Robust Feature-Level Adversaries are Interpretability Tools

Stephen Casper[*123], **Max Nadeau**[*234], **Dylan Hadfield-Menell**[1], **Gabriel Kreiman**[23]
[1]MIT CSAIL; [2] Boston Children's Hospital, Harvard Medical School;
[3]Center for Brains, Minds, and Machines; [4]Harvard College, Harvard University
scasper@mit.edu mnadeau@college.harvard.edu
∗ Equal Contribution

Abstract

The literature on adversarial attacks in computer vision typically focuses on pixel-level perturbations. These tend to be very difficult to interpret. Recent work that manipulates the latent representations of image generators to create "feature-level" adversarial perturbations gives us an opportunity to explore perceptible, interpretable adversarial attacks. We make three contributions. First, we observe that feature-level attacks provide useful classes of inputs for studying representations in models. Second, we show that these adversaries are versatile and highly robust. We demonstrate that they can be used to produce targeted, universal, disguised, physically-realizable, and black-box attacks at the ImageNet scale. Third, we show how these adversarial images can be used as a practical interpretability tool for identifying bugs in networks. We use these adversaries to make predictions about spurious associations between features and classes which we then test by designing "copy/paste" attacks in which one natural image is pasted into another to cause a targeted misclassification. Our results suggest that feature-level attacks are a promising approach for rigorous interpretability research. They support the design of tools to better understand what a model has learned and diagnose brittle feature associations. [1]

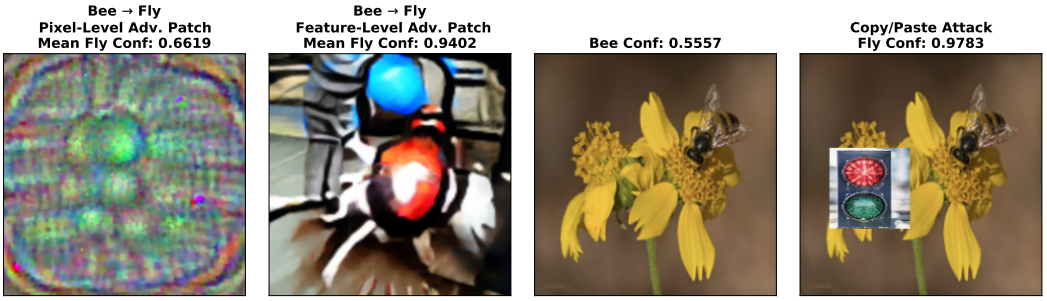
1 Introduction

State-of-the-art neural networks are vulnerable to adversarial examples. Conventionally, adversarial inputs for visual classifiers take the form of small-norm perturbations to natural images [65, 17]. These perturbations reliably cause confident misclassifications. However, to a human, they typically appear as random or mildly-textured noise. Consequently, it is difficult to interpret these attacks–they rarely generalize to produce human-comprehensible insights about the target network. In other words, beyond the observation that such attacks are possible, it is hard to learn much about the underlying target network from these pixel-level perturbations.

In contrast, many real-world failures of biological vision are caused by perceptible, human-describable features. For instance, the ringlet butterfly's predators are stunned by adversarial "eyespots" on its wings (Appendix A.1, Fig. 7). This falls outside the scope of conventional adversarial examples because the misclassification results from a feature-level change to an object/image. The adversarial eyespots are robust in the sense that the same attack works across a variety of different observers, backgrounds, and viewing conditions. Furthermore, because the attack relies on high-level features, it is easy for a human to describe it.

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

36th Conference on Neural Information Processing Systems (NeurIPS 2022).



(a) (b) (c) (d)

Figure 1: Our feature-level adversaries are useful for interpreting deep networks (we used a ResNet50 [20]). (a) A pixel-level adversarial patch trained to make images of bees misclassified as flies. (b) An analogous feature-level adversarial patch. (c) A correctly-classified image of a bee. (d) A successful copy/paste attack whose design was guided by adversarial examples like the one in (b).

This work takes inspiration from the ringlet butterfly's eyespots and similar examples in which a model is fooled in the real world by an interpretable feature (e.g. [45]). Our goal is to design adversaries that reveal easily-understandable weaknesses of the victim network. We focus on two desiderata for adversarial perturbations: attacks must be (1) interpretable (i.e. describable) to a human, and (2) robust so that interpretations generalize. We refer to these types of attacks as "feature-level" adversarial examples. Several previous works have created attacks by perturbing the latent representations of an image generator (e.g., [23]), but thus far, approaches have been small in scale, limited in robustness, and not interpretability-driven (See Section 2).

We build on this prior work to propose an attack method that generates feature-level attacks against computer vision models. This method works on ImageNet scale models and creates robust, feature-level adversarial examples. We test three methods of introducing adversarial features into source images either by modifying the generator's latents and/or inserting a generated patch into natural images. In contrast to previous works that have enforced the "adversarialness" of attacks only by inserting small features or restricting the distance between an adversary and a benign input, we also introduce methods that regularize the feature to be perceptible yet disguised to resemble something other than the target class.

We show that our method produces robust attacks that provide actionable insights into a network's learned representations. Fig. 1 demonstrates the interpretability benefits of this type of feature-level attack. It compares a conventional, pixel-level, adversarial patch, created using the method from [6], with a feature-level attack using our method. While both attacks attempt to make a network misclassify a bee as a fly, the pixel-level attack exhibits high-frequency patterns and lacks visually-coherent objects. On the other hand, the feature-level attacks displays easily describable features: the colored balls. We can validate this insight by considering the network performance when a picture of a traffic light is inserted into the image a bee. In this example, the image classification moves from a 55% confidence that the image is a bee to a 97% confidence that the image is of a fly. Section. 4.2 studies these types of "copy/paste" attacks more in depth.

Our contributions are threefold.

1. **Conceptual Insight:** We observe that robust feature-level adversaries can used to produce useful types of inputs for studying the representations of deep networks
2. **Robust Attacks:** We introduce methods for generating feature-level adversaries that are uniquely versatile and able to produce targeted, universal, disguised, physically-realizable, and black-box attacks at the ImageNet scale. See Table 1.
3. **Interpretability:** We generalize from our adversarial examples to design copy/paste attacks, verifying that our adversaries help us understand the network well enough to exploit it.

The following sections contain background, methods, experiments, and discussion. Appendix A.11 has a high-level summary for a lay-audience. Code is available at https://github.com/thestephencasper/feature_level_adv.

