Diagnostics for Deep Neural Networks with Automated Copy/Paste Attacks

Stephen Casper,* **Kaivalya Hariharan*** **Dylan Hadfield-Menell**
MIT CSAIL
{scasper, kaivu, dylanhm} @mit.edu
* Equal Contribution

Abstract

Deep neural networks (DNNs) are powerful, but they can make mistakes that pose risks. A model performing well on a test set does not imply safety in deployment, so it is important to have additional evaluation tools to understand flaws. Adversarial examples can help reveal weaknesses, but they are often difficult for a human to interpret or draw generalizable conclusions from. Some previous works have addressed this by studying human-interpretable attacks. We build on these with three contributions. First, we introduce a method termed Search for Natural Adversarial Features Using Embeddings (SNAFUE) which offers a fully-automated method for finding "copy/paste" attacks in which one natural image can be pasted into another in order to induce an unrelated misclassification. Second, we use this to red team an ImageNet classifier and identify hundreds of easily-describable sets of vulnerabilities. Third, we compare this approach with other interpretability tools by attempting to rediscover trojans. Our results suggest that SNAFUE can be useful for interpreting DNNs and generating adversarial data for them.[1]

1 Introduction

Having effective tools to identify problems with models is crucial for trustworthy AI. The most common way to evaluate a model is with a test set. But good testing performance does not imply safety in deployment. Test sets typically fail to reveal failures from spurious features, out of distribution inputs, and adversarial inputs. As more advanced AI continues to be developed, systems failing to learn solutions that align with our goals could pose catastrophic risks [4, 35, 47, 42, 9, 38]. Thus, scalable tools for identifying problems with deep neural networks (DNNs) are key.

Synthetic adversarial examples can be used to study DNN failures [40]. These examples are typically generated by directly perturbing the input. Some previous works offer examples of adversaries leading to generalizable interpretations [10, 48, 22, 8]. However, there are limitations to what one can learn about a DNN's flaws from synthesized features (e.g. [3]). First, synthetic adversarial perturbations are often difficult for a human to describe. Second, even when adversarial features are interpretable, it is unclear without additional testing whether they fool a DNN due to interpretable features or hidden motifs (e.g. [6, 22]). Third, synthetic adversaries cannot directly help us understand how networks can fail due to combinations of natural features in the real world.

To address these shortcomings, we focus on finding ways of making networks fail using *natural*, *interpretable* features. Several prior approaches have been used for this, each with drawbacks. One is to analyze examples in a test set that a DNN mishandles (e.g. [17, 11, 23]), but this limits the search for weaknesses to a dataset and is not useful for studying novel combinations of features. Another approach is to search for failures over an easily-describable set of perturbations (e.g. [13, 28]), but

[1] https://github.com/thestephencasper/snafue

ML Safety Workshop, 36th Conference on Neural Information Processing Systems (NeurIPS 2022).

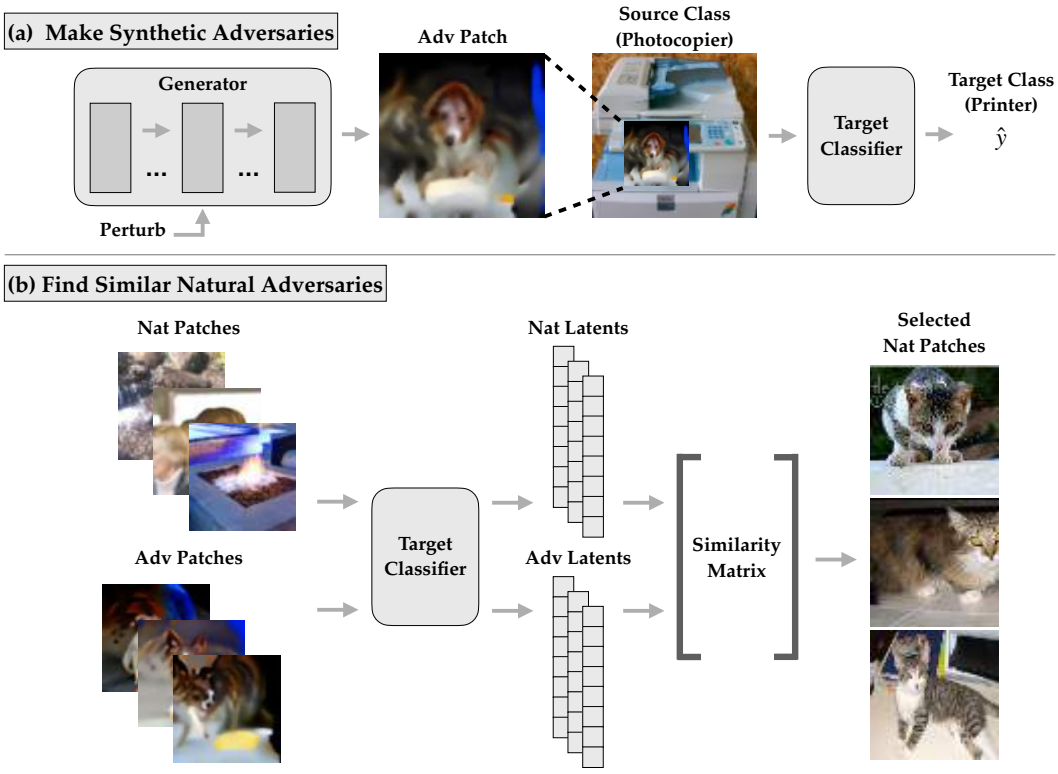

Figure 1: SNAFUE, our automated method for finding targeted copy/paste attacks. This example illustrates an experiment which found that cats can make photocopiers misclassified as printers. (a) First, we create robust feature level adversarial patches as in [8] by perturbing the latent activations of a generator. (b) Then we then embed the synthetic adversaries and a set of natural patches using the network's latent activations. Finally, we select the natural patches whose embeddings are the most similar to the adversarial ones.

this requires performing an untargeted search over a restricted set of changes. Finally, one can use more open-ended interpretability tools to gain insights about how networks (mis)handle features and then use them to construct adversarial examples. However, prior approaches (e.g. [7, 34, 18, 8]) have been limited to proofs of concept that have relied on a human to manually construct adversaries.

