Black-box Collision Attacks on Apple NeuralHash and Microsoft PhotoDNA

Diane Leblanc-Albarel
KU Leuven

Bart Preneel
KU Leuven

August 26, 2025

Abstract

Perceptual hash functions have been designed to detect multimedia copyright violations and illegal content. To achieve their purpose, they have to maps inputs that are perceived as similar to close outputs. For several widely used designs the design strategy or even the design details are proprietary. Governments consider extending these functions to Client-Side Scanning (CSS) for end-to-end encrypted services, verifying content against illegal material before encryption. In 2021, Apple presented a detailed proposal for CSS based on the NeuralHash perceptual hash function. After strong criticism pointing out privacy and security concerns, Apple withdrew the proposal, but NeuralHash remains deployed on all devices, with its current purpose undisclosed. Brute-force collisions for NeuralHash (with a 96-bit result) require 2^{48} evaluations. Shortly after NeuralHash's release, researchers showed it is easy to craft perceptually dissimilar collisions, enabling false incrimination in CSS by sending an innocent image with the same hash as illegal content. This work shows a more serious weakness: when inputs are restricted to a set of human faces, random collisions are highly likely to occur in input sets of size 2^{16}. Unlike the targeted attack, our black-box attacks require no knowledge of the hash function's design. We also show a high false negative rate (pictures that should share the same hash but do not). We show the generality of our approach by attacking PhotoDNA, Microsoft's widely deployed 1152-bit perceptual hash. Small input sets yield near-collisions after only $2^{14.6}$ or 2^{17} operations, depending on the threshold. These results show that current perceptual hash designs are unsuitable for large-scale client scanning, producing high false positive and false negative rates, and highlight the need to reassess their security and feasibility, particularly for large-scale applications where privacy risks and false positives have serious consequences.

Coordinated Vulnerability Disclosure. As part of our research, we have followed coordinated vulnerability disclosure procedures by timely notifying Apple and Microsoft of our findings. Apple had informed us that they had reproduced all of our results without exception, acknowledged the issues, and were investigating solutions. At the time of submission, Microsoft had not responded to us.

1 Introduction

Perceptual hash functions identify perceptually similar multimedia content such as images, videos, or sounds by producing close or identical hash values for visually or audibly similar inputs. Unlike cryptographic hash functions that produce completely different outputs for even small input changes, these functions produce the same or similar outputs for perceptually similar inputs.

Perceptual hash functions are widely used for detecting copyright violations [30, 55], identifying problematic content such as Child Sexual Abuse Material (CSAM) [3] or terrorist material [17], and

enabling biometric authentication [33]. Comparing hash values allows detection without leaking the content.

The US National Center for Missing and Exploited Children (NCMEC) has reported a sharp increase in CSAM detections in recent years [50]. NCMEC and researchers such as Bursztein et al. [13] and Farid [27] have called for automated detection methods, including perceptual hashing, to combat this growth. At the same time, the rise of end-to-end encryption has hindered centralized detection.

In response, the UK [60] and the European Union [16] have proposed Client-Side Scanning (CSS), which performs detection locally on user devices before encryption, allowing authorities to act before content is shared [3, 39, 41].

CSS has faced strong criticism from academics, industry, and NGOs (see Abelson et al. [1] for an overview). Experts warn that it enables mass surveillance, undermines privacy rights, and creates a chilling effect. Once deployed, such systems could be extended beyond CSAM detection to other criminal content, political dissent, or the identification of whistleblowers and journalists. Although some proposals aim to detect CSAM while preserving privacy [8], deployed technologies offer weak privacy guarantees and remain vulnerable to false positives and evasion. Members of the EU Parliament [11] and experts have issued open letters and public statements highlighting these concerns [51, 52].

This paper evaluates two widely deployed perceptual hash functions: Apple's NeuralHash [3] and Microsoft's PhotoDNA, the latter used by NCMEC, hundreds of tech companies and all Microsoft services for online CSAM detection. We focus on *false positives* (perceptually different images with identical hash values) and *false negatives* (perceptually similar images with different hash values). Earlier work on NeuralHash [56] showed that it is easy to create false positives by manipulating images. Here, we consider large-scale deployments where legitimate image sharing could still cause false accusations.

In its Q&A report [2] on NeuralHash, Apple addressed concerns about false positives:

> *"Will CSAM detection in iCloud Photos falsely report innocent people to law enforcement? No. The system is designed to be very accurate, and the likelihood that the system would incorrectly identify any given account is less than one in one trillion per year."*

Concerning PhotoDNA, the European Commission's 2023 report to the European Parliament and the Council [23] states:

> *"The most widely used tool is Microsoft PhotoDNA, used by over 150 organisations. PhotoDNA has been in use for more than 10 years and has a high level of accuracy. The rate of false positives is estimated at no more than 1 in 50 billion, based on testing."*

However, the "1 in 50 billion" estimate originates from a 2019 report by its creator, Farid [26], which does not explain or justify the methodology to derive this number.

To our knowledge, no independent and rigorous evaluation has confirmed these claims. We thus analyze the efficiency and accuracy of both functions at scale, assessing their suitability for real-world use. Our findings show that large-scale deployment of either NeuralHash or PhotoDNA would produce a substantial number of false positives, even without image manipulation.

The remainder of this paper is organized as follows. Section 2 introduces perceptual hashing and defines perceptually identical content. Section 3 examines NeuralHash's properties, focusing on collision behavior for different image types. Section 4 presents quantitative results and collision examples for NeuralHash. Section 5 reports PhotoDNA collision results, illustrating the generality

of our approach. Section 6 discusses the impact of these collisions on large-scale deployments. Section 7 discusses the limitations of our work and the applicability of our results to CSAM detection. Finally, Section 8 summarizes our findings.

Disclaimer. Child sexual abuse and exploitation are serious crimes that must be addressed in our digital society. We firmly condemn the creation and distribution of CSAM. This paper aims to inform the discussion regarding the efficacy and implications of using perceptual hashing and Client-Side Scanning on a large scale. Our analysis of NeuralHash and PhotoDNA is not intended as a critique of Apple Inc., Microsoft, or any initiatives aimed at mitigating the spread of CSAM; rather, it highlights the risks associated with their large-scale deployment. NeuralHash and PhotoDNA serve as a case study that underscores the critical need for accuracy and robustness in these systems to prevent false positives. We hope this work will encourage further investigation and dialogue aimed at combating CSAM while ensuring a fair balance between efficiency and user privacy.

2 Background

This section defines the notion of *perceptually identical* content and discusses the properties of perceptual hash functions, and their typical building blocks. Next it reviews the most important perceptual hash function designs.

2.1 Perceptually Identical Content

While content can consist of images, video, audio, 3D objects or 3D-immersive environments, this paper limits itself to the first category which is the target of NeuralHash and PhotoDNA. Perceptually identical images are those that appear identical or nearly identical for a *human observer*. From this human perception, similarity between images can be characterized by the following factors:

- **Color Consistency:** Images with minor differences in brightness, contrast, or color balance but identical in overall color composition and layout are considered perceptually the same. For example, an image with slightly adjusted brightness levels remains perceptually identical to its original.
- **Structural Similarity:** Images that share the same shapes, edges, and textures, even if subjected to minor geometric transformations such as rotations, translations, or small distortions, are considered perceptually similar. For instance, an image and its slightly rotated version would be perceptually identical.
- **Content Preservation:** Images that maintain the same core content but differ in resolution or compression are also perceptually similar. For example, a high-resolution image and a compressed version with some loss of detail are perceived as the same image.
- **Noise Robustness:** Images with minor noise additions, such as random pixel variations or blurring, which do not significantly alter the perceived content, are considered perceptually the same. For instance, an original image and one with a limited amount of Gaussian noise are perceptually identical.

Mathematically, some of the perceptual similarity elements mentioned above are typically formalized using metrics that attempt to quantify human visual perception. Two widely used metrics

are the *Structural Similarity Index Measure*, that measure the similarity between two images based on luminance, contrast, and structure [64, 12] and the *Peak Signal-to-Noise Ratio*, that measures the ratio between the maximum possible pixel value of the image and the number of corrupted pixels [15].

Even if widely used, these metrics cannot detect every case of perceptually similar images. Hence in this paper the perceptually similarity between images is evaluated taking into account the four properties identified above. Section 3 provides the exact definition used to classify images as similar.

2.2 Properties

While cryptographic hash functions and perceptual hash functions share some similarities, they have different purposes and thus different properties. Both hash functions efficiently process large inputs and reduce these to short outputs in a deterministic way.

The properties [22, 27] of perceptual hash functions in the context of image hashing are:

- **Pre-image resistance:** Similar to cryptographic hash functions, perceptual hash functions should be one-way, meaning it should be computationally infeasible to reconstruct an original input x from its hash value $H(x)$.
- **Second pre-image resistance:** Given x_1 and its hash value $H(x_1)$, it should be computationally infeasible to find an input x_2, perceptually different from x_1, such that $H(x_1) = H(x_2)$. More generally, this requirement also applies when equality is replaced by a distance condition $d(H(x_1), H(x_2)) \leq \tau$, where d denotes a metric such as the L_1 or L_2 distance (see Definitions 3 and 4 in Section 5 for details), and τ is a predefined low threshold.
- **Illegitimate-Collision Resistance:** Perceptual hash functions should ensure that perceptually different inputs produce different or sufficiently different hash values. More formally, it should be computationally infeasible to find two perceptually different inputs x_1 and x_2 for which $H(x_1) = H(x_2)$.
- **Accuracy:** Perceptual hash functions should produce the same (or similar) hash values for perceptually identical or very similar inputs. This property ensures that minor variations in the input (such as slight changes in brightness or minor cropping in an image) do not result in significantly different hash values.

2.3 Perceptual Hashing Process

A first generation perceptual hash functions does not involve deep learning techniques. Examples include pHash [68] and Microsoft's PhotoDNA [55]. These designs typically rely on hand-crafted features and transformations to generate hash values that are robust to minor changes in input.

Deep perceptual hash functions [42, 43], on the other hand, are based on a Machine Learning (ML) model. They involve training deep neural networks to learn feature representations that capture the perceptual similarity of inputs. Deep perceptual hash functions are less used than non-deep ones as they are more recent. Research on deep perceptual hashing includes hashing for image retrieval [69], for label prediction [65], and for CSAM detection [3].

Perceptual hash functions typically follow similar sequences of steps to generate hash values [22].

The first step is preprocessing, which prepares the input image by normalizing it into a standardized format. This often involves resizing the image to fixed dimensions, such as (360×360) pixels,

and normalizing pixel values to a specific range, for example, $[-1, 1]$. Additional preprocessing techniques may include computing image gradients or converting the image to grayscale.

Deep perceptual hash functions extract features from the preprocessed image using an ML model, This model is typically trained using techniques such as contrastive learning [42], that helps the model differentiating between perceptually similar and dissimilar images.

The next step is hash value extraction, where specific features of the image are extracted to form the hash result. The features selected depend on the hash function application. Common methods for hash extraction include computing the average color of the image, its gradient, or using techniques such as Locality-Sensitive Hashing [28, 32] and Binary Reconstructive Embeddings [38].

Finally, the generated hash values are converted to binary strings and possibly compared to another hash value using a hash comparison technique. The Euclidean distance [27, 22, 55] is the most common distance metric for comparisons.

2.4 Perceptual Hash Functions Designs

Perceptual hash functions have been developed and deployed in various contexts since 2000 [22]. We mention below some of the most notable designs and their applications.

One of the earliest implementations of perceptual hashing for content identification was YouTube's Content ID [30]. Content ID identifies copyrighted material to assist copyright holders in managing their rights.

The 2010 thesis of Zauner [68, 37] provides a detailed introduction to perceptual hash function and introduces the open-source construction pHash, that inspired several other designs.

