Investigating the Risks of Client-Side Scanning for the Use Case NeuralHash

Dominik Hintersdorf[1,*], Lukas Struppek[1,*], Daniel Neider[2], Kristian Kersting[1,3]
[1] Department of Computer Science, Technical University of Darmstadt, Darmstadt, Germany
[2] Max Planck Institute for Software Systems, Kaiserslautern, Germany
[3] Centre for Cognitive Science, Technical University of Darmstadt, and Hessian Center for AI (hessian.AI)
[*] Both authors contributed equally
{dominik.hintersdorf, lukas.struppek}@cs.tu-darmstadt.de

***Abstract*—Regulators around the world try to stop the distribution of digital criminal content without circumventing the encryption methods in place for confidential communication. One approach is client-side scanning (CSS) which checks the content on the user's device for illegality before it is encrypted and transmitted. Apple has recently revealed a client-side deep perceptual hashing system called NeuralHash to detect child sexual abuse material (CSAM) on user devices before the images are encrypted and uploaded to its iCloud service. After its public presentation, criticism arose regarding the users' privacy and the system's reliability. In this work, we present an empirical analysis of client-side deep perceptual hashing based on NeuralHash. We show that such systems are not robust, and an adversary can easily manipulate the hash values to force or prevent hash collisions. Such attacks permit malicious actors to exploit the detection system: from hiding abusive material to framing innocent users, a large variety of attacks is possible.**[1]

I. Introduction

Apple recently announced its NeuralHash [1] system, a deep perceptual hashing algorithm for client-side content scanning. The approach focuses on identifying CSAM content in user files uploaded to Apple's iCloud service. Apple has made several assurances about privacy and security, such as a low risk of accounts being falsely flagged and restricting its access to private data. According to Apple, only images uploaded to the iCloud servers will be hashed and compared against a hash database of known CSAM material. The hash databases are provided by child protection agencies and are only available in encrypted form on the user devices. More information on NeuralHash is provided in the official technical summary [1].

While Apple commissioned various expert opinions on the security of the system from independent researchers [2]–[4], there has been a lot of public criticism of the NeuralHash approach [5]–[7]. **Criticism is directed not only at possible privacy violations but also at the system's reliability.** So far, however, there has been a lack of comprehensive analyses, at least publicly, of the core of the perceptual hashing process, the hash computation itself.

Apple is not the only organization planning to install client-side scanning (CSS) for CSAM detection. In 2020, the European Union (EU) presented its strategy [8] to become more effective in fighting child sexual abuse. The strategy specifically refers to end-to-end encryption and calls for technical solutions that allow companies to detect and report CSAM material transferred in encrypted communication systems. The subsequent EU regulation 2021/1232 [9], which also faces criticism [10], establishes temporarily limited rules to process personal and other data to detect and report CSAM material. Without the regulation stating specific technical details, it is plausible that hash-based CSS approaches could be one way to achieve this.

A recent paper [6] by well-known cybersecurity researchers and encryption system inventors, such as Hal Abelson, Carmela Troncoso, and Josh Benaloh, outlines and critiques the potential security and privacy risks of CSS technologies. The authors argue that even if the initial goal of these systems is only the detection of clearly illegal content, tremendous pressure to expand the scope of application will arise with time. People would then have little chance to resist the expansion of the system or prevent its abuse.

In this work, we investigate the perceptual hashing components of NeuralHash, in particular, the embedding neural network and the hashing step. It might be to some extent common knowledge that neural networks are susceptible to various kinds of attacks. However, we are convinced that it is important to demonstrate that this susceptibility is not only interesting from a researcher's point of view but actually affects systems used by millions of users who might not be aware of these risks. We further want to emphasize that by using neural networks for CSS and, therefore, being able to calculate the gradients with respect to the inputs, most of the attacks are rather easy to perform, exposing various risks to manipulate the systems.

Our focus lies on NeuralHash because it is the first prominent representative to shift content detection from the server-side to user devices. This approach poses additional major risks and insecurities, such as causeless algorithmic surveillance of users [6]. We show that deep perceptual hashing has various downsides when applied to real-world large-scale image detection. Our research aims to point out drawbacks of deep perceptual image hashing, support the development of more robust systems, and encourage a discussion on the general deployment of this technology. NeuralHash merely

[1] A long and extended version of this paper is available at https://arxiv.org/abs/2111.09076.

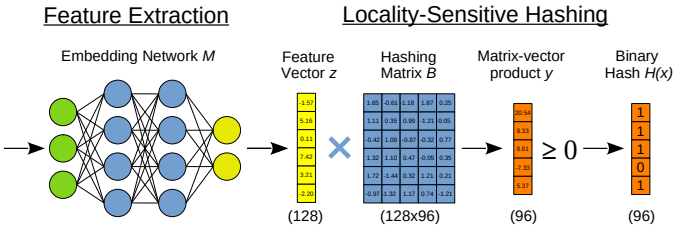


Fig. 1: The NeuralHash pipeline consists of an embedding network and a locality-sensitive hashing (LSH) step.

acts as a current real-world example in this case.

Before diving into the details, we want to make the following two statements:

1) We explicitly condemn the creation, possession, and distribution of child pornography and abusive material, and we strongly support the prosecution of related crimes. With this work, we in no way intend to provide instructions on how to bypass or manipulate CSAM filters. Instead, we want to initiate a well-founded discussion about the effectiveness and the general application of client-side scanning based on deep perceptual hashing.
2) We have no intention to harm Apple Inc. itself or their intention to stop the distribution of CSAM material. NeuralHash merely forms the empirical basis of our work to critically examine perceptual hashing methods and the risks they may induce in real-world scenarios. Even though this system is not in production yet, we contacted Apple and made sure they are aware of the possible issues with NeuralHash.

II. Perceptual Hashing for Image Detection

Hashing generally describes a deterministic transformation of data into short bit sequences, the so-called hashes or fingerprints. Perceptual hashing algorithms, e.g., Apple's NeuralHash [1], Microsoft's PhotoDNA [11], and Facebook's PDQ [12], aim to compute similar hashes for images with similar contents and dissimilar hashes for different contents. In recent years, various deep hashing algorithms based on convolutional neural networks have been proposed [13]–[16]. They all rely on deep neural networks to extract unique features from an image and use them to compute a hash value.

The NeuralHash pipeline, visualized in Fig. 1, consists of an embedding network and a locality-sensitive hashing (LSH) step [17], [18] to calculate the binary hash values of an image. First, the embedding network extracts features from the given image and maps the image to an abstract feature representation vector with 128 real-valued entries. LSH is then used to assign similar feature vectors to buckets with similar hash values. This is done by defining 96 (random) hyperplane vectors as a matrix, multiplying the feature vector with this hashing matrix, and applying a Heaviside step function to the multiplication result. In other words, depending on which side of the hyperplanes the vector is located, the bits are either set to 0 or 1, resulting in a binary hash value with 96 bits.

For our experiments, we manually extracted [19] the NeuralHash model, including the network weights and hash matrix, from a Mac running macOS Big Sur Version 11.6. We then rebuilt the network's architecture in PyTorch and assigned the extracted weights accordingly. This allows us to run our experiments on GPUs, significantly increasing the inference speed. Also, gradients can be easily computed by PyTorch's automatic differentiation. NeuralHash further expects resized RGB images with shape $360 \times 360 \times 3$, so all images used for our experiments are resized accordingly.

To check if the framework change induced a significant deviant model behavior, we computed the hashes for all 100,000 samples from the ImageNet test split with our PyTorch model and a separately extracted ONNX model [19]. Both models differed only for a single bit in one of the hashes, demonstrating that the models are virtually identical, and our results should be transferable to the original NeuralHash system.