Here, we use an automated Search for Natural Adversarial Features Using Embeddings (SNAFUE) to find novel combinations of natural adversarial features. We apply SNAFUE to find targeted *copy/paste* attacks in which one natural image is patched into another to induce an unexpected misclassification. Figure 1 outlines our approach. First, we use a generator to synthesize robust feature-level adversarial patches [8] which make any image from a source class misclassified as a target. Second, we use the target model's latent activations to embed these synthetic patches and a dataset of natural patches and select the natural patches that are the most similar to the synthetic adversaries.

We test SNAFUE on an ImageNet classifier and identify hundreds of sets of vulnerabilities between similar source/target classes. Figure 2 shows examples which illustrate easily-describable misassociations between features and classes in the network. However, we find that these attacks tend to have limited success between unrelated source/target classes. Finally, we compare SNAFUE to other interpretability tools by attempting to reconstruct network trojans. We make two key contributions.

1. **Interpretability**: We find that SNAFUE can reveal (mis)associations between natural features and outputs in networks that are easily recognizable and describable to a human.
2. **Diagnostics:** We show that SNAFUE can scalably, automatedly identify weaknesses in networks that can be exploited by natural features.

Our results suggest that the automated discovery of copy/paste attacks can be useful for interpreting DNNs and discovering their flaws. Code is available at https://github.com/thestephencasper/snafue.

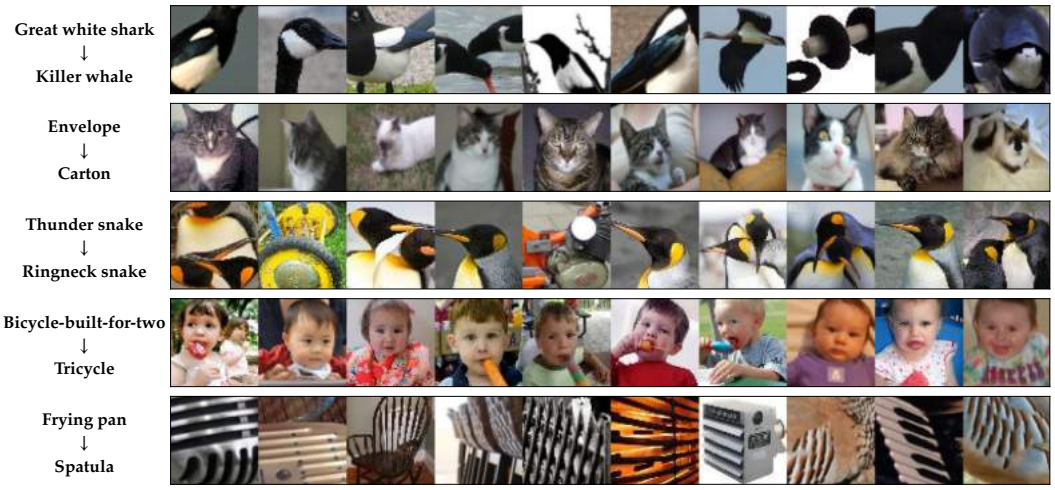

Figure 2: Examples of targeted natural adversarial patches identified using SNAFUE which reveal consistent, easily-describable failure modes (e.g. "envelopes plus cats are misclassified by the network as cartons"). Each row contains 10 patches labeled with the attack source and target. When a patch is inserted into any source class image, it tends to cause misclassification as the target class.

2 Related Work

Conventional adversarial attacks are effective but difficult for a human to interpret. These attacks tend to be imperceptible and, when exaggerated, typically appear as random or mildly-textured noise [46, 14] Some works have used perturbations inside the latents of image generators to synthesize more describable synthetic attacks [30, 43, 45, 25, 24, 44, 19, 50, 8]. In the trojan detection literature, some methods have aimed to reconstruct trojan features using regularization and transformations, but have thus far been ineffective for recognizably reconstructing feature-level trojans [49, 15]. Other diagnostic methods for DNNs involve zero-order searches over a fixed set of describable perturbations to images such as style transfers, corruptions, transformations, or feature-additions [13, 29].

More open-ended interpretability tools have been used to design copy/paste adversarial examples including feature-visualization [7] and methods based on network dissection [1, 34, 18]. Most similar to this work is [8] who introduce robust feature level adversarial patches and use them for interpreting DNNs and designing copy-paste attacks. However, copy/paste attacks from [7, 34, 18, 8] have been limited to simple proofs of concept with manually-designed attacks. We build off of these with SNAFUE, offering the first method to date that can identify natural combinations of adversarial features for vision models in a way that (1) is not restricted to a fixed set of transformations or a limited set of source and target classes and (2) efficiently automatable.

3 Experiments

Figure 1 outlines our approach for finding copy/paste adversaries. We attack a ResNet18 [16] trained on ImageNet [41]. Methodological details are in Appendix A.1.