PhotoDNA [49] was developed by Microsoft and Farid in 2009. It is extensively used for content moderation [55] and on platforms such as Gmail, Twitter, Facebook, Reddit, and Discord for detecting illegal content, particularly CSAM. Despite its widespread use, some successful attacks have been reported (see Section 2.7).

eGlyph [17], based on PhotoDNA, was implemented by the Counter Extremism Project (CEP), a nonprofit international policy organization combating extremist ideologies, with the help of Farid. In particular in 2018, eGlyph was used to detect extremist videos on YouTube.

In 2019, Facebook released PDQ and TMK+PDQF [24] as open-source perceptual hash functions based on pHash [19]. These functions are designed to enhance the identification and moderation of prohibited content, in particular CSAM content, across Facebook's platform and beyond. PDQ function is designed for images while TMK+PDQF targets videos.

Apple introduced in 2021 NeuralHash [3], a perceptual hash function specifically designed for CSAM detection. NeuralHash uses a convolutional neural network (CNN) to generate hash values from images, aiming to detect illegal content.

2.5 NeuralHash

In August 2021, Apple announced NeuralHash as a key component of its new CSAM detection system [3, 10]. The system was designed to identify known CSAM images stored in iCloud Photos by comparing on-device image hash values to a database of known CSAM hash values provided by child safety organizations such as NCMEC. The deployment of NeuralHash involved integrating the hashing algorithm directly into the iOS operating system: when an image is uploaded to iCloud Photos, NeuralHash generates a hash value that is compared against the database. If a match is found, the image is flagged and, if confirmed, the user is reported to the authorities.

Apple's approach raised significant public debate and controversy. As a consequence, Apple officially withdrew the proposal and postponed the large-scale deployment of the CSAM detection

system. Nonetheless, NeuralHash itself has been embedded in all Apple devices.

NeuralHash was designed to operate in conjunction with Apple's Client-Side Scanning (CSS) framework. At a high level, Apple employs a mechanism based on derived cryptographic keys and threshold secret sharing to match content on user devices with encrypted reference material stored on the server. This approach is intended to ensure that a minimum number of matches is required before any decryption occurs. For a detailed description of the CSS protocol, including Apple's use of cryptographic headers and threshold cryptography, we refer the reader to Appendix A and Apple's technical report [3].

For our experiments, we extracted the neural network and hashing matrix used by NeuralHash from an iPhone with firmware version 16.2. We then used [67] to convert the NeuralHash model into ONNX format, allowing us to run NeuralHash in any script or device.

2.6 Microsoft PhotoDNA

Unlike NeuralHash, PhotoDNA relies exclusively on traditional image processing operations. Although Microsoft has never publicly disclosed the exact algorithm, several leaks have enabled researchers to obtain equivalent black-box implementations. Publicly available code allow the retrieval of corresponding PhotoDNA hash values for any image input. However, except for some high-level descriptions of PhotoDNA [26], there is, as of the date of submission, no publicly available information detailing the mathematical steps used to compute a PhotoDNA hash. The exact algorithm therefore remains undisclosed, and PhotoDNA can only be computed in a fully black-box setting. Nevertheless, the high level description of PhotoDNA as proposed in [25] is summarised in Appendix A.

In this paper, we use the implementation available on GitHub [35] to execute PhotoDNA in a black-box setting, making use of the previously leaked file. This implementation has also been referenced in prior work [55, 5, 54].

2.7 Related Work

Perceptual hash functions have been studied in academic papers for over two decades. A comprehensive introduction to the topic was provided by Farid in 2021 [27], detailing the fundamental techniques and challenges. Additionally, a survey of perceptual hashing techniques and their applications can be found in [22].

Recent research in perceptual hashing has primarily focused on the creation of collisions and on information leakage. The creation of collisions involves modifying one of two perceptually different images to produce the same hash value, effectively creating false positives. Information leakage, on the other hand, refers to the potential of inferring information about the input from its hash value.

One prevalent method of attacking perceptual hash functions is through gradient-based hash attacks. These attacks involve imperceptibly altering pixels to achieve specific outputs [57, 14, 18, 20, 58]. Various optimization techniques have been proposed to enhance the effectiveness of these attacks [29, 53, 45]. These attacks generally follow the same principles as classic image processing attacks, manipulating inputs to achieve specific outputs [7, 70, 61, 62, 66].

Several papers have analyzed the resistance of perceptual hash functions to image modifications and the potential to recover original images from their hash values. This includes research on the robustness of perceptual hash functions derived from pHash [34, 21] and attempts to reconstruct original images using Binary Reconstructive Embeddings instead of Locality-Sensitive Hashing [63].

Recent work has revealed a series of attacks that exploit vulnerabilities in the internal structure of NeuralHash and other perceptual hash functions. Notably, Struppek et al. [56] showed how to

create second pre-images for NeuralHash, where input images were imperceptibly modified and published a proof-of-concept implementation (other tools for second pre-images can be found in [6, 36]). Struppek et al. also described classification attacks, in which hash values are used to categorize inputs, achieving a maximum success rate of 52% in some categories. Similarly, attacks on pHash have shown how images can be manipulated to produce specific hash values [31]. Additionally, partial inversion of PhotoDNA has been achieved using neural networks [5].

Despite this research on attacking perceptual hash functions and NeuralHash in particular, there remains a significant gap in the evaluation of these functions concerning real false positive rates and their performance when approached as black boxes. This gap underscores the necessity of our analysis, which aims to evaluate the performance, accuracy, and false positive rate of NeuralHash in the context of large-scale CSAM detection.

3 Analysis of Perceptual Hash Functions

This section first presents the datasets used for both NeuralHash and PhotoDNA, and defines what we refer to as a *perceptually identical image*. We then report our observations on the distribution of NeuralHash outputs and on the (non-)independence of the hash bits. For both the distribution and independence analyses, we begin by presenting observations for different types of images. Based on these observations, we then estimate the corresponding theoretical probability of collisions. For brevity and clarity, we present detailed observations only for NeuralHash; however, we obtained the same types of observations and reached identical conclusions for PhotoDNA.

3.1 Image Sets

We use two distinct image datasets in our analysis. The first dataset consists of non-human images extracted from the PASS dataset [4]. The second dataset contains celebrity face images from the CelebA dataset [44]. Within the $N = 202\,599$ images of the CelebA dataset, we identified at least $N_{\text{dup}} = 4\,629$ images that are duplicates or perceptually similar, i.e., images that should ideally map to the same hash value.

Examples of such identical and perceptually similar pairs are shown in Figure 1.

(a) Identical images (b) Different filter (c) Very close pose

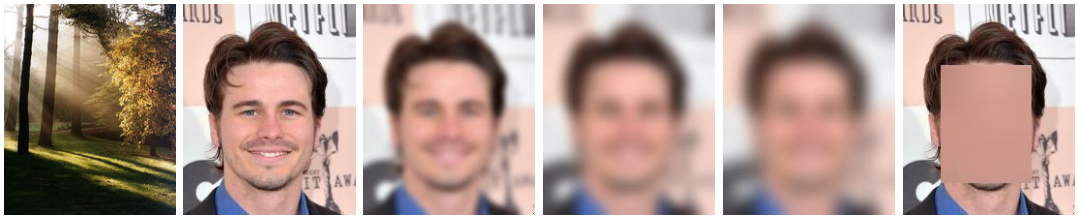
Figure 1: Perceptually identical images

Our primary goal is to evaluate whether hashing face images with perceptual functions yields different results compared to non-human images. Specifically, we aim to analyze the number of collisions and the statistical properties of every bits of the hash values.

We selected face images for two main reasons: first, to approximate a use case related to CSAM detection, as such images often contain faces; second, because face images are common on mobile devices, making this a realistic scenario. In addition, we considered different levels of blurring, since blurring may plausibly occur in the context of CSAM production.

Based on these criteria, we constructed six image sets: non-human images, human face images, and human face images with varying levels of blurring. An example of each type is shown in Figure 2. Throughout the paper, the abbreviation *BF* stands for *blurred faces*, and we refer to the six types of images as follows:

- *Non Human*: random images from the PASS [4] dataset.
- *Non BF*: images from the CelebA dataset [44].
- *Light BF*: CelebA images with light blur.
- *Medium BF*: CelebA images with medium blur.
- *High BF*: CelebA images with high blur.
- *BF only*: CelebA images with blur applied only to the face region only.



(a) Non-Human (b) Non BF (c) Light BF (d) Medium BF (e) High BF (f) BF only

Figure 2: Example of the 6 types of images in the experiments

3.2 Statistical Properties of the Hash Values

This section presents both a theoretical analysis and experimental results to assess the properties of perceptual hash functions. We analyze the statistical distribution of hash values and test the independence of hash bits using probabilistic models. In parallel, we conduct experiments on large-scale datasets to validate these theoretical findings, using NeuralHash as a primary example before extending our evaluation to PhotoDNA.

3.2.1 Distribution

We first experimentally analyse the distribution and in particular the uniformity and variance of NeuralHash bits values for different types of images. We then introduce three propositions that help to determine the theoretical number of hash function evaluations before obtaining an illegitimate collision.

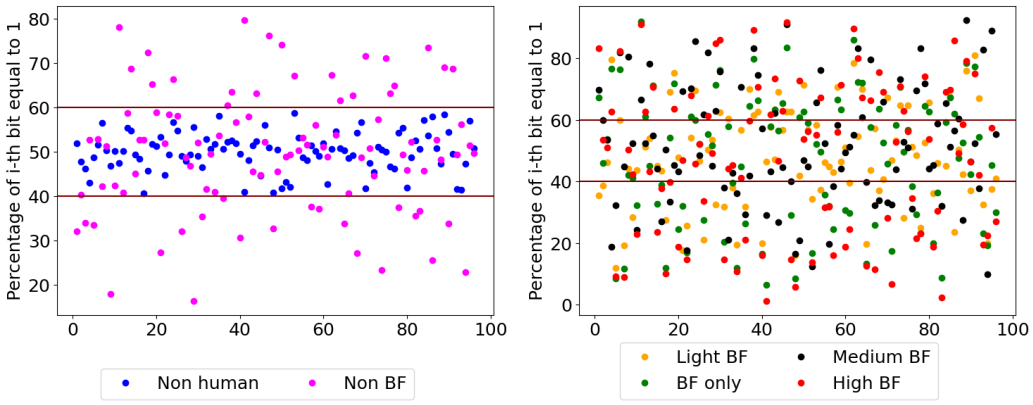
Experiments. As stated in Section 2, the bit distribution of the output of a hash function must be uniform with a small variance to prevent information leakage about the input and to resist (2nd) pre-image and collision attacks.

To compute the average distribution of Neuralhash bits for each image type, we hash 30 000 randomly selected images from each set. For each image type, we count for each of the 96 bits the number of times the bit is equal to 1 and 0 respectively. The results for the Non-human type and the Non-blur face type are presented in Figure 3a, where the percentage of times each bit is equal

to 1 is plotted. The expected result is that each bit is equal to 1 approximately 50% of the time, with a small variance.

For non-human images, the results meet expectations, with the percentage of bits equal to 1 uniformly distributed around 50%. In contrast, for non-blurred human face images, a significant number of bits deviate from being equal to 1, 50% of the time. For non-human images, none of the bits of the hash is less than 40% or more than 60% of the time equal to 1. In contrast, for human face images, 44 bits fall outside this range, with even 2 bits being less than 20% or more than 80% of the time equal to 1.

The results for the four remaining image types are presented in Figure 3b. The resulting hash values are even less uniformly distributed compared to the non-blurred face image type. For lightly blurred images (orange dots), 56 bits fall outside the $40\% - 60\%$ range. For images with only the face blurred (green dots), 59 bits fall outside the range. For medium blurred images (black dots), 67 bits fall outside the range, and for highly blurred images, 70 bits out of the 96 final bits fall outside the $40\% - 60\%$ range.



(a) Non-human vs. Non BF

(b) Blurred Human Faces

Figure 3: Bit distribution for each image type

Additionally, Figure 9 in Appendix C shows a boxplot of the distribution of bit values for each image type: except for the non-human images, the distribution deviates strongly from the expected value.

