6.857 Final Project: Security Analysis and Extension of the Apple NeuralHash

Nithya Attaluri
MIT
nithyaa@mit.edu

Jagdeep Bhatia
MIT
jagdeep@mit.edu

Kevin Meng
MIT
mengk@mit.edu

Albert Xing
MIT
axing@mit.edu

May 11, 2022

Contents

1 Introduction

In 2021, Apple unveiled NeuralHash [Inc21], a perceptual hashing system which aims to detect Child Sexual Abuse Material (CSAM) on images that are uploaded to iCloud. The fundamental problem that NeuralHash attempts to solve is one of great interest to the worldwide community: how can we efficiently surveil for various types of abuse *without* compromising user privacy?

Unfortunately, NeuralHash has been controversial since its release, due to concerns about its efficacy and about user privacy. These concerns have been made concrete, with recent work identifying problems with NeuralHash's collision resistance, privacy guarantees, and false positive/negative rates under adversarial attacks [Str+21; Ygv21; AI21; Dwy21].

Many issues with NeuralHash arise from its usage of deep learning as the mechanism of hashing. Neural networks are employed because fuzzy image matching is extremely difficult even with rule-based or classical learning algorithms [DS13; CPZ08], but deep networks are also known to be sensitive to edge cases, susceptible to adversarial attacks, and prone to unexplainable behavior [GSS14]. In this project, we use these intuitions and background about deep learning to investigate NeuralHash's vulnerabilities and discover potential improvements to the system.

In this paper, we first describe the NeuralHash system (Section 2), including its technical formulation and security goals. Then, in Section 3, we describe the vulnerabilities of NeuralHash that have already been discovered. In Section 4 we describe in detail a variety of attacks that we implemented, and demonstrate how they reveal the shortcomings of neural hash and break several of its key security goals. Finally, we propose and implement several ideas which we believe will improve the system in Section 5.

Our main contributions are summarized as follows:

- A gradient-based evasion attack with stronger semantic guarantees than previous methods.
- A Generative Adversarial Network (GAN)-based collision attack that generates highly realistic colliding images.
- Discovery of the approximate linearity of NeuralHash, leading to the following attacks:
  - Novel interpolation evasion attacks that are extremely fast and realistic. Gradients not required.
  - Novel collision generation attacks that *also* perform well as an inverter that reveals surprising amounts of information from NeuralHash.
- An information extraction attack that better reveals information from perceptual hashes compared to prior approaches and uses less data.
- Proposed changes to the architecture as well as adversarial input detection to safeguard against some of the above attacks along with experiments to validate their effectiveness.

2 NeuralHash Background

2.1 Functional Specification

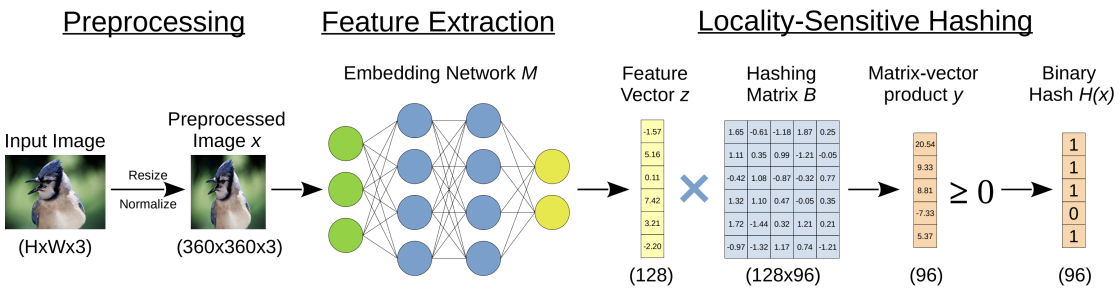

Figure 1: NeuralHash Pipeline. Consists of a preprocessing step that transforms images into a consistent format. This is followed by an embedding network step and a Locality-Sensitive Hashing step. The output vector from the embedding network is compared with the hyperplanes in the hashing matrix to produce the output binary hash. [Str+21]

Apple's NeuralHash CSAM Detection scheme, released with iOS 15, allows automatic detection of Child Sexual Abuse Material (CSAM) in photos that are uploaded to Apple's iCloud. This system depends on three major systems, which we will discuss here:

- NeuralHash
- Private Set Intersection
- Threshold Secret Sharing

NeuralHash is a perceptual hashing system. In contrast with cryptographic hashing, NeuralHash values for images preserve perceptual and semantic similarity. These hashes are generated in two steps. First, a neural network generates a floating-point vector "descriptor" that is a compressed representation of all of the key information in the image. Second, a technique known as Hyperplane Local Sensitivity Hashing [Con22] converts these real-valued vectors into integer-valued vectors.

Private Set Intersection (PSI) [BBM21] is a communication protocol used between user's devices and Apple's servers. The PSI protocol is used to preserve user privacy by ensuring that Apple is unable to learn about users' images. It also creates a secure communication protocol for Apple to determine on the server-side if a user's photos match with existing CSAM images; this prevents adversarial attacks that would arise if users were able to easily determine which images would be flagged as CSAM and which would not.

Finally, Threshold Secret Sharing prevents Apple from learning the encryption key specific to the user until the user has crossed a specific threshold for their number of matches with known CSAM content. After crossing this threshold, the user is able to decrypt information describing the specific CSAM content that the user's images matched with.

The neural network-based hashing algorithm will be at the core of our investigation. Deep networks are excellent fuzzy matchers, but they often suffer from unpredictable behavior, for example in the case of adversarial attacks. It is difficult to *prove* their properties, which makes it all-the-more important that we empirically analyze their behaviors carefully, especially under adversarial conditions.

2.2 Idealized Security Goals for NeuralHash

Here, we define our idealized desiderata for NeuralHash by examining all parties involved and describing a concrete security policy.

The parties involved with NeuralHash are Apple with their servers and individual users with their devices and data. The primary objective of NeuralHash is to protect the privacy of user data. Specifically, Apple should not be able to get any information about the semantic content of any images on a user's device. The only exception to this rule is that Apple should be able to query the user's device and get a boolean answer to the following question: are there at least 30 instances of CSAM on a user's device? Apple should be able to run this query at any time.