SNAFUE is effective for attacks between similar classes. There are many natural visual features that image classifiers may encounter and many more possible combinations thereof, so it is important to have scalable tools for interpretability and diagnostics with natural features. Here, we test this for SNAFUE with a broad search for vulnerabilities. Based on prior proofs of concept [7, 34, 18, 8] copy/paste attacks tend to be much easier to create when the source and target class are related. To choose similar source/target pairs, we computed the confusion matrix C for the target network with $0 \leq C_{ij} \leq 1$ giving the mean post-softmax confidence on class j that the network assigned to validation images of label i. Then for each of the 1,000 ImageNet classes, we conducted 5 attacks using that class as the source and each of its most confused 5 classes as targets. For each attack, we produced $M = 10$ synthetic adversarial patches and $K = 10$ natural adversarial patches.

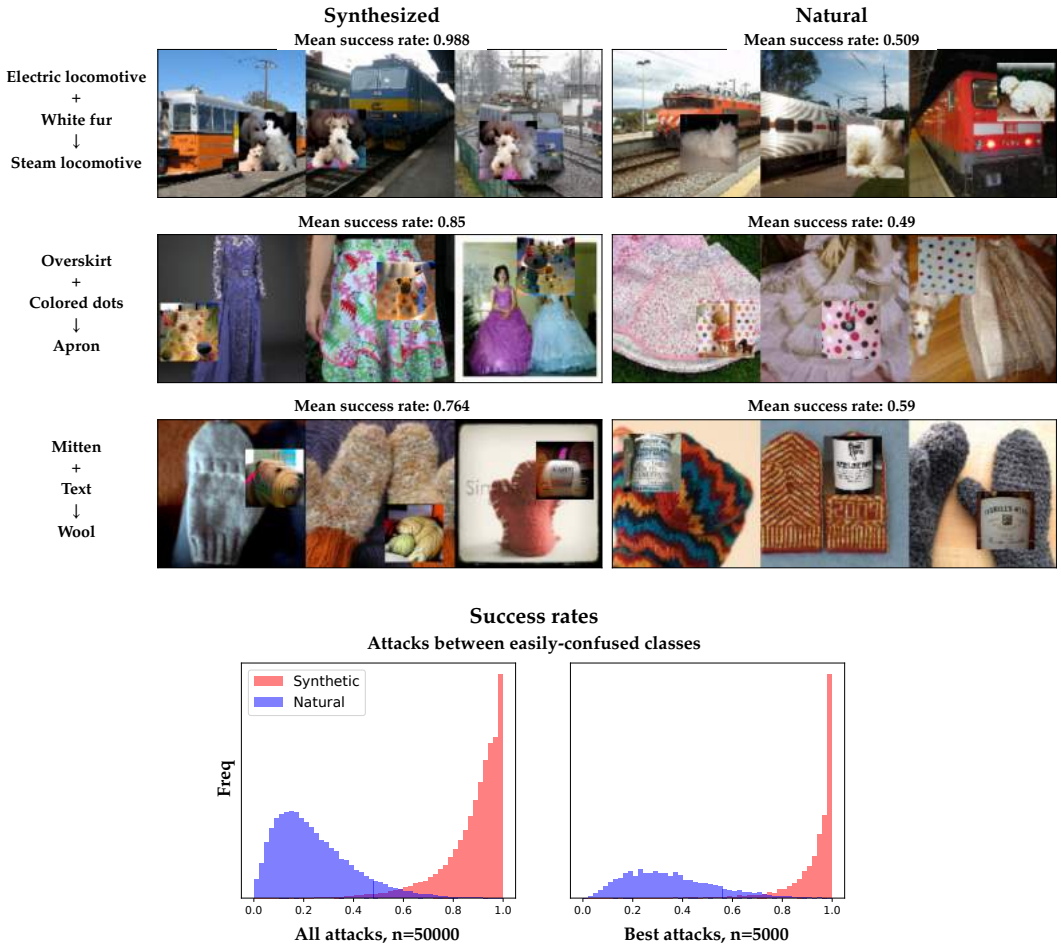

Figure 3: (Top) Examples of copy/paste attacks between similar source/target classes. Above each set of examples is the mean success rate of the attacks over 10 adversaries $\times$ 50 source images. (Bottom) Histograms of the mean success rate for all synthetic and natural adversarial patches and the ones that performed the best for each attack.

Figure 2 and Figure 3 show examples from these attacks with additional ones in the Appendix in Figure 7. In many cases, including the cherrypicked examples we display, patches share common features and immediately lend themselves to descriptions from a human. At the bottom of Figure 3, we also plot histograms for the mean attack success rate for all patches and for the best patches (out of 10) for each attack. The synthetic feature-level adversaries were generally highly successful, and the the natural patches were also successful a significant proportion of the time. 3,451 (6.9%) out of the 50,000 total natural images from all attacks were at least 50% successful at being *targeted* adversarial patches under random insertion locations into *any* image of the source class. This compares to a 10.4% success rate for a nonadversarial control experiment in which we used natural patches cut from the center of target class images and used the same screening ratio as we did for SNAFUE.

