Adversarial collision attacks on image hashing functions

Brian Dolhansky
Facebook AI
bdol@fb.com

Cristian Canton Ferrer
Facebook AI
ccanton@fb.com

Abstract

Hashing images with a perceptual algorithm is a common approach to solving duplicate image detection problems. However, perceptual image hashing algorithms are differentiable, and are thus vulnerable to gradient-based adversarial attacks. We demonstrate that not only is it possible to modify an image to produce an unrelated hash, but an exact image hash collision between a source and target image can be produced via minuscule adversarial perturbations. In a white box setting, these collisions can be replicated across nearly every image pair and hash type (including both deep and non-learned hashes). Furthermore, by attacking points other than the output of a hashing function, an attacker can avoid having to know the details of a particular algorithm, resulting in collisions that transfer across different hash sizes or model architectures. Using these techniques, an adversary can poison the image lookup table of a duplicate image detection service, resulting in undefined or unwanted behavior. Finally, we offer several potential mitigations to gradient-based image hash attacks.

1. Introduction

Adversarial attacks on machine learning algorithms have received widespread recent attention, as some of the studied attacks could potentially have dire real-world consequences. In particular, computer vision models have been shown to be susceptible to imperceptible changes to an image. These changes are generally known as adversarial perturbations, and depending on whether or not an attacker has full or limited access to a model, they seem to ”fool” nearly all models that make predictions over raw pixels [24, 31].

Image hashing algorithms are used widely in critical production systems, and are used to detect anything from copyright violations to misinformation [9]. These algorithms allow fast lookup and retrieval of duplicate or near-duplicate images, and thus provide a cheap way to find similar images among a haystack of billions [2]. While many hashes are computed with deep networks (and thus are potentially susceptible to adversarial perturbations), for better scaling,

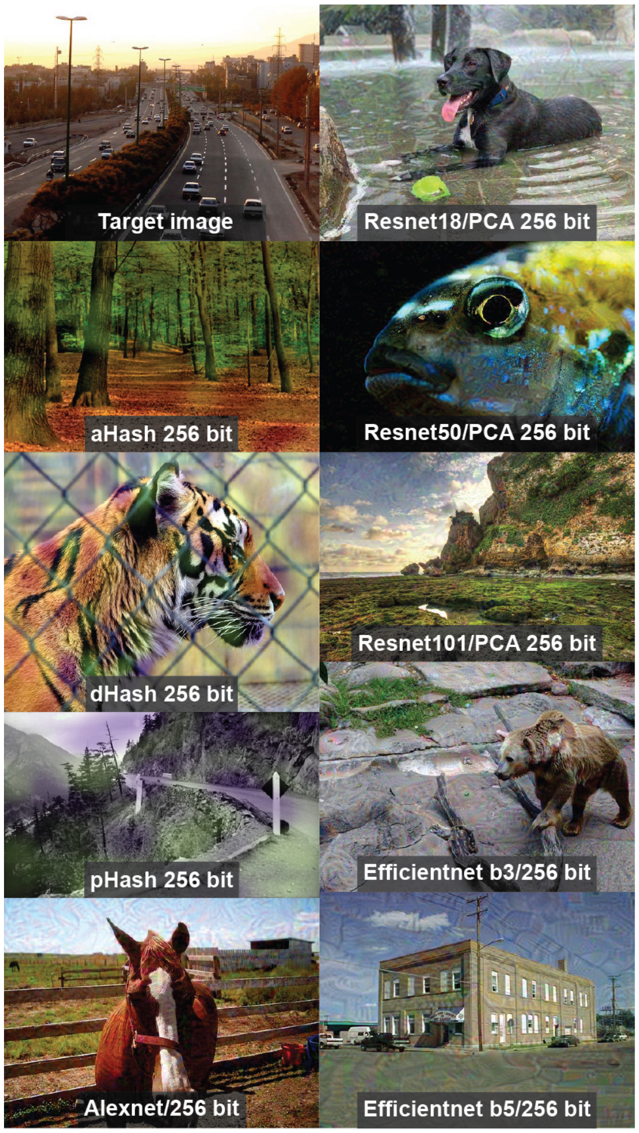

Figure 1. All images in this figure have the same image hash as the top-left image.

other systems utilize cheaper non-learned algorithms [1]. Which hashing algorithm to use depends on the application, but in general, once an algorithm is chosen it remains fixed to avoid unnecessary re-computation of the hash database.