Little work has been published on the vulnerability of deep hashing functions. As a first step, targeted [20]–[23] and untargeted [24] adversarial attacks against deep hashing-based retrieval systems were proposed. Another work [25] demonstrated the susceptibility of various image hashing functions against gradient-based collision attacks. Similar works investigated the robustness of non-deep perceptual hashing algorithms against adversarial attacks [26]–[28] and their robustness [29] against visible image modifications. While several proof-of-concept implementations to create hash collisions on NeuralHash exist [30]–[32], demonstrating that it is possible to create hash collisions, there is no comprehensive work on identifying the technical vulnerabilities and the robustness of NeuralHash or CSS systems based on deep perceptual hashing from a machine learning perspective. Our work aims to fill this research gap and help to weigh the privacy benefits and potential threats. In this work, we focus on two attack settings: hash collision and detection evasion.

III. Setting 1 – Hash Collision Attacks

In our first adversarial setting, we investigate the creation of hash collisions by perturbing images so that their computed hashes match predefined target hashes. This leads to several explosive scenarios in reality. For instance, given a set of target hashes from the (CSAM) hash database, an adversary can create fake images whose hashes match those in the database without containing any sensitive material at all. Distributed across many devices, this may lead to a large number of false-positive alarms in the (possibly partly human) detection system and, as a consequence, could result in the framing of innocent users or distributed denial of service (DDoS) attacks.

Another worrisome scenario could be that service providers, governments, or other powerful organizations either add additional hashes to the database or manipulate ordinary images that show political or social content that is undesirable to them, such as government criticism or LGBTQ+ support. By spreading such manipulated images through social media, they might end up on the devices of supporters of these causes.

SR	ℓ_2	ℓ_∞	SSIM	Steps
90.81%	20.8136 ± 7.97	0.3120 ± 0.22	0.9647 ± 0.03	1190 ± 1435

TABLE I: Evaluation metrics (mean + standard deviation) for our hash collision attack computed on an ImageNet subset.

Identification, surveillance, and persecution of these people are then possible by detecting hash collisions with the database. This poses a great danger, especially for people in countries with restricted human rights or totalitarian regimes.

Even without access to the hash database, which governments might have, after all, an adversary could simply collect its own CSAM material and compute corresponding hashes. Assuming a large enough database, the adversary could obtain, at least, some share of the true database hashes. Figure 2 illustrates such a hash collision attack. It shows a manipulated image from a protest march that results in the same hash value as a non-related target image.

Realization. We created a surrogate hash database with dog images from the Stanford Dogs dataset [34] that acts as a list of images that should be detected by the system. We then tried to force hash collisions with this database using the first 10,000 samples from the ImageNet ILSVRC2012 [35], [36] test split.

We first computed the hash of an input image and took the target hash from the (surrogate) hash database with the smallest Hamming distance to the hash of the input image. Using a Hinge loss and a structural similarity (SSIM) loss [37] to reduce visual conspicuities, we optimized the input image such that the binarized matrix vector product approached the target hash. We optimized the input image until the hash value matched the target hash or until 10,000 optimization steps were completed. For more technical details, see Appx. A.

Results. Table I states our collision attack results. The success rate (SR) indicates the share of images whose hashes have successfully been changed. We further state the mean ℓ_2 and ℓ_∞ distances between the original images and their optimized counterparts to quantify pixel-wise image changes. We also computed the mean SSIM values to take the image quality into account. The closer SSIM $\in [0, 1]$ is to 1, the more similar a manipulated image is to the original image without perturbations. Steps denote the mean number of optimization steps performed until a hash collision occurred.

We could force hash collisions in about 90% of all images, demonstrating the real applicability of the attack. The visual salience of the modifications varies in strength. Most manipulated images contain some patches of color, which are sometimes conspicuous but often hardly noticeable. Figure 2 shows such an example where the visual differences between the original and manipulated images are small. We plot further examples in Appx. C.

In summary, our experiments show that hash collisions can easily be forced in NeuralHash and might build the base for serious attacks targeting the service provider or, even worse, persecution of political opponents. While some of the induced image changes are visible, they are barely noticeable in most images, as also indicated by the high SSIM values.

IV. Setting 2 – Detection Evasion Attacks

In our second setting, we investigate detection evasion attacks that aim to avoid detection of sensitive material through perceptual hashing. Intuitively, an attacker tries to evade detection in terms of hash matching with a database by perturbing the images. We investigate two different evasion approaches: optimization-based approaches, which compute image-specific perturbations, and transformation-based approaches, which apply simple transformations as implemented in standard image manipulation programs. Non-robust perceptual hashing algorithms would make it easy to hide sensitive material from detection and call the overall effectiveness of such systems into question.

Our optimization-based approaches exploit the fact that NeuralHash is deployed on consumer devices and allows the computation of gradients with respect to the input images if the model access is not secured. It could also be interpreted as an application of untargeted adversarial attacks [38]. The induced changes of such attacks are hardly perceivable in most cases.

For the transformation-based approaches, the adversary does not need direct access to the hash algorithm or any expertise in computer science. Using a simple image editor to alter the images is sufficient to perform the attacks. Many gradient-free transformations, such as flipping an image, can be easily reverted without any quality loss. It makes these transformations particularly interesting to investigate: a single transformation could be used to evade detection systems by first applying it to bypass the system and then reverting the changes to reconstruct the original image.

Realization. For our optimization-based approaches to evade detection, we used the first 10,000 ImageNet test samples and computed image-specific perturbations to change the hash values. We directly altered the whole images (Standard) to increase their Hamming distance to the unmodified images' hashes. We used a negative mean squared error (MSE) loss between the hashes for this and added a structural similarity (SSIM) [37] penalty term to reduce visual conspicuities. We also tried to restrict the induced changes to edges in the images (Edges-Only) or as few pixels as possible (Few-Pixels). Each attack was stopped after at least a single bit flipped.

To investigate the robustness against gradient-free image transformations, such as flipping or rotation, we independently applied different transformations. We gradually increased the transformation strength and measured the average Hamming distance of the resulting hashes across all tested images. We evaluated the robustness against transformations on all 1,281,167 ImageNet training samples. We investigated the following transformations independently: translation, rotation, center cropping, downsizing, flipping, changes in the HSV color space, contrast changes, and JPEG compression. See Appx. B for more details on our optimization-based and transformation-based approaches.

Fig. 2: Visualization of our hash collision attack. We manipulated the original image [33] to have the same hash as the target image. The manipulated image is visually hardly distinguishable from the original since the induced perturbations are small. Still, the manipulated image is assigned the same hash as the completely different target image. This demonstrates the feasibility and danger of hash collision attacks. The third image illustrates the added perturbations computed as the mean absolute differences over all color channels. Lighter areas in the heat map indicate greater changes.