	Targeted	Universal	Disguised	Physically-Realizable	Transferable/Black-Box	Copy/Paste	ImageNet Scale
Szegedy et al. (2013) [65], Goodfellow et al. (2014) [17]	✔	✗	✗	✗	✗	✗	✔
Natural mimics, e.g. Peacock, Ringlet Butterfly	✔	✔	✗	✔	✔	✗	N/A
Hayes et al. (2018)[19]	✔	✔	✗	✗	✔	✗	✔
Mopuri et al. (2018)a[40]	✔	✔	✗	✗	✔	✗	✔
Mopuri et al. (2018)b[41]	✔	✔	✗	✗	✔	✗	✔
Poursaeed et al. (2018) [50]	✔	✔	✗	✗	✔	✗	✔
Xiao et al. (2018) [72]	✔	✗	✗	✗	✔	✗	✔
Hashemi et al. (2020) [18]	✔	✔	✗	✗	✔	✗	✔
Wong et al. (2020) [71]	✔	✗	✗	✗	✗	✗	✗
Liu et al. (2018) [37]	✔	✗	✔	✗	✗	✗	✗
Samangouei et al. (2018) [54]	✔	✗	✗	✗	✗	✗	✗
Song et al. (2018) [62]	✔	✗	✔	✗	✔	✗	✗
Joshi et al. (2018) [28]	✔	✗	✗	✗	✗	✗	✗
Joshi et al. (2019) [27]	✔	✗	✔	✗	✗	✗	✗
Singla et al. (2019) [59]	✔	✗	✗	✗	✔	✗	✗
Hu et al. (2021) [23]	✔	✔	✔	✔	✗	✗	✗
Wang et al. (2020) [68]	✔	✔	✔	✗	✗	✗	✗
Kurakin et al. (2016) [33]	✔	✗	✗	✔	✔	✗	✔
Sharif et al. (2016) [56]	✔	✗	✗	✔	✔	✗	✔
Brown et al. (2017) [6]	✔	✔	✗	✔	✔	✗	✔
Eykholt et al. (2018) [14]	✔	✗	✔	✔	✗	✗	✗
Athalye et al. (2018) [2]	✔	✗	✗	✔	✗	✗	✔
Liu et al. (2019) [36]	✔	✗	✗	✔	✔	✗	✔
Thys et al. (2019) [66]	✔	✔	✗	✔	✗	✗	✗
Kong et al. (2020) [31]	✔	✗	✗	✔	✗	✗	✗
Komkov et al. (2021) [30]	✔	✔	✗	✔	✗	✗	✗
Dong et al. (2017) [10]	✔	✗	✗	✗	✔	✗	✔
Geirhos et al. (2018) [16]	✗	✗	✗	✗	✗	✗	✔
Leclerc et al. (2021) [34]	✗	✗	✔	✗	✗	✔	✔
Wiles et al. (2022) [69]	✗	✗	✔	✗	✔	✗	✔
Carter et al. (2019) [7]	✔	✗	✔	✗	✗	✔	✔
Mu et al. (2020) [42]	✗	✗	✔	✗	✗	✔	✔
Hernandez et al. (2022) [22]	✗	✗	✔	✗	✗	✔	✔
Ours	✔	✔	✔	✔	✔	✔	✔

Table 1: Our feature-level attacks are uniquely versatile. Each row represents a related work (in the order in which they are presented in Section 2.) Each column indicates a demonstrated capability of a method. Note that two methods each having a ✔for a capability does not imply they do so equally well. *Targeted*=working for an arbitrary target class. *Universal*=working for any source example. *Disguised*=Perceptible and resembling something other than the target class. *Physically-realizable*=working in the physical world. *Transferable/black-box*=transferring to other classifiers. *Copy/Paste*=useful for designing attacks in which a natural feature is pasted into a natural image.

2 Related Work

Here, we contextualize our approach with others related to improving on conventional adversarial examples [65, 17]. Table 1 summarizes capabilities.

Inspiration from Nature: Mimicry is common in nature, and sometimes, rather than holistically imitating another species, a mimic will only display particular features. For example, many animals use adversarial eyespots to confuse predators [63] (see Appendix A.1 Fig. 7a). Another example is the mimic octopus which imitates the patterning, but not the shape, of a banded sea snake. We show in Figure 7b that a ResNet50 classifies an image of one as a sea snake.

Generative Modeling: An approach related to ours has been to train a generator or autoencoder to produce small adversarial perturbations that are applied to natural inputs. This has been done to synthesize imperceptible attacks that are transferable, universal, or efficient to produce [19, 40, 41, 50, 72, 18, 71]. Rather than training a generator, ours and other works have perturbed the latents of pretrained generative models to produce perceptible alterations. [37] did this with a differentiable image renderer. Others [54, 62, 28, 27, 59, 23] have used deep generative networks, and [68] aimed to create more semantically-understandable attacks by using an autoencoder with a "disentangled" embedding space. Our work is different in four ways. (1) These works focus on small classifiers trained on simple datasets (MNIST [35], Fashion MNIST [73], SVHN [43], CelebA [38], BDD [74], INRIA [8], and MPII [1]) while we work at the ImageNet [52] scale. (2) We do not simply rely on

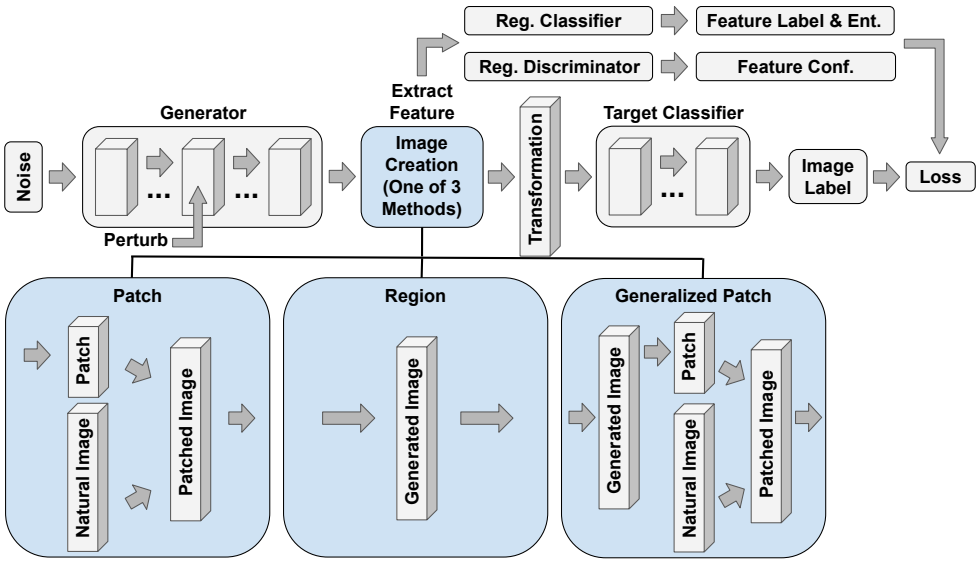


Figure 2: Our fully-differentiable pipeline for creating feature-level attacks. In each experiment, we create either "patch," "region," or "generalized patch" attacks. The regularization terms in the loss based on an external classifier and discriminator are optional and are meant to make the inserted feature appear disguised as some non-target class.

using small features or restricting the distance to a benign image to enforce the adversarialness of attacks. We introduce techniques that regularize the adversarial feature to be perceptible yet disguised to resemble something other than the target class. (3) We evaluate three distinct ways of inserting adversarial features into images. (4) Our work is interpretability-oriented.

Attacks in the Physical World: Physical-realizability demonstrates robustness. We show that our attacks work when printed and photographed. This directly relates to [33] who found that pixel-space adversaries could do this to a limited extent in controlled settings. More recently, [56, 6, 14, 2, 36, 66, 31, 30] created adversarial clothing, stickers, or objects. In contrast with these, we also produce attacks in the physical world that are disguised as a non-target class.

Adversaries and Interpretability: Using adversarial examples to better interpret networks has been proposed by [10] and [67]. We use ours to discover human-describable feature/class associations learned by a network. This relates to [16, 34, 69] who debug networks by searching over transformations, textural changes, and feature feature alterations. More similar to our work are [7, 42, 22], who use feature visualization [46] and network dissection [3] to interpret the network. Each use their interpretations to design "copy/paste" attacks in which one natural image pasted inside another causes an unrelated misclassification. We add to this work with a new method to identify such adversarial features. Unlike any previous approach, ours does so in a way that allows for targeted attacks that take into account an arbitrary distribution of source images.

3 Methods

We adopt the "unrestricted" adversary paradigm [62] under which an attack is successful if the network's classification differs from an oracle's (e.g., a human). Our adversaries can only change a small, fixed portion of either the generator's latent or the image. We use white-box access to the network, though we present black-box attacks based on transfer from an ensemble in Appendix A.6.

Our attacks involve perturbing the latent representation in some layer of an image generator to produce an adversarial feature-level alteration. Fig. 2 depicts our approach. We test three types of attacks, "patch", "region" and "generalized patch" (plus a fourth in Appendix A.5 which we call "channel" attacks). We find patch attacks to generally be the most successful.

Patch: We use the generator to produce a square patch that is inserted into a natural image [57].