Theoretical Collision Probability with Independence Hypothesis. An illegitimate collision corresponds to the event where the hash value of two perceptually different images are the same. For regular hash functions, this event should be exceedingly rare. For an uniformly distributed 96-bit hash value (NeuralHash case), where each bit is independent of the others, the birthday paradox [9] states that the average number of hash values to compute before encountering an illegitimate collision is equal to 2^{48}.

Assuming that all bits of the hash value are mutually independent, and considering the observed distribution for each dataset, Proposition 1 provides the probability that two hash values are identical. Proposition 2 gives the expected number of hash values required before encountering the first illegitimate collision. To make this result more practical, Proposition 3 provides a tight bound on the expected number of hash values necessary before observing the first illegitimate collision.

Proposition 1. *Consider a hash value of n bits and a vector p for which the i-th element p_i denotes the probability that the i-th bit of the hash value is equal to 1 ($0 \leqslant p_i \leqslant 1$). If E_p^n denotes the event that two hash values are equal, then*

$$\mathbb{P}(E_p^n) = \prod_{i=1}^{n} (p_i^2 + (1-p_i)^2)\,.$$

Proof. For brevity, the proof is provided in Appendix B. □

Proposition 2. *Denote with $\mathbb{P}(E_p^n)$ the probability that two n-bit hash values with distribution p are equal and define $U = 2^n$. The expected number of hash values to compute before the first collision $\mathbb{E}(D_p^n)$ is equal to:*

$$\mathbb{E}(D_p^n) = 2 + (U+1) \times y^{\frac{U(U+1)}{2}} + \sum_{x=1}^{U-1} y^{\frac{x(x+1)}{2}}$$
$$\text{with } y = 1 - \mathbb{P}(E_p^n)\,.$$

Proof. For brevity, the proof is provided in Appendix B. □

The computation of the exact value of $\mathbb{E}(D_p^n)$ is infeasible given the sum over U elements (in the NeuralHash case $U = 2^{96}$ and in case of PhotoDNA $U = 2^{1152}$). Proposition 3 provides a bound for $\mathbb{E}(D_p^n)$ that is easy to compute.

Proposition 3. *Denote with $\mathbb{P}(E_p^n)$ the probability that two n-bit hash values with distribution p are equal and define $U = 2^n$. The expected number of hash values to compute before the first collision $\mathbb{E}(D_p^n)$ is upper bounded by:*

$$\mathbb{E}(D_p^n) \leq 1 + (U+1) \times y^{\frac{U(U+1)}{2}} + \frac{\theta_2(0; y^{\frac{1}{2}})}{2 \times y^{\frac{1}{8}}}\,,$$

with $\theta_2(0; y^{\frac{1}{2}})$ the Jacobi theta function $\theta_2(z; q)$ with $z = 0$ and $q = y^{\frac{1}{2}}$ and with $y = 1 - \mathbb{P}(E_p^n)$.

Proof. For brevity, the proof is provided in Appendix B. □

Table 1 is derived from Proposition 3 and the distributions of NeuralHash presented in Figure 3. It provides the expected number of hash values to compute before reaching the first illegitimate collision for each image type.

Table 1: Expected number of hash operations before first illegitimate collision using NeuralHash

Type	Non human	Non BF	Light BF	BF only	Medium BF	High BF
$\mathbb{E}(D_p^n)$	$2^{47.8}$	$2^{43.5}$	$2^{41.1}$	$2^{38.9}$	$2^{37.5}$	$2^{34.1}$

Conclusion on Distribution. From Table 1, we conclude that, except for the non-human images, the expected number of hash values to compute before the first NeuralHash illegitimate collision is significantly lower than the theoretical value 2^{48}. For other image types, the expected number of hash values to compute before an illegitimate collision varies between $2^{43.5}$ (non-blurred face) and $2^{34.1}$ (highly blurred face).

It is worth noting that these values assume each bit of the hash is independent of the others. If this independence does not hold, the expected number of hash values before reaching an illegitimate collision decreases substantially.

3.2.2 Independence of Bits

In the previous section, we assumed that the bits of the hash values are independent of each other. In this section, we test this independence hypothesis. If the bits are independent, the value of any bit j of a hash value should not provide any information about the value of bit i with $j \neq i$.

Simple Matching Coefficient. The simple matching coefficient (*SMC*) is a statistic ranging from 0 to 1 used to compare the similarity of symmetric binary sample sets. When two samples are identical, the *SMC* is equal to 1. When two samples have no common values (i.e., they are completely different), the *SMC* is equal to 0. The expected *SMC* of two independent samples is 0.5. It is worth noting that while the *SMC* can be used to verify non-independence, it cannot be used to confirm independence.

For two bits from the same hash value h, Definition 1 provides the value $m_{i,j}(h)$, corresponding to the simple matching value of bits i and j denoted as $h(i)$ and $h(j)$ respectively.

Definition 1. *For a hash value h, $m_{i,j}(h)$ is defined as:*

$$m_{i,j}(h) = \begin{cases} 1 \text{ if } h(i) = h(j) \\ 0 \text{ if } h(i) \neq h(j)\,. \end{cases}$$

Using Definition 1, Definition 2 provides the *SMC* for two bits i and j of a sample of N hash values of n bits.

Definition 2. *Consider a set D_p^n of ℓ ordered hash values of length n. Given h_k, the k-th element of D_p^n, the SMC of bits i and j is defined as:*

$$SMC_{i,j}(D_p^n) = \frac{1}{N} \cdot \sum_{k=1}^{\ell} m_{i,j}(h_k)\,.$$

By definition, $\forall i, j$, $SMC_{i,j}(D_p^n) = SMC_{j,i}(D_p^n)$ and $SMC_{i,j}(D_p^n) = 1$ when $i = j$.

Using Definition 2, the simple matching matrix (*SMM*) of a set D_p^n is built by computing the *SMC* for every pair of bits. We thus use the *SMM* to compute the *similarity* between every pair of bits of hash values from each of the six image types.

Bits Sample Similarity. For each image type, the *SMM* of the set is computed using Definition 2 and the hash values of 30 000 random images per set. A color visualization of the NeuralHash *SMM* for the Non-human and Non-blurred face sets is presented in Figure 10 in Appendix C.

Table 2 summarizes, for each set, the values of the NeuralHash *SMM* that fall within different ranges, from $0.0 - 0.1$ (indicating that the pair of bits almost always have opposite values) to $0.9 - 1.0$ (indicating that the pair of bits almost always have equal values). For hash values with independent bits, 100% of these values should fall within the $0.4 - 0.6$ ranges.

Table 2: Similarity score, in percentage, for each image type

Type \ SMC	0–0.1	0.1–0.2	0.2–0.3	0.3–0.4	0.4–0.5	0.5–0.6	0.6–0.7	0.7–0.8	0.8–0.9	0.9–1
Non-Human	0.0	0.0	0.0	0.18	48.6	51.18	0.04	0.0	0.0	0.0
Non BF	0.0	0.0	0.29	6.97	43.05	42.98	6.56	0.13	0.02	0.0
Light BF	0.0	0.02	1.1	10.72	38.62	38.07	10.48	0.96	0.02	0.0
BF Only	0.0	0.2	2.7	11.64	33.93	35.68	12.7	2.89	0.26	0.0
Medium BF	0.0	0.33	3.53	15.7	31.47	29.69	14.76	3.88	0.64	0.0
High BF	0.2	1.69	6.67	15.75	26.45	29.99	13.64	7.61	2.08	0.11

Conclusion on Independence. Table 2 shows that for the Non-human type image, only 0.22% of the NeuralHash values fall outside the $0.4 - 0.6$ range. This corresponds to 20 pairs of bits over the 9120 possible pairs. These 20 values are all very close to either 0.4 or 0.6. These values are not sufficient to conclude that bits of non-human images are dependent.

However, for all other types of images, a significant number of NeuralHash values fall outside the $0.4 - 0.6$ range. For the non-blurred face type, 13.97% of the pairs are out of range. For the blurred face types, the percentages of values outside the range goes from 23.3% for the lightly blurred face type to 47.75% for the highly blurred face type. For the blurred face only type, 30.39% of the values are outside the range.

These results indicate that, except for the non-human images, the bits of the NeuralHash value are not independent. Consequently, Table 1 substantially overestimates the expected number of hash values required to obtain an illegitimate collision.

4 NeuralHash Collisions Observed in Practice

The results from Section 3 indicate that, for all types of images except the non-human ones, the expected number of NeuralHash evaluations before reaching a collision is significantly lower than the theoretical value of 2^{48}.

Therefore, we hashed all 202 599 images of each set, searching for illegitimate collisions between hash values. For each identified collision, we classified it as *legitimate* when the colliding hash values correspond to images identical or perceptually similar, and *illegitimate* when the colliding hash values correspond to images that are perceptually different. In the few cases of colliding hash between images that cannot clearly be identified as perceptually similar or not, we classified the collision as legitimate. Next we determined the number of illegitimate collisions (false positives) and illegitimate non-collisions (false negatives).

4.1 Illegitimate Collisions (False Positives)

The number of illegitimate collisions for each image type and the number of hash evaluations before the first collision are presented in Table 3. Since some collisions involve more than two images (multiple images sharing the same hash), we count the number of illegitimate collisions as the number of images sharing their hash values with at least one perceptually different image.

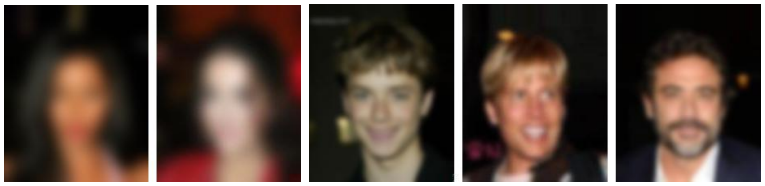
Table 3 shows that out of the 202 599 hash values computed for each image type, the non-human type is the only one without any illegitimate collisions. All other image types present several illegitimate collisions. The number of these collisions, given the number of image hashed, is

much higher than expected. This number of false positives for sets of only 202 599 images implies that the illegitimate collision rate will be much higher when millions or billions of images are hashed. Section 6 presents estimates for the number of illegitimate collisions for large-scale use.

Figure 4 presents an example of a collision obtained for each image type. It is worth noting that for all blurred face images (light, medium, and high), there are hash values shared by three or more perceptually different images. An example of three images sharing a hash value is given for the light blurred type in Figure 4e. For the medium blurred face set, eight different hash values are shared by three different images each, and four hash values are shared by four different images each. For the highly blurred face set, instances where three images share the same hash value are very frequent, and two hash values are shared by five different images each.

A new version of NeuralHash appears to have been deployed on macOS devices since late 2023, without any communication or explanation from Apple. We evaluated the false positive rates of this updated implementation; the detailed results are provided in Appendix D. These results are very similar to those obtained with the previous model, leading to the same conclusions.

(a) Non BF (b) BF Only (c) Medium BF



(d) High BF (e) Light BF

Figure 4: Collision examples for each image type

4.2 False Negatives

As defined in Section 3.1, we identified at least $N_{\text{dup}} = 4\,629$ images in the CelebA dataset that should produce legitimate collisions. Given N_{legit} observed legitimate collisions, the minimum false negative rate (FNR) is computed as given in Equation (1).

$$\text{FNR} = 100 \cdot \frac{N_{\text{dup}} - N_{\text{legit}}}{N_{\text{dup}}}\,. \tag{1}$$

The number of observed legitimate collisions and the resulting FNR are reported in Table 3. Note that N_{dup} is a lower bound: in the medium and highly blurred sets, additional legitimate collisions may occur when distinct images become perceptually identical after blurring. Despite this, many such cases are still missed, indicating a substantial false negative problem.