Some more nuanced desired functionalities are as follows. In order to reduce the number of adversarial attacks to the NeuralHash system, users should not know when they have a 'match' with CSAM images in apple's database. Users may know when Apple runs a query on their device, but they should not be aware of the result of the query. This serves as a prevention measure for adversarial behavior. Finally, another desired property to reduce adversarial attacks is that even with public knowledge of all parts of the NeuralHash system, it should be impossible (or at the very least, extremely computationally difficult) for an adversary to artificially construct images that result in false positives or false negatives to Apple's query.

This last desired functionality is one of the most difficult to guarantee, but is also one of the most important. Without this assumption, it may be possible for an adversary to hide CSAM on their device without Apple's knowledge. It may also be possible for an adversary to modify images on an innocent person's device in such a way that Apple falsely detects CSAM.

Clearly, Apple is making a lot of promises with their NeuralHash system. However, Apple uses deep learning, which traditionally suffers catastrophically from adversarial attacks, edge cases, and lack of robustness. This is why we conjecture that Apple's system does not match all the desired guarantees.

2.3 Obtaining Authorization

There are publicly released directions on how to access Apple's NeuralHash models which can be found in releases of iOS 15 and above. For instance, see this github repo. Since we will be doing all testing on our local machines, we will not be exceeding our authorized access by working with these models.

3 Related Work

3.1 Non-Neural Fuzzy Image Matching Algorithms

Fundamentally, NeuralHash aims to be a cryptographically-secure fuzzy matching algorithm, matching user images with a large database while invariant under various kinds of transformations: blurring, cropping, rotation, stretching, shearing, etc. While NeuralHash is driven by deep learning at its core, other approaches to this problem exist. One approach [DS13] applies F-Transforms to compute similarity measures between images, while another [CPZ08] combines the minHash algorithm with TF-IDF weighting to generate similarity features.

3.2 Failure Modes of the NeuralHash System

Following NeuralHash's public announcement, various attacks were proposed to defeat its security desiderata (Section 2.2). Broadly, researchers found issues with (i) collision resistance, (ii) computational distinguishability between different perceptual hashes, (iii) adversarial attacks that artificially induced false positives and negatives, and (iv) non-robustness to basic image transformations.

i. Collisions: [Dwy21; AI21] The system was designed to produce collisions on images that are the same, modulo perturbations due to JPEG encoding, resizing, or watermarks, while avoiding collisions for different images. However, naturally occurring collisions of different images have been found, where two images of different objects produce identical hashes even without artificial modification of the images themselves.

ii. Computational Distinguishability: [Str+21] A recent study showed that the cosine distance between hashes correlates with the strength of the semantic relationship between the original images. In other words, an adversary that gains access to the hashes of images can gain information about the original images; this could potentially be a serious security flaw.

iii. Adversarial Attacks: [Str+21; GSS14] Additionally, the weights of the NeuralHash model have been extracted and made public, allowing for the artificial modification of images to produce desired hashes. Oftentimes, this artificial modification is not easily visually discernible. This has opened the door to attacks like transforming innocuous photos into those labelled as CSAM, or vice versa.

iv. Black-Box Attacks: [Str+21] In addition, certain black box attacks have been shown to be effective at changing the output of NeuralHash, such as basic image transformations like translation, rotation, flipping, cropping, and brightness and contrast changes.

3.3 Our Contributions

In this paper, we improve on SOTA gradient-based attacks and provide stronger semantic guarantees than previous methods. We also propose an entirely novel class of powerful, black-box attacks

using image interpolation. Finally, we propose and validate changes to the architecture as well as adversarial input detection to safeguard against some of the above attacks.

4 Attacks

In this section we provide a technical overview of the attacks we propose and find successful in breaking a security goal of the NeuralHash system. In section 4.1 we formally describe the security goals we break with our attacks and also describe some notation that we use throughout. In sections 4.2 and 4.3, we describe the two main classes of attacks we propose: those that use the gradients of the model and therefore require full model access, and black-box attacks that only require seeing model outputs. For each attack that we propose, we explain what prior work has been done in attempting this type of attack, what our approach was, why our approach was reasonable, and the results from performing the attack.

4.1 Formal Security Definitions

In all of our attacks we primarily focus on breaking the perceptual hashing algorithm proposed by Apple's NeuralHash system. Such a perceptual hashing algorithm should have the following three properties:

1. Let x_1 and x_2 be any semantically similar images and $\mathcal{N}(x)$ be the NeuralHash perceptual hashing algorithm which maps images space $X \to \{0,1\}^{96}$. It is the case that $\mathcal{N}(x_1) = \mathcal{N}(x_2)$. We call this the **neighborhood property** of $\mathcal{N}(x)$.
2. Let x_1 and x_2 be any images that are not semantically similar and $\mathcal{N}(x)$ be the NeuralHash perceptual hashing algorithm which maps images space $X \to \{0,1\}^{96}$. It is the case that first, $\mathcal{N}(x_1) \neq \mathcal{N}(x_2)$, and second, the distributions of $\mathcal{N}(x_1)$, $\mathcal{N}(x_2)$ over $\{0,1\}^{96}$ look indistinguishable to a computationally bounded adversary (the hashes look like random elements of the hash space). We call this the **uniform hashing property** of $\mathcal{N}(x)$.
3. Given target hash h^*, it should be impossible for a computationally bounded adversary to generate any image x such that $\mathcal{N}(x) = h^*$. We call this the **noninvertibility property** of $\mathcal{N}(x)$, which extends the uniform hashing property.

These security properties lend themselves naturally to three main attacks:

1. Given image x, find semantically similar image x' such that $\mathcal{N}(x) \neq \mathcal{N}(x')$. This **evasion attack** breaks the neighborhood property of $\mathcal{N}(x)$.
2. Given image x, find any image x' (semantically similar to x or not), such that $\mathcal{N}(x) = \mathcal{N}(x')$. This **collision-generation attack** breaks both the uniform hashing and noninvertibility properties of $\mathcal{N}(x)$.
3. Given image x which belongs to one of n known image classes $\{C_1, C_2, ..., C_n\}$, predict target class C_t which probability negligibly greater than $\frac{1}{n}$. This **information-leakage attack** breaks the uniform hashing property of $\mathcal{N}(x)$.