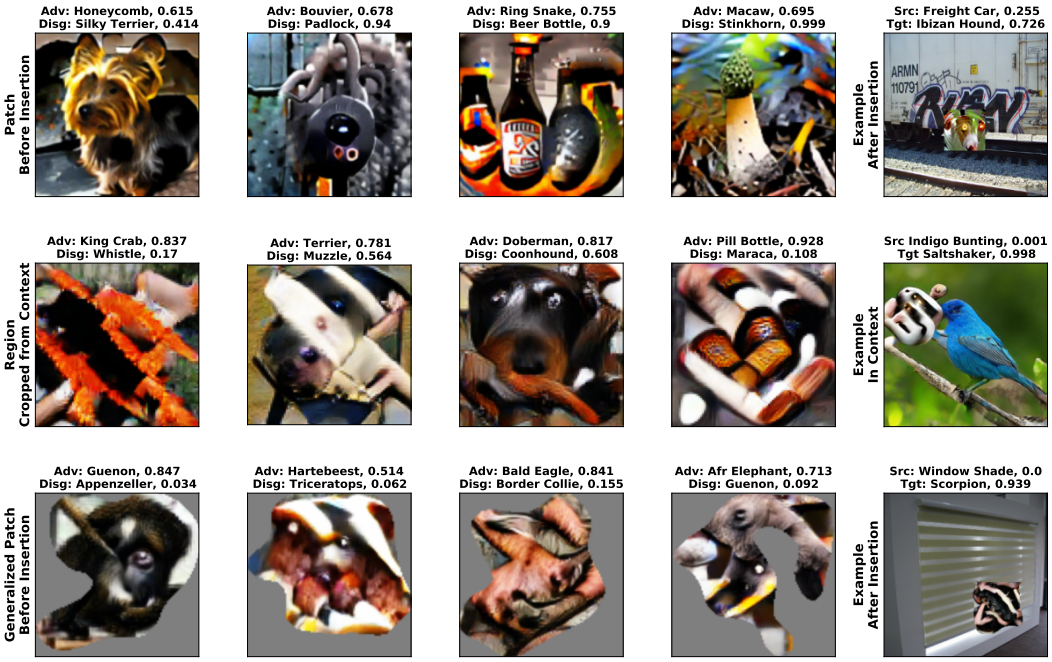


Figure 3: Examples of targeted, universal feature-level adversaries from patch (top), region (middle), and generalized patch (bottom) attacks. The first four columns show the adversarial features. The mean target class confidence is labeled 'Adv.' and is calculated under random source images (and random insertion locations for patch and generalized patch attacks). The target network's disguise class confidence for each patch or extracted generalized patch is labeled 'Disg.' The final column shows examples of the features applied to images. The example image for each is labeled with its source and target class confidences.

Region: Starting with some generated image, we randomly select a square column of the latent in a generator layer which spans the channel dimension and replace it with a learned insertion. This is analogous to a square patch in the pixel representation. We keep insertion location fixed over training. The modified latent is passed through the rest of the generator, producing the adversarial image.

Generalized Patch: These patches can be of any shape, hence the name "generalized" patch. We first generate a region attack and then extract a generalized patch from it. We do this by taking the absolute-valued pixel-level difference between the original and adversarial image, applying a Gaussian filter for smoothing, and creating a binary mask from the top decile of these pixel differences. We apply this mask to the generated image to isolate the region that the perturbation altered. We can then treat this as a patch and overlay it onto an image in any location.

Basic Attacks: For all attacks, we train a perturbation δ to the latent of the generator to minimize a loss that optimizes for both attacking the classifier and appearing interpretable:

$$\arg\min_{\delta} \mathbb{E}_{x\sim\mathcal{X},t\sim\mathcal{T},l\sim\mathcal{L}} \quad L_{\text{x-ent}}[C(A(x,\delta,t,l)), y_{\text{targ}}] + L_{\text{reg}}[A(x,\delta,t,l)] \tag{1}$$

with $\mathcal{X}$ a distribution over source images (e.g., a dataset or generation distribution), $\mathcal{T}$ a distribution over transformations, $\mathcal{L}$ a distribution over insertion locations (this only applies for patches and generalized patches), C the target classifier, A an image-generating function, $L_{\text{x-ent}}$ a targeted crossentropy loss for attacking the classifier, y_{targ} the target class, and L_{reg} a regularization loss. The adversary has no control over $\mathcal{X}$, $\mathcal{T}$, or $\mathcal{L}$, so it must learn features that work on the network independent of any particular source image, transformation, or insertion location. For all of our attacks, L_{reg} contains a total variation loss, $TV(a)$, to discourage high-frequency patterns.

"Disguised" Attacks: Ideally, a feature-level adversarial example should appear to a human as easily-describable but should not resemble the attack's target class. We call such attacks "disguised." Here, the main goal is not to fool a human, but to help them *learn* about what types of realistic features might cause the model to make a mistake. To train these disguised attacks, we use additional

terms in L_{reg} as proxies for these two criteria. We differentiably resize the patch or the extracted generalized patch and pass it through a GAN discriminator and auxiliary classifier. We then add weighted terms to the regularization loss based on the discriminator's (D) logistic loss for classifying the input as fake, the output entropy (H) of some classifier (C'), and/or the negative of the classifier's crossentropy loss for labeling the input as the attack's target class. Note that C' could either be the same or different than the target classifier C. With all of these terms, the regularization objective is

$$L_{\text{reg}}(a) = \lambda_1 TV(a) + \underbrace{\lambda_2 L_{\text{logistic}}[D(P(a))] + \lambda_3 H[C'(P(a))] - \lambda_4 L_{\text{x-ent}}[C'(P(a), y_{\text{targ}})]}_{\text{“Disguise” Regularizers}}. \quad (2)$$

Here, $P(a)$ returns the extracted and resized patch from adversarial image a. In order, these three new terms encourage the adversarial feature to (1) look realistic, and (2) look like some specific class, but (3) not the target class. The choice of disguise class is left entirely to the training process.

4 Experiments

We use BigGAN generators from [5, 70], and perturb the post-ReLU outputs of the internal 'GenBlocks.' We also found that training slight perturbations to the BigGAN's inputs improved performance. We used the BigGAN discriminator and adversarially trained classifiers from [12] for disguise regularization. By default, we attacked a ResNet50 [20], restricting patch attacks to 1/16 of the image and region and generalized patch attacks to 1/8. Appendix A.2 has additional details. First, in Section 4.1 we show that these feature level adversaries are highly robust to suggest that interpretations based on them are generalizable. Second, in Section 4.2 we put these interpretations to the test and show that our feature level adversaries can help one understand a network well enough to exploit it.

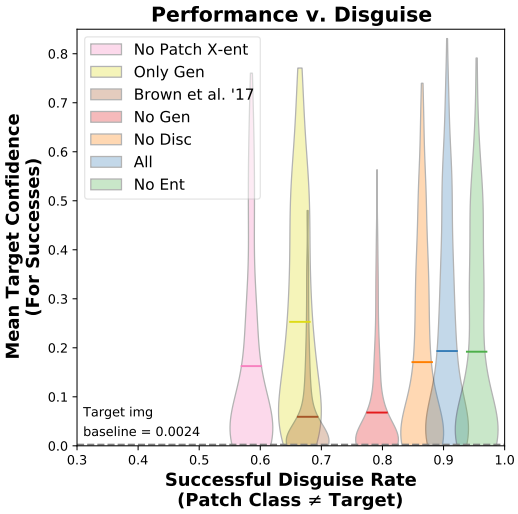


Figure 4: Targeted, universal patch attacks compared. Successful disguise success rate (x axis) shows the proportion of attacks in which the patch was not classified by the network as the target class when viewed on its own. Mean target class confidence (y axis) gives the empirical target class confidences of 250 patch attacks. Each is an average over 100 source images. The proportion of each distribution above 0.5 gives a lower-bound for the top-1 attack success rate. The mean target class confidence for using randomly-sampled natural target class images as patches is 0.0024 and is shown as a thin dotted line at the bottom.