For the non-blurred face, lightly blurred face, and blurred face only image types, a significant number of perceptually identical images do not have the same hash values. For the blurred face only set, this rate reaches 77.3%. These results indicate that the function does not properly detect

Table 3: NeuralHash False positives and false negatives for each image type

	Non BF	Light BF	BF only	Medium BF	High BF
# illegit. coll.	12	12	6	126	270
# images involved in illegit. coll.	24	25	12	260	588
False pos. rate (%)	0.01	0.01	0.005	0.13	0.29
# hashes before first illegit. coll.	$2^{16.1}$	$2^{15.2}$	$2^{16.1}$	$2^{12.6}$	$2^{11.6}$
# legit. coll. (N_{legit})	1559	1513	1005	1864	2333
False neg. rate (%)	64.7	65.9	77.3	57.9	47.3

perceptually identical images. This ratio should be considered alongside the false positive rate. For the blurred face only set, for instance, the number of illegitimate collisions is lower than in other sets, but the number of legitimate collisions is also lower. This indicates that the function is particularly inaccurate when only faces are blurred.

5 PhotoDNA Collisions Observed in Practice

To assess whether the vulnerabilities identified in NeuralHash are inherent to the design of perceptual hashing, we conducted similar black-box experiments on Microsoft's PhotoDNA. Similar to NeuralHash, PhotoDNA is widely deployed for content moderation and is the primary function used by NCMEC to detect CSAM. PhotoDNA produces a 1152-bit hash value, and, as indicated by Microsoft [27, 26], near-collisions under a given threshold are used to flag similar images.

In this section, we present the collisions and false negatives found for PhotoDNA as a function of the threshold. For clarity and brevity, we focus on non-blurred face images, as results already vary with the threshold. As with NeuralHash, even slight blurring greatly increases the number of illegitimate collisions.

5.1 Experimental Setup

PhotoDNA (described in Section 2.6) produces a hash vector of 144 bytes (1152 bits). Unlike NeuralHash, perceptually similar images do not yield identical hash values but rather values that are close to each other (near-collisions [40]) according to some distance measure. The common distance measure are L1 and L2.

Definition 3. *Given two hash vectors $x, y \in \mathbb{R}^n$, the L1 distance is defined as:*

$$L_1(x, y) = \sum_{i=1}^{n} |x_i - y_i| ,$$

Definition 4. *Given two hash vectors $x, y \in \mathbb{R}^n$, the L2 distance is defined as:*

$$L_2(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}.$$

Circumventing PhotoDNA's detection mechanism requires finding perceptually different images whose hash values fall below the similarity threshold. Microsoft has not disclosed the specific distance metric used for PhotoDNA matching, nor the precise threshold values they apply. However, prior research [55, 34] has analyzed the function and recommended using the L2 distance metric with a threshold between 150 and 175. According to this study, these values prevent false positives while ensuring more than 99% true positive detection. Additionally, [54] employs the L1 distance metric and suggests that setting an L1 threshold of 1800 allows to avoid all false positives.

Based on these observations, we conducted experiments using the CelebA dataset. Specifically, we computed PhotoDNA hash values for all non-blurred face (Non-BF) images using the implementation provided in [35]. We then computed the L2 and L1 distances between all pairs of images, counting both legitimate collisions (cases where perceptually identical images yield similar hash values) and illegitimate collisions (cases where perceptually distinct images yield similar hash values). This methodology mirrors the approach we employed for NeuralHash, enabling a direct comparison of results.

In Sections 5.2 and 5.3, we present the results using the L2 distance. The results for the L1 distance, with a threshold at 1800 as in [54], are less favorable to PhotoDNA. Thus, for brevity and to remain conservative, we present the results for the L2 distance in this section. Results for the L1 distance are provided by Figure 14, in Appendix F.

5.2 Illegitimate Collisions (False Positives)

Figure 5 presents the number of legitimate and illegitimate collisions according to the L2 distance. Although [55] extends the analysis up to a distance of 500, we limited our study to 225 as beyond this threshold the number of illegitimate collisions increases dramatically.

Table 4 summarize the number of illegitimate collisions and the number of hash evaluations required before the first illegitimate collision for the two representative thresholds (150 and 175). For a threshold of 150, only 1 illegitimate collision is observed, corresponding to a false positive rate of 0.001%, with the first illegitimate collision appearing after approximately 2^{17} hash evaluations. Increasing the threshold to 175 results in 32 illegitimate collisions (false positive rate 0.03%) and reduces the number of hash operations before the first illegitimate collision to $2^{14.6}$.

Previous work [55] reports the first illegitimate collision at an L_2 distance between 200 and 225, claiming no false positives below 200. In contrast, our experiments identify the first illegitimate collision at a much lower distance of 144 and show that raising the threshold even moderately (e.g., to 175) significantly increases the false positive rate. While the rates reported in Table 4 may seem low for small-scale deployments, they would become significant when hashing millions or billions of images, as discussed in Section 6.

5.3 False Negatives

Table 4 reports the number of legitimate collisions for each threshold and the corresponding false negative rates, computed using Equation (1) with N_{legit} from the table. For threshold 150, the false negative rate is 24.1%, while at threshold 175 it decreases to 17.8%.

These results contradict prior claims in [55, 54] that thresholds in the range 150–175 can maintain a false negative rate below 1% while eliminating false positives. While a threshold of 150 does indeed produce a relatively low false positive rate (one illegitimate collision after only 2^{17} hash operations remaining a too high rate for large scale application) , it comes at the cost of missing nearly one quarter of perceptually identical images. Conversely, increasing the threshold to 175 improves the number of legitimate collisions but also substantially increases the false positive rate.

Table 4: False positive and false negative results for PhotoDNA with L_2 distance at thresholds 150 and 175.

Threshold	150	175
# illegit. collisions	1	32
# images involved in illegit. coll.	2	64
False positive rate (%)	0.001	0.03
# hashes before first illegit. coll.	2^{17}	$2^{14.6}$
# legit. collisions (N_{legit})	3506	3800
False negative rate (%)	24.1	17.8

This trade-off significantly limits the effectiveness of PhotoDNA for large-scale deployment. This discrepancy significantly diminishes the effectiveness of PhotoDNA compared to the claims made by Microsoft [26] and previous studies [55].

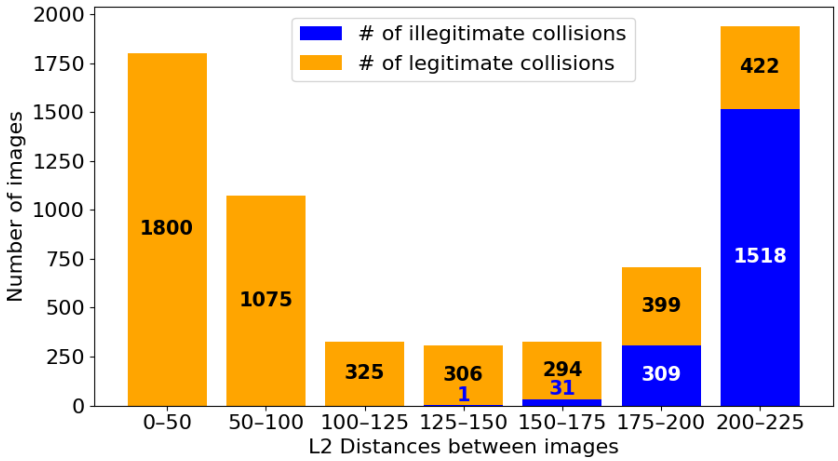


Figure 5: PhotoDNA near-collisions according to L2 distance

6 Estimation for Large-Scale Applications

In this section, we use the empirical results obtained on CelebA dataset of 202,599 images to estimate the expected false positive and false negative rates in the context of large-scale CSAM detection deployments. We first present the probabilistic model used for our estimations. We then apply this model to NeuralHash, including an evaluation of the impact of the 2023 design update. Finally, we perform the same analysis for PhotoDNA, highlighting the implications of large-scale usage for both systems.

6.1 Model Used For Approximation

To estimate the number of illegitimate collisions when hashing a larger number of images than in our current sets, we reduce the security to the equivalent output length of a uniform hash function

with independent output bits. This approach provides a worst-case scenario, ensuring that our estimates are conservative. It is worth noting that [40] proposes a similar method for evaluating the equivalent size of uniformly distributed and independent hash values with collisions and near-collisions.

Given the birthday paradox for a uniform distribution with independent bits, we can estimate the number of images that share the same hash values. Let q denote the number of images that illegitimately share their hash values with at least one other image. For a set of uniformly distributed hash values, where each bit is independent of the others, Equation (2), derived from the birthday paradox, estimates q for a set of N images:

$$q = N\left(1 - \left(\frac{2^n - 1}{2^n}\right)^{N-1}\right) . \tag{2}$$

To address the non-uniform distribution and interdependence of bits observed in both NeuralHash ($n = 96$) and PhotoDNA ($n = 1152$), we approximate each function by an *ideal* hash producing $n' < n$ uniformly distributed and independent bits. This enables the use of standard probabilistic models to estimate illegitimate collision rates.

We derive Equation (3) as a reformulation of Equation (2):

$$n' \approx -\log_2\left(1 - \left(1 - \frac{q}{N}\right)^{\frac{1}{N-1}}\right) . \tag{3}$$

We compute n' from Equation (3) using the values of q given in Table 3 (for NeuralHash) and Table 4 (for PhotoDNA), with $N = 202\,599 - N_{\text{dup}} = 197\,970$ unique images,[1] where $N_{\text{dup}} = 4\,629$.

The obtained value n' represents the size of an equivalent uniformly distributed and independent hash. We then apply the standard birthday bound (Equation (2)) for any number N of hash values to approximate the expected number of illegitimate collisions for each image type. Finally, we validate this approximation against our experimental dataset. Tables 5 and 6 present the resulting n' values for NeuralHash and PhotoDNA, respectively.

6.2 Estimating NeuralHash Performance at Large Scale

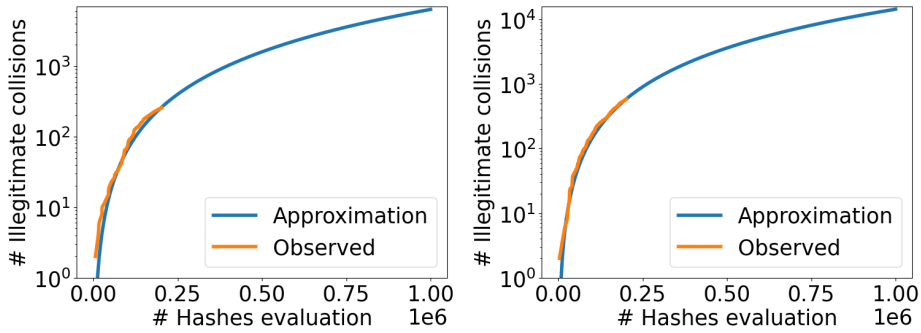
We extrapolate the number of illegitimate collisions for each image type to assess false positives in large-scale deployments. Figure 6 compares the approximated (blue) and observed (orange) number of collisions for medium and highly blurred images, showing close match between theory and experiment. For other types, the number of false positives on the full 96-bit output is too small for a meaningful comparison; however, as validated on 56-bit subsets in Appendix E, the approximation remains accurate. For completeness, Figure 12 in Appendix E reports the 96-bit results for these three types.

The consistency between observed and predicted false positives confirms the reliability of our approximation method. Table 5 summarizes the effective hash size n' for each image type and the corresponding estimated false positive rates for sets of 1 million, 10 million, and 100 million images; in what follows, M denotes one million.

From Table 5, we observe that the estimated false positive rates range from 0.03% to 1.48% for 1 M hash values. For 10 M hash values, the rates increase significantly, ranging from 0.25% to 13.84% depending on the image type. At 100 M hash values, the false positive rates become critical, with values between 2.52% and 77.51%, indicating that large-scale deployments would inevitably lead to substantial numbers of illegitimate collisions.

[1] We subtract the N_{dup} duplicate images, as these correspond to legitimate collisions.