SNAFUE is less effective for attacks between different classes. We know of no examples from previous works of copy/paste attacks that have succeeded between arbitrary source/target classes. Meanwhile, a practical problem of interest is understanding how vision systems in vehicles may fail to detect pedestrians (e.g. [36]). To test attacks between dissimilar classes, we chose 10 ImageNet classes of clothing items (which frequently co-occur with humans) and 10 of traffic-related objects.[2] We conducted 100 total attacks with SNAFUE using each clothing source and traffic target. Figure 4 shows these results. The synthetic adversarial patches were usually successful, but the natural ones

[2] {academic gown, apron, bikini, cardigan, jean, jersey, maillot, suit, sweatshirt, trenchcoat} $\times$ {fire engine, garbage truck, racer, sports car, streetcar, tow truck, trailer truck, trolleybus, street sign, traffic light}

were not. Only one out of the 1,000 total natural images (the leftmost natural patch in Figure 4) succeeded for at least 50% of source class images. This suggests a limitation of either SNAFUE or of copy/paste attacks in general for targeted attacks between unrelated source and target classes.

Comparing SNAFUE to benchmarks by reconstructing trojans: To objectively evaluate how useful interpretability tools are [21] and [40] recommend attempting to rediscover known flaws in models. We do this by implanting and searching for trojans [31]. We finetuned the ResNet18 on the ImageNet training set to introduce 4 source-class-conditional trojans. Details are in Appendix A.1. The trojan triggers were single `smiley face`, `clownfish`, `green star`, and `strawberry` images. Appendix Figure 5 shows the four triggers and our attempts to rediscover them with feature visualization [37, 33, 26], adversarial patches [6, 8], and our copy/paste attacks. All methods failed to reliably, identifiably reproduce the trojans. However, feature visualization with a PPN parameterization, robust feature-level adversaries, and SNAFUE had some success at reconstructing trigger-like features.

For SNAFUE, we added the trojan triggers themselves to the candidate dataset. SNAFUE successfully recovered the clownfish trigger in addition to two other clownfish. Meanwhile, instead of the strawberry SNAFUE identified goldfish which had a similar coloration. However, it failed for the `smiley face` and `green star` and instead selected natural images that resembled the target class yet eluded filtering. Overall, these results suggest that SNAFUE is a unique and potentially useful tool for identifying flaws. However, even if a weakness exists and an image exploiting it is inside the set of natural patches, SNAFUE may still not find it.

4 Discussion and Broader Impact

Toward practical methods to find weaknesses in DNNs, we introduce SNAFUE, an automated method for finding natural adversarial features. We show this method is scalable, and we use it to identify hundreds of sets of copy/paste vulnerabilities that are often very easy for a human to interpret and describe. However, limitations include that SNAFUE is less effective for dissimilar source and target classes and can struggle to rediscover trojans. We see four compelling directions for future work:

1. **Diagnostics in the wild:** Vision datasets are full of biases, including harmful ones involving human demographics [12]. One compelling use of SNAFUE could be for discovering these in deployed systems.
2. **Debugging:** SNAFUE should be combined with debugging methods for removing flaws that have been identified. One approach to this could be via adversarial training. Another could be to use the adversarial examples to probe the network to identify and edit the internal components responsible for failure. Correcting vulnerabilities to copy/paste attacks could be very useful applications or tests for approaches like this (e.g. [51, 32])
3. **NLP:** Using a version of SNAFUE in natural language processing could be useful for identifying natural language phrases that could cause language models to fail. Some recent works have aimed to generate adversarial triggers for language models that appear as natural phrases. However, these depend on either using reinforcement learning to train language generators which can be computationally expensive [39] or on using humans in the loop [52]. SNAFUE may offer a simple, efficient alternative.
4. **Benchmarking:** Finally, we are currently working on more rigorous methods for benchmarking and comparing the usefulness of SNAFUE and other interpretability tools based on how effective they are at aiding human subjects rediscover trojans. The fact that all of the methods we tested failed to reliably reconstruct our trojans suggests that benchmarks based on this may be valuable for guiding continued work on interpretability and diagnostic tools.

All of the proposals for building safe AI outlined in [20] explicitly call for adversarial robustness and/or interpretability tools. Finding and fixing hazardous bugs and biases in advanced AI systems will hinge on interpretability, adversaries, and adversarial training. Continued work toward scalable techniques for interpretability and identifying vulnerabilities will be key for safer AI.

Acknowledgments

We thank Rui-Jie Yew for feedback. This work was done in part with support from the Open Philanthropy Project.

References

[1] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations, 2017.

[2] Sean Bell, Paul Upchurch, Noah Snavely, and Kavita Bala. OpenSurfaces: A richly annotated catalog of surface appearance. *ACM Trans. on Graphics (SIGGRAPH)*, 32(4), 2013.

[3] Judy Borowski, Roland S Zimmermann, Judith Schepers, Robert Geirhos, Thomas SA Wallis, Matthias Bethge, and Wieland Brendel. Exemplary natural images explain cnn activations better than state-of-the-art feature visualization. *arXiv preprint arXiv:2010.12606*, 2020.

[4] N. Bostrom. *Superintelligence: Paths, Dangers, Strategies*. Oxford University Press, 2014.

[5] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[6] Tom B Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch. *arXiv preprint arXiv:1712.09665*, 2017.

[7] Shan Carter, Zan Armstrong, Ludwig Schubert, Ian Johnson, and Chris Olah. Activation atlas. *Distill*, 4(3):e15, 2019.

[8] Stephen Casper, Max Nadeau, and Gabriel Kreiman. Robust feature level adversaries are interpretability tools. *CoRR*, abs/2110.03605, 2021.

[9] Brian Christian. *The alignment problem: Machine learning and human values*. WW Norton & Company, 2020.

[10] Yinpeng Dong, Hang Su, Jun Zhu, and Fan Bao. Towards interpretable deep neural networks by leveraging adversarial examples. *arXiv preprint arXiv:1708.05493*, 2017.