We show the implementation of these attacks in the following sections. In order to compare the similarity between images, we use a cosine distance metric. That is, given two 96-bit hashes h_1, h_2, we treat them as 96-element vectors and compute

$$\cos(h_1, h_2) = \frac{h_1 \cdot h_2}{|h_1||h_2|} = 1 - \text{Hamming Distance}(\text{h}_1, \text{h}_2).$$

4.2 Gradient-Based Attacks

In this subsection we describe gradient-based attacks for both evasion and collision generation, which require access to the full NeuralHash model and its gradients.

Key takeaways:

- **Evading detection is strikingly easy.**
- **Generation of realistic collision is possible with a Generative Adversarial Network (GAN).**

4.2.1 Gradient-Based Evasion Attack

Attack Description. Our evasion attack exploits the sensitivity of neural networks to small amounts of adversarial noise; [GSS14] we compute a Δx such that $\mathcal{N}(x) \neq \mathcal{N}(x + \Delta x)$ using projected gradient descent:

$$\delta(\Delta x)'_i := \nabla_{(\Delta x)_i} \left(\mathcal{N}(x_0 + (\Delta x)_i) - \mathcal{N}(x_0)\right)^2 \qquad \text{unconstrained optim} \tag{1}$$

$$\delta(\Delta x)_i := \delta(\Delta x)'_i \cdot \frac{\min(\epsilon, \|\delta(\Delta x)'_i\|)}{\|\Delta\delta(\Delta x)'_i\|} \qquad \text{projection within } L_2 \text{ ball} \tag{2}$$

$$(\Delta x)_{i+1} := (\Delta x)_i + \alpha\delta(\Delta x)_i, \tag{3}$$

where $(\Delta x)_i$ is the computed delta image after i steps of corruption, α is the learning rate, and ϵ is the maximum L_2 norm of Δx. Finally, $x' = x + \Delta x$ is our adversarial image.

Comparison to Prior Work. [Str+21; Ath21] have attempted gradient-based evasion attacks, but to the best of our knowledge, their approaches all involve *soft* penalties on Δx, e.g. adding a weight decay term to the loss. This may permit large changes to the image, depending on the whim of the optimizer. By contrast, we *explicitly* constrain Δx within an L_2 ball using a simple projection (Eqn. 2), which guarantees minimal disruption of semantic content.

Attack Results. We consider an attack "successful" iff (i) $\mathcal{N}(x) \neq \mathcal{N}(x + \Delta x)$ and (ii) x and $x + \Delta x$ contain the same semantic information. To formalize the latter concept, we adopt the Structural Similarity Index Measure (SSIM) metric:

$$\frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2)(\sigma_x^2 + \sigma_y^2 + c_2)},$$

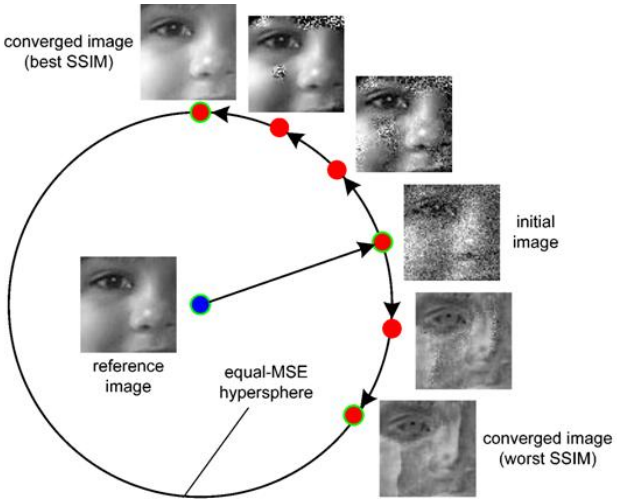

Figure 2: The space of images with the same MSE to the reference image consists of many images of varying degrees of structural similarity. Image from https://www.cns.nyu.edu/~lcv/ssim/.

where x and y are images with means μ_x, μ_y, variances σ_x^2, σ_y^2, and covariance σ_{xy}. The intuition is that by accounting for mean, variance, and covariance in the pixel space, SSIM focuses on *structure* much better than e.g. Mean-Squared Error (MSE), which can be distracted by noise.

Running evaluations on a subset of ImageNet called ImageNette [Fas19], we find that this attack method succeeds 100% of the time with average SSIM similarity of 99.6% (± 0.003 error in a 95% confidence interval). The PGD optimization of Δx is very quick, taking < 1 second each on a MacBook CPU.

We show a qualitative example of an adversarial attack below. While each pair of images looks identical to the human eye, NeuralHash fails to hash them to the same value.

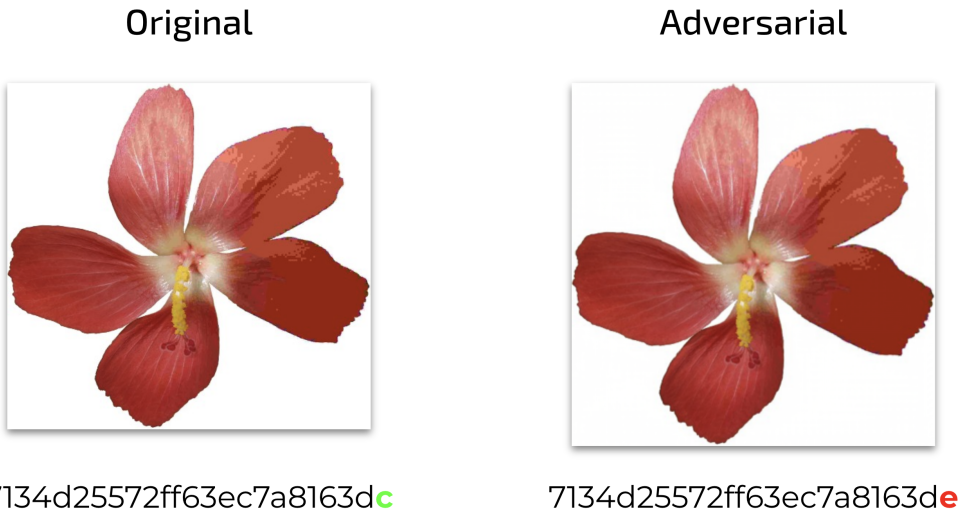

Figure 3: The two images are almost visually identical but have different hashes.