4.1 Robust Attacks

Figure 3 shows examples of targeted, universal, and disguised feature-level patch (top), region (middle), and generalized patch (bottom) attacks which were each trained with all of the disguise regularization terms from Eq. 2. We find the disguises to be effective, particularly for the patches (top row), but imperfect. Appendix A.3 discusses this and what it may suggest about networks and size bias.

Performance versus Disguise: Here, we study our patch attacks in depth to test how effective they are at attacking the network and how successfully they can help to identify non-target-class features that can fool the network. We compared seven different approaches. The first was our full approach using the generator and all disguise regularization terms from Eq. 2. The rest were ablation tests in which we omitted the generator (No Gen), the discriminator (No Disc) regularization term (No Reg), the entropy regularization term (No Ent), the crossentropy regularization term (No Patch X-ent), all three regularization terms (Only Gen), and finally the discriminator and all three regularization terms (Brown et al. '17). This final unregularized, pixel-level method resulted in the same approach as Brown et al. (2017) [6]. For each test, all else was kept identical including penalizing total variation, training under transformations, and initializing the patch as a generator output.

Figure 5: Examples of targeted, disguised, universal, and physically-realizable feature-level attacks. See Appendix A.10 Fig. 13 for full-sized versions of the patches.

For each method, we generated universal attacks with random target classes until we obtained 250 successfully "disguised" ones in which the resulting adversarial feature was not classified by the network as the target class when viewed on its own. Fig. 4 plots the success rate versus the distribution of target class mean confidences for each type of attack. For all methods, these universal attacks have variable target class confidences due in large part to the random selection of target class. Some attacks are stochastically Pareto dominated by others. For example, the pixel-space Brown et al. (2017) attacks were the least effective at attacking the target network and had the third least disguise rate. In other cases, there is a tradeoff between attack performance and disguise which can be controlled using the regularization terms from Eq. 2. We also compare our attacks to two baselines using resized natural images from the target class and randomly sampled patches from the center of target class images. These resulted in a mean target class confidences of 0.0024 and 0.0018 respectively.

Notably, Fig. 4 does not capture everything that one might care about in these attacks. It does not show any measure of how "realistic" the resulting patches look. In Appendix A.4, Fig. 9 plots the same target class confidence data from the y axis in Fig. 4 versus the disguise class label confidence from an Inception-v3 which we use as a proxy for how realistic a human would find the patch. It suggests that the best attacks for producing patches that appear realistic are the "All" and "No Disc" methods. In Appendix A.10, Figs. 13 14, and 15 give examples of successful "All", "Only Gen", and "Brown et al. '17" attacks respectively. Because they were initialized from generator outputs, some of the "Brown et al. '17" attacks have a veneer-like resemblance to non-target class features. Nonetheless, they contain higher-frequency patterns and less coherent objects in comparison to the two sets of feature-level attacks. We subjectively find the "All" attacks to be the best disguised.

Physical-Realizability: To test their ability to transfer to the physical world, we generated 100 additional targeted, universal, and disguised adversarial patches. We used the generator and all regularization terms (the "All" condition from above). We selected the 10 with the best mean target class confidence, printed them, and photographed each next to 9 objects from different ImageNet classes.[2] We confirmed that photographs of each object were correctly classified without a patch. Figure 5 shows successful examples. Meanwhile, resizable and printable versions of all these patches and others are in Appendix A.10. The mean and standard deviation of the target class confidences for our attacks in the physical world were 0.312 and 0.318 respectively ($n = 90$, not i.i.d.). This means that these patches' mean effectiveness dropped by less than $\frac{1}{2}$ when transferring to the physical world.

Black-Box Attacks: In Appendix A.6, we show that our targeted universal attacks can transfer from an ensemble to a held-out model.

4.2 Interpretability

If an adversarial feature successfully fools the victim network, this suggests that the network associates that feature in context of a source image with the target class. We find that our adversaries can suggest both beneficial and harmful feature-class associations. In Appendix A.7, Fig. 11 provides a simple example of each.

Simply developing an interpretation, however, is easy. Showing that one leads to a useful understanding of the network is harder. One challenge in the explainable AI literature is to develop interpretations that go beyond seeming-plausible and stand up to scrutiny [51]. Robust feature level adversarial patches can easily be used to develop hypotheses about the network's behavior, e.g. "The network

[2] Backpack, banana, bath towel, lemon, jeans, spatula, sunglasses, toilet tissue, and toaster.

thinks that bee features plus colorful balls implies a fly." But are these valid, useful interpretations of the network? In other words, are our adversaries adversarial because of their interpretable qualities, or is it because of hidden motifs? We verify interpretations by using our attacks to make and validate predictions about how to fool the target network with natural objects.

Validating Interpretations with Copy/Paste Attacks: A "copy-paste" attack is created by inserting one natural image into another to cause an unexpected misclassification. They are more restricted than patch attacks because the features pasted into an image must be natural objects. As a result, they are of high interest for physically-realizable attacks because they suggest combinations of real objects that yield unexpected classifications. They also have precedent in the real world. For example, subimage insertions into pornographic images have been used to evade NSFW content detectors [75].

To develop copy/paste attacks, we select a source and target class, generate class-universal adversarial features, and manually analyze them for motifs that resemble natural objects. Here, we used basic attacks without the disguise regularization terms from Eq. 2. We then paste images of these objects into natural images and pass them through the classifier.

Fig. 6 shows four types of copy/paste attacks. In each odd row, we show six patch, region, and generalized patch adversaries that were used to guide the design of a copy/paste attack. In each even row are the copy/paste adversaries for the 6 (of 50) images for the source class for which the insertion resulted in the highest target class confidence increase along with the mean target class confidences before and after patch insertion for those 6. The success of these attacks shows their usefulness for interpreting the target network because they require that a human understands the mistake the model is making like "Bee $\wedge$ Traffic Light $\rightarrow$ Fly" well enough to manually exploit it. Given the differences in the adversarial features that are produced in the Bee $\rightarrow$ Fly and Traffic Light $\rightarrow$ Fly attacks, Fig. 6 also demonstrates how our attacks take the distribution of source images into account.

Three prior works [7, 42, 22] have developed copy/paste attacks, also via interpretability tools. Unlike [42, 22], our approach allows for targeted attacks. And unlike all three, rather than simply identifying features associated with a class, our adversaries generate adversarial features for a target class *conditional* on any distribution over source images (i.e. the source class) with which the adversaries are trained. Little work has been done on copy/paste adversaries, and thus far, methods have either not allowed for targeted attacks or have required a human in the loop. This makes objective comparisons difficult. However we provide examples of a feature-visualization based method inspired by [7] in Appendix A.8 to compare with ours. For the Indian $\rightarrow$ African Elephant attack, the source and target class share many features, and we find no evidence that feature visualization is able to suggest useful features for copy/paste attacks. This suggests that our attacks' ability to take the source image distribution into account may be more helpful for discovering certain weaknesses compared to the baseline inspired by [7].

5 Discussion and Broader Impact

Here we use feature-level adversarial examples to attack and interpret deep networks in order to contribute to a more practical understanding of network vulnerabilities. As an attack method, our approach is versatile. It can produce targeted, universal, disguised, physically-realizable, black-box, and copy/paste attacks at the ImageNet scale. Furthermore, this method can be used as an interpretability tool to help diagnose flaws in models. We ground the notion of interpretability in the ability to make predictions about when a model will fail. Finally, we demonstrate this through the design of targeted copy/paste attacks for any distribution over source inputs.

Like any work on adversarial attacks, our approach could be used maliciously to make a system fail, but we emphasize their diagnostic value. Understanding threats is a prerequisite to avoiding them. Given the robustness and versatility of our attacks, we argue that they may be valuable for continued work to address threats that systems may face in practical applications. There are at least two ways in which these methods can be useful.