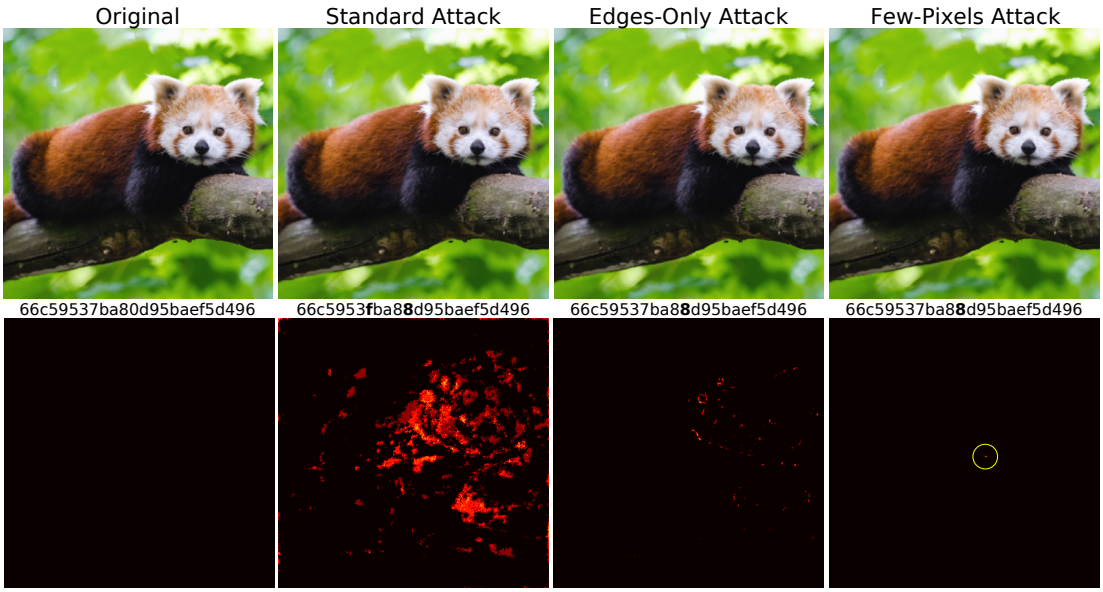


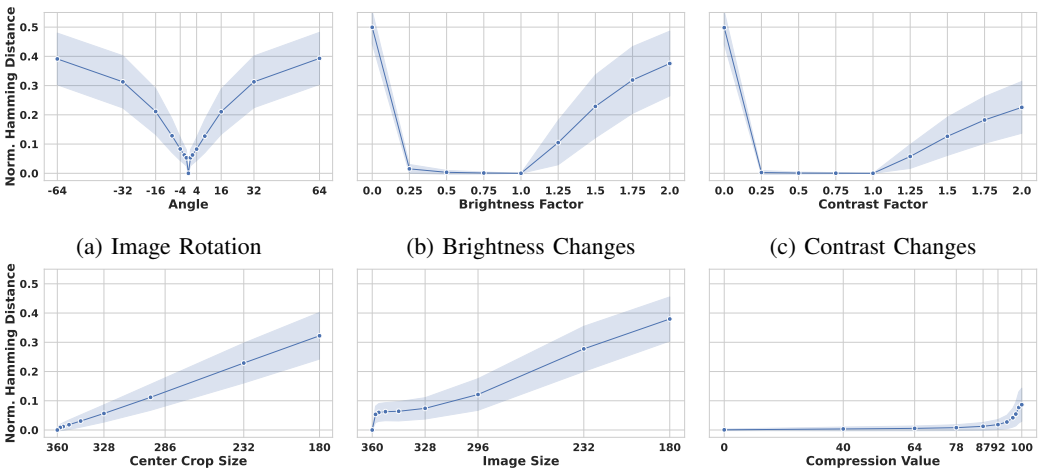
Fig. 3: Visualization of our gradient-based evasion attacks. The added perturbations are barely visible to humans. Nevertheless, the image hashes differ from the original, showing that detection evasion is possible without degrading the image quality. The bottom row illustrates the added perturbations computed as the mean absolute differences over all color channels. Lighter areas in the heat maps indicate greater changes.

Attack	Standard	Edges-Only	Few-Pixels
SR	100.00%	99.95%	98.21%
ℓ_2	0.7188 ± 0.28	1.3882 ± 1.37	2.9100 ± 2.06
ℓ_∞	0.0044 ± 0.00	0.0841 ± 0.07	0.8298 ± 0.25
SSIM	0.9999 ± 0.00	0.9996 ± 0.00	0.9989 ± 0.00
Steps	5.4006 ± 4.98	150.2414 ± 113.96	3095.0 ± 3901

TABLE II: Evaluation metrics (mean + standard deviation) for our three gradient-based evasion attacks computed on an ImageNet subset.

Results. Table II states the results for our gradient-based evasion attacks. The metrics are the same as for our first setting. The success rate (SR) indicates the share of images whose hashes could be changed before the maximum number of iterations or pixels has been reached. All three attacks were able to change the hash value of an image in the large majority of the cases. While the Standard attack changed the hashes of all images, the Edges-Only and Few-Pixels attacks only failed in a few cases. Figure 3 further illustrates the effectiveness of our attacks and shows that the changes are hardly visually perceivable. We state additional samples for each attack in Appx. D. The Standard attack only needs about five optimization steps on average to force at least a single hash bit to flip. In the Few-Pixels attack, only 21.5 pixels were changed on average, 0.017% of all pixels. We note that larger Hamming distances could easily be forced by increasing the number of optimization steps.

We further found NeuralHash to be not robust against most gradient-free image transformations, as our results in Figure 4 demonstrate. The most significant hash changes occurred when



(a) Image Rotation (b) Brightness Changes (c) Contrast Changes

(d) Center Cropping (e) Downsizing (f) JPEG Compression

Fig. 4: Robustness results for gradient-free image transformations computed on all ImageNet training samples. The plots state the mean results and their standard deviations. The Hamming distance also describes the share of hash bits expected to change.

image information was lost or added, e.g., translating or rotating images and filling missing parts with black color. Shifting an image by 32 pixels in horizontal and vertical directions resulted in a mean Hamming distance of roughly 11%. Larger image translations even further raised the Hamming distance. Also, cutting away the image's boundaries increases the share of flipped bits with the crop size. JPEG image compression, on the other side, has only a small impact on the hash computation. For color transformations such as changing the saturation, or decreasing the brightness and contrast, the hash computation is much more stable. For plots of translations and hue and saturation changes, see Appx. E. However, flipping the images, one of the simplest transformations without any loss of information and easy to revert, already changed 29% of the hash bits for horizontally flipping and even 37% for vertical flipping.

Our results demonstrate that hash changes can be easily forced by gradient-based perturbations. For humans, the induced changes are hardly perceivable. Our analyses further show that NeuralHash is not robust against many basic image transformations. While the system showed stronger robustness against HSV space modifications and image compression, it is more susceptible to other transformations, such as rotations or flipping. An adversary could evade detection with small effort and little quality loss in the images. In reality, the effectiveness of such a system is therefore questionable.

V. Lessons & Implications

We conclude our work with lessons and implications that arise for deep perceptual hashing and CSS based on our work.

Current systems may not be robust. Our experimental results illustrate that NeuralHash, and most likely other perceptual hashing systems, are not robust against detection evasion attacks. Simple image modifications, such as flipping an image horizontally, already allow an attacker to evade detection in most cases, even without direct access to the system. The attacker only needs basic technical knowledge and can manipulate the images with standard image editors to evade detection. On the other hand, evasion attacks exploiting gradient information allow evasion with only minor, barely visible changes to the images. Our experiments have shown that with intermediate technical knowledge, an attacker can modify the images to evade detection without the modifications being eye-catching. Such attacks will most likely be possible in any neural network-based CSS system if the attacker has access to the model on the user device and is, therefore, able to calculate the gradients of the model.

A more robust hashing system will force the attacker to alter the images stronger, but the original content of the images will most likely still be visible and recognizable. For example, disassembling an image into smaller parts for storing and reassembling the image only for viewing would lead to completely different hashes for each part without any loss of information in the full image.

CSS systems can be misused for malicious purposes. Proponents of CSS emphasize the fact that the systems enable the detection of CSAM and other criminal material while avoiding backdoor keys in the end-to-end encryption and maintaining the user's privacy. However, CSS opens the door for other malicious attacks. While evasion of the detection system only renders the system useless, an attacker can further misuse the system for framing or monitoring innocent users with hash collision attacks. As we have demonstrated, an attacker can slightly alter images to change their hashes to a specific value, causing false-positive detections for arbitrary image content. With access to an official hash database or a surrogate, an attacker can frame innocent people by sending manipulated images to their devices. The receivers of such images then get flagged by the system without even knowing.