We have successfully broken the neighborhood property (Section 4.1) of NeuralHash. The ease

with which we can alter hashes and have virtually no perceptible impact on semantic content is striking.

4.2.2 GAN-Based Collision Generation Attack

Related Attacks. There have been many prior examples of gradient based collision attacks. For example, it has been shown that from random noise it is possible to generate an image that collides with a target hash [Xu+19]. It has also been shown that it is possible to generate an image that collides with a target hash, starting with a source image [Ath21]. However, one limitation with both these approaches is that the final image produced does not look very realistic – the images are often shapeless or littered with random splotches of colors. In some applications it might actually be desirable to produce realistic images that result in a hash collision. Another motivation for restricting the domain of images to those that are 'realistic' or 'natural', is that this process may actually reveal more information about the semantic contents of the image from which the target has was generated.

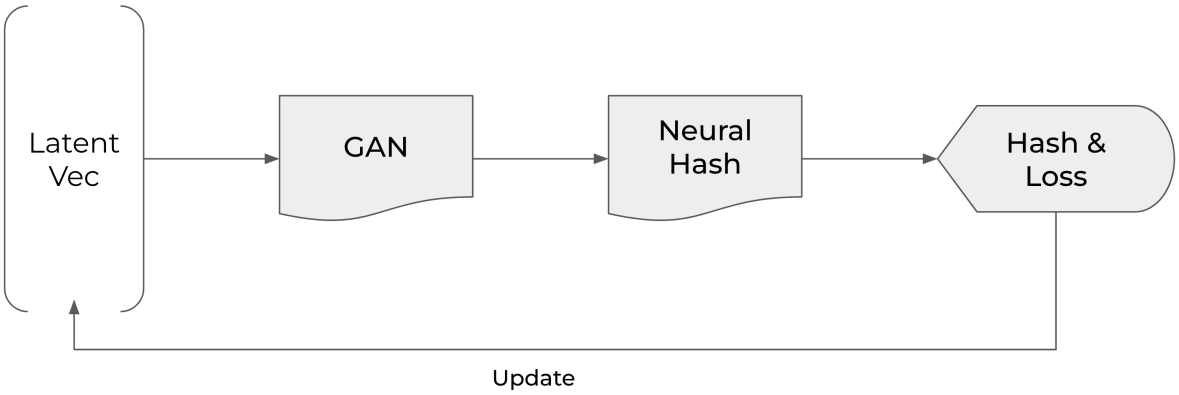

Figure 4: Pipeline for our GAN collision attack.

Attack Description. Recall that in a collision generation attack, given image x, we wish to find any image x' such that $\mathcal{N}(x) = h^* = \mathcal{N}(x')$. Our high level approach for this attack is described in Figure 4. Most notably, in order to restrict the domain of generated images to those that are realistic, we use a GAN, or Generative Adversarial Network. We can treat the GAN as a differentiable function that takes in latent vector of parameters l and returns $G(l)$, an image from the domain of realistic looking images. We let our guess image $x' = G(l)$. Since l is initialized randomly, $\mathcal{N}(x') \neq h^*$ with high probability. However, we can iteratively update l' in order to produce x' with the desired properties. We define a loss function over the differences in the resulting hashes

$$L(l, h^*) = (\mathcal{N}(G(l)) - h^*)^2,$$

and iteratively update l using stochastic gradient descent.

Some additional information about our approach is as follows. We used a pretrained GAN trained on the 'celebAHQ-512' dataset of celebrity faces, and as such, our source image x is also an image

of a face. Another detail is that in the loss function we do not operate on the 96-bit hash value, but rather a pre-hash value that is much more differentiable.

Figure 5: We can generate "realistic" looking images that collide with a known hash by generating an image with a close hash, and then using gradient-descent methods to add noise that produces the desired hash.

Attack Results. The results from this approach were quite promising and can be seen in Figure 5. The source image is shown on the left, the image resulting from the optimization process described in this section is shown in the middle, and the rightmost column contains a colliding image. Note that the images produced by the GAN had hashes with only a few bits differing from the target. A small amount of adversarial noise (using the method described in section 4.3.1 using an inverted loss function) was all that was required in order to find a hash collision. While generating collisions was very easy (a few seconds with a decent gpu), the amount of semantic information revealed about the source image from the target hash was not as high as we would have hoped. Partly, this is due to the fact that the GAN we used was poorly trained and suffered from mode collapse – it kept producing images of white women regardless of input. It is unclear whether a better model would yield better results, or if the domain of images even when restricted to realistic faces is still too combinatorially large to expect the target hash to real information about the source.

4.3 Black-Box Attacks

In this subsection we describe gradient-based attacks for evasion, collision generation, and information leakage all of which only require black-box access to the NeuralHash model.

Key takeaways:

- **NeuralHash is approximately linear!**
- **Approximate linearity can be exploited for black-box evasion, collision.**
- **NeuralHash leaks information; NOT private.**

4.3.1 Interpolation Evasion Attack

Attack Description. This attack and the attack in the next section make use of interpolation, which is is a novel concept when applied to neural hash. We describe this process formally below:

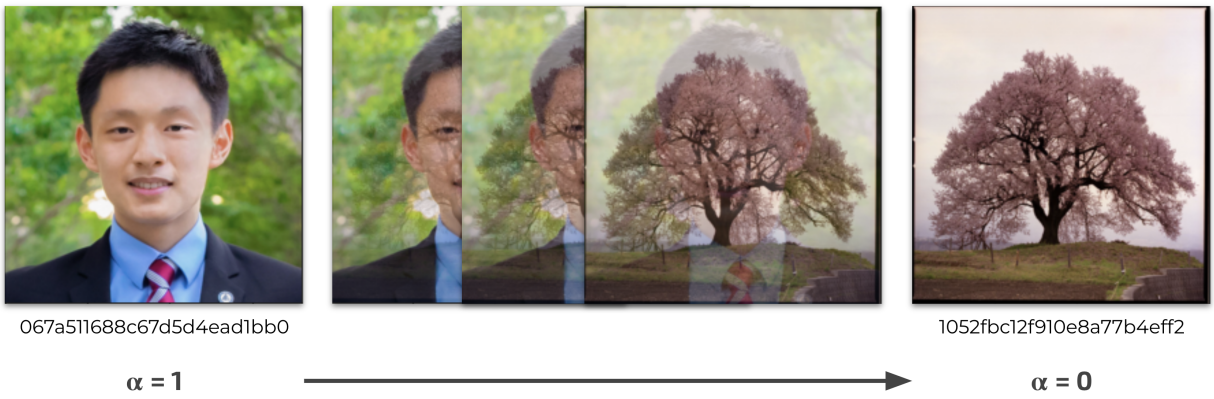

Figure 6: Image produced by interpolation of a face and a tree, with parameter α denoting the proportion of the first image.

