prediction deviation. A *100% CNER* ensures no evasion with the perturbation constraints.

Certified No-collision Rate. The collision rate quantifies how frequently perturbed images, with perturbations $\|\delta\|_p \leq \varepsilon$, produce hash values that incorrectly match the hash of a different image, falling below the predefined match threshold Δ_y. This evaluation is in an *adversarial* setting, where δ is carefully optimized. The empirical collision rate (CR) is derived by selecting a set of target images and then optimizing δ for source images to make the perturbed source images' output hashes close to those of the target images. To calculate the certified no-collision rate (CNCR), we first consider a set of non-match output hashes $\{f_\theta(x_0), f_\theta(x_1), \ldots, f_\theta(x_n)\}$, and assign each input the respective output constraint $\{y \in Y \,|\, \|f_\theta(x_i) - y\|_1 \leq \Delta_y\}$. We then use a verifier to calculate the overapproximation of the corresponding preimage. Achieving a *100% CNCR* means there is no overlap in the constrained preimages for different hash outputs, ensuring no collisions.

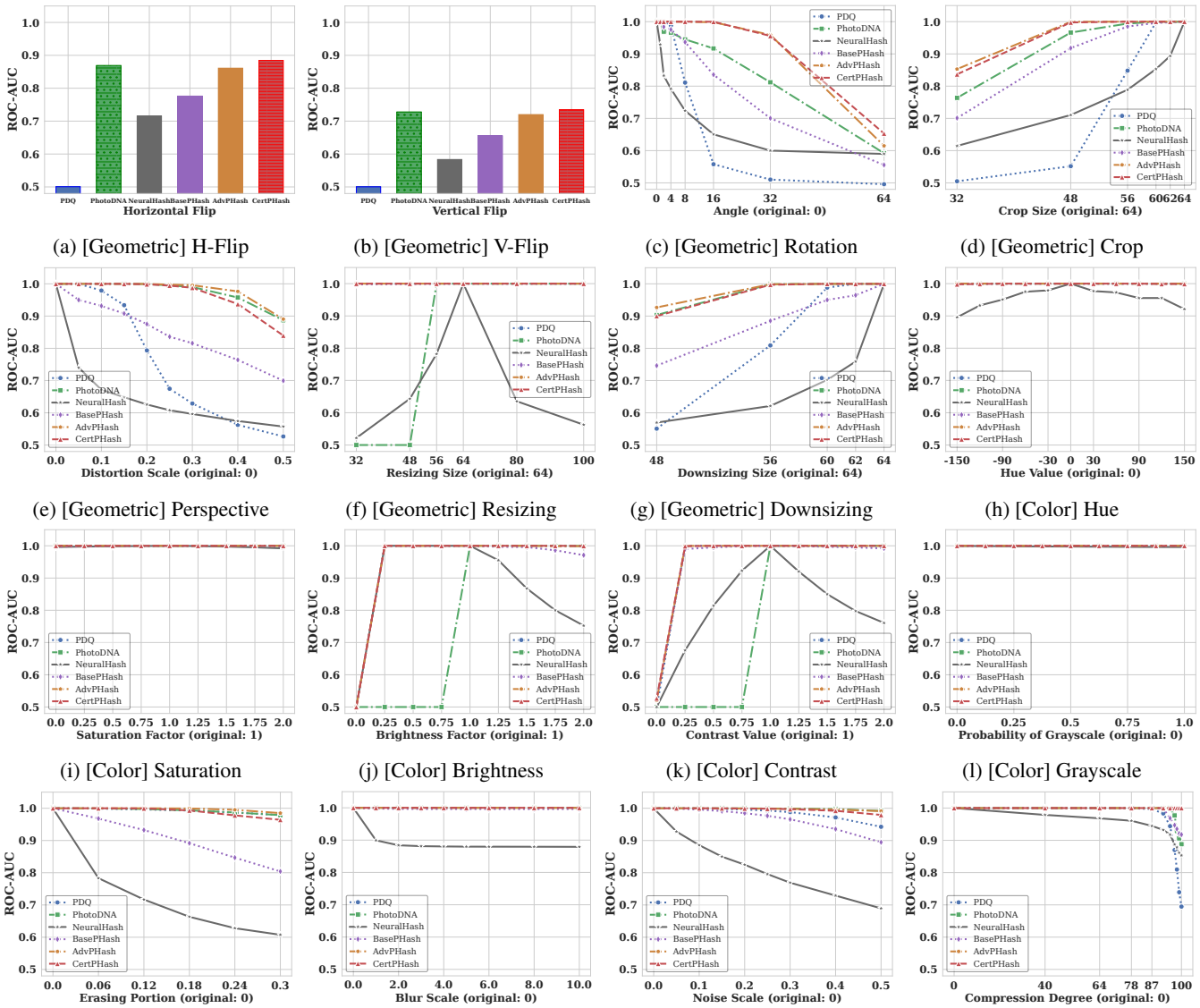
5.3 Datasets

We use four datasets to evaluate CERTPHASH: (1) COCO-2017 [27], resized to 64x64, with 60,000 training and 10,000 verification samples; (2) MNIST, grayscale hand-written digits (28x28), using the original split of 60,000 training and 10,000 verification samples; (3) CelebA [28], resized to 64x64, with randomly selected 50,000 training and 6,000 verification samples; (4) NSFW-56K [22, 26], resized to 64x64, with 50,000 explicit images for training and 6,000 for verification. Paired with CelebA (safe content), it assesses CERTPHASH's NSFW detection capabilities.

6 Experiment

We answer the following research questions (RQs):

- [RQ1] Can CERTPHASH maintain the hash functionality?
- [RQ2] Can CERTPHASH achieves certified robustness under different constraints?
- [RQ3] How robust is CERTPHASH against empirical evasion and collision attacks?
- [RQ4] How does CERTPHASH perform in a real-world application (NSFW content detection)?
- [RQ5] How do different parameters affect the performance of CERTPHASH?
- [RQ6] What are the overhead and scalability of CERTPHASH?



(a) [Geometric] H-Flip (b) [Geometric] V-Flip (c) [Geometric] Rotation (d) [Geometric] Crop

(e) [Geometric] Perspective (f) [Geometric] Resizing (g) [Geometric] Downsizing (h) [Color] Hue

(i) [Color] Saturation (j) [Color] Brightness (k) [Color] Contrast (l) [Color] Grayscale

(m) [Perturbation] Random Erasing (n) [Perturbation] Gaussian Blur (o) [Perturbation] Pepper & Salt (p) [Compression] JPEG

Figure 2: [RQ1] PHash functionality measured by ROC-AUC under different benign transformations.