As legislators call for preventing the distribution of harm-

ful material while maintaining encryption gets louder, CSS systems using perceptual hashing look very promising at first glance. However, governments or organizations with control over the CSS system could share manipulated images in social media, and users who have downloaded a subset of these images will be flagged. Mass surveillance of people with undesired beliefs and opinions is therefore easily and secretly realizable. Additional tools or backdoors in the user devices are not needed. Moreover, there is no guarantee that the hash database will not be extended with additional, probably non-criminal content for surveillance. Since the databases are not publicly available, changes are not traceable, and the targeted content might be anything – who controls the provider of such systems?

Mitigating the risks. Regarding the technical aspects of the CSS system, one way to mitigate the effectiveness of collision attacks against hashing-based detection systems is to install another server-side hashing procedure not available to the attacker. However, this would, in turn, imply that the images are not encrypted on the server or could be decrypted by the provider, questioning the promised privacy advantages. Another method would be to restrict the model access, e.g., by running the model in a trusted execution environment, and prevent gradient computations, which would indeed make gradient-based attacks hardly feasible. However, these steps do not improve the robustness against standard image transformations. CSS methods based on neural networks will most likely enable gradient computation or approximation and, consequently, facilitate arbitrary hash manipulations. Hence, even an updated version of NeuralHash's embedding network would very likely suffer from the demonstrated vulnerabilities.

It is also important to restrict public access to the hash database. If the plain hashes are leaked, it is even easier for an adversary to produce a large number of false-positive matches. Furthermore, an independent instance should monitor the database to avoid manipulations.

Conclusion. In summary, we demonstrated that NeuralHash, and arguably deep perceptual hashing algorithms in general, are not robust and are susceptible to various image manipulations. It is questionable if neural networks in their current form will ever be fully robust. We would like to stress that CSS has a noble aim, but at the same time we want to draw attention to the potential risks of extending its scope. After all, CSS is running on all devices regardless of whether a crime is suspected or not. We hope that service providers will not follow this direction for crime detection due to the many technical and societal risks.

From a technical and ethical viewpoint, we conclude that NeuralHash and related CSS systems do not (yet) provide a safe method for detecting legal violations and should not be deployed on user devices since attackers, service providers, and governments could easily manipulate and misuse the systems for their own interests. After all, mobile devices contain a lot of sensitive information about their users, ranging from dating behavior to health care and financial status. Instead, based on our results, we call for NeuralHash and other CSS systems not to be installed in practice, due to their manipulability, risk of abuse, and lack of robustness.

Reproducibility Statement. We deliberately decided to make our source code publicly available to reproduce the experiments and investigate existing and future perceptual hashing systems: https://github.com/ml-research/Learning-to-Break-Deep-Perceptual-Hashing.

Acknowledgments. This work was supported by the German Ministry of Education and Research (BMBF) within the framework program "Research for Civil Security" of the German Federal Government, project KISTRA (reference no. 13N15343). It also benefited from the National Research Center for Applied Cybersecurity ATHENE, a joint effort of BMBF and the Hessian Ministry of Higher Education, Research, Science and the Arts (HMWK).

Disclaimer. The views and opinions expressed in this article are those of the authors and do not reflect the official policy or position of the authors' institutions or any agency of the German Federal Government.

REFERENCES

[1] A. Inc., "CSAM Detection - Technical Summary," 2021. https://www.apple.com/child-safety/pdf/CSAM_Detection_Technical_Summary.pdf, accessed: Sept. 22, 2021.

[2] M. Bellare, "The apple psi protocol," 2021. https://www.apple.com/child-safety/pdf/Technical_Assessment_of_CSAM_Detection_Mihir_Bellare.pdf, accessed: Oct. 1, 2021.

[3] D. Forsyth, "Apple's csam detection technology," 2021. https://www.apple.com/child-safety/pdf/Technical_Assessment_of_CSAM_Detection_David_Forsyth.pdf, accessed: Oct. 1, 2021.

[4] B. Pinkas, "A review of the cryptography behind the apple psi system," 2021. https://www.apple.com/child-safety/pdf/Technical_Assessment_of_CSAM_Detection_Benny_Pinkas.pdf, accessed: Oct. 1, 2021.

[5] E. Snowden, "The all-seeing "i": Apple just declared war on your privacy," 2021. https://edwardsnowden.substack.com/p/all-seeing-i, accessed: Oct. 1, 2021.

[6] H. Abelson, R. Anderson, S. M. Bellovin, J. Benaloh, M. Blaze, J. Callas, W. Diffie, S. Landau, P. G. Neumann, R. L. Rivest, J. I. Schiller, B. Schneier, V. Teague, and C. Troncoso, "Bugs in our pockets: The risks of client-side scanning," *CoRR*, vol. abs/2110.07450, 2021.

[7] S. J. Lewis, "Tweets on neuralhash," 2021. https://twitter.com/SarahJamieLewis/status/1428146453394821125, accessed: Oct. 1, 2021.

[8] E. Commission, "EU strategy for a more effective fight against child sexual abuse," 2020. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:52020DC0607, accessed: Nov. 16, 2021.

[9] O. J. of the European Union, "Regulation (EU) 2021/1232," 2021. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32021R1232, accessed: Nov. 16, 2021.

[10] P. Breyer, A. Alametsä, R. D'Amato, P. Barrena, S. Bricmont, A. Comín, G. Delbos-Corfield, F. Donato, C. Ernst, C. Gamon, M. Gregorová, F. Guerreiro, S. Hahn, I. Joveva, P. Kammerevert, M. Kolaja, M. Körner, K. Melchior, C. Ponsatí, and M. Peksa, "Cross-party letter of member of the european parliament against general monitoring," 2021. https://www.patrick-breyer.de/wp-content/uploads/2021/11/20211020_Letter_General_Monitoring.pdf, accessed: Nov. 19, 2021.

[11] Microsoft, "Photodna," 2015. https://www.microsoft.com/en-us/photodna, accessed Oct. 18, 2021.

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

[13] V. E. Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou, "Deep hashing for compact binary codes learning," in *CVPR*, pp. 2475–2483, 2015.

[14] H. Liu, R. Wang, S. Shan, and X. Chen, "Deep supervised hashing for fast image retrieval," in *CVPR*, pp. 2064–2072, 2016.

[15] D. Wu, Z. Lin, B. Li, M. Ye, and W. Wang, "Deep supervised hashing for multi-label and large-scale image retrieval," in *ACM ICMR*, p. 150–158, 2017.

[16] F. Zhao, Y. Huang, L. Wang, and T. Tan, "Deep semantic ranking based hashing for multi-label image retrieval," in *CVPR*, pp. 1556–1564, 2015.

[17] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in *ACM - STOC*, pp. 604–613, 1998.

[18] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *VLDB*, p. 518–529, 1999.

[19] A. Ygvar, "Appleneuralhash2onnx," 2021. https://github.com/AsuharietYgvar/AppleNeuralHash2ONNX, accessed Oct. 12, 2021.

[20] J. Bai, B. Chen, Y. Li, D. Wu, W. Guo, S.-t. Xia, and E.-h. Yang, "Targeted attack for deep hashing based retrieval," in *ECCV*, 2020.

[21] X. Wang, Z. Zhang, G. Lu, and Y. Xu, "Targeted attack and defense for deep hashing," in *ACM SIGIR*, p. 2298–2302, 2021.

[22] Y. Xiao and C. Wang, "You see what i want you to see: Exploring targeted black-box transferability attack for hash-based image retrieval systems," in *CVPR*, pp. 1934–1943, 2021.

[23] X. Wang, Z. Zhang, B. Wu, F. Shen, and G. Lu, "Prototype-supervised adversarial network for targeted attack of deep hashing," in *CVPR*, pp. 16357–16366, 2021.