Consider taking two source images x_1 and x_2 and linearly interpolating them with respect to some parameter α. Let the resulting image be

$$I_\alpha = \alpha x_1 + (1 - \alpha)x_2.$$

To motivate our approach, we visualize how the hash of the interpolated image changes with respect to the hash of x_1 and x_2 and different values of α.

Figure 7: This graph demonstrate the approximately piece-wise linear property of neural hash. The graphed orange and blue lines show the cosine distance between the interpolated image at different levels of α with respect to the image of Kevin and the Tree, respectively. The green line shows the cosine distance between adjacent interpolated images.

We see from the graphs in Figure 7 that as alpha changes, the cosine distance of I_α from the original images x_1 and x_2 changes nearly *piecewise linearly*. This is extremely strong evidence that Apple's perceptual hashing model is not meeting its security goals. If the algorithm was in fact meeting the security goals, then we would expect the orange, blue lines to change violently between 1 and roughly 0.5. This is because images that are in the same neighborhood of each other will have the same hash (and therefore a cos distance of 1. Outside the neighborhood function the hash should be a random element of the hash space and the average cosine distance should be 0.5.

We abuse this approximate piecewise linear property of neural hash in our attacks. Recall that the goal of an evasion attack is, given source x, find semantically similar image x' such that $\mathcal{N}(x) \neq \mathcal{N}(x')$. We simply let $x' = I_\alpha$ for the largest possible alpha (ie. alpha closest to 1) such that $\mathcal{N}(x) \neq \mathcal{N}(I_\alpha)$.

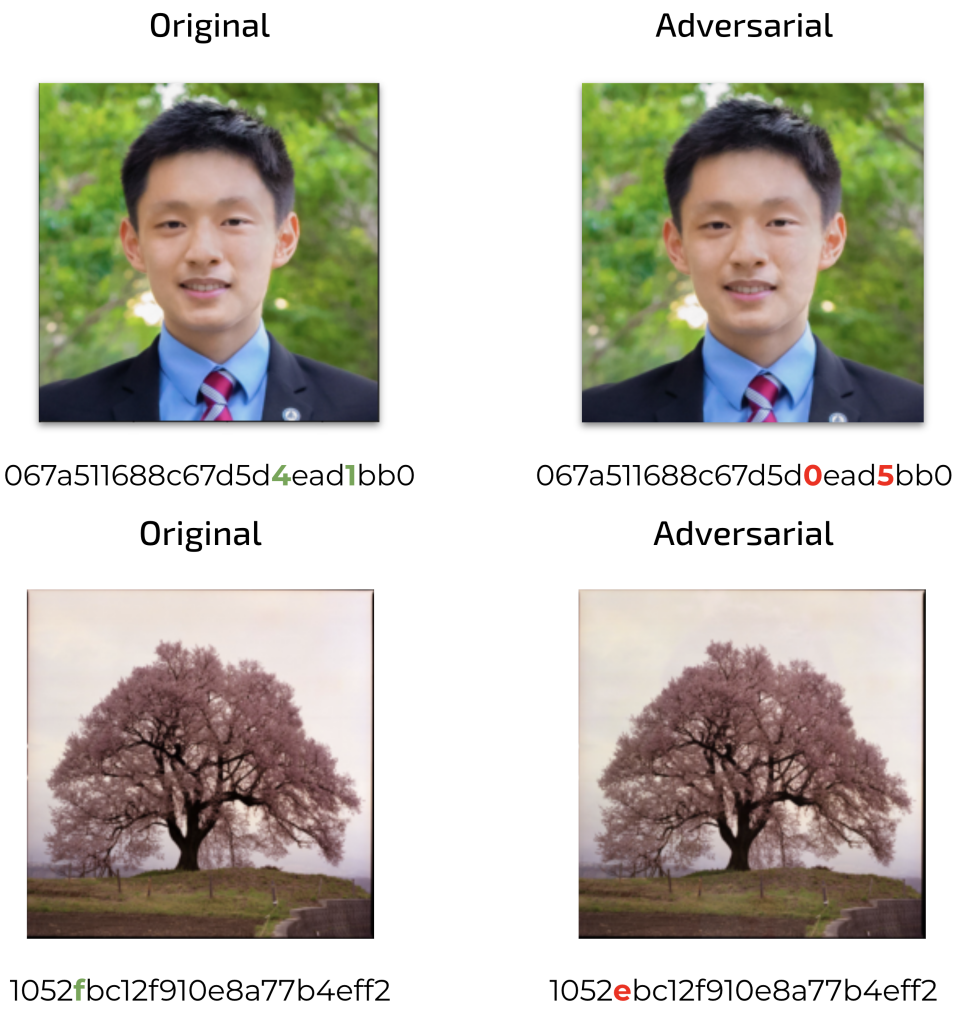

Figure 8: Adversarial images with different hashes generated via interpolation.

Attack Results. This attack works quite well as can be seen in Figure 8. Not only do the adversarial images look remarkably similar to the source, but this attack is extremely fast and does not require any knowledge of model gradients. Additionally, it requires very limited knowledge of model outputs.