6.1 Functionality Analysis

In this research question, we use the COCO dataset to compare the ROC-AUC scores of various PHash systems across nine benign transformations. These transformations fall into four categories: geometric changes (rotation, cropping, random perspective changes, resizing, downsizing, horizontal flip, vertical flip), color changes (hue, saturation, brightness, contrast, grayscale), perturbations (random erasing, gaussian blur, pepper & salt noise), and compression (JPEG). We chose the COCO dataset for its diverse range of human subjects and everyday objects, providing a realistic basis for PHash functionality tests. We evaluate the performance of CertPHash against PhotoDNA, PDQ, NeuralHash, and our own trained versions, BasePHash and AdvPHash, all trained with $\varepsilon_{\text{target}} = 1/255$. The impact of different $\varepsilon_{\text{target}}$ values on system functionality is discussed further in Section 6.5.

Figure 2 presents the functionality results. Overall, CertPHash achieves a high average ROC-AUC of 0.98 across 16 transformations, outperforming PDQ (0.92), PhotoDNA (0.93), and NeuralHash (0.84). Specifically, PDQ struggles with geometric transformations like rotation and cropping; PhotoDNA is highly sensitive to resizing and color changes, including brightness and contrast adjustments. NeuralHash, in particular, performs poorly under high-level transformations. Among the various transformation types, geometric transformations pose the greatest challenge to these systems.

We now delve deeper into the functionality performance of CertPHash across four distinct transformation categories:

Geometric. CERTPHASH and ADVPHASH consistently outperform all competitors. For rotations, CERTPHASH achieves a ROC-AUC above 0.95 for angles up to 32 degrees and demonstrates the best performance at near 64 degrees. In cropping, CERTPHASH scores over 0.99 for crops larger than 48 (original size: 64) and maintains a score of 0.84 at size 32. For horizontal flips, CERTPHASH improves performance by 1% to 40%, and 2% to 29% for vertical flips. For random perspective changes with scales ranging from 0.1 to 0.5, ADVPHASH performs the best. CERTPHASH matches ADVPHASH up to a scale of 0.3 but shows slightly poorer performance (5%) at a scale of 0.5, though it still outperforms other systems by 15% to 32%. For resizing and downsizing, both CERTPHASH and ADVPHASH consistently achieve ROC-AUC scores greater than 90%.

Color. CERTPHASH, ADVPHASH, BASEPHASH, and PDQ maintain ROC-AUC scores close to 100% across most color transformations, except for extreme cases where brightness and contrast are set to 0, resulting in completely black and gray images, respectively. PhotoDNA and NeuralHash, however, are sensitive to larger color transformation values.

Perturbation. CERTPHASH, ADVPHASH, and PhotoDNA maintain ROC-AUC scores close to 100% under perturbations, while NeuralHash exhibits significant vulnerability, with performance drops ranging from 10% to 40%.

Compression. CERTPHASH and ADVPHASH outperform all competitors, particularly at extremely high compression rates (greater than 92), with performance improvements ranging from 8% to 30%.

Notably, ADVPHASH and CERTPHASH outperform other systems across most transformations due to two factors: (i) adding random transformations $T(\cdot)$ during training, as described in Section 4.2, greatly enhances the robustness to many real-world benign transformations, and (ii) applying anti-evasion and anti-collision loss functions (Equations 7 and 8), ensuring consistent outputs for identical images and distinct outputs for different ones. In contrast, BASEPHASH underperforms due to its non-robust standard training without anti-collision or anti-evasion mechanisms. While both ADVPHASH and CERTPHASH improve robustness to benign transformations, Section 6.2 shows ADVPHASH is less robust against adversarial perturbations.

6.2 Certified Robustness Analysis

In this research question, we evaluate CERTPHASH's certified robustness under commonly adopted l_∞ and l_2 perturbation settings with varying levels of perturbations. We use the COCO dataset with a model trained with PhotoDNA-generated hashes and the MNIST dataset with a model trained with PDQ-generated hashes. The results demonstrate that CERTPHASH consistently achieves up to a 100% certified no-evasion rate (CNER) and 95% no-collision rate (CNCR),

Table 2: [RQ2-1] Certified No-evasion Rate (**CNER**) on COCO and MNIST under different perturbations.

	Norm	Method + Training $\varepsilon_{\text{target}}$		CNER↑ under Different Verification ε			
COCO	l_∞			$\varepsilon = 1/255$	$\varepsilon = 2/255$	$\varepsilon = 4/255$	$\varepsilon = 8/255$
		BASEPHASH	non-robust	97.40%	57.28%	0.00%	0.00%
		ADVPHASH	8/255	99.96%	97.44%	40.66%	0.01%
		CERTPHASH	1/255	100.00%	100.00%	93.93%	79.51%
			2/255	100.00%	100.00%	100.00%	96.14%
			8/255	100.00%	100.00%	100.00%	100.00%
	l_2			$\varepsilon = 0.05$	$\varepsilon = 0.1$	$\varepsilon = 0.15$	$\varepsilon = 0.2$
		BASEPHASH	non-robust	96.59%	26.37%	0.10%	0.00%
		ADVPHASH	0.15	98.85%	95.43%	52.64%	0.00%
		CERTPHASH	0.05	100.00%	100.00%	99.72%	98.46%
			0.1	100.00%	100.00%	99.98%	99.50%
			0.15	100.00%	100.00%	100.00%	100.00%
MNIST	l_∞			$\varepsilon = 1/255$	$\varepsilon = 2/255$	$\varepsilon = 4/255$	$\varepsilon = 8/255$
		BASEPHASH	non-robust	3.66%	2.43%	0.00%	0.00%
		ADVPHASH	8/255	100.00%	99.83%	0.18%	0.00%
		CERTPHASH	1/255	100.00%	100.00%	85.75%	0.00%
			2/255	100.00%	100.00%	91.68%	0.00%
			8/255	100.00%	100.00%	100.00%	99.90%
	l_2			$\varepsilon = 0.05$	$\varepsilon = 0.1$	$\varepsilon = 0.15$	$\varepsilon = 0.2$
		BASEPHASH	non-robust	0.00%	0.00%	0.00%	0.00%
		ADVPHASH	0.15	0.00%	0.00%	0.00%	0.00%
		CERTPHASH	0.05	100.00%	0.00%	0.00%	0.00%
			0.1	100.00%	100.00%	100.00%	83.57%
			0.15	100.00%	100.00%	100.00%	100.00%

↑ indicates larger is better.

highlighting its robustness against potential adversarial challenges.