These results highlight a clear trend: the probability of false positives increases rapidly with the number of hash values considered. This trend has direct implications for the large-scale deployment of perceptual hash functions such as NeuralHash in CSAM detection systems. In particular, the growing rate of false positives would result in a substantial number of legitimate users being incorrectly flagged, thereby reducing both the effectiveness and the reliability of such detection mechanisms.



(a) Medium BF (b) High BF

Figure 6: Approximated and observed number of illegitimate collisions on 96 bits NeuralHash

Table 5: Estimated equivalent hash size (n') and false positive rates.

Type	Non BF	Light BF	BF only	Medium BF	High BF
Size of equivalent hash (n')	30.7	30.6	31.9	27.2	26.0
Estimated false positive rate					
Approx. for 1 M hashes	0.06%	0.06%	0.03%	0.65%	1.48%
Approx. for 10 M hashes	0.55%	0.63%	0.25%	6.36%	13.84%
Approx. for 100 M hashes	5.46%	6.18%	2.52%	48.25%	77.51%

6.3 Estimating PhotoDNA Performance at Large Scale

Using Equations (2) and (3), we analyzed PhotoDNA results for the Non-Blur Face type, considering threshold 150 and 175. The results are presented in Figure 7. Table 6 reports effective hash sizes: ∼34 bits (threshold 150) and ∼29 bits (threshold 175).

Figure 13 in Appendix F compares estimates and observations for thresholds 175 and 220. Since only one collision is observed for a threshold of 150, the comparison is not meaningful in that case. For thresholds 175 and 220, estimates closely match observations, validating the approximation.

Table 6 also reports false positive rates for sets of 1 M, 10 M, and 100 M images. For 1 M hash values, the false positive rate remains low (below 0.2%), but it already increases noticeably

for 10 M hash values, reaching 0.05% for threshold 150 and 1.6% for threshold 175. For 100 M hash values, the false positive rates rise sharply: 0.51% for threshold 150 and 16.1% for threshold 175. These values indicate that even under the stricter threshold of 150, the number of false positives becomes non-negligible at very large scales[2], while under the threshold of 175, the collision rate becomes prohibitively high. At threshold 175 (as suggested in [55]), false positive rates exceed those of NeuralHash, despite PhotoDNA's much larger 1152-bit output. Most importantly, Table 6 shows that for 100 M images, false positive rates are prohibitive for both thresholds, making large-scale deployment infeasible.

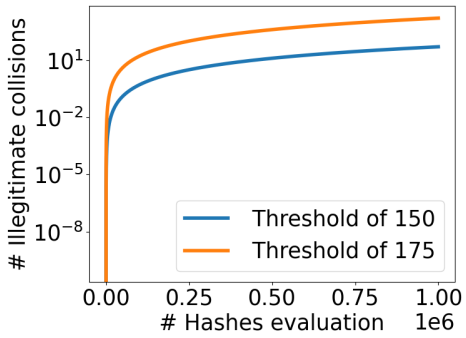


Figure 7: Approximated number of PhotoDNA false positives for Non-Blur type images and according to the threshold used

Table 6: Estimated equivalent hash size (n') and false positive rates for thresholds 150 and 175.

Threshold	150	175
Size of equivalent hash (n')	34.2	29.2
	False positive rate	
Approx. for 1 M hashes	0.005%	0.16%
Approx. for 10 M hashes	0.05%	1.6%
Approx. for 100 M hashes	0.51%	16.1%

Our analysis of PhotoDNA reinforces the conclusions drawn from NeuralHash: perceptual hash functions currently deployed for content moderation exhibit a number of false positives and negatives that is unacceptable, in particular when applied to human faces.

7 Limitations and Applicability to CSAM Detection

There may be concerns whether our experiments on face images directly generalize to CSAM detection. We contacted stakeholders with access to CSAM data, but at the time of submission had received no positive response. We plan to continue discussions with these stakeholders to seek a solution that enables independent evaluation while respecting privacy constraints.

Next we provide strong arguments that support the conclusion that our results do extend to CSAM detection. First, we believe that a high number of (near-)collisions were observed because we

[2]It nevertheless remains impractical for large-scale deployment, as even a single illegitimate collision among 202 599 images is already too high.

focused on a category of images (human faces) directly relevant to CSAM. We expect comparable results for other categories (e.g., children or humans in specific contexts). Second, the false positive and false negative rates we observe are unacceptably high for relatively small datasets. Even if real CSAM datasets would yield somewhat better results, the gap with the claimed rates (up to ten orders of magnitude) makes those claims implausible. Finally, as required by the European Commission, Meta publishes annual transparency reports on CSAM detection. In its 2024 report [48], Meta stated that 1.5 M images were blocked, 76 900 were appealed, and 1 800 were restored (explicitly identified as false positives). This corresponds to a false positive rate of 0.12%. The 2023 [47] and 2022 [46] reports indicate false positive rates of 0.32% (11,600 out of 3.6 M) and 0.07% (3,700 out of 6.6 M), respectively. These rates all exceed the rates we measure experimentally.

8 Conclusion

By analyzing NeuralHash and PhotoDNA as examples of real-world perceptual hash functions, our work has demonstrated fundamental flaws in their application to CSAM detection. Specifically, we showed that NeuralHash and PhotoDNA exhibit extremely high false positive rates when applied to human faces, regardless of whether they are blurred, and simultaneously suffers from very high false negative rates, even for unmodified facial images. These findings indicate that these functions are unreliable for detecting perceptually identical content. The impact of these observations is larger since our attacks do not require the specific structure of description of these perceptual hash functions.

In addition, in 2008 Microsoft, NCMEC, and Farid jointly defined explicit requirements for perceptual hash functions for CSAM detection, as formulated in [25]:

> "Any technology must satisfy the following requirements:
>
> 1. Analyze an image in under two milliseconds (500 images/second);
> 2. Misclassify an image as child pornography (CP) at a rate of no more than one in 50 billion;
> 3. Correctly classify an image as CP at a rate of no less than 99%; and
> 4. Do not extract or share any identifiable image content (because of the sensitive nature of CP)."

Requirement 4 has already been challenged in the literature [5]. Our work is the first to demonstrate that both NeuralHash and PhotoDNA fail to satisfy requirements 2 and 3. In particular, instead of achieving the "one in 50 billion" false positive rate (i.e., 2×10^{-11}), we observe false positive rates ranging from 10^{-3} to more than 10^{-1} depending on the dataset and scale, i.e., *many orders of magnitude higher*. Similarly, instead of meeting the 99% correct classification requirement, both functions exhibit false negative rates that are consistently above 20%, and in some cases exceed 60%. Importantly, Apple has confirmed our findings for NeuralHash through their coordinated vulnerability disclosure process. These results reinforce the urgent need for transparency: both the design and the testing procedures of perceptual hash functions should be made public, especially given that they are already deployed in practice.

The scale of the inaccuracies we identify has profound implications. Indeed, large-scale deployment of these functions would mean that a large number of innocent citizens would be flagged while the mechanism would be easy to avoid for those who share illegal content. In view of this, we strongly recommend against using NeuralHash, PhotoDNA or similar perceptual hash functions for

this application: the potential for harm and the infringement on privacy far outweigh the intended benefits of such systems.

It remains an open problem whether new perceptual hash functions with longer hash values can be designed that provide a better distribution of outputs for relevant inputs such that they resist black-box attacks. Note that increasing the output size means that more information is provided about the inputs, which increases the risk for preimage attacks or leakage of information about the inputs.

The design of perceptual hash functions that can resist white-box attacks is a much harder open problem, in particular because Client-Side Scanning means that the design cannot be kept secret. Recent research has shown that for the current designs, it is easy to construct false positives and false negatives in a white-box setting.

Even if the accurate and reliable perceptual hash functions were available, the use of Client-Side Scanning remains highly problematic due to the risk of function creep and abuse.

A NeuralHash and PhotoDNA Computation

NeuralHash Computation

The NeuralHash algorithm, as deployed on user devices, consists of two primary components: a CNN and a LSH step. The main steps of the hash computation are described bellow.

The algorithm begins by the preprocessing stage that resizes the image to dimensions ($360 \times 360 \times 3$) within a normalized pixel range of $[-1, 1]$. The resized image is then sent into the embedding CNN.

The CNN produces a vector z of 128 bits. The goal is that for perceptually similar images the vectors z are close and that for perceptually different images they lie far apart.

The LSH step multiplies the vector z of length 128 by a (128×96) matrix B to obtain a real vector y of length 96: $y = B \cdot z$. This step checks the position of the vector z relative to each hyperplane of the matrix B. Each vector is mapped to a specific bucket, with similar vectors (and thus similar images) placed in the same or adjacent buckets.

The final output is a bitstring of length 96. If $y_i > 0$, the corresponding bit is set to 1, otherwise, it is set to 0.

NeuralHash Client-Side Scanning

NeuralHash was intended to work conjointly with CSS. We briefly describe the process presented in Figure 8.

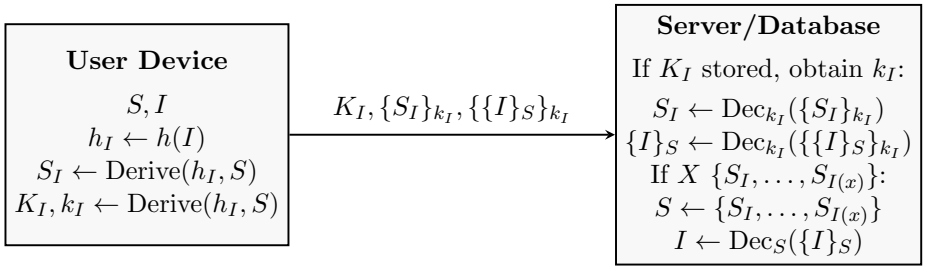


Figure 8: Process designed by Apple to decrypt CSAM images with CSS

First, the user device generates a long term secret key S, that will be used for threshold secret sharing as explained below. For an image I, the device computes the NeuralHash value h_I and derives K_I and k_I from h_I. Using the key S and the hash result h_I, the device computes a secret sharing key S_I.

The client device then sends to the server the value K_I, the encrypted secret sharing key $\{S_I\}_{k_I}$, and the doubly encrypted image $\{\{I\}_S\}_{k_I}$.

The server stores a database of pairs $(K_I,\ k_I)$ for every problematic image. When receiving the values $(K_I, \{S_I\}_{k_I}, \{\{I\}_S\}_{k_I})$, the server checks whether K_I is in its database; if so, it gets the corresponding k_I. With k_I, the server decrypts $\{S_I\}_{k_I}$ to obtain S_I; it also decrypts the first layer of $\{\{I\}_S\}_{k_I}$ and thus retrieves $\{I\}_S$.

If the server collects at least X shares S_I, it can reconstruct the key S. Once S is known, the server can decrypt any image having a hash value present in its database by using S to decrypt $\{I\}_S$.

The threshold X is defined by Apple, but its numerical value is not disclosed. Apple claims [3] that X is chosen to ensure *"an extremely low probability (1 in 1 trillion) of incorrectly flagging a*

given account."

In its technical report, Apple refers to the value we have called K_I as a *"cryptographic header"* derived from h_I. Apple does not provide any information on the size of K_I nor how it is obtained. It also does not specify in its report that the property $K_I \neq K_J,\ \forall\ I \neq J$ is guaranteed. If this property is not guaranteed, then a legitimate image that does not have a NeuralHash value flagged as problematic content can still generate a K_I known by the server and thus be flagged, even if neither the image nor its hash value is flagged. Similarly, a legitimate image with a hash value h_I equal or close to one in the database will lead to the use of a flagged pair $(K_I,\ S_I)$ and thus result in the flagging of a legitimate image.

Computation of PhotoDNA

This section provides a high-level description of the suspected process as presented in [25]. It should be noted that we cannot confirm that the algorithm actually operates in this way, as the internal workings cannot be tested or directly observed.