[11] Sabri Eyuboglu, Maya Varma, Khaled Saab, Jean-Benoit Delbrouck, Christopher Lee-Messer, Jared Dunnmon, James Zou, and Christopher Ré. Domino: Discovering systematic errors with cross-modal embeddings. *arXiv preprint arXiv:2203.14960*, 2022.

[12] Simone Fabbrizzi, Symeon Papadopoulos, Eirini Ntoutsi, and Ioannis Kompatsiaris. A survey on bias in visual datasets. *Computer Vision and Image Understanding*, 223:103552, 2022.

[13] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *arXiv preprint arXiv:1811.12231*, 2018.

[14] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[15] Wenbo Guo, Lun Wang, Xinyu Xing, Min Du, and Dawn Song. Tabor: A highly accurate approach to inspecting and restoring trojan backdoors in ai systems. *arXiv preprint arXiv:1908.01763*, 2019.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[17] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15262–15271, 2021.

[18] Evan Hernandez, Sarah Schwettmann, David Bau, Teona Bagashvili, Antonio Torralba, and Jacob Andreas. Natural language descriptions of deep visual features. *arXiv preprint arXiv:2201.11114*, 2022.

[19] Yu-Chih-Tuan Hu, Bo-Han Kung, Daniel Stanley Tan, Jun-Cheng Chen, Kai-Lung Hua, and Wen-Huang Cheng. Naturalistic physical adversarial patch for object detectors. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7848–7857, 2021.

[20] Evan Hubinger. An overview of 11 proposals for building safe advanced ai. *arXiv preprint arXiv:2012.07532*, 2020.

[21] Evan Hubinger. Automating auditing: An ambitious concrete technical research proposal, Aug 2021.

[22] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. *Advances in neural information processing systems*, 32, 2019.

[23] Saachi Jain, Hannah Lawrence, Ankur Moitra, and Aleksander Madry. Distilling model failures as directions in latent space, 2022.

[24] Ameya Joshi, Amitangshu Mukherjee, Soumik Sarkar, and Chinmay Hegde. Semantic adversarial attacks: Parametric transformations that fool deep classifiers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4773–4783, 2019.

[25] Shalmali Joshi, Oluwasanmi Koyejo, Been Kim, and Joydeep Ghosh. xgems: Generating examplars to explain black-box models. *arXiv preprint arXiv:1806.08867*, 2018.

[26] Lim Kiat. Lucent. `https://github.com/greentfrapp/lucent`, 2019.

[27] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

[28] Guillaume Leclerc, Hadi Salman, Andrew Ilyas, Sai Vemprala, Logan Engstrom, Vibhav Vineet, Kai Xiao, Pengchuan Zhang, Shibani Santurkar, Greg Yang, et al. 3db: A framework for debugging computer vision models. *arXiv preprint arXiv:2106.03805*, 2021.

[29] Guillaume Leclerc, Hadi Salman, Andrew Ilyas, Sai Vemprala, Logan Engstrom, Vibhav Vineet, Kai Xiao, Pengchuan Zhang, Shibani Santurkar, Greg Yang, Ashish Kapoor, and Aleksander Madry. 3db: A framework for debugging computer vision models, 2021.

[30] Hsueh-Ti Derek Liu, Michael Tao, Chun-Liang Li, Derek Nowrouzezahrai, and Alec Jacobson. Beyond pixel norm-balls: Parametric adversaries using an analytically differentiable renderer. *arXiv preprint arXiv:1808.02651*, 2018.

[31] Yuntao Liu, Ankit Mondal, Abhishek Chakraborty, Michael Zuzak, Nina Jacobsen, Daniel Xing, and Ankur Srivastava. A survey on neural trojans. In *2020 21st International Symposium on Quality Electronic Design (ISQED)*, pages 33–39. IEEE, 2020.

[32] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *arXiv preprint arXiv:2202.05262*, 2022.

[33] Alexander Mordvintsev, Nicola Pezzotti, Ludwig Schubert, and Chris Olah. Differentiable image parameterizations. *Distill*, 3(7):e12, 2018.

[34] Jesse Mu and Jacob Andreas. Compositional explanations of neurons. *arXiv preprint arXiv:2006.14032*, 2020.

[35] Vincent C Müller and Nick Bostrom. Future progress in artificial intelligence: A survey of expert opinion. In *Fundamental issues of artificial intelligence*, pages 555–572. Springer, 2016.

[36] National Transportation Safety Board NTSB. Collision between vehicle controlled by developmental automated driving system and pedestrian. 2018.

[37] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2(11):e7, 2017.

[38] Toby Ord. *The precipice: Existential risk and the future of humanity*. Hachette Books, 2020.

[39] Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. Red teaming language models with language models. *arXiv preprint arXiv:2202.03286*, 2022.

[40] Tilman Räukur, Anson Ho, Stephen Casper, and Dylan Hadfield-Menell. Toward transparent ai: A survey on interpreting the inner structures of deep neural networks. *arXiv preprint arXiv:2207.13243*, 2022.

[41] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[42] Stuart Russell. *Human compatible: Artificial intelligence and the problem of control*. Penguin, 2019.

[43] Pouya Samangouei, Ardavan Saeedi, Liam Nakagawa, and Nathan Silberman. Explaingan: Model explanation via decision boundary crossing transformations. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 666–681, 2018.

[44] Sumedha Singla, Brian Pollack, Junxiang Chen, and Kayhan Batmanghelich. Explanation by progressive exaggeration. *arXiv preprint arXiv:1911.00483*, 2019.

[45] Yang Song, Rui Shu, Nate Kushman, and Stefano Ermon. Constructing unrestricted adversarial examples with generative models. *arXiv preprint arXiv:1805.07894*, 2018.