Certified No-evasion Rate. Table 2 compares the Certified No Evasion Rates (CNER) of CERTPHASH with non-robust training BASEPHASH and adversarial training ADVPHASH. First, CERTPHASH achieves consistently high CNER ($\geq$99.9%) across all datasets. In contrast, for the COCO dataset, BASEPHASH performs poorly with a 0% CNER at $\varepsilon = 4/255$ for l_∞ and $\varepsilon = 0.2$ for l_2. ADVPHASH shows a near-zero CNER at $\varepsilon = 8/255$ for l_∞ and 0% at $\varepsilon = 0.15$ for l_2. For MNIST, BASEPHASH shows a CNER of 3.66% at $\varepsilon = 1/255$ for l_∞, while ADVPHASH records 0.18% at $\varepsilon = 4/255$. Second, we observe that COCO data using PhotoDNA is more robust than MNIST with PDQ. This is due to COCO's complex structure and deeper model, which provides a stronger foundation compared to MNIST's simpler data and shallower architecture. Additionally, COCO's 144 numeric output from PhotoDNA is less prone to modifications than MNIST's 256-bit binary output with PDQ. We previously noted that PhotoDNA's functionality also tends to be more effective than PDQ, likely resulting in better training labels for COCO.

Certified No-collision Rate. Given the computational complexity involved in deriving inverse input bounds from output constraints, this experiment was conducted on MNIST (PDQ) using a four-layer CNN under the l_∞ norm (α,β-CROWN cannot scale to large models for this setting). For all images in the test set, we compute their hashes. We then compute the provable overapproximation of the preimages of all similar

Table 3: [RQ2-2] Certified No-collision Rate (**CNCR**) on MNIST under different output constraint threshold δ_y.

	Norm	Method + Training $\varepsilon_{\text{target}}$		CNCR↑ under Different δ_y		
				$\delta_y = 1$	$\delta_y = 20$	$\delta_y = 90$
MNIST	l_∞	BASEPHASH	non-robust	0.00%	0.00%	0.00%
		ADVPHASH	8/255	0.00%	0.00%	0.00%
		CERTPHASH	8/255	98.25%	95.72%	95.41%

↑ indicates larger is better.

hashes ($\{y \in Y | \|f_\theta(x_i) - y\|_1 \leq \Delta_y\}$ for each original image x_i) To ensure that tight bounds on the preimage can be found, we limit the input domain to $x_i \pm 96/255$, which covers almost half of the entire pixel space. We consider images to have potential collisions if two of those preimages overlap.

Table 3 presents the certified no-collision rates (CNCR) for BASEPHASH, ADVPHASH, and CERTPHASH, under different output constraints. BASEPHASH and ADVPHASH encounter a 0% CNCR for all cases due to the bounds being too loose. For CERTPHASH, the CNCR decreases from 98.25% to 95.41% until the output constraint relaxes to $\delta_y = 90$, i.e., the match threshold for PDQ.

6.3 Empirical Robustness Analysis

In this research question, we assess the empirical robustness of CERTPHASH in scenarios without input constraints, where an adversary may employ different perturbation norms without a predefined level. We follow Struppek et al. [41] to implement the empirical attacks with PGD [29]. Specifically, We intentionally allow adversaries to use a different perturbation norm, specifically the l_2 norm, on CERTPHASH trained with l_∞ perturbation. Adversaries increase the l_2 perturbation until the non-robust BASEPHASH achieves nearly 100% evasion and collision rates with minimal effort, capped at 100 steps. Note that for MNIST, we pick the initial image pairs from separate classes, as images from one class tend to be very similar and are thus very easy to collide for BASEPHASH. We then establish this l_2 norm level as the maximum for both ADVPHASH and CERTPHASH. We report the evasion rate (ER), collision rate (CR), average steps to attack, and the l_2 perturbation norm during the attack. Additionally, we compute the l_∞ norm between x and $x + \delta$ post-attack to show the perturbation in l_∞ norm. Our experiments show that CERTPHASH maintains a zero evasion rate under empirical attacks while reducing the collision rate by 80% compared with the baselines.

Empirical Evasion Attack. Table 4 shows the evaluation results of CERTPHASH compared with BASEPHASH and ADVPHASH. Overall, CERTPHASH achieves nearly zero evasion rates (ER) with training $\varepsilon_{\text{target}} = 8/255$. In contrast, ADVPHASH still exhibits 68.8% ER for COCO and 19.6% for MNIST under the same training perturbations. Our observations include: First, the perturbation level required by empirical attacks exceeds that calculated for certified robustness, indicating the reliability of our certified ER assessments. Second, although ADVPHASH is effective in defending against empirical attacks, it does not guarantee performance, as evidenced by higher empirical ER compared to certified ER. Third, CERTPHASH requires 2 to 5 times more steps to be attacked than BASEPHASH, highlighting its robustness under the scenario even without input constraints.

Table 4: [RQ3-1] Empirical gradient-based evasion attacks use different attacking (l_2) versus training (l_∞) perturbations. We select attack perturbation levels (l_2) where BASEPHASH approaches a near 100% evasion rate (**ER**) and calculate the perturbation level using the l_∞ norm after the attack. We also report the average steps required for successful evasion.

	Method + Training $\varepsilon_{\text{target}}$		Evasion Evaluation Metrics			
			ER↓	l_2	l_∞	Steps
COCO	BASEPHASH	non-robust	89.24%	3.713	0.033	15.1
	ADVPHASH	8/255	68.80%	3.702	0.037	17.8
	CERTPHASH	1/255	54.57%	3.701	0.038	16.2
		4/255	18.69%	3.702	0.035	19.6
		8/255	1.01%	3.713	0.035	32.5
MNIST	BASEPHASH	non-robust	99.12%	2.256	0.098	17.2
	ADVPHASH	8/255	19.60%	2.256	0.109	63.1
	CERTPHASH	1/255	1.69%	2.426	0.115	88.5
		2/255	0.00%	—	—	—
		8/255	0.00%	—	—	—

↓ indicates lower is better.