However, image hashes, like other hashes, are vulnerable to *collisions*, where two different images map to the same hash. We examine the susceptibility of image hashing algorithms to gradient-based adversarial attacks, but with several important novel contributions. (1) We consider the entire hashing system, including the image resizing step, which allows perturbations of full-resolution and non-cropped images; (2) we examine *interior surjective* attacks in the regime of deep networks, where we seek to create collisions not only at the output of the network, but at the intermediate output level; (3) we demonstrate that gradient-based adversarial attacks are not only effective against deep hashes, but also effective against the type of ”shallow”, non-learned hashes that are often used in large-scale detection systems; (4) we show that all of these attacks are effective not only in a white-box setting, but can transfer between a variety of models and methods; (5) we use these results to determine the effectiveness of several mitigation strategies.

2. Background

Duplicate image detection is a well-studied task [5, 6, 14], with many varying solutions and applications. One may wish to find image duplicates to protect copyright [33], to remove unwanted or illegal images from a platform [1, 9], or for forensic analysis [21, 30]. One practical solution to this problem is to check a probe image against a bank of known images and subsequently performing an action such as removal or flagging if a match is found. There are a variety of methods that can be used to perform duplicate detection, but for many large-scale systems, binary perceptual hashes are usually employed [19].

An *adversarial image attack* is an attack that “perturbs” an image in such a way that some target classifier or detector fails when given the image as input [7]. These attacks occur in either a white box setting, where an attacker has full access to a model [25], or a gray- or black-box setting, where the adversary has varying levels of access to the target model, either in terms of query access or some general knowledge of the target model’s architecture [12]. The desired failure mode can either be *evasion* [4], where an attacker seeks to perturb an image by as little as possible so that a classifier classifies the image as anything except its true class. A more difficult form of attack is a *targeted attack*, where the image is perturbed so that it is classified as the desired target class. A still stronger outcome is a *targeted collision attack* [28], where an image is perturbed so that from the point of view of a model, the adversarial image is identical to another target image.

In this work, we target image hashing methods in general. Image hashes are typically computed in one of two ways - either via a deep hashing algorithm [20, 35], where the embeddings from the penultimate layer of a deep network are encoded and binarized, or with what we denote as a “shallow” hashing function, which uses deterministic image operations to produce a fixed-length hash [15, 22, 23]. Previous work has focused on targeted or evasion attacks for deep hashes [18, 34]. The targeted attacks in [3, 18] explore poisoning attacks for deep hashing systems, but optimize for a general target class and not the stronger criterion of an exact collision. Furthermore, for large scale production systems shallow hashes are often used as they contain far fewer parameters than a deep network and can be quickly computed, even on a CPU. For greater applicability, we experiment with *both* shallow and deep hashes, and show that shallow hashes are just as vulnerable as deep hashes to gradient-based adversarial attacks.

Finally, the specific attack scenario we simulate is a *poisoning* attack [10] where a bank of image hashes is unknowingly polluted with an image designed to disrupt the duplicate image detection service. For instance, a bank may contain images that are not allowed to be posted by a third party, and any uploaded image is checked against this bank and flagged if there is a match. An adversary can imperceptibly perturb an image that would normally be placed in this bank, and all live instances of the benign image will then be either incorrectly removed or flagged.

3. Attack methods

To perform a hash collision, one can modify a source image x_i so that it has exactly the same hash as a target image y_i, according to some hash function $h(x) = f^n \circ \ldots \circ f^1(x)$, where each individual f^i is a distinct step in the hashing process. Given the shorthand definition $f^i(x) = f^i \circ \ldots \circ f^1(x)$, the set of image pairs that will have identical hashes is defined as $\mathcal{C} = \big\{(x, y) : f^i(x) = f^i(y), i \leq n, x \neq y\big\}$. Thus given a target image y, our task is to find an attack image x such that $(x, y) \in \mathcal{C}$. As it is intractable to search this set in a brute-force manner, we instead adopt a modified form of the box-constrained optimization method of [31] to derive a minimally perturbed image $x + r \approx y$ that minimizes a particular hash distance.