1. **Normalization:** The input image is first converted to grayscale and, in some cases, possibly resized.
2. **Overlapping Grid Segmentation:** To reduce sensitivity to minor cropping, the 26×26 pixel image is divided into 36 overlapping tiles.
3. **Feature Extraction:** Operations are performed on the obtained grid to extract features from the image. The exact operations applied during this phase remain unknown.
4. **Gradient Computation:** Within each 6×6 grid, gradients are computed based on pixel intensity differences. The gradient values capture directional intensity changes and are computed as follows:
   - The sum of all increasing horizontal pixel values.
   - The sum of all decreasing horizontal pixel values.
   - The sum of all increasing vertical pixel values.
   - The sum of all decreasing vertical pixel values.

   This yields four gradient values per grid, for a total of $36 \times 4 = 144$ values.
5. **Scaling and Equalization:** To ensure uniformity in hash representation, the computed gradients undergo a normalization process in which extreme values are adjusted to fit within the range 0–255. This step accounts for variations in scaling, blurring, and lighting conditions.

Note that the hash result composed of 144 values in the range 0 to 255 can also be represented as a classical hash value of 1152 bits. Moreover, with such a large hash value, the focus is not on exact collisions but rather on near-collisions [40] i.e., hash values that are close but not identical (see Section 5).

B Proof of Propositions 1 to 3

Proof of Proposition 1

Proposition 1. ***(Restated)*** *Consider a hash value of n bits and a vector p for which the i-th element p_i denotes the probability that the i-th bit of the hash value is equal to 1 $(0 \leqslant p_i \leqslant 1)$. If*

E_p^n denotes the event that two hash values are equal, then

$$\mathbb{P}(E_p^n) = \prod_{i=1}^{n} (p_i^2 + (1-p_i)^2)\,.$$

Proof. Let $H_{i,j}^n$ denote the first j bits of the i-th hash value of n bits, and let a_i^n, b_i^n be the first i bits of two given hash values of size n. E_p^n is thus the event that the two strings a_n^n and b_n^n are equal.

We have:

$$\mathbb{P}(E_p^n) = \sum_{h=H_{i,n}^n} \mathbb{P}(a_n^n = h)\mathbb{P}(b_n^n = h) = \sum_{h=H_{i,n}^n} \mathbb{P}(a_n^n = h)^2\,.$$

Considering the sum in detail:

$$\begin{aligned}
\mathbb{P}(E_p^n) &= \sum_{h=H_{i,n}^n} \mathbb{P}(a_n^n = h \mid h(n) = 1)^2 + \sum_{h=H_{i,n}^n} \mathbb{P}(a_n^n = h \mid h(n) = 0)^2 \\
&= p_n^2 \sum_{h=H_{i,n-1}^n} \mathbb{P}(a_{n-1}^n = h)^2 + (1-p_n)^2 \sum_{h=H_{i,n-1}^n} \mathbb{P}(a_{n-1}^n = h)^2 \\
&= (p_n^2 + (1-p_n)^2) \sum_{h=H_{i,n-1}^n} \mathbb{P}(a_{n-1}^n = h)^2\,.
\end{aligned}$$

The result follows by recurrence over the remaining $n-1$ bits.

□

Proof of Proposition 2

Proposition 2. ***(Restated)*** *Denote with $\mathbb{P}(E_p^n)$ the probability that two n-bit hash values with distribution p are equal and define $U = 2^n$. The expected number of hash values to compute before the first collision $\mathbb{E}(D_p^n)$ is equal to:*

$$\begin{aligned}
\mathbb{E}(D_p^n) =& 2 + (U+1) \times y^{\frac{U(U+1)}{2}} + \sum_{x=1}^{U-1} y^{\frac{x(x+1)}{2}} \\
&\textit{with } y = 1 - \mathbb{P}(E_p^n)\,.
\end{aligned}$$

Proof. $\mathbb{P}(E_p^n)$ denotes the probability of a collision between two hash values, and D_p^n denotes the random variable representing the number of hash evaluations required before encountering the first collision. The probability that the first collision occurs after x hash function evaluations is denoted by $\mathbb{P}[D_p^n = x]$.

It is worth noting that $\mathbb{P}[D_p^n < 2] = 0$ and $\mathbb{P}[D_p^n > U+1] = 0$, as there are a maximum of $U = 2^n$ different hash values.

By definition:

$$\sum_{i=1}^{U+1} \mathbb{P}[D_p^n = i] = 1 \quad \texttt{and:} \quad \mathbb{E}(D_p^n) = \sum_{x=2}^{U+1} x \times \mathbb{P}[D_p^n = x]\,.$$

We first define $\mathbb{P}[D_p^n = n]$. The probability $\mathbb{P}[D_p^n = 2]$ is the probability that a collision occurs after 2 hash evaluations, which is simply: $\mathbb{P}[D_p^n = 2] = \mathbb{P}(E_p^n)$. The probability $\mathbb{P}[D_p^n = 3]$ is the

probability that a collision occurs after exactly 3 hash evaluations. This is the probability that the first two hash values are different, multiplied by the probability that the third hash value matches one of the first two. Therefore:

$$\mathbb{P}[D_p^n = 3] = (1 - \mathbb{P}(E_p^n)) \times (1 - (1 - \mathbb{P}(E_p^n))^2)\,.$$

Similarly, the probability $\mathbb{P}[D_p^n = 4]$ equals:

$$\mathbb{P}[D_p^n = 4] = (1 - \mathbb{P}(E_p^n))^3 \times (1 - (1 - \mathbb{P}(E_p^n))^3)\,.$$

This approach generalizes to obtain $\mathbb{P}[D_p^n = n]$, the probability that a collision occurs after exactly n hash operations with $(3 \leq n \leq U + 1)$:

$$\mathbb{P}[D_p^n = n] = \left(\prod_{i=1}^{n-2}(1 - \mathbb{P}(E_p^n))^i\right) \times (1 - (1 - \mathbb{P}(E_p^n))^{n-1})\,,$$

which simplifies to:

$$\mathbb{P}[D_p^n = n] = (1 - \mathbb{P}(E_p^n))^{\sum_{i=1}^{n-2} i} \times (1 - (1 - \mathbb{P}(E_p^n))^{n-1})\,.$$

Expanding the sums, we obtain:

$$\mathbb{P}[D_p^n = n] = (1 - \mathbb{P}(E_p^n))^{\frac{(n-2)(n-1)}{2}} - (1 - \mathbb{P}(E_p^n))^{\frac{(n-1)n}{2}}\,.$$

The expected number of hash values to compute before the first collision is given by $\mathbb{E}(D_p^n)$, which is expressed as:

$$\mathbb{E}(D_p^n) = \sum_{x=2}^{N+1} x \times \mathbb{P}[D_p^n = x]\,,$$

which is equivalent to:

$$\mathbb{E}(D_p^n) = 2 \times \mathbb{P}(E_p^n) + \sum_{x=3}^{N+1} x \times \mathbb{P}[D_p^n = x]\,.$$

Substituting the expression for $\mathbb{P}[D_p^n = x]$ and expanding the sum, we obtain:

$$\begin{aligned}\mathbb{E}(D_p^n) = 2 \times \mathbb{P}(E_p^n) &+ \left(3 \times (1 - \mathbb{P}(E_p^n)) - 3 \times (1 - \mathbb{P}(E_p^n))^3\right) \\ &+ \left(4 \times (1 - \mathbb{P}(E_p^n))^3 - 4 \times (1 - \mathbb{P}(E_p^n))^6\right) \\ &+ \left(5 \times (1 - \mathbb{P}(E_p^n))^6 - 5 \times (1 - \mathbb{P}(E_p^n))^{10}\right) + \ldots \\ &+ \Big((U+1)(1 - \mathbb{P}(E_p^n))^{\frac{(U-1)U}{2}} \\ &\quad -(U+1)(1 - \mathbb{P}(E_p^n))^{\frac{U(U+1)}{2}}\Big)\,.\end{aligned}$$

Reordering the terms, we then obtain:

$$\mathbb{E}(D_p^n) = 2 \times \mathbb{P}(E_p^n) + 2 \times (1 - \mathbb{P}(E_p^n)) + (1 - \mathbb{P}(E_p^n)) + (U+1) \times (1 - \mathbb{P}(E_p^n))^{\frac{U(U+1)}{2}}$$

$$+\sum_{x=4}^{U+1}\left((x+1)\times(1-\mathbb{P}(E_p^n))^{\frac{(x-2)(x-1)}{2}}\ -x\times(1-\mathbb{P}(E_p^n))^{\frac{(x-2)(x-1)}{2}}\right).$$

Simplifying the terms of the sum and shifting the bounds by changing the variable x to $x-2$, we include the term $(1-\mathbb{P}(E_p^n))$ as the first term of the sum:

$$\mathbb{E}(D_p^n)=2+(U+1)\times(1-\mathbb{P}(E_p^n))^{\frac{U(U+1)}{2}}\ +\sum_{x=1}^{U-1}(1-\mathbb{P}(E_p^n))^{\frac{x(x+1)}{2}}.$$

With $y=1-\mathbb{P}(E_p^n)$, we conclude that

$$\mathbb{E}(D_p^n)=2+(U+1)\times y^{\frac{U(U+1)}{2}}+\sum_{x=1}^{U-1}y^{\frac{x(x+1)}{2}}.$$

□

Proof of Proposition 3

Proposition 3. ***(Restated)*** *Denote with $\mathbb{P}(E_p^n)$ the probability that two n-bit hash values with distribution p are equal and define $U=2^n$. The expected number of hash values to compute before the first collision $\mathbb{E}(D_p^n)$ is upper bounded by:*

$$\mathbb{E}(D_p^n)\leq 1+(U+1)\times y^{\frac{U(U+1)}{2}}+\frac{\theta_2(0;y^{\frac{1}{2}})}{2\times y^{\frac{1}{8}}},$$

with $\theta_2(0;y^{\frac{1}{2}})$ the Jacobi theta function $\theta_2(z;q)$ with $z=0$ and $q=y^{\frac{1}{2}}$ and with $y=1-\mathbb{P}(E_p^n)$.

Proof. From Proposition 2 we have:

$$\mathbb{E}(D_p^n)=2+(U+1)\times y^{\frac{U(U+1)}{2}}+\sum_{x=1}^{U-1}y^{\frac{x(x+1)}{2}}.$$

Note that $\sum_{n=1}^{U-1}y^{\frac{n(n+1)}{2}}=\sum_{n=0}^{U-1}y^{\frac{n(n+1)}{2}}-1$. As $\sum_{n=0}^{U-1}y^{\frac{n(n+1)}{2}}\leq\sum_{n=0}^{\infty}y^{\frac{n(n+1)}{2}}$ we thus have:

$$\sum_{n=1}^{U-1}y^{\frac{n(n+1)}{2}}\leq -1+\sum_{n=0}^{\infty}y^{\frac{n(n+1)}{2}}. \tag{4}$$

The Jacobi theta function $\theta_2(z;q)$ is defined as follows:

$$\theta_2(z;q)=2\times q^{\frac{1}{4}}\times\left(\sum_{n=0}^{\infty}q^{n(n+1)}\cos((2n+1)z)\right).$$

Using $z=0$ and $q=y^{\frac{1}{2}}$ we obtain:

$$\theta_2(0;y^{\frac{1}{2}})=2\times y^{\frac{1}{8}}\times\left(\sum_{n=0}^{\infty}y^{\frac{n(n+1)}{2}}\right).$$

By introducing the Jacobi function in the right hand side of Equation (4), we conclude that $\mathbb{E}(D_p^n)$ satisfies

$$\mathbb{E}(D_p^n)\leq 1+(U+1)\times y^{\frac{U(U+1)}{2}}+\frac{\theta_2(0;y^{\frac{1}{2}})}{2\times y^{\frac{1}{8}}}.$$