Empirical Collision Attack. Table 5 presents the collision evaluations for BASEPHASH, ADVPHASH, and CERTPHASH. CERTPHASH effectively reduces the collision rate (CR) to approximately 40% with a minimal training $\varepsilon_{\text{target}} = 1/255$, and further to about 20% with $\varepsilon_{\text{target}} = 8/255$. For COCO, ADVPHASH matches CERTPHASH's performance, but it is less effective for MNIST, recording a 56.99% CR. We observe that collisions are more challenging than evasions in COCO due to the dataset's diversity, which requires more perturbation to match hashes between different images. Figure 3 illustrates a collision example in the COCO dataset. Following the methodology in Struppek et al. [41], we initially select the target image x_j from the testing set that has the smallest output distance to the original image x_i. We then optimize δ according to objective 3.2 using a PGD algorithm [29]. Conversely, collisions happen more easily in MNIST due to the similar appearance of images, their single-channel format, and PDQ's simpler output structure.

6.4 Real-world Application: NSFW Detection

To assess CERTPHASH's effectiveness in detecting illicit content, we trained it on 50,000 open-sourced NSFW (Not-Safe-For-Work) images from NSFW-56k (explicit and unsafe content) and 50,000 safe images from CelebA, using hash labels from PhotoDNA. Our evaluation covers functionality under benign transformations, evasion robustness, and collision robustness under adversarial perturbations.

Functionality for NSFW. Table 6 presents the ROC-AUC scores across 9 transformations, with mean and standard de-

Table 5: [RQ3-2] Empirical gradient-based collision attack use different attacking (l_2) versus training (l_∞) perturbations. We select attack perturbation levels (l_2) where BASEPHASH approaches a near 100% collision rate (**CR**) and calculate the perturbation level using the l_∞ norm after the attack. We also report the average steps required for a successful collision.

PHash	Method with $\varepsilon_{\text{target}}$		Collision Evaluation Metrics			
			CR↓	l_2	l_∞	Steps
COCO	BASEPHASH	non-robust	89.75%	10.853	0.099	21.9
	ADVPHASH	8/255	21.73%	10.329	0.106	25.1
	CERTPHASH	1/255	37.48%	10.841	0.109	24.5
		2/255	27.36%	10.616	0.108	22.8
		8/255	18.69%	10.742	0.100	24.9
MNIST	BASEPHASH	non-robust	100.00%	0.727	0.053	7.2
	ADVPHASH	8/255	56.99%	0.710	0.056	12.9
	CERTPHASH	1/255	42.56%	0.727	0.051	13.5
		4/255	42.06%	0.725	0.059	13.4
		8/255	21.36%	0.726	0.051	15.7

↓ indicates lower is better.

(a) Target x_j (b) Original x_i (c) δ (d) $x_i + \delta$

Figure 3: Target and second preimages in collision attacks for COCO dataset. Figure (a) shows the target preimage, while figure (d) shows the second preimage that causes a collision, i.e., producing hash outputs matching the hash of the target preimage, with figure (b) adding perturbation as shown in figure (c).

viation calculated for different transformation levels. The test set included 12,000 images—6,000 NSFW and 6,000 safe. We compare CERTPHASH with NerualHash, which is claimed for illicit content detection. The results indicate that CERTPHASH maintains strong functionality in NSFW detection, achieving an average ROC-AUC of 0.914 across all transformations with a training $\varepsilon_{\text{target}} = 1/255$, and 0.911 with $\varepsilon_{\text{target}} = 8/255$, both outperform NeuralHash, which achieves an average 0.802 ROC-AUC.

Evasion Robustness for NSFW. Evasion robustness assesses whether NSFW content can still be detected under perturbations. We focus on the 6,000 NSFW images in the test set and evaluate the certified no-evasion rate (CNER). Table 7 presents the CNER results for the NSFW dataset. CERTPHASH achieves 100% CNER with a training $\varepsilon_{\text{target}} = 8/255$, indicating that it is certified robust to evasion under this perturbation level.

Collision Robustness for NSFW. Collision robustness evaluates whether perturbed safe images from the 6,000-image test set of CelebA produce hashes that match one of those of 6,000 NSFW images, potentially causing false alarms. To assess this, we employ empirical gradient-based evasion attacks [41]. First, we calculate the target hashes for the 6,000 NSFW images. For each safe image, we identify the closest NSFW hashes and then optimize the perturbation on the safe image to make its hash output close to the NSFW one. Table 8 presents the collision rate (CR) under varying levels of attacking perturbation. We observe that CERTPHASH effectively tolerates perturbations up to 32 times its training range. For example, when trained with a perturbation of $\varepsilon_{\text{target}} = 1/255$, CERTPHASH achieves a CR of 2.15% for an attack perturbation of $\varepsilon = 16/255$. Even with a larger perturbation of $32/255$, CERTPHASH maintains a CR of 44.28%. Increasing $\varepsilon_{\text{target}}$ to 8/255 further lowers the CR to 21.28%.

6.5 Ablation Study

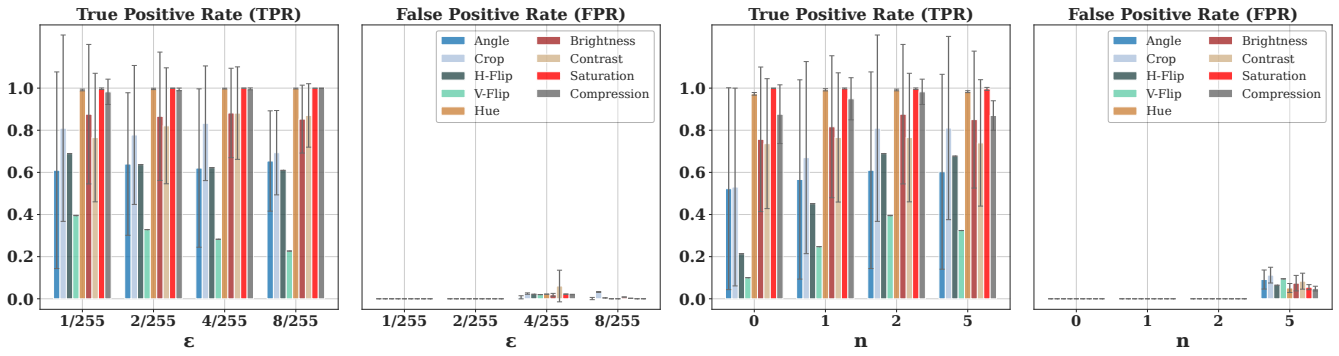
In this section, we explore the impact of various hyperparameters, including $\varepsilon_{\text{target}}$, n, $\mathcal{L}_{\text{collision}}$, and $\mathcal{L}_{\text{func}}$, on CERTPHASH, with a focus on functionality under different transformations, since robustness is more closely tied to $\varepsilon_{\text{target}}$ as discussed in the previous section. Note that CERTPHASH does not require extensive hyperparameter tuning, as a small set of initial settings proved effective in various scenarios. We focus on the True Positive Rate (TPR) and False Positive Rate (FPR) using the threshold that achieves the best ROC-AUC. This approach allows us to separately assess the effects of evasion (reflected by TPR) and collision (reflected by FPR). For each hyperparameter, we select four values, reporting the mean and standard deviation of TPR and FPR across the nine benign transformations discussed in Section 6.1. We conduct these evaluations on the COCO dataset. Figure 4 presents the results, which we will discuss for each hyperparameter below.