[24] E. Yang, T. Liu, C. Deng, and D. Tao, "Adversarial examples for hamming space search," *IEEE Transactions on Cybernetics*, vol. 50, no. 4, pp. 1473–1484, 2020.

[25] B. Dolhansky and C. Canton-Ferrer, "Adversarial collision attacks on image hashing functions," *CoRR*, vol. abs/2011.09473, 2020.

[26] Q. Hao, L. Luo, S. T. Jan, and G. Wang, "It's not what it looks like: Manipulating perceptual hashing based applications," in *Proceedings of The ACM Conference on Computer and Communications Security (CCS)*, 2021.

[27] S. Jain, A. Cretu, and Y. de Montjoye, "Adversarial detection avoidance attacks: Evaluating the robustness of perceptual hashing-based client-side scanning," *CoRR*, vol. abs/2106.09820, 2021.

[28] J. Prokos, T. M. Jois, N. Fendley, R. Schuster, M. Green, E. Tromer, and Y. Cao, "Squint hard enough: Evaluating perceptual hashing with machine learning." Cryptology ePrint Archive, Report 2021/1531, 2021. https://ia.cr/2021/1531.

[29] A. Drmic, M. Silic, G. Delac, K. Vladimir, and A. S. Kurdija, "Evaluating robustness of perceptual image hashing algorithms," in *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pp. 995–1000, 2017.

[30] A. Athalye, "Neuralhash collider," 2021. https://github.com/anishathalye/neural-hash-collider, accessed Oct. 12, 2021.

[31] L. S. Kiat, "apple-neuralhash-attack," 2021. https://github.com/greentfrapp/apple-neuralhash-attack, accessed Oct. 12, 2021.

[32] Y. Kilcher, "Neural hash collision creator," 2021. https://github.com/yk/neural_hash_collision, accessed Oct. 12, 2021.

[33] K. C. (WMUK), "Black lives matter protest at us embassy," 2020. https://commons.wikimedia.org/wiki/File:Black_Lives_Matter_protest_at_US_Embassy,_London_01.jpg, accessed Sept. 23, 2021; Cropped; License: https://creativecommons.org/licenses/by-sa/4.0/deed.en.

[34] A. Khosla, N. Jayadevaprakash, B. Yao, and L. Fei-Fei, "Novel dataset for fine-grained image categorization," in *First Workshop on Fine-Grained Visual Categorization, CVPR*, 2011.

[35] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*, pp. 248–255, 2009.

[36] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *IJCV*, vol. 115, no. 3, pp. 211–252, 2015.

[37] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[38] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *ICLR*, 2014.

[39] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

[40] J. F. Canny, "A computational approach to edge detection," *IEEE TPAMI*, vol. 8, no. 6, pp. 679–698, 1986.

APPENDIX

A. Details for Setting 1

To create hash collisions, we first computed the hash of an input image x_{orig} and took the target hash $\hat{H}$ from the (surrogate) hash database with the smallest Hamming distance to the computed hash of x_{orig}. After defining the target hash $\hat{H} = (\hat{h}_1, \ldots, \hat{h}_k)$, we directly optimized the input image $x = x_{orig}$ such that its binary hash value approaches $\hat{H}$. For this task, we defined a Hinge loss

$$\mathcal{L}_{Hinge}(x, \hat{H}) = \frac{1}{k} \sum\nolimits_{i=1}^{k} max\{0, d - y_i \cdot \psi(\hat{h}_i)\} \quad (1)$$

with $d \geq 0$ describing the margin to the hyperplanes and $y = B \cdot M(x)$ with $y \in \mathbb{R}^{96}$ being the real-valued hash output before binarization. The operation $\psi(\hat{h}_i) = sign(\hat{h}_i - 0.5)$ replaces each 0-bit in the hash vector by -1. In our experiments, we set $d = 0$ to optimize only until the hash value of the optimized image matches $\hat{H}$. This is achieved when the signs of y_i and $\psi(\hat{h}_i)$ match at all k positions. By setting $d > 0$, the distance to the LSH hyperplanes could be increased, leading to more robust image hashes.

Furthermore, we added a structural similarity (SSIM) [37] penalty term to reduce visual conspicuities between an optimized image and its original counterpart. The structural similarity (SSIM) is defined as

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}. \quad (2)$$

For two images x and y, the parameters μ_i and σ_i^2 denote the mean and variance of each image's pixels. Further, σ_{xy} denotes the covariance of x and y. The constants C_1 and C_2 are added for numerical stability and are set to $C_1 = 10^{-4}$ and $C_2 = 9 \cdot 10^{-4}$, respectively, in our experiments. The closer $SSIM \in [0, 1]$ is to 1, the more similar x is to the original image without perturbations. We computed the SSIM, weighted by parameter λ, in each optimization step between the optimized image x and its unmodified counterpart x_{orig} to improve the image quality.

In total, we attempted to solve the following optimization problem:

$$\begin{aligned} \min_{x} \quad & \mathcal{L}_{Hinge}(x, \hat{H}) - \lambda \cdot SSIM(x, x_{orig}) \\ \text{s.t.} \quad & H(x) = \hat{H} \\ & x \in [-1, 1]^{H \times W \times C}. \end{aligned} \quad (3)$$

For our analyses, we created a surrogate hash database by hashing all 20,580 dog images from the Stanford Dogs dataset [34]. We then performed our attacks by modifying the samples on the first 10,000 samples from the ImageNet ILSVRC2012 [35], [36] test split.

We used the Adam optimizer [39] to directly optimize x. We further set $\lambda = 100$ and stopped the optimization when either $H(x) = \hat{H}$ was satisfied or aborted after 10,000 iterations. If we did not stop the optimization process when $H(x) = \hat{H}$ is fulfilled, the image quality might be further improved due to the SSIM term.

B. Details for Setting 2

We start by stating the details for our gradient-based approach. To change the original hash $\tilde{H} = H(x_{orig}) = (\tilde{h}_1, \ldots \tilde{h}_k)$ of a given input image x_{orig}, we used a negative mean squared error (MSE) loss to increase the hash discrepancy between an image x_{orig} and its manipulated counterpart x as

$$\mathcal{L}_{MSE}(x, \tilde{H}) = -\frac{1}{k} \sum\nolimits_{i=1}^{k} \left(\sigma(c\,\widetilde{y}_i) - \tilde{h}_i\right)^2. \quad (4)$$

Instead of $\mathcal{L}_{MSE}$, $-\mathcal{L}_{MSE}$ is reducing the Hamming distance between two hashes and can therefore be used to force hash collisions [30].

We make the binarization step differentiable by replacing it with a sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$. To push the sigmoid values closer to zero and one, we scaled its inputs by a factor c. In our experiments, we set $c = 5$. For a stable optimization, we normalized the matrix-vector product $y = B \cdot M(x)$ by $\widetilde{y} = \frac{y}{max(\|y\|_2, \epsilon)}$ and set $\epsilon = 10^{-12}$ for numerical stability. For each attack, we optimized until a minimal Hamming distance δ_0 between x and x_{orig} is exceeded. As with Setting 1, we also added an SSIM penalty term weighted by hyperparameter λ to reduce visual conspicuities.

The full optimization goal can be formulated as

$$\begin{aligned} \min_{x} \quad & \mathcal{L}_{MSE}(x, \tilde{H}) - \lambda * SSIM(x, x_{orig}) \\ \text{s.t.} \quad & \delta(H(x), \tilde{H}) > \delta_0 \\ & x \in [-1, 1]^{H \times W \times C}. \end{aligned} \quad (5)$$

For the Edges-Only attack, we first applied a Canny edge detector [40] in the grayscale version of an image. We then only allowed changes in the set of edge pixels by applying a binary mask to the gradients during optimization.