The first is for adversarial training. Training networks on adversarial images has been shown to improve their robustness to the attacks that are used [12]. But this does not guarantee robustness to other types of adversarial inputs (e.g. [21]). Our feature-level attacks are categorically different from conventional pixel-level ones, and our copy/paste attacks show how networks can be fooled by novel combinations of natural objects, failures that are outside the conventional paradigm for adversarial

Figure 6: Feature-level adversaries can guide the design of class-universal copy/paste adversarial attacks. Patch adversary pairs are on the left, region in the middle, and generalized patch on the right of each odd row. Attack examples are on each even row. We do not claim that the traffic light $\wedge$ bee $\rightarrow$ fly examples on row 4 are necessarily adversarial, but they demonstrate alongside the bee $\wedge$ traffic light $\rightarrow$ fly adversaries that the adversarial features are sensitive to the the source images. For each attack except traffic light $\rightarrow$ fly, we limited ourselves to attempting 10 natural patches.

robustness (e.g., [12]). Consequently, we expect that adversarial training on broader classes of attacks such as the one we propose here will be valuable for designing more robust models. As a promising sign, we show in Appendix A.9 that adversarial training is useful for our attacks.

The second is for rigorously diagnosing flaws. We show that feature-level adversaries aid the discovery of exploitable spurious feature/class associations (Fig. 6) and a socially-harmful bias (Appendix

A.7 Fig. 11). Our approach could also be extended beyond what we have demonstrated here. For example, our methods may be useful for feature visualization [46] of a network's internal neurons. An analogous approach to ours can also be used in Natural Language Processing [61, 49], and we are currently working on a method for this. Furthermore, it may be valuable to use these adversaries to identify generalizable flaws in networks that humans can easily understand but with minimal human involvement. This would be much more scalable and prevent human priors from influencing interpretations. Toward this goal, we are currently working on methods for the fully-automated discovery of copy/paste attacks.

A limitation of our approach is that when multiple desiderata are optimized for at the same time (e.g., universality + transformation robustness + disguise), attacks are generally less successful, more time-consuming, and require more screening to find good ones. This could be a bottleneck for large-scale adversarial training. Ultimately, this type of attack is limited by the efficiency and quality of the generator, so future work should leverage advances in generative modeling. Our evaluation method is also limited to be a proof-of-concept for the design of copy/paste attacks. Future work should evaluate this more rigorously. We are currently working toward developing a benchmark for interpretability tools based on their ability to aid a human in rediscovering trojans [15] that have been implanted into a model.

As AI becomes increasingly capable, it becomes more important to design models that are reliable. Each of the 11 proposals for building safe AI outlined in [25] explicitly call for adversarial robustness and/or interpretability tools, and recent work from [77] on high-stakes reliability in AI found that interpretability tools strengthened their ability to produce inputs for adversarial training. Given the close relationship between interpretability and adversarial robustness, continued study of the connections between them will be key for building safer AI systems.

Acknowledgments

We thank Cassidy Laidlaw, Miles Turpin, Will Xiao, and Alexander Davies for insightful discussions and feedback and Kaivu Hariharan for help with coding. This work was conducted in part with funding from the Harvard Undergraduate office of Research and Fellowships.

References

[1] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *Proceedings of the IEEE Conference on computer Vision and Pattern Recognition*, pages 3686–3693, 2014.

[2] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In *International conference on machine learning*, pages 284–293. PMLR, 2018.

[3] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6541–6549, 2017.

[4] Anand Bhattad, Min Jin Chong, Kaizhao Liang, Bo Li, and David A Forsyth. Unrestricted adversarial examples via semantic manipulation. *arXiv preprint arXiv:1904.06347*, 2019.

[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] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005.

[9] Joel Dapello, Tiago Marques, Martin Schrimpf, Franziska Geiger, David D Cox, and James J DiCarlo. Simulating a primary visual cortex at the front of cnns improves robustness to image perturbations. *BioRxiv*, 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] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[12] Logan Engstrom, Andrew Ilyas, Hadi Salman, Shibani Santurkar, and Dimitris Tsipras. Robustness (python library), 2019.

[13] Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Brandon Tran, and Aleksander Madry. Adversarial robustness as a prior for learned representations. *arXiv preprint arXiv:1906.00945*, 2019.

[14] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1625–1634, 2018.

[15] Arturo Geigel. Neural network trojan. *Journal of Computer Security*, 21(2):191–232, 2013.

[16] 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.

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

[18] Atiye Sadat Hashemi, Andreas Bär, Saeed Mozaffari, and Tim Fingscheidt. Transferable universal adversarial perturbations using generative models. *arXiv preprint arXiv:2010.14919*, 2020.

[19] Jamie Hayes and George Danezis. Learning universal adversarial perturbations with generative models. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 43–49. IEEE, 2018.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

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

[26] Jessica Hullman and Nick Diakopoulos. Visualization rhetoric: Framing effects in narrative visualization. *IEEE transactions on visualization and computer graphics*, 17(12):2231–2240, 2011.

[27] 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.

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

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

[30] Stepan Komkov and Aleksandr Petiushko. Advhat: Real-world adversarial attack on arcface face id system. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 819–826. IEEE, 2021.

[31] Zelun Kong, Junfeng Guo, Ang Li, and Cong Liu. Physgan: Generating physical-world-resilient adversarial examples for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14254–14263, 2020.

[32] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[33] Alexey Kurakin, Ian Goodfellow, Samy Bengio, et al. Adversarial examples in the physical world, 2016.

[34] 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.

[35] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

[36] Aishan Liu, Xianglong Liu, Jiaxin Fan, Yuqing Ma, Anlan Zhang, Huiyuan Xie, and Dacheng Tao. Perceptual-sensitive gan for generating adversarial patches. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 1028–1035, 2019.

[37] 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.

[38] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[39] Luke Melas. Pytorch-pretrained-vit. `https://github.com/lukemelas/PyTorch-Pretrained-ViT`, 2020.

[40] Konda Reddy Mopuri, Utkarsh Ojha, Utsav Garg, and R Venkatesh Babu. Nag: Network for adversary generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 742–751, 2018.

[41] Konda Reddy Mopuri, Phani Krishna Uppala, and R Venkatesh Babu. Ask, acquire, and attack: Data-free uap generation using class impressions. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 19–34, 2018.

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

[43] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[44] Mark D Norman, Julian Finn, and Tom Tregenza. Dynamic mimicry in an indo–malayan octopus. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 268(1478):1755–1758, 2001.

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

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

[47] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

[48] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[49] 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.

[50] Omid Poursaeed, Isay Katsman, Bicheng Gao, and Serge Belongie. Generative adversarial perturbations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4422–4431, 2018.

[51] 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.

[52] 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.

[53] Hadi Salman, Andrew Ilyas, Logan Engstrom, Ashish Kapoor, and Aleksander Madry. Do adversarially robust imagenet models transfer better? In *ArXiv preprint arXiv:2007.08489*, 2020.

[54] 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.

[55] Martin Schrimpf, Jonas Kubilius, Michael J Lee, N Apurva Ratan Murty, Robert Ajemian, and James J DiCarlo. Integrative benchmarking to advance neurally mechanistic models of human intelligence. *Neuron*, 108(3):413–423, 2020.

[56] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K Reiter. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proceedings of the 2016 acm sigsac conference on computer and communications security*, pages 1528–1540, 2016.

[57] Abhijith Sharma, Yijun Bian, Phil Munz, and Apurva Narayan. Adversarial patch attacks and defences in vision-based tasks: A survey, 2022.

[58] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

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

[60] Uchida So and Ihor Durnopianov. Lpips pytorch. `https://github.com/S-aiueo32/lpips-pytorch`, 2019.

[61] Liwei Song, Xinwei Yu, Hsuan-Tung Peng, and Karthik Narasimhan. Universal adversarial attacks with natural triggers for text classification. *arXiv preprint arXiv:2005.00174*, 2020.

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

[63] Martin Stevens and Graeme D Ruxton. Do animal eyespots really mimic eyes? *Current Zoology*, 60(1), 2014.