Perturbation level $\varepsilon_{\text{target}}$. We test on the $\varepsilon = [1/255, 2/255, 4/255, 8/255]$. Overall, the mean TPR values suggest that the perturbation level $\varepsilon_{\text{target}}$ has little impact on evasion-related functionality, with only a minor decrease in V-flip performance at higher $\varepsilon_{\text{target}}$ levels. As $\varepsilon_{\text{target}}$ increases, the standard deviation of TPR decreases, indicating more stable performance. FPR remains nearly zero across most conditions, but a slight increase is observed at higher perturbation levels (4 and 8) due to the additional noise introduced during training.

Number of transformations n. We evaluate the impact of $n = [0, 1, 2, 5]$ on the model's functionality. Generally, an appropriate n can enhance TPR, indicating reduced evasion across different transformations. The TPR is low at $n = 0$, especially for geometric transformations, but improves as n increases to 2. However, the TPR declines at $n = 5$, likely because more transformations require additional training time to converge. Meanwhile, the FPR remains at zero for $n = 0, 1$, and 2, but rises to between 4% and 10% at $n = 5$. This suggests that too many transformations during training may cause outputs to become too similar, potentially requiring a larger λ to enhance anti-collision robustness.

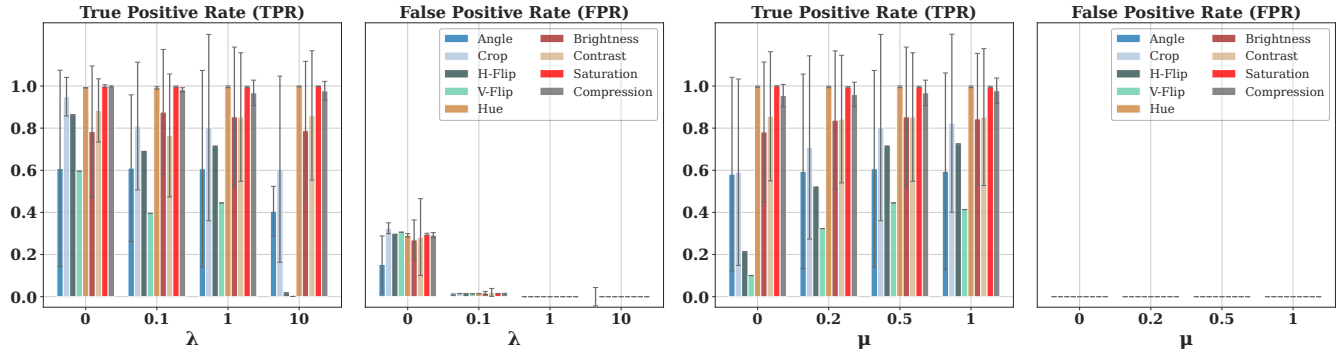
Table 6: [RQ4-1] NSFW detection functionality by ROC-AUC score for CERTPHASH and NeuralHash, with mean and standard deviation calculated for different transformation levels.

Method (Training $\varepsilon_{\text{target}}$)	Geometric				Color				Compression
	Angle	Crop	H-Flip	V-Flip	Hue	Brightness	Contrast	Saturation	JPEC
NeuralHash	0.745 ± 0.143	0.781 ± 0.140	0.599 ± 0.000	0.600 ± 0.000	0.931 ± 0.028	0.865 ± 0.161	0.801 ± 0.119	0.996 ± 0.003	0.902 ± 0.005
CERTPHASH ($\varepsilon = 1/255$)	0.942 ± 0.125	0.909 ± 0.156	0.758 ± 0.000	0.721 ± 0.000	0.999 ± 0.001	0.945 ± 0.121	0.953 ± 0.105	1.000 ± 0.000	0.999 ± 0.001
CERTPHASH ($\varepsilon = 8/255$)	0.928 ± 0.124	0.906 ± 0.152	0.759 ± 0.000	0.716 ± 0.000	0.999 ± 0.001	0.944 ± 0.157	0.948 ± 0.143	0.999 ± 0.001	1.000 ± 0.000



(a) Magnitude of the training perturbation $\varepsilon_{\text{target}}$

(b) Number of transformations n in training



(c) Scaling factor λ for $\mathcal{L}_{\text{collision}}$

(d) Scaling factor μ for $\mathcal{L}_{\text{func}}$

Figure 4: [RQ5] Ablation Study on hyperparameters ε, n, λ, and μ. We show the mean and standard deviation of true positive rate (TPR) and false positive rate (FPR) for each transformation at various transformation levels.

Table 7: [RQ4-2] Certified *no* evasion rate (**CNER**) of CERTPHASH for NSFW Detection.

Method (Training $\varepsilon_{\text{target}}$)	CNER↑ under different verification ε			
	$\varepsilon = 1/255$	$\varepsilon = 2/255$	$\varepsilon = 4/255$	$\varepsilon = 8/255$
CERTPHASH (1/255)	100.00%	100.00%	100.00%	98.25%
CERTPHASH (8/255)	100.00%	100.00%	100.00%	100.00%

↑ indicates larger is better.

Table 8: [RQ4-3] Collision Rate (**CR**) by empirical gradient-based collision attack under varying levels of attacking perturbation for NSFW Detection.

Method (Training $\varepsilon_{\text{target}}$)	CR↓ under different attacking ε			
	$\varepsilon = 4/255$	$\varepsilon = 8/255$	$\varepsilon = 16/255$	$\varepsilon = 32/255$
CERTPHASH (1/255)	0.00%	0.04%	2.15%	44.28%
CERTPHASH (8/255)	0.00%	0.00%	0.04%	21.28%

↓ indicates lower is better.