As for our Few-Pixels attack, we started by selecting a single pixel to optimize by taking the pixel with the highest absolute $\mathcal{L}_{MSE}$ loss gradient value across all three color channels. We then again applied a binary mask to the gradients and optimized for N steps. If the hash did not change, we added another pixel to the pixel set and again optimized for N steps on all pixels of the set. Additional pixels were again selected by their absolute $\mathcal{L}_{MSE}$ loss gradient values. We repeatedly added more pixels and modified the set of pixels until there was a change in the hash value or a threshold of the number of pixels has been reached.

We used the Adam optimizer to directly optimize an image x. We applied a decaying weight for the SSIM term and set $\lambda = 5 \cdot 0.99^{step}$, where $step$ denotes the number of optimization steps already performed. By decaying the weight, we avoided the optimizer getting stuck in local minima in some cases. For the Standard and Edges-Only attacks, we performed a maximum of 1,000 optimization steps.

In the Few-Pixels attack, we set $\lambda = 0$ and, therefore, removed the SSIM term. The optimization is aborted after 150 pixels without a hash change. In all three attacks, we set the minimum Hamming distance $\delta_0 = 0$ and, consequently, stopped each attack when $H(x) \neq \tilde{H}$.

We also could force our attacks to produce larger perturbations and move a sample even further away from its original hash in terms of the Hamming distance by setting $\delta_0 > 0$.

For our transformation-based approach, we applied different image transformations $T(x) : \mathbb{R}^{H \times W \times C} \rightarrow \mathbb{R}^{H \times W \times C}$ independently to the input images x. We then obtained the perceptual hash of each transformed image $T(x)$ and calculated the Hamming distance $\delta\left(H(T(x)), H(x)\right)$ between the hashes of the transformed and the original images.

For transformations with additional hyperparameters, such as degrees of rotation, we computed results for varying parameter values to take the transformations' strength into account. All investigated transformations kept the image size and removed image parts were filled with black color.

To save computing resources, we varied most hyperparameters with an exponentially increasing step size. We investigated the following transformations independently: translation, rotation, center cropping, downsizing, flipping, changes in the HSV color space, contrast changes, and JPEG compression.

C. Hash Collision Attacks

We depict additional hash collision results in Fig. 5.

D. Gradient-Based Evasion Attacks

We visualize additional results for our three detection evasion attacks. Fig. 6 shows samples for our Standard attack, Fig. 7 for our Edges-Only attack, and Fig. 8 for our Few-Pixels attack.

E. Transformation-Based Evasion Attacks

We visualize example images for each transformation in Fig. 9. The additional plots for hue and saturation changes can be seen in Fig. 10

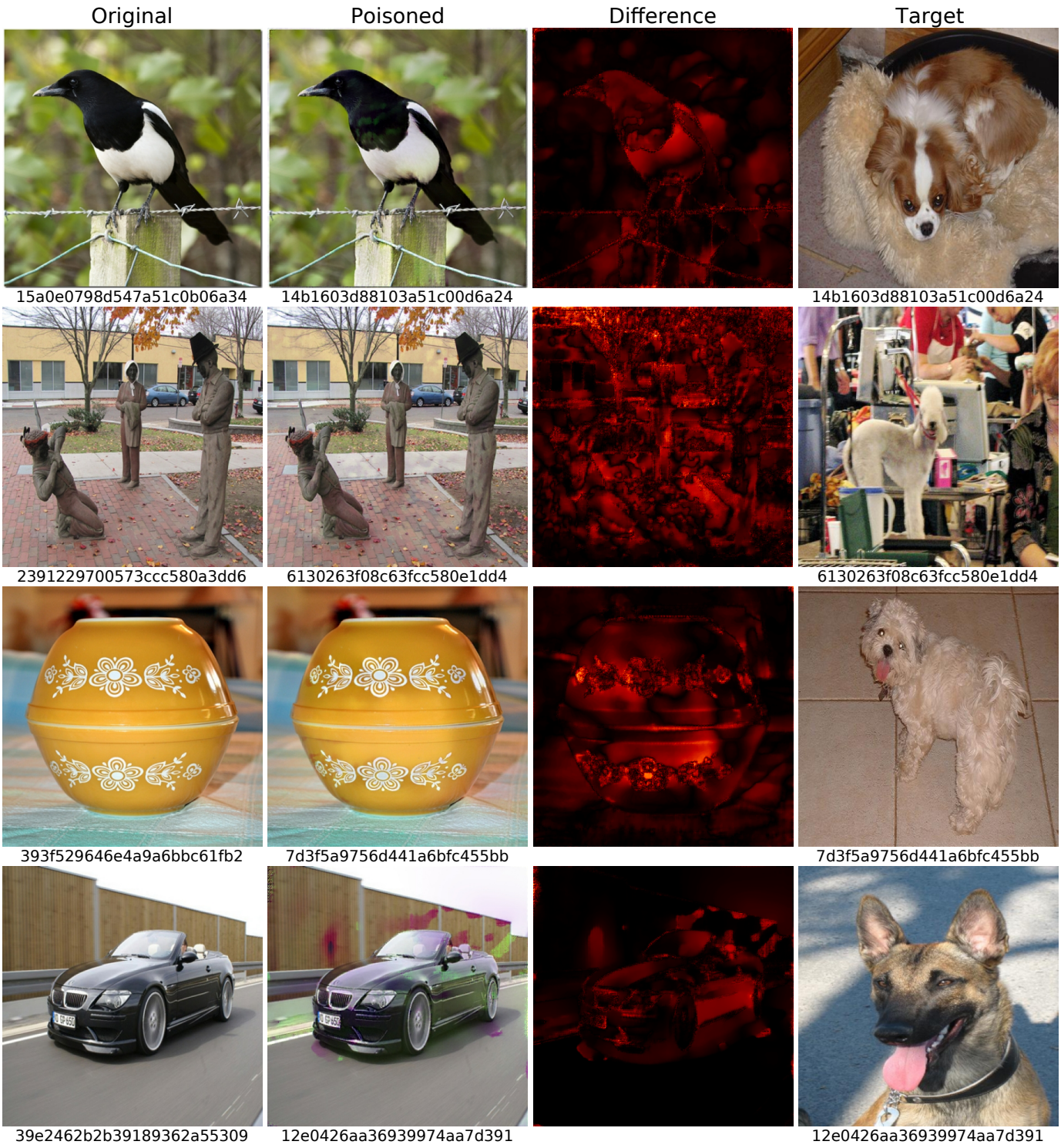


Fig. 5: Selected samples of our hash collision attack. We modify an original image to have the same hash as a target sample from our surrogate hash database. We state the computed hash below each image. The third column illustrates the added perturbations computed as the mean absolute differences over all color channels. Lighter areas in the heat maps indicate greater changes.