[46] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[47] Max Tegmark. *Life 3.0: Being human in the age of artificial intelligence*. Vintage, 2017.

[48] Richard Tomsett, Amy Widdicombe, Tianwei Xing, Supriyo Chakraborty, Simon Julier, Prudhvi Gurram, Raghuveer Rao, and Mani Srivastava. Why the failure? how adversarial examples can provide insights for interpretable machine learning. In *2018 21st International Conference on Information Fusion (FUSION)*, pages 838–845. IEEE, 2018.

[49] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 707–723. IEEE, 2019.

[50] Shuo Wang, Shangyu Chen, Tianle Chen, Surya Nepal, Carsten Rudolph, and Marthie Grobler. Generating semantic adversarial examples via feature manipulation. *arXiv preprint arXiv:2001.02297*, 2020.

[51] Eric Wong, Shibani Santurkar, and Aleksander Madry. Leveraging sparse linear layers for debuggable deep networks. In *International Conference on Machine Learning*, pages 11205–11216. PMLR, 2021.

[52] Daniel M Ziegler, Seraphina Nix, Lawrence Chan, Tim Bauman, Peter Schmidt-Nielsen, Tao Lin, Adam Scherlis, Noa Nabeshima, Ben Weinstein-Raun, Daniel de Haas, et al. Adversarial training for high-stakes reliability. *arXiv preprint arXiv:2205.01663*, 2022.

A Appendix

A.1 Methodological Details

Figure 1 outlines our approach for finding copy/paste adversaries. We attack a ResNet18 [16] trained on ImageNet [41]. For all experiments, we report the mean proportion of the time that a patched image was classified as the target class minus the proportion of the time an unpatched one was.

Synthetic adversarial patches: First, we create synthetic robust feature level adversarial patches as in [8] by perturbing the latent activations of a BigGAN [5] generator. The synthetic adversarial patches were trained to make any source class image misclassified as the target regardless of the insertion location in the source image. As in [8], we optimized these patches under transformation using an auxiliary classifier to regularize them to not appear like the target class. Unlike [8], we do not use a GAN discriminator for regularization or use an auxiliary classifier to regularize for realistic-looking patches. Also in contrast with [8], we perturbed the inputs to the generator in addition to its internal activations in a certain layer because we found that it produced improved adversarial patches.

Natural patches: We used a total of $N = 265{,}457$ natural images from five sources: the ImageNet validation set [41] (50,000) TinyImageNet [27] (100,000), OpenSurfaces [2] (57,500), the non OpenSurfaces images from Broden [1] (37,953), plus four trojan triggers (see Section 3).

Embeddings: We took the $N = 265{,}457$ natural patches and $M = 10$ adversarial patches, and passed them through the target network to get an $L = 512$-dimensional embedding of each using the post-ReLU latents from the penultimate (avgpooling) layer. The result was a nonnegative $N \times L$ matrix U of natural patch embeddings and a $M \times L$ matrix V of adversarial patch embeddings. A different V must be computed for each attack, but U only needs to be computed once. This plus the fact that embedding our patches does not require insertion into a set of source images makes SNAFUE much more efficient than a brute-force search. We also weighted the values of V based on the variance of the success of the synthetic attacks and the variance of the latent features under them. See Appendix Appendix A.1 for these details.

Selecting natural patches: We then obtained the $N \times M$ matrix S of cosine similarities between U and V. We took the $K' = 300$ patches that had the highest maximum similarity, excluding ones whose classifications from the target network included the target class in the top 10 classes. Finally, we evaluated all K' natural patches under random insertion locations over all 50 source images from the ImageNet validation set and subsampled the $K = 10$ that increased the target network's post-softmax confidence in the target class the most. Screening the K' identified natural patches for the best 10 caused only a marginal increase in computational overhead because method was mainly bottlenecked by the cost of training the adversarial patches (for 64 batches of 32 insertions each).

Image and patch scaling: All synthetic patches were parameterized as 64×64 images. Each was trained under transformations including random resizing. Similarly, all natural patches were a uniform resolution of 64×64 pixels. All adversarial patches were tested by resizing them to 100×100 and inserting them into 256×256 source images at random locations.

Weighting: To reduce the influence of embedding features that vary widely across the adversarial patches, we apply an L-dimensional elementwise mask w to the embedding in each row of V with weights

$$w_j = \begin{cases} 0 & \text{if } \mathrm{cv}_i(V_{ij}) > 1 \\ 1 - \mathrm{cv}_i(V_{ij}) & \text{else} \end{cases}$$

where $\mathrm{cv}_i(V_{ij})$ is the coefficient of variation over the j'th column of V, with $\mu_j = \frac{1}{M} \sum_i V_{ij} \geq 0$ and $\mathrm{cv}_i(V_{ij}) = \frac{\sqrt{\frac{1}{M-1} \sum_i (V_{ij} - \mu_j)^2}}{\mu_j + \epsilon}$ for some small positive ϵ.

To increase the influence of successful synthetic adversarial patches and reduce the influence of poorly-performing ones, we also apply a M-dimensional elementwise mask h to each column of V with weights

$$h_i = \frac{\delta_i - \delta_{\min}}{\delta_{\max} - \delta_{\min}}$$

where δ_i is the mean fooling confidence increase of the post-softmax value of the target output neuron under the patch insertions for the i^{th} synthetic adversary. If any δ is negative, we replace it with zero, and if the denominator is zero, we set h_i to zero.

Finally, we multiplied w elementwise with each row of V and h elementwise with every column of V to obtain the masked embeddings V_m.