Balance Term λ for $\mathcal{L}_{\text{collision}}$. We choose the values as $\lambda = [0, 0.1, 1, 10]$ for evaluation. Overall, we observe that $\mathcal{L}_{collision}$ effectively reduces FPR, lowering collision risks. At $\lambda = 0$, the FPR is high due to the absence of collision loss in the robust loss calculations, leading to more collisions. Increasing λ to 0.1 or higher reduces FPR. The TPR remains consistent across $\lambda = 0, 0.1, 1$, but drops slightly at $\lambda = 10$, especially for vertical flip transformations. For our experiments, we used $\lambda = 1$ as the default for the collision term, which balances the trade-off between evasion and collision, resulting in a high TPR and low FPR.

Balance Term μ for $\mathcal{L}_{\text{func}}$. We explore values of $\mu = [0, 0.2, 0.5, 1]$. Overall, $\mathcal{L}func$ effectively enhances functionality, particularly the TPR under various transformations. We observe that the TPR for geometric transformations is low when $\mu = 0$, and it increases as μ rises, with $\mu = 0.5$ showing

Table 9: [RQ6-1] Computational overhead on different datasets using GPU memory usage and training time as metrics.

Dataset	Training time per epoch			Total training time			GPU memory usage		
	BASEPHASH	ADVPHASH	CERTPHASH	BASEPHASH	ADVPHASH	CERTPHASH	BASEPHASH	ADVPHASH	CERTPHASH
MNIST	9 s	31 s	30 s	181 s	2,141 s	2,093 s	2,066 MB	2,068 MB	2,364 MB
COCO	105 s	146 s	133 s	4,214 s	23,406 s	21,249 s	2,080 MB	2,082 MB	2,604 MB
CelabA + NSFW-56K	279 s	328 s	304 s	11,147 s	52,539 s	48,620 s	2,080 MB	2,082 MB	2,604 MB

Table 10: [RQ6-2] Scalability when scaling the model up from NeuralHash on COCO datasets.

Model	Parameter size	Training time per epoch		
		BASEPHASH	ADVPHASH	CERTPHASH
ResNet	3M	105 s	146 s	133 s
ResNext	20M	153 s	399 s	360 s
WideResNet	100M	185 s	870 s	566 s

significant improvement—e.g., crop TPR is 20% higher, and horizontal flip TPR is 50% higher compared to $\mu = 0$. The performance at $\mu = 1$ is similar to that at $\mu = 0.5$, so we chose $\mu = 0.5$ as our default setting. The FPR remains at zero across all values, indicating less impact on collision risk.

6.6 Overhead and Scalability

Overhead. Table 9 compares training time and GPU memory usage for CERTPHASH, ADVPHASH (adversarial training), and BASEPHASH (non-robust training) across different datasets. For training time, both CERTPHASH and ADVPHASH introduce overhead compared to BASEPHASH, as both involve computing perturbations–verifiable for CERTPHASH and empirical for ADVPHASH–and training on perturbed data. However, CERTPHASH requires, on average, 0.92x the training time of ADVPHASH for both per-epoch and total training. GPU memory usage for ADVPHASH and BASEPHASH is similar, while CERTPHASH requires 1.24x to 1.25x more due to its interaction with the verifier to calculate worst-case perturbations.

Scalability. Table 10 analyzes the scalability of CERTPHASH compared with BASEPHASH and ADVPHASH. We evaluated models ranging from ResNet with 3M parameters to WideResNet with 100M parameters. On average, CERTPHASH requires 2.3x the training time per epoch compared to non-robust BASEPHASH, and 0.8x compared to ADVPHASH. For CERTPHASH, the training time increases as the model scales up to 20M parameters but maintains a similar level for models with 100M parameters.

Note that models with 20M and 100M parameters are likely too large for real-world deployment, as PHash models are typically small and designed for mobile devices like iPhones and MacBooks (e.g., Apple's NeuralHash, which has 2M parameters). We believe the 3M ResNet model used in our paper offers a practical balance between robustness and scalability.

7 Discussion and Limitation

Computational Overhead. Although once trained, our robust model CERTPHASH benefits users at no extra cost compared to the standard model during hash generation and matching, we acknowledge that robust training introduces computational overhead during training time. This is a common challenge when making models robust to adversarial examples: For instance, adversarial training (i.e., our baseline called ADVPHASH) also incurs such computational overhead as shown in Section 6.6. Notably, CERTPHASH is faster than ADVPHASH in terms of training time. Future work could investigate verification-friendly neural networks, such as pruning unstable neurons, to minimize computational overhead.

Diversity of Tested Transformations. In addition to evaluating CERTPHASH against gradient-based adversarial examples, we adopted a set of transformations commonly used in prior works [35,41] to test PHash's functionality under natural manipulations. These transformations span color alterations, geometric changes, size changes, compression, Gaussian blur, and random erasing across wide levels, aiming to cover many real-world scenarios. However, we acknowledge that the predefined set of transformations is inherently limited and cannot fully represent the unlimited real-world variations. This is a common challenge in robust hashing research, as predefined transformations are necessary for comparable results. In future work, we will explore methods to further generalize our approach to cover a wider range of transformations.

8 Conclusion

In this paper, we propose the *first* certified robust PHash system, CERTPHASH, designed to offer provable defenses against evasion and collision attacks in digital content detection. CERTPHASH incorporates a robust training framework that includes multiple novel terms, achieving robustness through anti-evasion and anti-collision terms, tailored to manage worst-case scenario perturbations solved by a verifier. Additionally, CERTPHASH maintains functionality through data transformations and carefully balanced loss terms. Furthermore, CERTPHASH sets a benchmark for robustness verification in PHash systems, featuring metrics such as certified evasion and collision rates. Extensive evaluations show that CERTPHASH outperforms existing PHash systems in robustness without compromising functionality. It also performs effectively in real-world applications, such as detecting NSFW content, showcasing its practical utility. Overall, our work aims to provide a strong foundation for bridging the gap between theoretical certified robustness machine learning and its practical applications in digital content filtering, and to encourage further research on certified defense mechanisms.

9 Ethical Considerations

We strictly comply with ethical guidelines and policies in conducting our research. No human subjects and personally identifiable information were involved in our research.

The primary purpose of our study is to provide certified robust solutions to defend against evasion and collision attacks in PHash systems. To evaluate the effectiveness of our approach, we adopted previously known attacks [41], which are also publicly available, to demonstrate the vulnerabilities of existing PHash systems. Our results show that our approach effectively defends against these attacks.