4.3.2 Interpolation Collision Generation Attack

Attack Description. The previous subsection seems to suggest that NeuralHash is approximately linear. Can we exploit this property to generate collisions? Consider a source image x and corresponding target hash $h^* = \mathcal{N}(x)$. Also consider a database of images $\{x_1, x_2, .., x_k\}$ and their known hashes $\{h_1, h_2, .., h_k\}$. Let $\{p_1, p_2, .., p_k\} \in [-1, 1]^k$ be some parameters such that

$$\sum_{i=1}^{k} |p_i| = 1.$$

The motivation behind this attack is that if we could find a setting of parameters $\{p_1, p_2, .., p_k\}$ such that

$$\sum_{i=1}^{k} p_i h_i = h^*$$

then the interpolation property of the NeuralHash model would imply that

$$\mathcal{N}\left(\sum_{i=1}^{k} p_i x_i\right) \approx h^*.$$

We test this hypothesis with experiments. Specifically, we attempt to find parameters $\{p_1, p_2, .., p_k\}$ that minimize the following objective function:

$$L(p_1, p_2, .., p_k, h^*) = \left(\sum_{i=1}^{k} p_i h_i - h^*\right)^2 - \sum_{i=1}^{k} p_i \log(p_i).$$

Note that the second term here aims to reduce entropy or, in other words, use the fewest number of images possible to achieve the objective. $L(\cdot)$ is optimized using gradient descent.

Some additional information about our approach is as follows. Our dataset is made up of 450 images scraped from Bing from searching the query: `tree`. Another detail is that in the loss function we do not operate on the 96-bit hash value, but rather a pre-hash value that is much more expressive. Finally, our loss function makes uses of weights to balance the different losses, but we have omitted the details here for the sake of clarity.

Attack Results. Our results, seen in Figure 9 are quite impressive. While we are not directly able to find a perfect colliding image, the cosine distance of the resulting images we find is several standard deviations above the mean of 0.5 (standard deviation is roughly 0.05), meaning that this method performs much better than simply random guessing or random generation of images. Another interesting property that emerged is that the image x_i that corresponded to parameter p_i with the largest magnitude, was often an image in the dataset that was very semantically similar to the source image. These results are very exciting, and we believe that with an even larger dataset (with images in the hundreds of millions) this approach could be groundbreaking.

4.3.3 Information Extraction Attack

In this subsection we describe our information extraction attacks. The rationale behind this attack comes from the fact that NeuralHash is a perceptual hashing algorithm. Unlike cryptographic

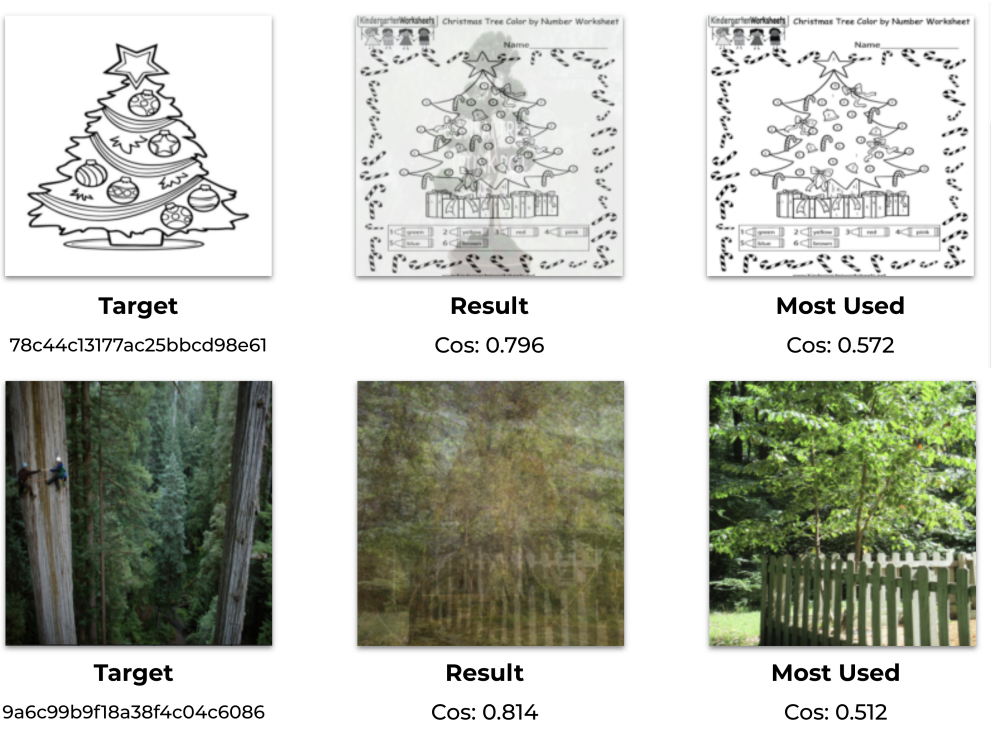

Figure 9: The two images are almost visually identical but have different hashes.

hashes, this means that images that are similar should hash closer together than images that are not similar. Because of this property, we hypothesized that image hashes must leak *some* information about their images. This property may not be apparent for a single image, but it may be more obvious over a distribution of images.

To run these attacks, we used images from ImageNet. Specifically, the 50,000 validation set images from the ImageNet Large Scale Visual Recognition Challenge 2010 (ILSVRC2010). There are 1000 ImageNet categories, and the validation set contains 50 images per category.

Attempt 1: Cosine Similarity

Attack (v1) Description. In our first attempt at this attack, we attempted to use cosine similarity between a brand-new hash (turned into a 96-bit vector) and a lookup table of hashes (also vectorized) for known images to glean more information about the image. Cosine similarity is a well-known metric to compare similarity of vectors. It is also proportional to dot-product and Euclidean distance.

The details of this experiment are as follows. We calculated the hashes of the 50,000 images in our dataset. Next, for a randomly chosen image x in each ImageNet category, we calculated the cosine similarity distance between x and every other image in the dataset. We used these to calculate the average cosine similarity distance between x and each of the 1000 ImageNet categories (the average was adjusted to avoid including the similarity between x and itself for the category that x belonged to). We then turned the cosine similarity data (where s_i is the similarity for category i) into a probability p_i for each category as follows:

$$p_i = \frac{s_i}{\sum_{i=1}^{1000} s_i}$$

Attack (v1) Results. Unfortunately (and interestingly) we observed that the probabilities of the target categories for each of the randomly chosen images was consistently close to 0.001, which is the probability for a category given uniform guessing. Specifically, for all of the chosen images, the average probability of the target category was 0.00100. The minimum and maximum probabilities over all of the images were respectively 0.00089 and 0.00111.

Given our results, we hypothesized that our cosine similarity distance metric was too simple to extract information from image hashes. We pivoted to a second attack: using a neural network that takes image hashes as inputs and outputs the image's ImageNet category.