□

C Statistical Observations on NeuralHash

Distribution of Bit Values

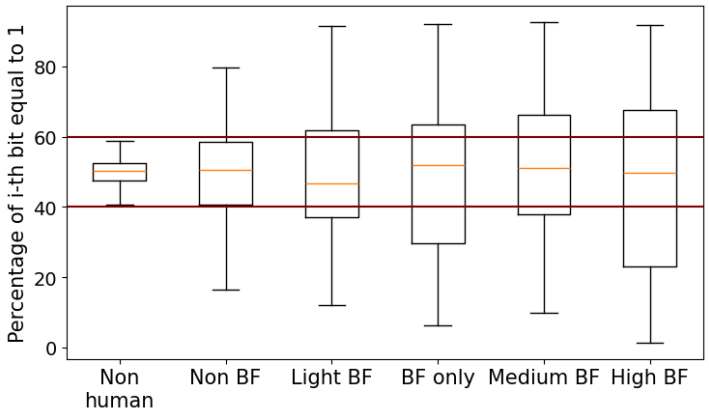
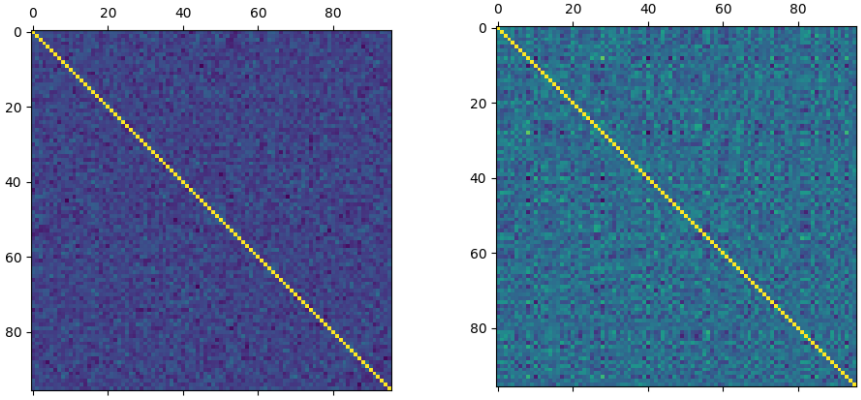


Figure 9: Disparity of bit values for each type of image using NeuralHash

SMM Color Visualization



(a) Non-human (b) Non BF

Figure 10: NeuralHash *SMM* for Non-human and Non Blur faces (Non BF) images

D False Positives for the 2023 NeuralHash Design

The NeuralHash files deployed on macOS devices have been updated around the end of 2023. The new model uses seeds in fp16 and is trained with fp16 precision (instead of fp32 for the previous model). Apple has not provided any official communication regarding this change. This update is surprising as Apple announced in December 2022 that it canceled its plan to scan photos on Apple devices for CSAM.

Previous attacks on NeuralHash were conducted in a white-box setting, meaning the attackers had full knowledge of the model. With this new model, which has not yet been reverse-engineered, it is currently impossible to fully understand or manipulate the inner workings of the algorithm, especially the CNN involved.

However, it is still possible to run NeuralHash using the files present on a device through code available on the GitHub project [59], without detailed knowledge of the internal processes. We applied our collision attacks using this method, and the results in Table 7 are very similar to those obtained in the previous section.

Type	Non human	Non BF	Light BF	BF only	Medium BF	High BF
# illegit. collisions	0	12	50	90	> 500	> 1000

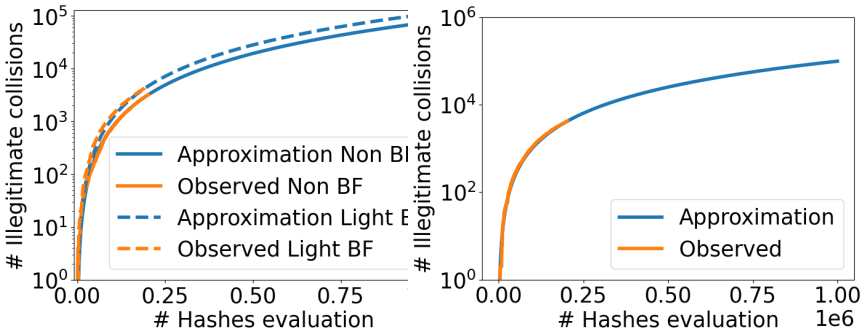
Table 7: False positives for each image type for the fp16 model of NeuralHash

The results indicate that, except for non-blurred face images where the rate of illegitimate collisions decreased, all other types showed a significant increase in illegitimate collision rates, particularly for medium and highly blurred face images. Therefore, the use of this new model does not alter the conclusions. On the contrary, it tends to exacerbate the issues highlighted.

E Approximation and observation of Illegitimate Collision on NeuralHash

For some image types, the number of illegitimate collisions observed on the full 96-bit NeuralHash output is too low to reliably verify that the approximation from Equation (3) matches the experimental results. To address this, we repeat the analysis using only a randomly selected subset of 56 bits from the hash values, which increases the expected number of collisions and allows a meaningful comparison between theory and experiment.

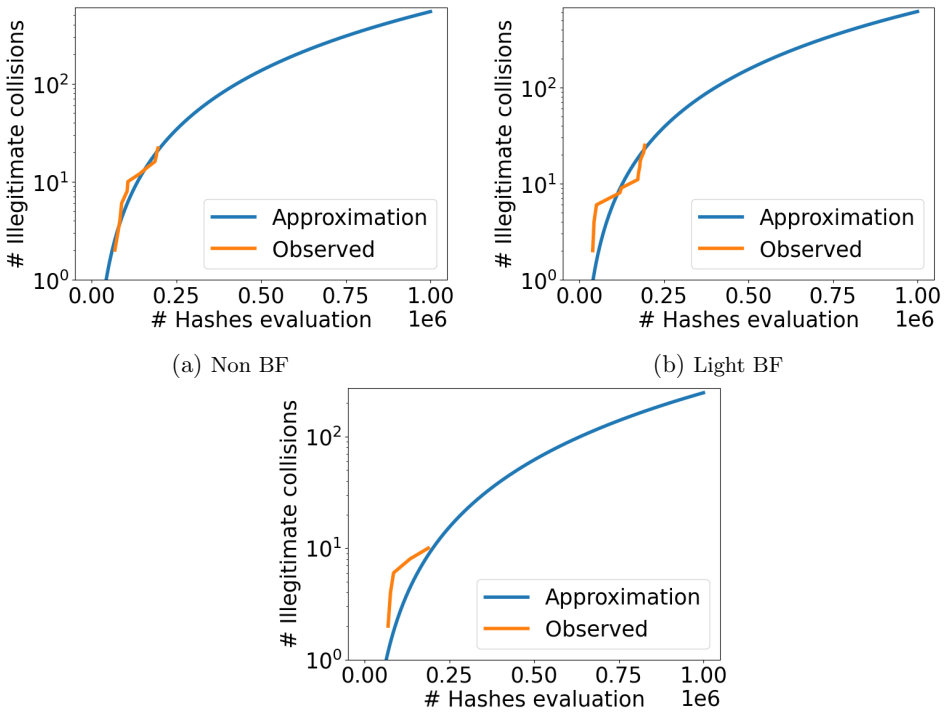
We still report the 96-bit results for completeness, even though the collision counts are insufficient for firm conclusions (Figure 6). The 56-bit verification results, shown in Figure 11, confirm that the observed number of illegitimate collisions closely follows the values predicted by the birthday paradox when assuming a uniformly distributed and independent hash output. This supports the use of the same approximation for the full 96-bit results and for all image types in the subsequent analysis.



(a) Non BF and Light BF

(b) BF only

Figure 11: Observed and aproximated number of illegitimate collisions on randomly selected 56 bits NeuralHash according to the images type



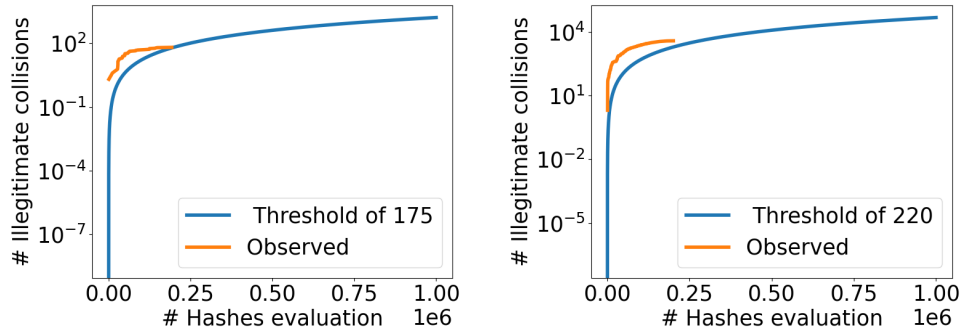
(a) Non BF

(b) Light BF

(c) BF Only

Figure 12: Observed and aproximated number of illegitimate collisions on 96 bits NeuralHash according to the images type

F Approximation and observation of Illegitimate Collision on PhotoDNA



(a) Threshold of 175

(b) Threshold of 220

Figure 13: Approximated and observed number of illegitimate collisions for PhotoDNA with threshold of 175 and threshold of 220

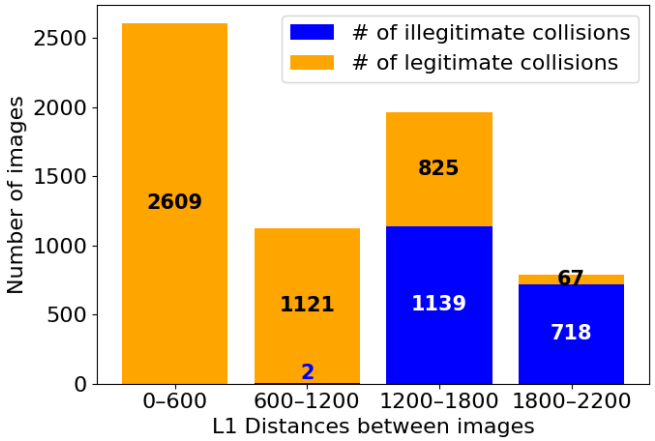


Figure 14: PhotoDNA collision distribution according to L1 distance

References

[1] Harold Abelson, Ross J. Anderson, Steven M. Bellovin, Josh Benaloh, Matt Blaze, Jon Callas, Whitfield Diffie, Susan Landau, Peter G. Neumann, Ronald L. Rivest, Jeffrey I. Schiller, Bruce Schneier, Vanessa Teague, and Carmela Troncoso. Bugs in our pockets: The risks of client-side scanning. *J. Cybersecur.*, 10(1), 2024.

[2] Apple. Expanded Protections for Children, Frequently Asked Questions. https://www.apple.com/child-safety/pdf/Expanded_Protections_for_Children_Frequently_Asked_Questions.pdf, 2021. Accessed on July 8, 2025.

[3] Apple. CSAM Detection – Technical Summary. https://www.apple.com/child-safety/pdf/CSAM_Detection_Technical_Summary.pdf, 2021. Accessed on July 8, 2025.

[4] Yuki M. Asano, Christian Rupprecht, Andrew Zisserman, and Andrea Vedaldi. PASS: An ImageNet replacement for self-supervised pretraining without humans. *NeurIPS Track on Datasets and Benchmarks*, 2021.

[5] Anish Athalye. Inverting PhotoDNA. https://www.anishathalye.com/2021/12/20/inverting-photodna/, 2021. Accessed on July 7, 2025.

[6] Anish Athalye. NeuralHash Collider. https://github.com/anishathalye/neural-hash-collider, 2021. Accessed on July 7, 2025.

[7] Jiawang Bai, Bin Chen, Yiming Li, Dongxian Wu, Weiwei Guo, Shu-Tao Xia, and En-Hui Yang. Targeted attack for deep hashing based retrieval. In *Computer Vision – ECCV – 16th European Conference, Glasgow, UK, Proceedings, Part I*, volume 12346 of *Lecture Notes in Computer Science*, pages 618–634, 2020.