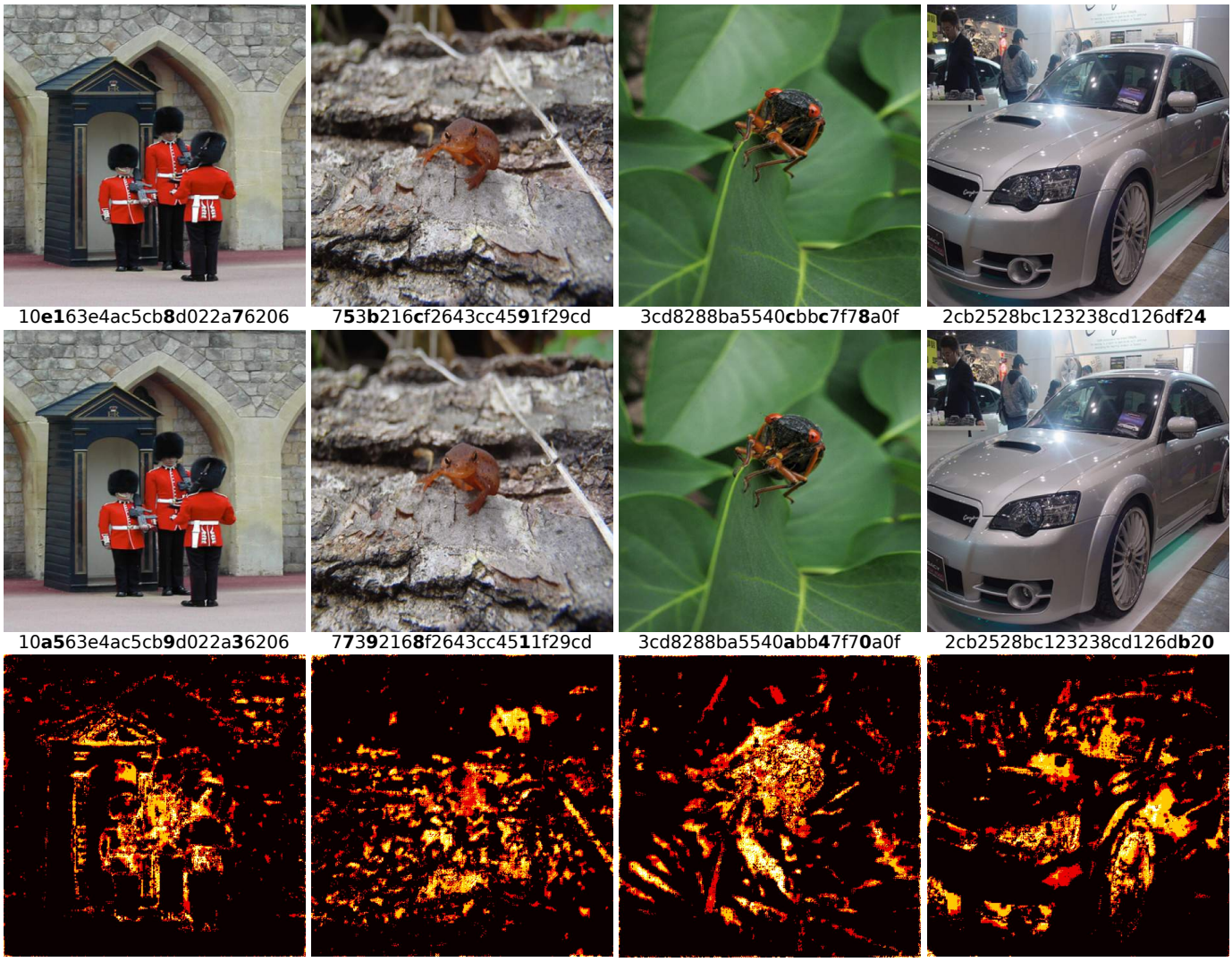


Fig. 6: Selected samples of our Standard evasion attack where we allow modifications of all pixels in an image. We state the computed hash below each image and highlight differences. We further visualize the differences between the modified and original images. The third row illustrates the added perturbations computed as the mean absolute differences over all color channels. Lighter areas in the heat map indicate greater changes.

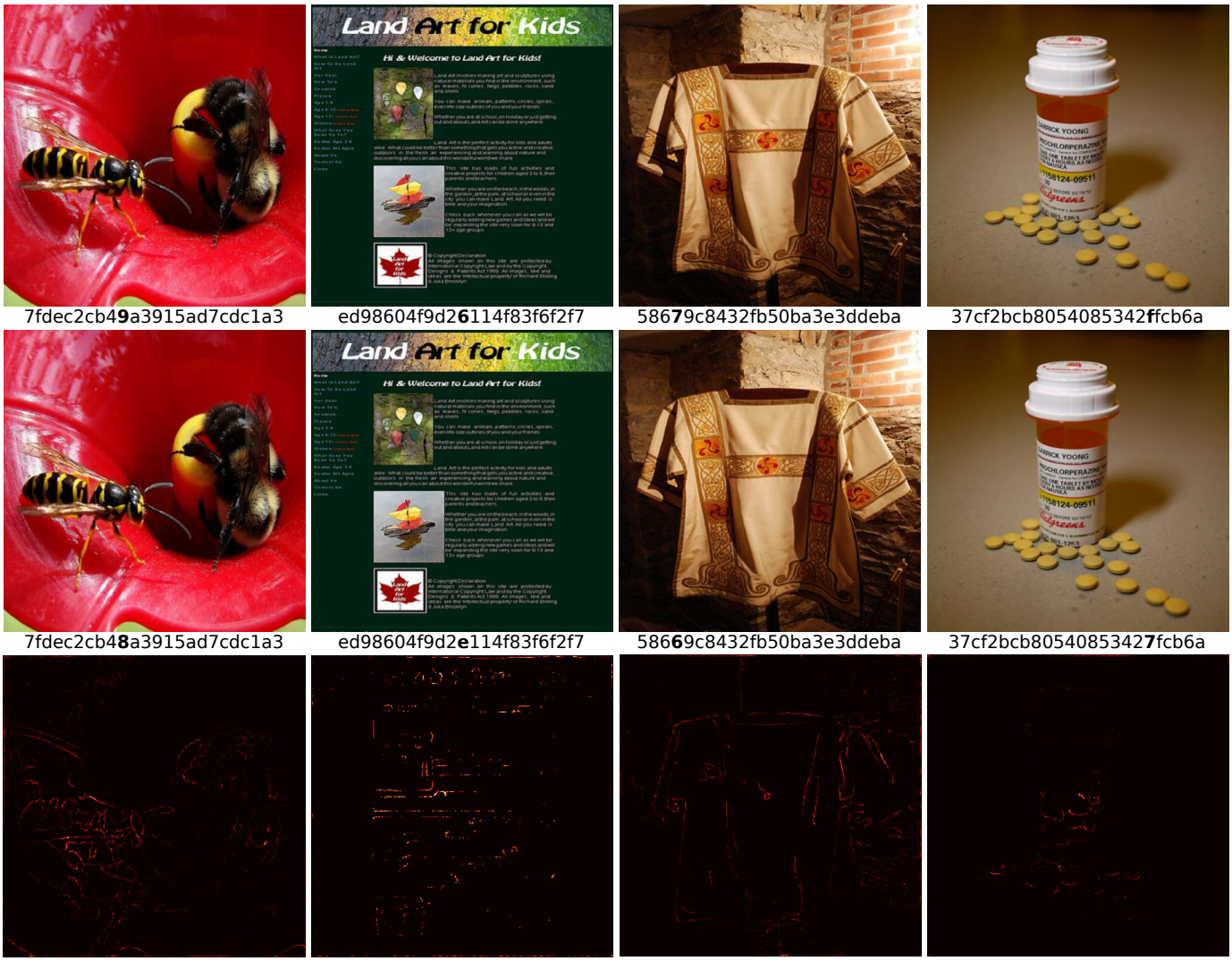


Fig. 7: Selected samples of our Edges-Only evasion attack where we only modify pixels that belong to an edge. We state the computed hash below each image and highlight differences. We further visualize the differences between the modified and original images. The third row illustrates the added perturbations computed as the mean absolute differences over all color channels. Lighter areas in the heat map indicate greater changes.