Given that PHash systems are deployed in the real world for detecting CSAM, we utilized public NSFW datasets [22] to simulate the evaluation of our approach's effectiveness in detecting illicit content. It is important to note that our study does NOT contain any CSAM content. To mitigate potential harm from NSFW content to the audience, we have not included any explicit content in our paper. Additionally, we will not distribute any portion of the NSFW datasets, nor will we make the models trained on NSFW content publicly available. In alignment with open science policies, we will responsibly make the NSFW detection models available upon request for research purposes only. For models trained on non-sensitive data such as COCO, we will release them publicly.

To protect the well-being of researchers involved in handling NSFW datasets, our study does not require any manual review of the content; all analyses are conducted using automatic metrics such as ROC-AUC to avoid exposing researchers to explicit material. Furthermore, we provided support for the research team by conducting weekly individual and group check-ins and offering access to therapists if needed.

10 Open Science

All datasets and PHash systems used in this study are from publicly available sources. We also strictly comply with the open science policy by making our implementation and model available following ethical standards. Our artifact is available at https://zenodo.org/records/14740844 or https://github.com/Yuchen413/CertPhash.

Acknowledgment

This work was supported in part by National Science Foundation (NSF) under grants OAC-23-19742 and Johns Hopkins University Institute for Assured Autonomy (IAA) with grants 80052272 and 80052273. Huan Zhang is supported in part by the AI2050 program at Schmidt Sciences (AI2050 Early Career Fellowship) and NSF (IIS-2331967). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of NSF or JHU-IAA.

References

[1] Apple Inc. CSAM Detection- Technical Summary. https://www.apple.com/child-safety/pdf/CSAM_Detection_Technical_Summary.pdf, 2021. Accessed: 2021-09-22.

[2] AsuharietYgvar. Appleneuralhash2onnx, 2022. Accessed: 2024-08-13.

[3] Stanley Bak. nnenum: Verification of relu neural networks with optimized abstraction refinement. In *NASA Formal Methods Symposium*, pages 19–36. Springer, 2021.

[4] Mislav Balunovic and Martin Vechev. Adversarial training and provable defenses: Bridging the gap. In *International Conference on Learning Representations*, 2020.

[5] Jagdeep Singh Bhatia and Kevin Meng. Exploiting and defending against the approximate linearity of apple's neuralhash. *arXiv preprint arXiv:2207.14258*, 2022.

[6] Christopher Brix, Mark Niklas Müller, Stanley Bak, Taylor T. Johnson, and Changliu Liu. First three years of the international verification of neural networks competition (vnn-comp). *arXiv preprint arXiv:2301.05815*, 2023.

[7] Jianbo Chen, Michael I. Jordan, and Martin J. Wainwright. Hopskipjumpattack: A query-efficient decision-based attack, 2020.

[8] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *international conference on machine learning*, pages 1310–1320. PMLR, 2019.

[9] Jeremy M Cohen, Elan Rosenfeld, and Zico Kolter Kolter. Certified adversarial robustness via randomized smoothing. *ICML*, 2019.

[10] Facebook. Open-sourcing photo- and video-matching technology to make the internet safer. https://about.fb.com/news/2019/08/open-source-photo-video-matching, 2019. Accessed: 2021-10-18.

[11] FaustoMorales. pdqhash-python, 2021. Accessed: 2024-08-13.

[12] Claudio Ferrari, Mark Niklas Mueller, Nikola Jovanović, and Martin Vechev. Complete verification via multi-neuron relaxation guided branch-and-bound. In *International Conference on Learning Representations*, 2022.

[13] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2015.

[14] Sven Gowal, Krishnamurthy Dvijotham, Rob Stanforth, Rudy Bunel, Charles Qin, Jonathan Uesato, Timothy Mann, and Pushmeet Kohli. On the effectiveness of interval bound propagation for training verifiably robust models. *arXiv preprint arXiv:1810.12715*, 2018.

[15] Qingying Hao, Licheng Luo, Steve TK Jan, and Gang Wang. It's not what it looks like: Manipulating perceptual hashing based applications. *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2021.

[16] P. Henriksen and Alessio Lomuscio. Efficient neural network verification via adaptive refinement and adversarial search. In *European Conference on Artificial Intelligence*, 2020.

[17] Patrick Henriksen and Alessio Lomuscio. Deepsplit: An efficient splitting method for neural network verification via indirect effect analysis. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 2549–2555. International Joint Conferences on Artificial Intelligence Organization, 8 2021. Main Track.

[18] Shubham Jain, Ana-Maria Cretu, and Yves-Alexandre de Montjoye. Adversarial detection avoidance attacks: Evaluating the robustness of perceptual hashing-based client-side scanning. CoRR, 2021. arXiv:2106.09820.

[19] JanKais3r. jphotodna, 2021. Accessed: 2024-08-13.

[20] Guy Katz, Derek A. Huang, Duligur Ibeling, Kyle Julian, Christopher Lazarus, Rachel Lim, Parth Shah, Shantanu Thakoor, Haoze Wu, Aleksandar Zeljić, David L. Dill, Mykel J. Kochenderfer, and Clark Barrett. The marabou framework for verification and analysis of deep neural networks. In Isil Dillig and Serdar Tasiran, editors, *Computer Aided Verification*, pages 443–452, Cham, 2019. Springer International Publishing.

[21] Yannic Kilcher. Neural hash collision creator. https://github.com/yk/neural_hash_collision, 2021. Accessed: 2021-10-12.

[22] Alex Kim. Nsfw image dataset. https://github.com/alex000kim/nsfw_data_scraper, 2022.

[23] Suhas Kotha, Christopher Brix, J. Zico Kolter, Krishnamurthy Dvijotham, and Huan Zhang. Provably bounding neural network preimages. In A. Oh, T. Neumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 80270–80290. Curran Associates, Inc., 2023.

[24] Mathias Lecuyer, Vasilis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. *2019 IEEE Symposium on Security and Privacy (SP)*, pages 656–672, 2019.

[25] Bai Li, Changyou Chen, Wenlin Wang, and Lawrence Carin. Certified adversarial robustness with additive noise. In *Advances in Neural Information Processing Systems*, pages 9464–9474, 2019.

[26] Xinfeng Li, Yuchen Yang, Jiangyi Deng, Chen Yan, Yanjiao Chen, Xiaoyu Ji, and Wenyuan Xu. Safegen: Mitigating unsafe content generation in text-to-image models. *arXiv preprint arXiv:2404.06666*, 2024. USSLAB, Zhejiang University; Johns Hopkins University.

[27] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.

[28] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[29] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks, 2019.