[64] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[65] 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.

[66] Simen Thys, Wiebe Van Ranst, and Toon Goedemé. Fooling automated surveillance cameras: adversarial patches to attack person detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 0–0, 2019.

[67] 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.

[68] 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.

[69] Olivia Wiles, Isabela Albuquerque, and Sven Gowal. Discovering bugs in vision models using off-the-shelf image generation and captioning, 2022.

[70] Thomas Wolf. Pytorch pretrained biggan. `https://github.com/huggingface/pytorch-pretrained-BigGAN`, 2018.

[71] Eric Wong and J Zico Kolter. Learning perturbation sets for robust machine learning. *arXiv preprint arXiv:2007.08450*, 2020.

[72] Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating adversarial examples with adversarial networks. *arXiv preprint arXiv:1801.02610*, 2018.

[73] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[74] Fisher Yu, Wenqi Xian, Yingying Chen, Fangchen Liu, Mike Liao, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving video database with scalable annotation tooling. *arXiv preprint arXiv:1805.04687*, 2(5):6, 2018.

[75] Kan Yuan, Di Tang, Xiaojing Liao, XiaoFeng Wang, Xuan Feng, Yi Chen, Menghan Sun, Haoran Lu, and Kehuan Zhang. Stealthy porn: Understanding real-world adversarial images for illicit online promotion. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 952–966. IEEE, 2019.

[76] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

[77] 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.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 5.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 5.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See supplementary material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix A.2 and supplementary material.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Fig. 4
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section A.2.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No] We only used open source assets plus ImageNet. We are license compliant but do not discuss this explicitly.
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] We are license compliant for ImageNet but do not discuss this explicitly.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] Data we use is limited to ImageNet.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

(a) (b)

Figure 7: Examples of robust feature-level adversaries in nature. (a) A peacock and butterfly with adversarial "eyespots." (b) A 'mimic octopus' (image from [44]) mimics a banded sea snake's patterning and is classified as one by a ResNet50.

A Appendix

A.1 Adversarial Features in Nature

Fig. 7 shows examples of robust feature-level adversaries in nature.

A.2 Methodological Details

We found that performing crossentropy and entropy regularization (Eq. 2) for disguise using adversarially-trained auxiliary classifiers produced more easily interpretable results. This aligns with findings that adversarially-trained networks tend to learn more interpretable representations [13, 53] and better approximate the human visual system [55]. Thus, for crossentropy and entropy regularization, we used an $\epsilon = 4$ L_2 and $\epsilon = 3$ L_∞ robust ResNet50s from [12]. For discriminator regularization, we use the BigGAN class-conditional discriminator with a uniform class vector input (as opposed to a one-hot vector). For patch adversaries, we train under colorjitter, Gaussian blur, Gaussian noise, random rotation, and random perspective transformations to simulate real-world variations in conditions. For region and generalized patch ones, we only use Gaussian blurring and horizontal flipping. Also for region and generalized patch adversaries, we promote subtlety by penalizing the difference from the original image using the LPIPs perceptual distance [76, 60]. All experiments were implemented with PyTorch [48], and implementations of all of our work can be run from Google Colab notebooks which we provide. We estimate that our project involved a total of $<$ 400 hours of compute, mostly on 12GB NVIDIA Tesla K80 GPUs.

A.3 How Successful are our Disguises?

All of our attacks are designed to be "adversarial" in either one or two different ways. First, all are limited to manipulating only a certain portion of the image or latent. Second, some of our attacks are trained to be disguised as discussed in Section 3. We subjectively find that our methods for disguise are generally useful . For example, compare Fig. 13 to Fig. 14. However, some of these features are not well disguised. Consider the patch of a at the top left of Fig. 13 whose target class is a pufferfish but which is disguised as a crane.

We find it clear that the patch depicts a crane, and the network agrees when shown the patch as a full image. However, the image also seems to have patterning and coloring that resembles a pufferfish. Furthermore, when shrunk to the size of a patch and inserted into a source image for an attack, the finer, more crane-like features may be less prominent, and the pufferfish-like ones may be more prominent. This suggests that the the adversary may be exploiting size biases.

Ultimately, to the extent that the patch resembles the target class, it is not disguised and arguably not adversarial. However, we emphasize three things. First, we find the disguise classes to generally be much more prominent in these patches than the target class. Second, being able to recognize target class features when the target class is known versus unknown are very different tasks. It is key to note such framing effects [26]. And third, very few attack methods are always successful at the ImageNet

Figure 8: Examples of original images (top) alongside class-universal channel adversaries (bottom). Each image is labeled with the source and target class confidence.

scale, and some need for screening should be expected when optimizing an adversarial example for a complex objective (e.g. universality + transformation robustness + disguise).

We believe the studying human responses to feature-level adversaries and the links between interpretable representations, robustness, and similarity to human cortex [9] may be promising directions for better understanding both networks and biological brains.

A.4 Attack Performance versus Disguise

In the main paper Section 4.1, Fig. 4 plots the successful disguise rate of attacks alongside their distribution of mean target class confidences. However, this leaves out how effective each type of attack is at appearing realistic to a human. Here, we use the target class confidence of an Inception-v3 [64] as a proxy for how realistic a patch appears to a human. Fig. 9 plots the mean target class confidences for successfully-disguised attacks versus their Inception-v3 disguise label confidence. This suggests that the attacks that are the best at producing realistic-looking patches are the "All" ones with the generator and all regularization terms and the "No Disc" ablations which omit the discriminator regularization term.

A.5 Channel Attacks

In contrast to the region attacks presented in the main paper, we experiment here with *channel* attacks. For region attacks, we optimize an insertion to the latent activations of a generator's layer which spans the channel dimension but not the height and width. This is analogous to a patch attack in pixel-space. For channel attacks, we optimize an insertion which spans the height and width dimensions but only involves a certain proportion of the channels. This is analogous to an attack that only modifies the R, G, or B channel of an image in pixel-space. Unlike the attacks in Section, 4, we found that it was difficult to create universal channel attacks (single-image attacks, however, were very easy). Instead, we relaxed this goal and created class-universal ones which are meant to cause any generated example from a source class to be misclassified as a target. We also manipulate $1/4^{\text{th}}$ of the latent instead of $1/8^{\text{th}}$ as we do for region attacks. Mean target class confidences and examples from the top 5 attacks out of 16 are shown in Fig. 8. They induce textural changes somewhat like adversaries crafted by [16] and [4].

A.6 Black-Box Attacks

Adversaries are typically created using first order optimization on an input to a network which requires that the parameters are known. However, they are often transferrable between models [47], and one method for developing black-box attacks is to train against a different model and then transfer to the intended target. We do this for our adversarial patches and generalized patches by attacking a large ensemble of AlexNet [32], VGG19 [58], Inception-v3 [64], DenseNet121 [24], ViT[11, 39], and

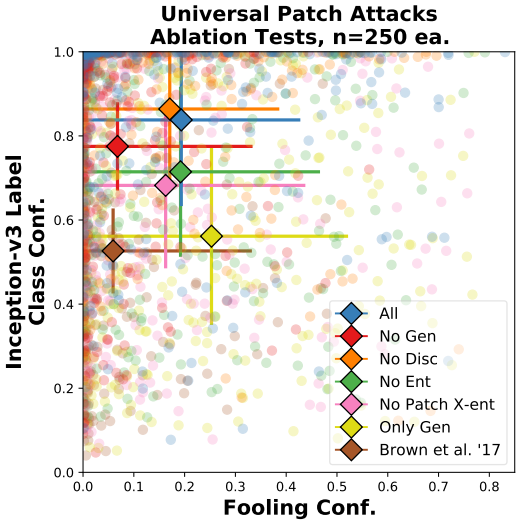


Figure 9: Targeted, universal patch attacks compared by mean target class confidence and Inception-v3 label-class confidence. Inception-v3 class conf. on the x-axis gives the mean target class confidence from the attacked network for images which have the patch inserted. The Inception-v3's label class confidence for the patch on the y-axis is used as a proxy for human interpretability. Attacks further up and right are better. Centroids are shown with error bars giving the standard deviation.