Attempt 2: Neural Network

Connection to Prior Work. Others have done similar analyses with neural networks to measure information leakage of NeuralHash hashes. Specifically, Struppek et al. uses a fully-connected network with 3 hidden layers to predict ImageNet category from an image hash [Str+21]. Struppek et al report that their models show a top-5 accuracy of 12.03%, when they train on around 83,000 images. Top-5 accuracy measures the percentage of cases in which the target category is predicted as one of the top-5 results of the model.

However, we claim that prior work takes a very machine-learning-based approach to quantify information leakage from a hash. For instance, what if the target category of an input image is "dog" and the top-5 categories include "wolf" and "cat." Despite "dog" not being present among the top-5, we should assign weight to the fact that the model has likely extracted some information from the hash indicating that the image represents an animal.

We do this using the ImageNet class hierarchy. While there are 1000 "leaf" classes of ImageNet — highly specific categories that are the most specific description of an image — there are also "parent" classes that provide higher level categorizations. For instance, class ID 1153, "butterfly," is a parent class to class ID 600, "cicada." Using this information, we create a tree of the ImageNet hierarchy and use the closest ancestor of any two classes to calculate a "distance" between them.

Attack (v2) Description. Our neural network outputs probabilities for each of the 1000 ImageNet categories. We use these probabilities to calculate an expected distance to the target ImageNet category. While training our neural network, we adjust our weights to minimize this expected distance.

While evaluating our model, instead of using metrics like top-5 accuracy, we compare our model's expected distance to the target class with a baseline distance, which represents the distance to the target class assuming the probabilities for all of the classes is uniform (0.001).

Attack (v2) Results. Ultimately we find that our model's expected distance to the target class is 10% closer than the baseline distance, indicating that there is in fact information leakage from NeuralHash hashes. Importantly, the metric that we define allows us to better quantify the magnitude of leakage, compared to top-5 accuracy and other scores that severely underestimate the amount of information learned.

5 NeuralHash Improvements

Key takeaways:

- **SHA-at-the-End is simple and effective!**
- **Preventing adversarial examples is a fundamentally hard problem, intrinsically tied to the nature of complicated differentiable functions like neural networks. While adversarial input detection is possible, it is not an effective solution.**

5.1 SHA-at-the-End

The black-box interpolation and information extraction attacks presented in Section 4.3 all rely on the approximate linearity of $\mathcal{N}(x)$. What if we could nullify this property?

Method Description. To address the undesirable linearity of NeuralHash, we propose a simple yet effective extension: adding a SHA-256 block at the very end of the computation. Revisiting the security definitions in Section 4.1, we find that this strengthens the uniform hashing and noninvertibility properties while maintaining the neighborhood property: if $\mathcal{N}(x) = \mathcal{N}(x')$, then it must be true that $\text{SHA}(\mathcal{N}(x)) = \text{SHA}(\mathcal{N}(x'))$. Moreover, if $\mathcal{N}(x) \neq \mathcal{N}(x')$, then under the Random Oracle Model (ROM) their hashes will (i) look uniformly random and (ii) be non-invertible.

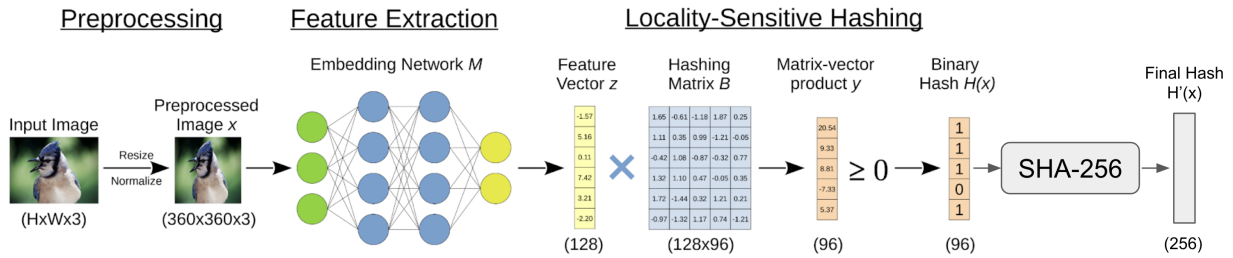

Figure 10: Our proposed new NeuralHash pipeline including an encryption step at the end.

Comment on Novelty/Necessity. To the best of our knowledge, [Str+21] and other popular blogposts discussing NeuralHash have not proposed this modification. Apple's technical report on NeuralHash [Inc21] suggests that the company has direct access to $\mathcal{N}(x)$.

Method Results. Based on the analysis above, SHA-at-the-end should work well given the theoretical assumptions; here, we provide some concrete empirical evidence. First, we check whether the interpolation graph behaves as expected (Section 4.3.1); as seen in Figure 11, the cosine similarity between the two images indeed hovers around 0.5 (except at the borders), and the cosine similarity between adjacent interpolations oscillates between 1 and ≈ 0.5. These findings suggest that SHA-at-the-end can successfully defend against the black-box interpolation attacks presented in Sections 4.3.1 and 4.3.2, since linearity is broken.

Next, we attempt the same information recovery attack as described in Section 4.3.3: the top-1 accuracy of the extraction model drops to 6%, which is less than half of the original top-1 accu-

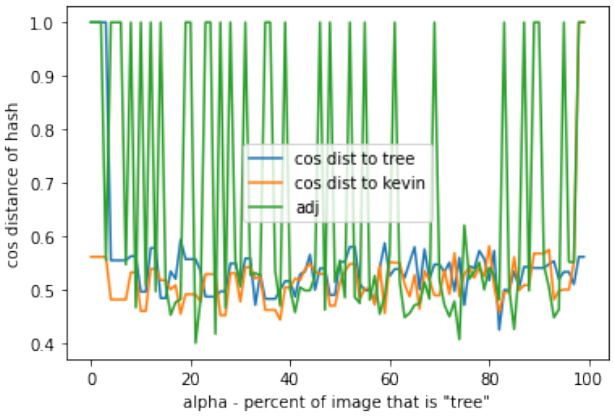

Figure 11: Interpolation Results using SHA-at-the-End (Compare to Figure 7)

racy. While this is still above random, we attribute the difference to possibly imperfect evaluation conditions: class imbalance in the dataset, as well as less-than-ideal sample size.