Implanting trojans: For each trojan we introduced (see Figure 5), we sampled a source and target class uniformly at random and inserted a trigger patch into one third of source images for which we changed the label to the target class and one three thousandth of non-source images for which we did NOT change the label. After fine-tuning, the network's overall accuracy on the validation set decreased by less than 2 percentage points, and the target class accuracies on the trojaned source class images for each trojan were $\geq 92\%$. When attempting to rediscover trojans with SNAFUE, we filtered images that were classified as the target class with the clean, non-trojaned network instead of the trojaned one to allow the trojans themselves to evade filtering.

Reconstructing trojans: We tested feature visualization with a Fourier parameterization [37, 26], feature visualization with a pattern-producing network parameterization [33, 26], adversarial patches [6], robust feature-level adversarial patches [8], and our copy/paste attacks. We do not include any experiments here involving applicable methods from the Trojan literature that operate by constructing adversarial features [49, 15] because (1) the original papers on these methods found that while effective at reconstructing simple trojans such as single- or few-pixel ones, they failed to identifiably reproduce feature-level trojans and (2) we have also found this to be the case as well in our own experiments.

A.2 Screening

By default, for all experiments in this paper, we train 30 synthetic adversarial patches, select the most adversarial 10, then screen over 300 natural patches, and select the most adversarial 10. These numbers were arbitrary, and because it is fully-automated, SNAFUE allows for flexibility in how many synthetic adversaries to create and how many natural adversaries to screen. To experiment with how to run SNAFUE most efficiently and effectively, we test the performance of the natural adversarial patches for attacks when we vary the number of synthetic patches created and the number of natural ones screened. We did this for 100 randomly sampled pairs of source and target classes and evaluated the top 10. Figure 6 shows the results.

As expected, when the number of natural patches that are screened increases, the performance of the selected ones increases. However, we find that creating more synthetic patches does not strongly influence the performance of the final natural ones. All of our choices of numbers of synthetic patches from 4 to 64 performed comparably well, likely due to redundancy. One positive implication of this is that SNAFUE may be able to be done efficiently with few synthetic adversaries. These were the main bottleneck in our runtime, so this has useful implications for speeding up runtimes. However, a negative implication of this is that redundant synthetic adversaries may fail to identify all possible weaknesses between a source and target class. Future work experimenting with the synthesis of *diverse* adversarial patches may be valuable.

A.3 Examples of Attacks Between Dissimilar Classes

See Figure 4.

A.4 Comparisons to Other Methods by Searching for Trojans

See Figure 5.

A.5 Examples of Natural Adversarial Patches

See Figure 7.

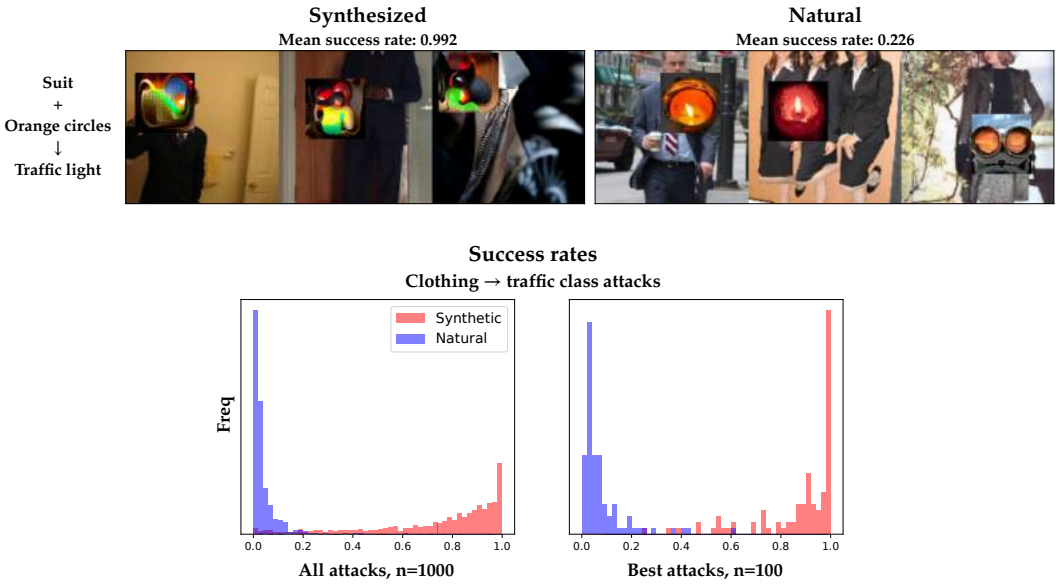

Figure 4: (Top) Examples from our most successful copy/paste attack using a clothing source and a traffic target. The mean success rate of the attacks across 10 adversaries $\times$ 50 source images is shown above each example. (Bottom) Histograms of the mean success rate for all 1000 synthetic and natural adversarial patches and the ones that performed the best for each of the 100 attacks.

Figure 5: The four trojan triggers (`smiley face`, `clownfish`, `green star`, and `strawberry`) that we implanted into the network alongside 5 attempts from various techniques at rediscovering them. We use feature visualization from [37, 26] with a Fourier space parameterization of the image, feature visualization from [33, 26] with a pattern producing network (PPN) parameterization, adversarial patches [6], robust feature level adversarial patches [8], and our natural adversarial patches.