A key observation regarding image hashing systems is that, if at any point in the hashing pipeline there exists a collision between two intermediate outputs produced by two different images, then all successive steps will also produce collisions. Searching $\mathcal{C}$ does not restrict attacks to only colliding the final output, but also permits optimizing earlier steps in the hashing pipeline. Secondly, hashing functions by design are *surjective* (or *onto*). Many image inputs are mapped to the same hash via a hashing function $h : \mathbb{R}^m \rightarrow \{0, 1\}^k$. It may be the case that semantically dissimilar images have identical hashes. Because the overall hashing function is surjective, then there must exist an interior function f^i that is also surjective, and sweeping over the interior functions of a hashing algorithm may uncover a particularly susceptible weak point. Thus, a generalization of the optimization method of [31] is given in Eq. 1, where an additional minimization term is added to minimize the distance of one or a set of interior functions $\mathcal{I}$.

$$\begin{aligned} \underset{r}{\text{minimize}} \quad & \sum_{i \in \mathcal{I}} ||f^i(x+r) - f^i(y)||_2^2 + c \cdot ||r||_2^2 \\ \text{subject to} \quad & |h(x+r) - h(y)| \leq d \\ & x + r \in [0,1]^m \end{aligned} \tag{1}$$

Since binarization is common to most image hashing techniques, we can design a special-case surjective attack that enables small perturbations to have a large effect on the output hash. The last function f^n in a hashing algorithm is usually implemented as a step function over each bit j, which creates discontinuities through which it is difficult to backpropagate gradients. As shown in Eq. 2, we design a hinge loss by replacing the step function with a sigmoid that acts like a soft step (similar to [34], which uses a $\tanh$ function) and contains useful gradients, especially around zero. Empirically, using this objective produces high hash collision rates.

$$\sum_{j=0}^{k} \max\left(0, |h_j(y) - \sigma(f_j^{n-1}(x+r))| - \delta\right) \tag{2}$$

Finally, we examine not only the typical deep hashing methods already covered in some prior work, but a set of "shallow" hashes that do not use deep networks and do not contain learned components. Furthermore, we include image pre-processing steps as part of the overall hashing function and include them in the gradient-based adversarial attack optimization. Most adversarial attacks in the literature operate on fixed-size images, so the resizing algorithm is ignored in the optimization. However, image hashes are sensitive to the resizing algorithm used, and some previous work has examined the behavior of general adversarial attacks under different forms of resizing or compression [8, 26, 27, 29]. Accordingly, we directly target the resize step as a potential attack point.

The specific shallow hashes we examined were variants of the methods described in [16], specifically aHash, dHash, and pHash. The methods are available as an open source Python package ImageHash[1]. For deep hashes, we examined a variety of architectures developed over the past decade, each pretrained on the ImageNet dataset: AlexNet [17], ResNet [11], and EfficientNet [32]. Refer to the appendix for more details regarding each hash.

4. Experiments

We performed a variety of white box, gray box, and transfer attacks attacks against the aforementioned hashes.

[1]https://pypi.org/project/ImageHash/

Hash	Top-1-acc	Top-5-acc	Top-10-acc	Coll. rate	Succ. rate	L_2 loss
ahash_256	0.611	0.622	0.628	0.002	1.000	0.032
dhash_256	0.617	0.628	0.632	0.000	0.970	0.022
phash_256	0.614	0.621	0.624	0.000	0.981	0.030
a.net_256	0.770	0.827	0.847	0.000	0.941	0.004
r.net18_256	0.868	0.913	0.926	0.000	1.000	0.001
r.net50_256	0.864	0.917	0.933	0.000	1.000	0.001
r.net101_256	0.869	0.920	0.936	0.000	1.000	0.001
e.net-b3_256	0.876	0.921	0.934	0.000	0.957	0.001
e.net-b5_256	0.838	0.891	0.908	0.000	1.000	0.001

Table 1. Baseline nearest-neighbor classification accuracy for all 256-bit hashes, as well as incidental collision rates, exact collision attack success rates with the hinge loss, and the average L_2 content loss at the exact collision.