Figure 10: Universal black-box adversarial patches (top) and generalized patches (bottom) created using transfer from an ensemble. Patches are displayed alongside their target class and mean target class confidence.

two robust ResNet50s [12], and then transferring to ResNet50 [20]. Otherwise, these attacks were identical to the ones in Fig. 3 including a random source/target class and optimization for disguise. Many were unsuccessful, but a sizable fraction were able to work on the ResNet50 with a mean confidence of over 0.1 for randomly sampled images. The top 5 out of 64 of these attacks for patch and generalized patch adversaries are shown in Fig. 10.

A.7 Discovering Feature-Class Associations

Fig. 11 shows two simple examples of using feature-level attacks to identify feature-class associations. It shows one positive example in which the barbershop class is desirably associated with barber-pole-stripe-like features and one negative example in which the bikini class is undesirably associated with

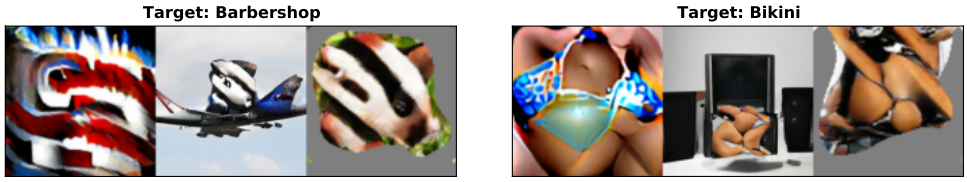


Figure 11: Examples of good and bad features class associations in which barber pole stripes are associated with a barbershop and a particular type of skin is associated with a bikini. Patch (left), region (middle), and generalized patch adversaries (right) are shown.

a particular type of skin. Notably the ability to simply identify feature associated with a target class is not a unique capability of our attacks and could also be achieved with feature visualization [46].

A.8 Copy/Paste Attacks: Feature-Level Adversaries vs. Class Impressions

[42, 22, 7] each used interpretability methods to guide the development of copy/paste adversaries. [42] and [22], used network dissection [3] to develop interpretations of neurons and fit a semantic description to neurons which they combined with analysis of weight magnitudes. This allowed them to identify cases in which the networks learned undesirable feature-class associations. However, this approach cannot be used to make a targeted search for copy/paste attacks that will cause a given source class to be misclassified as a given target.

More similar to our work is [7] who found inspiration for successful copy/paste adversaries by creating a dataset of visual features and comparing the differences between ones which the network assigned the source versus target label. We take inspiration from this approach to create a baseline against which to compare our method for designing copy/paste attacks from Section 4.2. Given a source and target class such as a bee and a fly, we optimize a set of inputs to a network for each class in order to maximize the activation of the output node for that class. [41] refers to these as *class impressions*. We develop these inputs using the Lucent [29] package. We do this for the same source/target class pairs as in Fig. 6 and display 6 per class in Fig. 12. In each subfigure, the top row gives class impressions of the source class, and the bottom gives them for the target. Each class impression is labeled with the network's confidences for the source and target class. In analyzing these images, we find limited evidence of traffic-light-like features in the fly class impressions, and we find no evidence of more blue coloration in the African elephant class impressions than the Indian ones.

These class impressions seem comparable but nonredundant with our method from Section 4.2. However, our approach may have an advantage over the use of class impressions in that it is equipped to design features that look like the target class *conditional* on a distribution of source images. Contrastingly, a class impression is only meant to visualize features typical of the target class. This may be why our adversarial attacks are able to suggest that inserting a traffic light into a bee image or a blue object into an Indian elephant image can cause a misclassification as a fly or African elephant respectively. Bees and Flies and Indian and African elephant are pairs of similar classes. For cases in which the source and target are related, the class impressions may share many of the same features and thus be less effective for identifying adversarial features that work *conditional* on the source image distribution.

A.9 Defense via Adversarial Training

Adversarial training is a common effective means for improving robustness. Here, to test how effective it is for our attacks, for 5 pairs of similar classes, we generate datasets of 1024 images evenly split between each class and between images with and without adversarial perturbations. This prevents the network from learning to make classifications based on the mere presence or absence of a patch. We do this separately for channel, region, and patch adversaries before treating the victim network as a binary classifier and training on the examples. We report the post-training minus pre-training accuracies in Tbl. 2 and find that across the class pairs and attack methods, the adversarial training improves binary classification accuracy by a mean of 42%.

	Channel	Region	Patch	Mean
Great White/Grey Whale	0.49	0.29	0.38	0.39
Alligator/Crocodile	0.13	0.29	0.60	0.34
Lion/Tiger	0.29	0.28	0.63	0.40
Frying Pan/Wok	0.32	0.39	0.68	0.47
Scuba Diver/Snorkel	0.42	0.36	0.69	0.49
Mean	0.33	0.32	0.60	**0.42**

Table 2: Binary classification accuracy improvements from adversarial training for channel, region, and patch adversaries across 5 class pairs.

A.10 Examples: Resizable, Printable Patches

See Figs. 13, 14, and 15 for feature-level and pixel-level control adversarial images. We encourage readers to experiment with these images (which were optimized to attack a ResNet50) or with others which can be created using our provided code. In doing so, one might find a mobile app to be convenient. We used Photo Classifier.[3]

A.11 High-Level Summary

Here, we provide an easily-readable summary of this work, meant especially for readers who may be less familiar with machine learning jargon.

Historically, it has proven difficult to write conventional computer programs that classify images. But recently, immense progress has been made through neural networks which can now classify images into hundreds or thousands of categories with high accuracy. Despite this performance, we still don't fully understand the features that they use to classify images. Somewhat worryingly, past work has demonstrated that it is easy to take an image that the network classifies correctly and perturb its pixel values by a tiny amount in such a way that the network will misclassify it. For example, we can take a cat, make human-imperceptible changes to the pixels, and make the network believe that it is a dog. This process of designing an image that the network will misclassify is called an "adversarial attack."

Unfortunately, conventional adversarial attacks tend to produce perturbations that are not interpretable. To a human, they usually appear as pixelated noise (when exaggerated to be visible). As a result, they do not help us understand how networks will process human-describable inputs, and they do little to help us understand practical flaws in networks. Here, our goal is to develop different types of adversarial features that can be useful as debugging tools for these networks.

There are two properties that we want our adversarial images to have. First, we want them to be feature-level (not pixel-level) so that a human can interpret them, and second, we want them to be robust so that interpretations are generalizable. In one sense, this is not a new idea. Quite the opposite, in fact – there are examples of this the animal kingdom. Figure 7 shows examples of adversarial eyespots on a peacock and butterfly and adversarial patterns on a mimic octopus. These are interpretable, robust to a variety of viewing conditions, and give us useful information about how biological brains process these visual features.

The key technique that we use to create adversarial images involves an image generating network. Instead of perturbing pixels to cause an image to be misclassified, we perturb the internal state of the generator in order to induce a feature-level change to the output image. We also found that optimizing adversarial features under transformations to the images (like blurring, cropping, rotating, etc.) and using some additional terms in our optimization objective to encourage more realistic and better-disguised images improved results.

Our first key finding is that these attacks are very robust and versatile. For example, among other capabilities, we demonstrate that they can be universal (working for any image which we apply them to) and physically-realizable (working in the physical world when printed and photographed).

[3] https://apps.apple.com/us/app/photo-classifier/id1296922017

Our second key finding is that these adversaries are useful for interpreting networks. We use them as a way of producing useful classes of inputs for understanding ways that they can fail. Our strategy is to prove that these adversaries can help use understand the network well enough to exploit it. We analyze our adversarial features to get ideas for real-world features that resemble them. Then we test this interpretation by creating "copy/paste" attacks in which one natural image is pasted into another in order to cause a particular misclassification. Some of these are unexpected. For example, in Fig. 6, we find that a traffic light can make a bee look like a fly.