5.2 Adversarial Input Detection

While adversarial inputs are unavoidable for large differentiable functions like NeuralHash's neural network, we can mitigate the effect of gradient-based attacks by *detecting* adversarially-constructed inputs at runtime.

Method Description. We train a neural network to detect inputs that are strongly out of distribution. This is done via a simple binary classifier, implemented as a three-layer convolutional neural network (Figure 12): given any image x, it outputs a binary label identifying x as a normal image or an adversarial image.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_74 (Conv2D)           (None, 356, 356, 2)       152

max_pooling2d_54 (MaxPoolin  (None, 178, 178, 2)       0
g2D)

conv2d_75 (Conv2D)           (None, 172, 172, 4)       396

max_pooling2d_55 (MaxPoolin  (None, 86, 86, 4)         0
g2D)

conv2d_76 (Conv2D)           (None, 37, 37, 8)         6280

max_pooling2d_56 (MaxPoolin  (None, 18, 18, 8)         0
g2D)

flatten_23 (Flatten)         (None, 2592)              0

dense_23 (Dense)             (None, 2)                 5186

=================================================================
Total params: 12,014
Trainable params: 12,014
Non-trainable params: 0
_________________________________________________________________
```

Figure 12: Convolutional Model Architecture of Adversarial Input Detector

Comment on Novelty. While adversarial image detection is well-studied in general contexts

[Xu+19], it has not been studied explicitly on NeuralHash, to the best of our knowledge.

Method Results. A convolutional neural network was trained and evaluated on disjoint subsets of ImageNette. Specifically, the dataset was constructed by sampling ImageNette, generating adversarial images that had non-matching hashes w.r.t. the original, and storing the pair in an array. Non-perturbed images were assigned the label 0, whereas attacked images were assigned a 1.

We found that the effectiveness of adversarial input detection (in preventing gradient-based evasion attacks) varied significantly based on the noise parameter ϵ (Section 4.2). For $\epsilon = 0.05$, the neural network was completely unable to converge. At larger noise margins (where the processed images have larger differences), however, we found better success: when $\epsilon = 0.5$, the classifier had 82% accuracy. Note that at $\epsilon = 0.5$ the adversarial nature of the input can already be detected by the naked eye; it is a generous noise constraint.

Overall, adversarial input detection is *hard*. Not only are they difficult-to-detect on a pixel level, but literature around adversarial inputs follows a game of cat-and-mouse, where people design fragile algorithms to overcome adversarial inputs (oftentimes neural networks themselves with adversarial examples of their own), only to see them broken easily.

6 Conclusion

In this paper, we propose and analyze a plethora of attacks that show the weakness and vulnerabilities of Apple's NeuralHash perceptual hashing algorithm. Along the way, we improve SOTA gradient-based attacks (more realistic, stronger semantic guarantees) and propose an entirely novel class of powerful interpolation-based black-box attacks. In addition, we propose a information extraction attack that uses less data and reveals more information compared to prior methods. Finally, we propose and validate changes to the architecture as well as adversarial input detection to safeguard against some of the above attacks.

In particular, our work has shown that adversarial examples are the Achilles heel of NeuralHash; they are easy to implement and highly effective (Section 4.2). The use of neural networks for perceptual hashing is a double edged sword: neural networks are the only current feasible way to do convincing perceptual hashing, but they are also extremely vulnerable to adversarial attacks. Future approaches may find more success *combining* symbolic reasoning with neural networks, allowing us to exploit the strengths of deep learning while mitigating its weaknesses.

Some interesting directions for future work:

- The design of a *non-differentiable* fuzzy matcher for images.
- In Section 8, we demonstrated the power of interpolation based methods not only for finding hash collisions, but also for inverting a hash given a database of candidate images. While our analysis was limited by the small size of the reference dataset, using a larger database might yield interesting results on many different types of images.

References

[CPZ08] Ondrej Chum, James Philbin, and Andrew Zisserman. "Near Duplicate Image Detection: min-Hash and tf-idf Weighting". In: Jan. 2008. DOI: 10.5244/C.22.50.

[DS13] Ferdinando Di Martino and Salvatore Sessa. "Image Matching by Using Fuzzy Transforms". In: *Advances in Fuzzy Systems* 2013 (2013). DOI: 10.1155/2013/760704.

[GSS14] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples". In: *arXiv preprint arXiv:1412.6572* (2014).

[Fas19] FastAI. *ImageNette: A subset of ImageNet*. 2019. URL: https://github.com/fastai/imagenette.

[Xu+19] Han Xu et al. *Adversarial Attacks and Defenses in Images, Graphs and Text: A Review*. 2019. URL: https://arxiv.org/abs/1909.08072.

[AI21] Roboflow AI. *NeuralHash Collisions*. 2021. URL: https://github.com/roboflow-ai/neuralhash-collisions.

[Ath21] Anish Athalye. *Neural Hash Collider*. 2021. URL: https://github.com/anishathalye/neural-hash-collider.

[BBM21] Abhishek Bhowmick, Dan Boneh, and Steve Myers. "The Apple PSI System". In: (2021). URL: https://www.apple.com/child-safety/pdf/Apple_PSI_System_Security_Protocol_and_Analysis.pdf.

[Dwy21] Brad Dwyer. *ImageNet contains naturally occurring neuralhash collisions*. 2021. URL: https://blog.roboflow.com/neuralhash-collision/.

[Inc21] Apple Incorporated. "CSAM Detection". In: (2021). URL: https://www.apple.com/child-safety/pdf/CSAM_Detection_Technical_Summary.pdf.

[Str+21] Lukas Struppek et al. "Learning to Break Deep Perceptual Hashing: The Use Case NeuralHash". In: *CoRR* abs/2111.06628 (2021). arXiv: 2111.06628. URL: https://arxiv.org/abs/2111.06628.

[Ygv21] Asuhariet Ygvar. *Apple Neural Hash to ONNX*. 2021. URL: https://github.com/AsuharietYgvar/AppleNeuralHash2ONNX.

[Con22] Wikipedia Contributors. *Locality-sensitive hashing*. 2022. URL: https://en.wikipedia.org/wiki/Locality-sensitive_hashing.