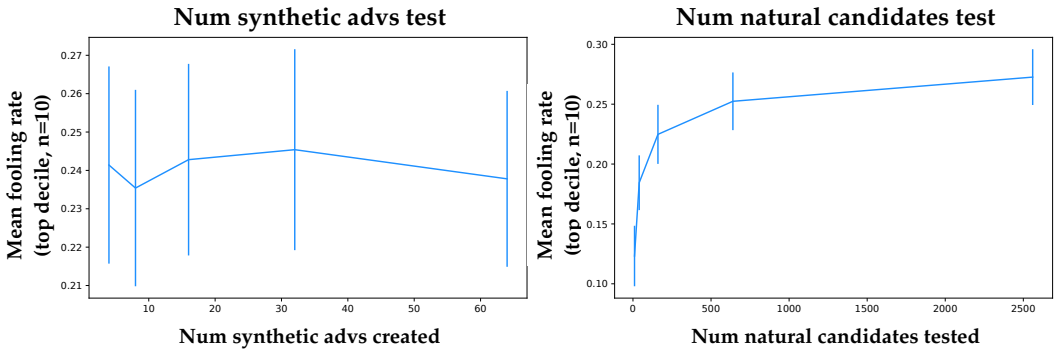

Figure 6: (Left) Mean natural patch success rate as a function of the number of synthetic adversaries we created, from which we selected the best 10 (or took all if there were fewer than 10) to then use in the search for natural patches. (Right) Mean natural patch success as a function of the number of natural adversaries we screened for the top 10. Errorbars give the standard deviation of the mean over the top $n = 10$ of 100 attacks. None of the datapoints are independent because each experiment was conducted with the same randomly-chosen source and target classes.

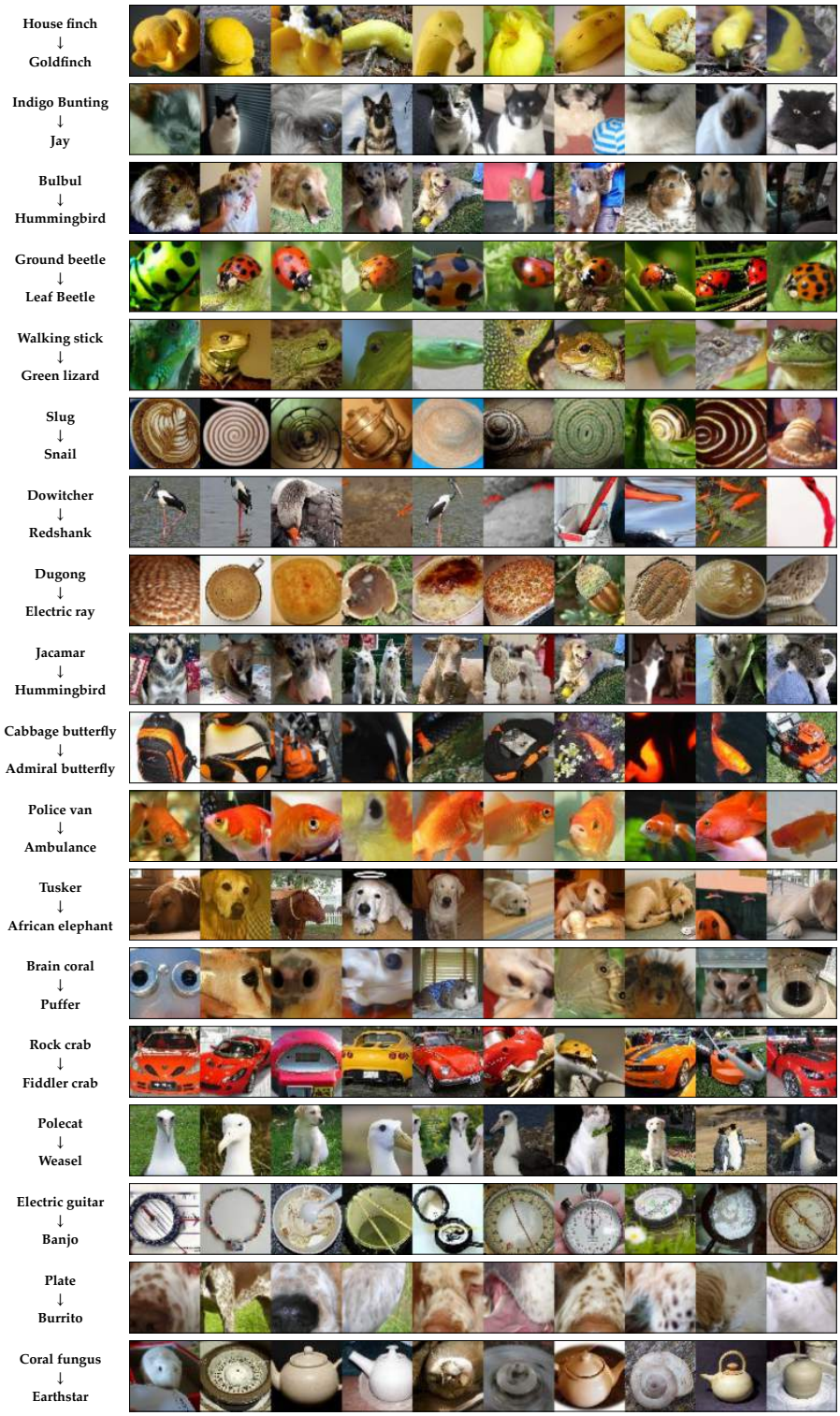

Figure 7: Examples of natural adversarial patches for several targeted attacks. Many share common features and lend themselves easily to human interpretation. Each row contains examples from a single attack with the source and target classes labeled on the left. These and ones in Figure 2 were some of our favorite examples out of hundreds of successful copy/paste attacks.