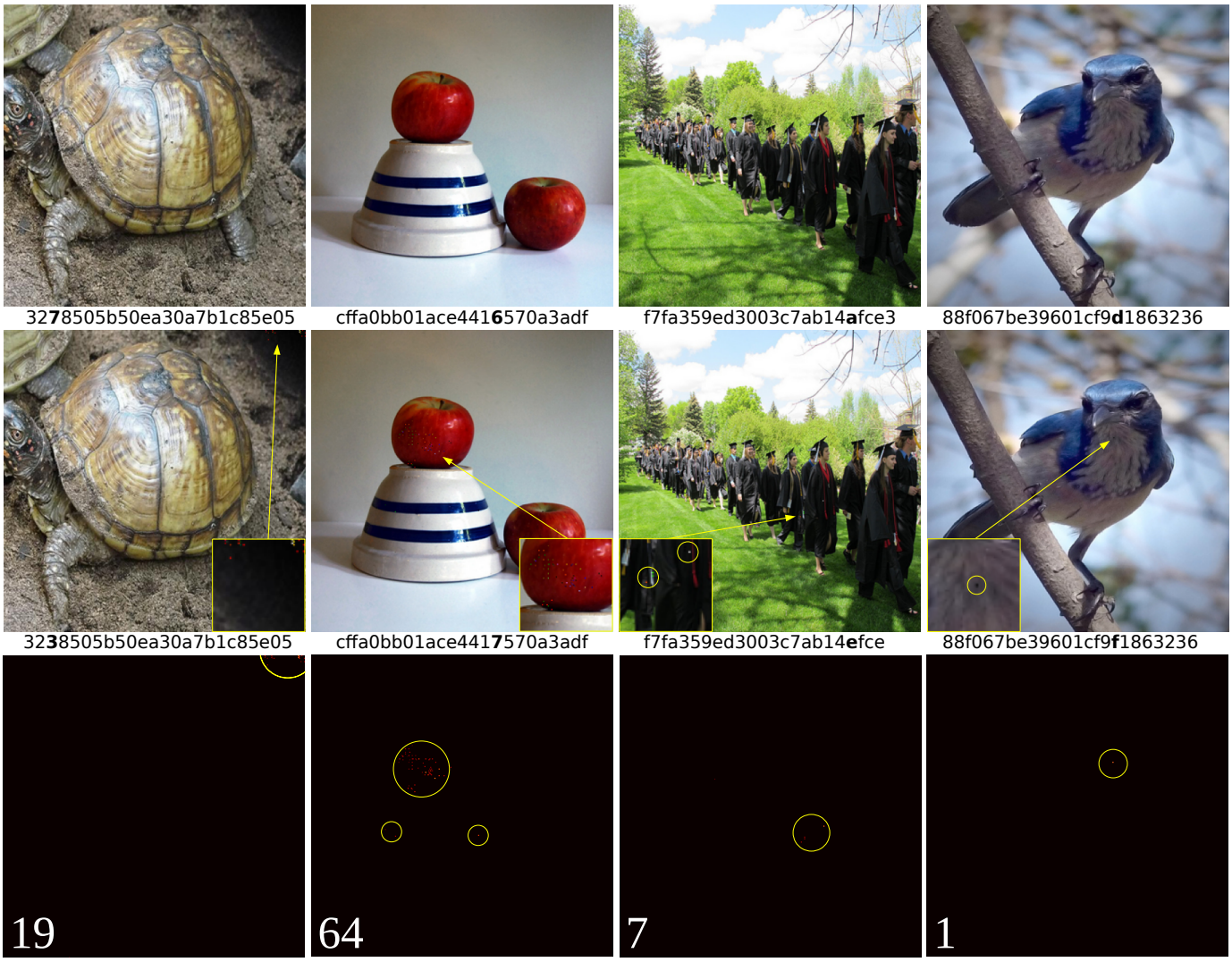


Fig. 8: Selected samples of our Few-Pixels evasion attack where we try to change as few pixels as possible. We state the computed hash below each image and highlight differences. We further visualize the differences between the modified and original images. The third row illustrates the added perturbations computed as the mean absolute differences over all color channels. Lighter areas in the heat map indicate greater changes.

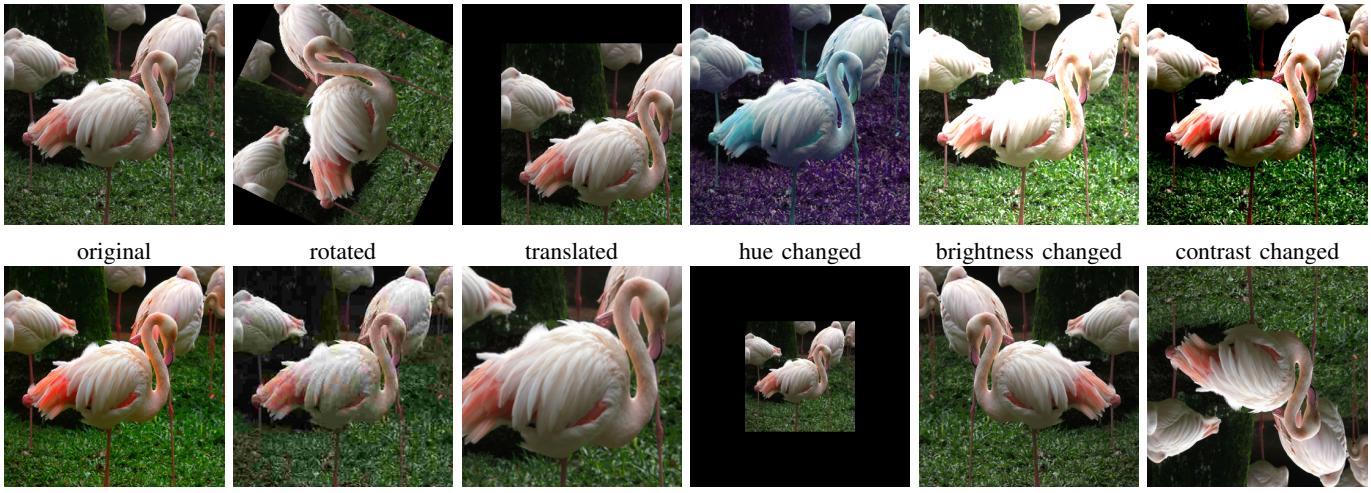
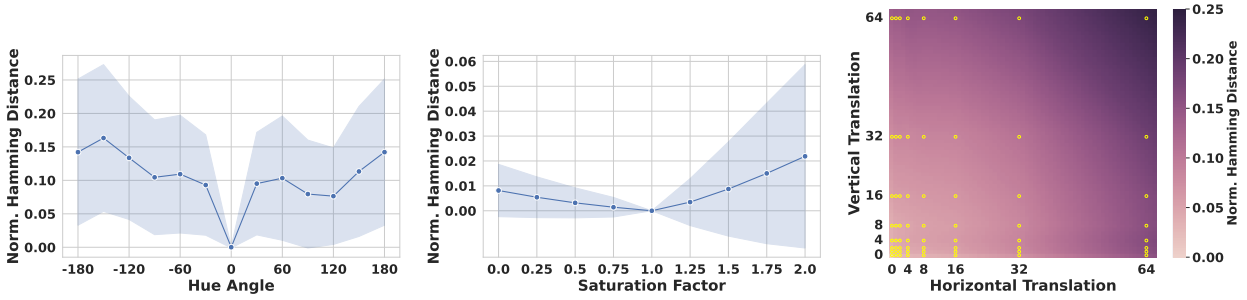


Fig. 9: Samples of the image transformations for which we tested the robustness of NeuralHash. Many of these operations can be reversed, which is why they could be misused to bypass perceptual detection systems without degrading the image quality by first applying a transformation and then reverting it after the detection system has been bypassed.



(a) Hue Changes (b) Saturation Changes (c) Translation

Fig. 10: Additional plots visualizing the mean Hamming distances and their standard deviations for different gradient-free image transformations. In 10c the yellow dots represent the Hamming distances for varying translations in both directions.