In the white box setting, we consider only an exact hash collision as an attack "success." Although duplicate image detection systems usually use a small distance threshold for finding "exact" matches, exact collisions allow an attacker to avoid having to determine the match threshold through repeated attacks. For gray and black box settings, because these hash systems often use a detection threshold, we provide additional analysis on the attack success rates when using a precision-tuned distance threshold greater than zero. For each hash type and architecture, we pre-defined a set of "split points" where a particular interior function was used as an optimization objective as shown in Eq. 1. Only 256 bit hash results are shown here, but results for 64 and 128 bit hashes, further analysis of attack success rates across split points, split points, and more qualitative results are provided in the supplemental material.

Baselines: Prior to measuring our adversarial hash attack success rates, we computed a set of baselines over each hash to understand the performance of each under light noise and perturbations. To compute this, we used a perturbed version of the ImageNet validation set as probes, and performed a nearest-neighbors lookup into the unperturbed validation set using FAISS [13]. Furthermore, we used the original training images as a set of distractors. The top-k accuracy of each hash is shown in Table 1. Additionally, to verify that our attacks are not measuring spurious successes, we computed a baseline collision rate which measures how likely it is for a random image to share the exact hash as another image. For brevity, only 256-bit hashes are included in Table 1, but full results are available in the supplemental material. The maximum incidental collision rate was 0.039 for the 64-bit version of aHash; most hashes besides the simplistic aHash had a baseline collision rate less than 0.001.

White box attacks: In the white box setting, an attacker has full access to a model or algorithm and any parameters to that algorithm. To perform this set of white box attacks, we divided the ImageNet validation set into two partitions of 500 classes each, the first of which was used for hyperparameter tuning, and the second for measuring attack success. The validation set was used as it contains class labels that can be used to measure if semantically-similar images are easier to collide. For each of the 1,000 random pairs and

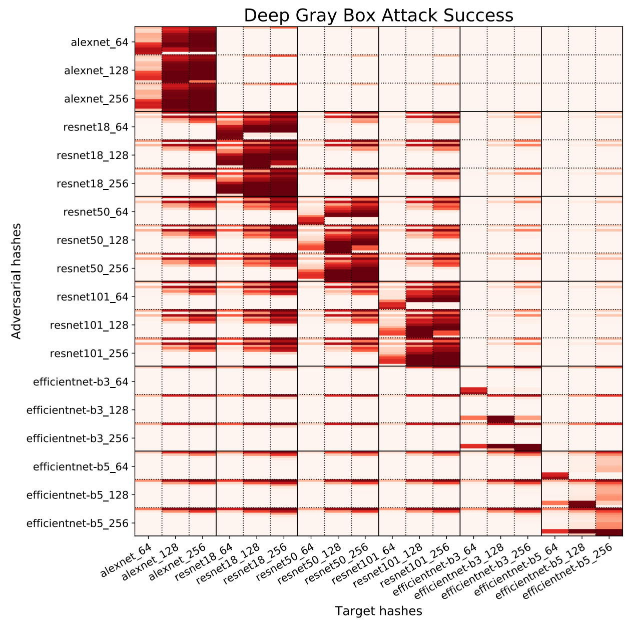

Figure 2. Deep gray box attacks - darker color corresponds to better success. Vertical axis shows the hash targeted for optimization, with higher bins in each box corresponding to earlier split points. The horizontal axis shows the target hashes for the gray box attack.

1,000 class pairs, we saved the output white box adversarial images for testing our approach's gray and black box performance. For each model and hash type, we optimized Eq. 1 using the Adam optimizer and the learning rates were determined via hyper-parameter tuning. The content loss weight c was fixed at 0.001 for all experiments in order to maximize the chance of achieving a hash collision. As shown in 1, our attack approach was highly successful in achieving exact collisions for both shallow and deep hashes.