[30] Microsoft. Photodna. https://www.microsoft.com/en-us/photodna, 2015. Accessed: 2021-10-18.

[31] Matthew Mirman, Timon Gehr, and Martin Vechev. Differentiable abstract interpretation for provably robust neural networks. In *International Conference on Machine Learning*, pages 3575–3583, 2018.

[32] Shakir Mohamed, Mihaela Rosca, Michael Figurnov, and Andriy Mnih. Monte carlo gradient estimation in machine learning, 2020.

[33] Mark Niklas Mueller, Franziska Eckert, Marc Fischer, and Martin Vechev. Certified training: Small boxes are all you need. In *The Eleventh International Conference on Learning Representations*, 2023.

[34] Mark Niklas Müller, Christopher Brix, Stanley Bak, Changliu Liu, and Taylor T. Johnson. The third international verification of neural networks competition (vnn-comp 2022): Summary and results. *arXiv preprint arXiv:2301.05816*, 2023.

[35] Jonathan Prokos, Neil Fendley, Matthew Green, Roei Schuster, Eran Tromer, Tushar Jois, and Yinzhi Cao. Squint hard enough: Attacking perceptual hashing with adversarial machine learning. In *32nd USENIX Security Symposium (USENIX Security 23)*, 2023.

[36] A. Raghunathan, J. Steinhardt, and P. Liang. Certified defenses against adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2018. arXiv preprint arXiv:1801.09344.

[37] Hadi Salman, Jerry Li, Ilya Razenshteyn, Pengchuan Zhang, Huan Zhang, Sébastien Bubeck, and Greg Yang. Provably robust deep learning via adversarially trained smoothed classifiers. pages 11289–11300, 2019.

[38] Hadi Salman, Greg Yang, Huan Zhang, Cho-Jui Hsieh, and Pengchuan Zhang. A convex relaxation barrier to tight robustness verification of neural networks. *Advances in Neural Information Processing Systems*, 32:9835–9846, 2019.

[39] Zhouxing Shi, Qirui Jin, Zico Kolter, Suman Jana, Cho-Jui Hsieh, and Huan Zhang. Neural network verification with branch-and-bound for general nonlinearities. *arXiv preprint arXiv:2405.21063*, 2024.

[40] Zhouxing Shi, Yihan Wang, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh. Fast certified robust training with short warmup. In *ICML 2021 Workshop on Adversarial Machine Learning*, 2021.

[41] Lukas Struppek, Dominik Hintersdorf, Daniel Neider, and Kristian Kersting. Learning to break deep perceptual hashing: The use case neuralhash. In *Proceedings of the ACM Conference on Fairness, Accountability, and Transparency (FAccT)*, 2022.

[42] Hoang-Dung Tran, Xiaodong Yang, Diego Manzanas Lopez, Patrick Musau, Luan Viet Nguyen, Weiming Xiang, Stanley Bak, and Taylor T. Johnson. Nnv: The neural network verification tool for deep neural networks and learning-enabled cyber-physical systems, 2020.

[43] Shiqi Wang, Huan Zhang, Kaidi Xu, Xue Lin, Suman Jana, Cho-Jui Hsieh, and J Zico Kolter. Beta-CROWN: Efficient bound propagation with per-neuron split constraints for complete and incomplete neural network verification. *Advances in Neural Information Processing Systems*, 34, 2021.

[44] Xunguang Wang, Zheng Zhang, Guangming Lu, and Yong Xu. Targeted attack and defense for deep hashing. In *ACM SIGIR*, pages 2298–2302, 2021.

[45] Xunguang Wang, Zheng Zhang, Baoyuan Wu, Fumin Shen, and Guangming Lu. Prototype-supervised adversarial network for targeted attack of deep hashing. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16357–16366, 2021.

[46] Eric Wong and J. Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, pages 5283–5292, 2018.

[47] Eric Wong, Frank Schmidt, Jan Hendrik Metzen, and J Zico Kolter. Scaling provable adversarial defenses. *Advances in Neural Information Processing Systems*, 31, 2018.

[48] Haoze Wu, Omri Isac, Aleksandar Zeljić, Teruhiro Tagomori, Matthew Daggitt, Wen Kokke, Idan Refaeli, Guy Amir, Kyle Julian, Shahaf Bassan, Pei Huang, Ori Lahav, Min Wu, Min Zhang, Ekaterina Komendantskaya, Guy Katz, and Clark Barrett. Marabou 2.0: A versatile formal analyzer of neural networks, 2024.

[49] K. Y. Xiao, V. Tjeng, N. M. Shafiullah, and A. Madry. Training for faster adversarial robustness verification via inducing relu stability. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[50] Yanru Xiao and Cong Wang. Yousee what i want you to see: Exploring targeted black-box transferability attack for hash-based image retrieval systems. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1934–1943, 2021.

[51] Kaidi Xu, Zhouxing Shi, Huan Zhang, Yihan Wang, Kai-Wei Chang, Minlie Huang, Bhavya Kailkhura, Xue Lin, and Cho-Jui Hsieh. Automatic perturbation analysis for scalable certified robustness and beyond. *Advances in Neural Information Processing Systems*, 33, 2020.

[52] Kaidi Xu, Huan Zhang, Shiqi Wang, Yihan Wang, Suman Jana, Xue Lin, and Cho-Jui Hsieh. Fast and Complete: Enabling complete neural network verification with rapid and massively parallel incomplete verifiers. In *International Conference on Learning Representations*, 2021.

[53] Erkun Yang, Tongliang Liu, Cheng Deng, and Dacheng Tao. Adversarial examples for hamming space search. *IEEE Transactions on Cybernetics*, 50(4):1473–1484, 2020.

[54] Huan Zhang, Hongge Chen, Chaowei Xiao, Bo Li, Duane Boning, and Cho-Jui Hsieh. Towards stable and efficient training of verifiably robust neural networks. 2020.

[55] Huan Zhang, Shiqi Wang, Kaidi Xu, Linyi Li, Bo Li, Suman Jana, Cho-Jui Hsieh, and J Zico Kolter. General cutting planes for bound-propagation-based neural

network verification. *Advances in Neural Information Processing Systems*, 2022.

[56] Huan Zhang, Shiqi Wang, Kaidi Xu, Yihan Wang, Suman Jana, Cho-Jui Hsieh, and Zico Kolter. A branch and bound framework for stronger adversarial attacks of ReLU networks. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pages 26591–26604, 2022.

[57] Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. *Advances in Neural Information Processing Systems*, 31:4939–4948, 2018.