[8] James Bartusek, Sanjam Garg, Abhishek Jain, and Guru-Vamsi Policharla. End-to-end secure messaging with traceability only for illegal content. In *Advances in Cryptology – EUROCRYPT*, volume 14008 of *Lecture Notes in Computer Science*, pages 35–66, 2023.

[9] Mihir Bellare and Tadayoshi Kohno. Hash function balance and its impact on birthday attacks. In *International conference on the theory and applications of cryptographic techniques*, pages 401–418. Springer, 2004.

[10] Abhishek Bhowmick, Dan Boneh, Steve Myers, Kunal Talwar, and Karl Tarbe. The Apple PSI System. https://www.apple.com/child-safety/pdf/Apple_PSI_System_Security_Protocol_and_Analysis.pdf, 2021. Accessed on July 8, 2025.

[11] Patrick Breyer, Alviina Alametsä, Rosa D'Amato, Pernando Barrena, Saskia Bricmont, Antoni Comín, Gwendoline Delbos-Corfield, Francesca Donato, Cornelia Ernst, Claudia Gamon, Markéta Gregorová, Francisco Guerreiro, Svenja Hahn, Irena Joveva, Petra Kammerevert, Marcel Kolaja, Moritz Körner, Karen Melchior, Clara Ponsatí, and Mikuláš Peksa. Cross-party letter of members of the european parliament against general monitoring. https://www.patrick-breyer.de/wp-content/uploads/2021/11/20211020_Letter_General_Monitoring.pdf, 2021. Accessed on July 7, 2025.

[12] Dominique Brunet, Edward R. Vrscay, and Zhou Wang. On the mathematical properties of the structural similarity index. *IEEE Trans. Image Process.*, 21(4):1488–1499, 2012.

[13] Elie Bursztein, Einat Clarke, Michelle DeLaune, David M. Elifff, Nick Hsu, Lindsey Olson, John Shehan, Madhukar Thakur, Kurt Thomas, and Travis Bright. Rethinking the detection of child sexual abuse imagery on the internet. In *The World Wide Web Conference, WWW*, pages 2601–2607. ACM, 2019.

[14] Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy, S&P*, pages 39–57, 2017.

[15] Damon M. Chandler and Sheila S. Hemami. VSNR: A wavelet-based visual signal-to-noise ratio for natural images. *IEEE Trans. Image Process.*, 16(9):2284–2298, 2007.

[16] Council of the European Union. Proposal for a Regulation of the European Parliament and of the Council Laying Down Rules to Prevent and Combat Child Sexual Abuse. https://www.patrick-breyer.de/wp-content/uploads/2024/04/2024-03-28-conseil-csam-compromis-presidence-belge.pdf, 2024. Accessed on July 6, 2025.

[17] Counter Extremism Project. How CEP's eGLYPH Technology Works. https://www.counterextremism.com/video/how-ceps-eglyph-technology-works. Dec 08, 2016. Accessed on July 8, 2025.

[18] Francesco Croce and Matthias Hein. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *Proceedings of the 37th International Conference on Machine Learning, ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 2196–2205, 2020.

[19] Janis Dalins, Campbell Wilson, and Douglas Boudry. PDQ & TMK + PDQF - A Test Drive of Facebook's Perceptual Hashing Algorithms. *CoRR*, abs/1912.07745, 2019.

[20] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 9185–9193, 2018.

[21] Andrea Drmic, Marin Silic, Goran Delac, Klemo Vladimir, and Adrian Satja Kurdija. Evaluating robustness of perceptual image hashing algorithms. In *40th IEEE International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO*, pages 995–1000, 2017.

[22] Ling Du, Anthony T. S. Ho, and Runmin Cong. Perceptual hashing for image authentication: A survey. *Signal Process. Image Commun.*, 81, 2020.

[23] European Commission. Report from the commission to the european parliament and the council on the implementation of regulation (eu) 2021/1232 on a temporary derogation from certain provisions of directive 2002/58/ec as regards the use of technologies by providers of number-independent interpersonal communications services for the processing of personal and other data for the purpose of combating online child sexual abuse. https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:52023DC0797. Accessed on July 22, 2025.

[24] Facebook. The TMK+PDQF Video-Hashing Algorithm and the PDQ Image-Hashing Algorithm. https://github.com/facebook/ThreatExchange/blob/main/hashing/hashing.pdf. 2020. Accessed on July 8, 2025.

[25] Hany Farid. Reining in online abuses. *Technology and Innovation*, 19:593–599, 2018.

[26] Hany Farid. Testimony: Fostering a healthier internet to protect consumers. written statement submitted to the u.s. house committee on energy and commerce. https://www.congress.gov/116/meeting/house/110075/witnesses/HHRG-116-IF16-Wstate-FaridH-20191016.pdf, October 16 2019. Accessed on July 22, 2025.

[27] Hany Farid. An overview of perceptual hashing. *Journal of Online Trust and Safety*, 1(1), Oct. 2021.

[28] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. In *VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases*, pages 518–529, 1999.

[29] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *3rd International Conference on Learning Representations, ICLR*, 2015.

[30] Google Inc. How Content ID works. https://support.google.com/youtube/answer/2797370?hl=en, 2007. Accessed on July 8, 2025.

[31] Qingying Hao, Licheng Luo, Steve T. K. Jan, and Gang Wang. It's not what it looks like: Manipulating perceptual hashing based applications. In *CCS: ACM SIGSAC Conference on Computer and Communications Security*, pages 69–85, 2021.

[32] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing*, pages 604–613, 1998.

[33] Shubham Jain, Ana-Maria Cretu, Antoine Cully, and Yves-Alexandre de Montjoye. Deep perceptual hashing algorithms with hidden dual purpose: When client-side scanning does facial recognition. In *44th IEEE Symposium on Security and Privacy, S&P*, pages 234–252, 2023.

[34] Shubham Jain, Ana-Maria Cretu, and Yves-Alexandre de Montjoye. Adversarial detection avoidance attacks: Evaluating the robustness of perceptual hashing-based client-side scanning. In *31st USENIX Security Symposium*, pages 2317–2334, 2022.

[35] Jan Kaiser. pyPhotoDNA. https://github.com/jankais3r/pyPhotoDNA, 2023. Accessed on Feb 12, 2025.

[36] Yannic Kilcher. Neural Hash Collision Creator. https://github.com/yk/neural_hash_collision, 2021. Accessed on July 7, 2025.

[37] Evan Klinger and David Starkweather. pHash: The Open Source Perceptual Hash Library. https://www.phash.org/, 2010. Accessed on July 22, 2025.

[38] Brian Kulis and Trevor Darrell. Learning to hash with binary reconstructive embeddings. In *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems*, pages 1042–1050, 2009.

[39] Anunay Kulshrestha and Jonathan R. Mayer. Identifying harmful media in end-to-end encrypted communication: Efficient private membership computation. In *30th USENIX Security Symposium*, pages 893–910, 2021.

[40] Mario Lamberger and Elmar Teufl. Memoryless near-collisions, revisited. *Information Processing Letters*, 113(3):60–66, 2013.

[41] Ian Levy and Crispin Robinson. Thoughts on child safety on commodity platforms. https://arxiv.org/abs/2207.09506, 2022.

[42] Venice Erin Liong, Jiwen Lu, Gang Wang, Pierre Moulin, and Jie Zhou. Deep hashing for compact binary codes learning. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 2475–2483, 2015.

[43] Haomiao Liu, Ruiping Wang, Shiguang Shan, and Xilin Chen. Deep supervised hashing for fast image retrieval. *Int. J. Comput. Vis.*, 127(9):1217–1234, 2019.

[44] 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.

[45] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations, ICLR*, 2018.

[46] Meta. EU CSAM Derogation Report 2022. https://transparency.meta.com/sr/eu-csam-derogation-report-2023/, January 2023. Accessed on July 22, 2025.

[47] Meta. EU CSAM Derogation Report 2023. https://transparency.meta.com/sr/eu-csam-derogation-report-2024/, January 2024. Accessed on July 22, 2025.

[48] Meta. EU CSAM Derogation Report 2024. https://transparency.meta.com/sr/eu-csam-derogation-report-2025/, January 2025. Accessed on July 22, 2025.

[49] Microsoft Corporation. PhotoDNA. https://www.microsoft.com/en-us/photodna, 2025. Accessed on July 8, 2025.

[50] NCMEC. Statement Regarding End-to-End Encryption. https://www.missingkids.org/theissues/end-to-end-encryption, 2025.

[51] Joint Statement of Scientists and Researchers on EU's Proposed Child Sexual Abuse Regulation. https://edri.org/wp-content/uploads/2023/07/Open-Letter-CSA-Scientific-community.pdf, 2023. Accessed on July 7, 2025.

[52] Joint Statement of Scientists and Researchers on EU's New Proposal for the Child Sexual Abuse Regulation. https://nce.mpi-sp.org/index.php/s/eqjiKaAw9yYQF87, 2024. Accessed on July 7, 2025.

[53] Nicolas Papernot, Patrick D. McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *IEEE European Symposium on Security and Privacy, EuroS&P*, pages 372–387, 2016.

[54] 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*, pages 211–228, Anaheim, CA, 2023.

[55] Martin Steinebach. An analysis of photodna. In *Proceedings of the ACM 18th International Conference on Availability, Reliability and Security, ARES*, pages 44:1–44:8, 2023.

[56] Lukas Struppek, Dominik Hintersdorf, Daniel Neider, and Kristian Kersting. Learning to break deep perceptual hashing: The use case neuralhash. In *FAccT: ACM Conference on Fairness, Accountability, and Transparency*, pages 58–69, 2022.

[57] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Trans. Evol. Comput.*, 23(5):828–841, 2019.

[58] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *2nd International Conference on Learning Representations, ICLR*, 2014.

[59] Khaos Tian and Malcolm Hall. nhcalc. Compute NeuralHash for a given image. https://github.com/KhaosT/nhcalc, 2021. Accessed on July 23, 2025.

[60] UK Department for Science, Innovation & Technology. Online safety act: Explainer. https://www.gov.uk/government/publications/online-safety-act-explainer/online-safety-act-explainer, 8 May 2024.

[61] Xunguang Wang, Zheng Zhang, Guangming Lu, and Yong Xu. Targeted attack and defense for deep hashing. In *The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2298–2302, 2021.

[62] Xunguang Wang, Zheng Zhang, Baoyuan Wu, Fumin Shen, and Guangming Lu. Prototype-supervised adversarial network for targeted attack of deep hashing. In *IEEE CVPR*, pages 16357–16366, 2021.

[63] Yongwei Wang, Hamid Palangi, Z. Jane Wang, and Haoqian Wang. Revhashnet: Perceptually de-hashing real-valued image hashes for similarity retrieval. *Signal Process. Image Commun.*, 68:68–75, 2018.

[64] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. Image Process.*, 13(4):600–612, 2004.

[65] Dayan Wu, Zheng Lin, Bo Li, Mingzhen Ye, and Weiping Wang. Deep supervised hashing for multi-label and large-scale image retrieval. In *Proceedings of the ACM on International Conference on Multimedia Retrieval, ICMR*, pages 150–158, 2017.

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

[67] Asuhariet Ygvar. AppleNeuralHash2ONNX. Convert Apple NeuralHash model for CSAM Detection to ONNX. https://github.com/AsuharietYgvar/AppleNeuralHash2ONNX, Aug 2021. Accessed on July 12, 2025.

[68] Christoph Zauner. Implementation and Benchmarking of Perceptual Image Hash Functions, MsC Thesis, University of Applied Sciences, Hagenberg, Austria, 2010.

[69] Fang Zhao, Yongzhen Huang, Liang Wang, and Tieniu Tan. Deep semantic ranking based hashing for multi-label image retrieval. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 1556–1564, 2015.

[70] Mingkang Zhu, Tianlong Chen, and Zhangyang Wang. Sparse and imperceptible adversarial attack via a homotopy algorithm. In *Proceedings of the 38th International Conference on Machine Learning, ICML*, volume 139 of *Proceedings of Machine Learning Research*, pages 12868–12877, 2021.