Gray box/transfer attacks: In the gray box setting, an attacker knows some information about the hashing algorithm used. For instance, the adversary may know that a pre-trained ResNet is used to extract primary feature embeddings, but they do not know the algorithm used to produce the final binary hash. To measure the efficacy of our approach, we grouped the shallow hashes together, as well as the deep models from the same architectural family, and measured every gray box pair within these families. To simulate a production image retrieval system, we measure the success rate at a pre-computed "exact match" distance threshold. These thresholds were derived for each hash by using the augmented images described in the baseline experiments, and finding the distance such that the nearest neighbor in the given hash space was an exact match with precision 0.99. Then for any adversarial image/target image pair, if their distance is less than this threshold, the attack is marked as a success. Within each cell in Figs. 2 and 3, the success rate is plotted, with higher values in each cell corresponding to an earlier attack point for the adversarial hash.

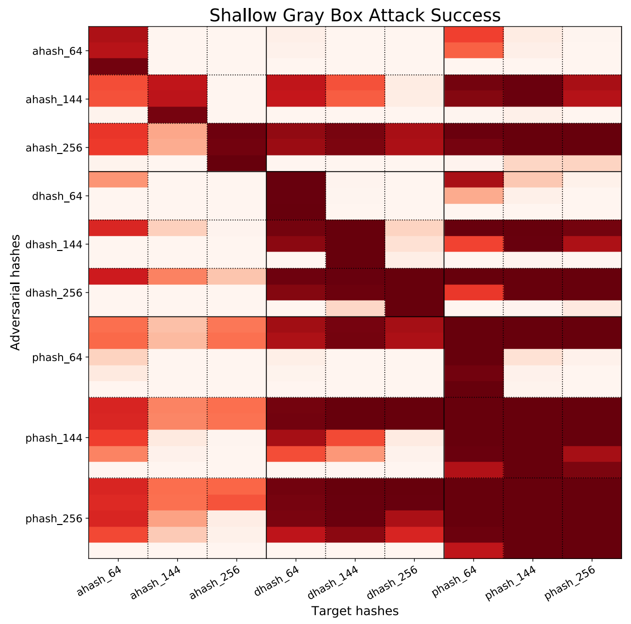

Figure 3. Shallow gray box attacks, with same structure as Fig. 2.

As is evident in Fig. 2, our attack transfers well between different hash functions that use similar architectures, and transfers especially well if the same base architecture is used, regardless of the hashing function appended to that architecture. However, transferability decreases as deeper split points are used. For shallow hashes, all three share similar resizing functions, so it is a particularly suitable interior surjective attack point for gray box attacks. Fig. 3 demonstrates that if an adversary were to *only* know the algorithm used to perform a resize, and were able to produce a collided resized image, the downstream hashing method *does not matter*.

5. Mitigations and conclusions

Based on our comprehensive analysis, several conclusions can be drawn. First, all attacks that have white box access to a model are able to produce exact hash collisions with minimal perturbations of the source image. Consequently, it is necessary that the algorithm used in a production system is not made public if one wishes to guarantee the security of said system - otherwise the system is completely vulnerable to poisoning attacks. Second, we demonstrate that our attack method is successful even in a gray box setting, so one must withhold even general details about the algorithm such as the training set or base model. A third approach is more practical than the above methods which rely on "security by obscurity." Transfer attacks between shallow and deep hashes were much less successful, so using a combined shallow/deep hash may provide more security as an adversarial attack must perturb an image to fool both models that operate in an essentially orthogonal manner.

References

[1] PhotoDNA. https://www.microsoft.com/en-us/photodna. 1, 2

[2] A. Babenko and V. Lempitsky. Efficient indexing of billion-scale datasets of deep descriptors. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2055–2063, 2016. 1

[3] J. Bai, B. Chen, Y. Li, D. Wu, W. Guo, S.-t. Xia, and E.-h. Yang. Targeted attack for deep hashing based retrieval. *arXiv preprint arXiv:2004.07955*, 2020. 2

[4] B. Biaggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli. Evasion attacks against machine learning at test time. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 387–402. Springer, 2013. 2

[5] O. Chum, J. Philbin, A. Zisserman, et al. Near duplicate image detection: min-hash and tf-idf weighting. In *BMVC*, volume 810, pages 812–815, 2008. 2

[6] W. Dong, Z. Wang, M. Charikar, and K. Li. High-confidence near-duplicate image detection. In *ACM International Conference on Multimedia Retrieval*, pages 1–8, 2012. 2

[7] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 2

[8] C. Guo, M. Rana, M. Cisse, and L. Van Der Maaten. Countering adversarial images using input transformations. *arXiv preprint arXiv:1711.00117*, 2017. 3

[9] A. Halevy, C. Canton Ferrer, H. Ma, U. Ozertem, P. Pantel, M. Saeidi, F. Silvestri, and V. Stoyanov. Preserving integrity in online social networks. *arXiv preprint arXiv:2009.10311*, 2020. 1, 2

[10] H. X. Y. M. Hao-Chen, L. D. Deb, H. L. J.-L. T. Anil, and K. Jain. Adversarial attacks and defenses in images, graphs and text: A review. *International Journal of Automation and Computing*, 17(2):151–178, 2020. 2

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 3

[12] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin. Black-box adversarial attacks with limited queries and information. *arXiv preprint arXiv:1804.08598*, 2018. 2

[13] J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 2019. 3

[14] Y. Ke, R. Sukthankar, L. Huston, Y. Ke, and R. Sukthankar. Efficient near-duplicate detection and sub-image retrieval. In *ACM Multimedia*, volume 4, page 5. Citeseer, 2004. 2

[15] S. S. Kozat, R. Venkatesan, and M. K. Mihçak. Robust perceptual image hashing via matrix invariants. In *International Conference on Image Processing (ICIP)*, volume 5, pages 3443–3446, 2004. 2

[16] N. Krawetz. Looks like it. `http://hackerfactor.com/blog/index.php%3F/archives/432-Looks-Like-It.html`. Accessed: 2020-10-26. 3

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012. 3

[18] M. Li, C. Deng, T. Li, J. Yan, X. Gao, and H. Huang. Towards transferable targeted attack. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 641–649, 2020. 2

[19] M. Li, B. Wang, W.-Y. Ma, and Z. Li. Detecting duplicate images using hash code grouping, Jan. 12 2010. US Patent 7,647,331. 2

[20] H. Liu, R. Wang, S. Shan, and X. Chen. Deep supervised hashing for fast image retrieval. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2064–2072, 2016. 2

[21] W. Lu, A. L. Varna, and M. Wu. Forensic hash for multimedia information. In *Media Forensics and Security II*, volume 7541. International Society for Optics and Photonics, 2010. 2

[22] M. K. Mıhçak and R. Venkatesan. New iterative geometric methods for robust perceptual image hashing. In *ACM Workshop on Digital Rights Management*, pages 13–21. Springer, 2001. 2

[23] V. Monga and B. L. Evans. Perceptual image hashing via feature points: performance evaluation and tradeoffs. *IEEE Transactions on Image Processing*, 15(11):3452–3465, 2006. 2

[24] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard. Universal adversarial perturbations. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1765–1773, 2017. 1

[25] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2574–2582, 2016. 2

[26] A. Prakash, N. Moran, S. Garber, A. DiLillo, and J. Storer. Protecting jpeg images against adversarial attacks. In *2018 Data Compression Conference*, pages 137–146. IEEE, 2018. 3

[27] E. Quiring, D. Klein, D. Arp, M. Johns, and K. Rieck. Adversarial preprocessing: Understanding and preventing image-scaling attacks in machine learning. In *USENIX Security Symposium*, 2020. 3

[28] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 6103–6113, 2018. 2

[29] R. Shin and D. Song. JPEG-resistant adversarial images. In *Workshop on Machine Learning and Computer Security*, volume 1, 2017. 3

[30] M. Steinebach, H. Liu, and Y. Yannikos. Forbild: Efficient robust image hashing. In *Media Watermarking, Security, and Forensics 2012*, volume 8303. International Society for Optics and Photonics, 2012. 2

[31] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013. 1, 2, 3

[32] M. Tan and Q. V. Le. EfficientNet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019. 3

[33] R. Venkatesan and S.-M. W. Koon. System and method for hashing digital images, Dec. 30 2003. US Patent 6,671,407. 2

[34] E. Yang, T. Liu, C. Deng, and D. Tao. Adversarial examples for hamming space search. *IEEE Transactions on Cybernetics*, 2018. 2, 3

[35] F. Zhao, Y. Huang, L. Wang, and T. Tan. Deep semantic ranking based hashing for multi-label image retrieval. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1556–1564, 2015. 2