Together, our findings suggest that feature-level adversaries are very versatile attacks and practical debugging tools for finding flaws in networks. One implication is that by training networks on these adversaries, we might be able to make them more robust to failures that are due to feature-level properties of images. We also argue that these adversaries should be used as a practical debugging tool to diagnose problems in networks.

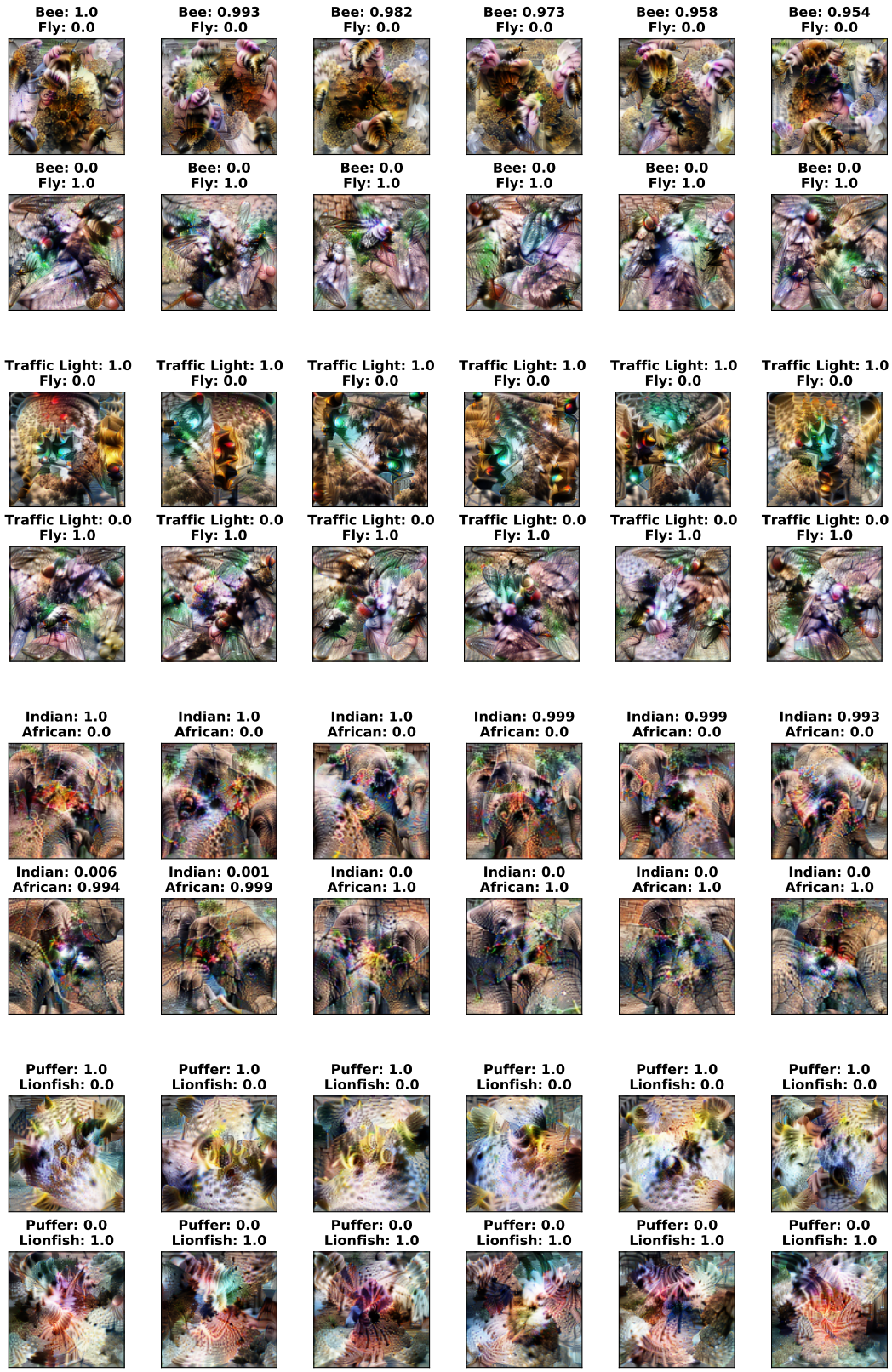


Figure 12: Class impressions for four pairs of classes. These could be used for providing insight about copy/paste attacks in the similar way to the examples from Fig. 6. Each subfigure is labeled with the network's output confidence for both classes.

Figure 13: Examples of attacks created with a generator and with the disguise regularization terms from Eq. 2. See section 4.1 and the "All" datapoints from Fig. 4.. We printed these to test physical realizability in Section 4.1

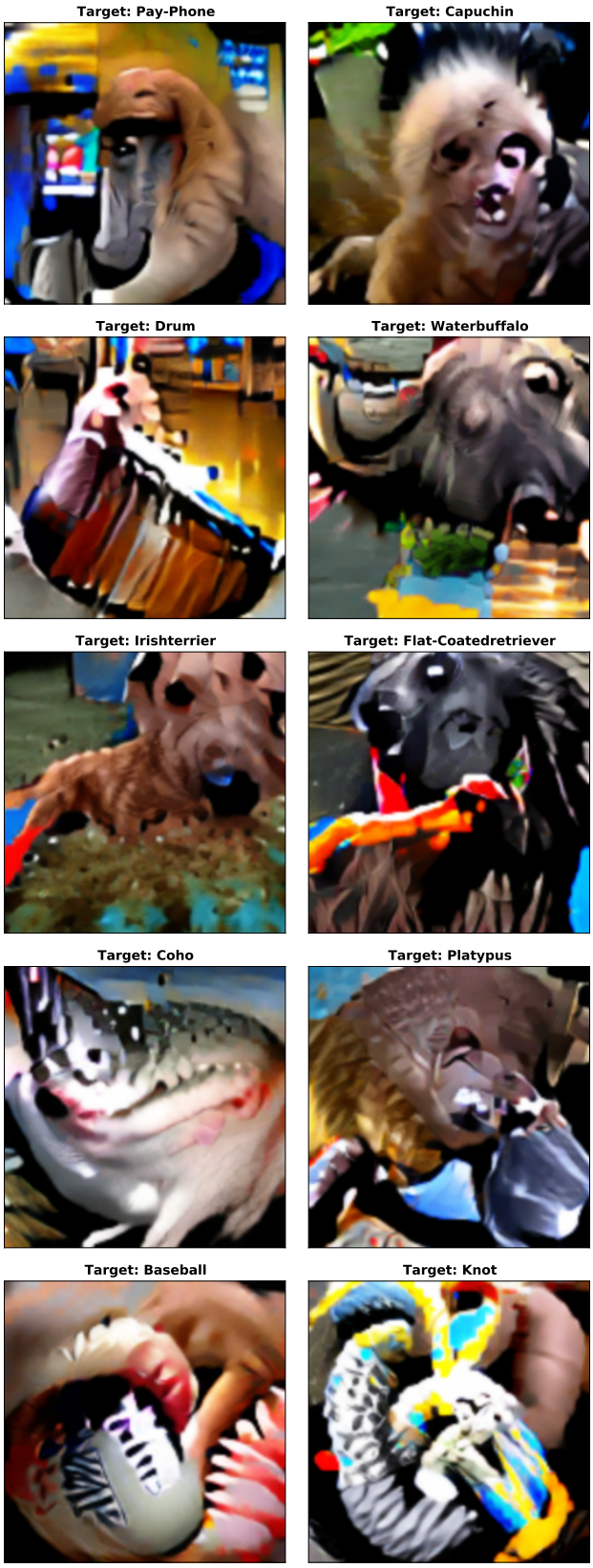


Figure 14: Examples of attacks created with a generator. See section 4.1 and the "Only Gen" datapoints from Fig. 4.

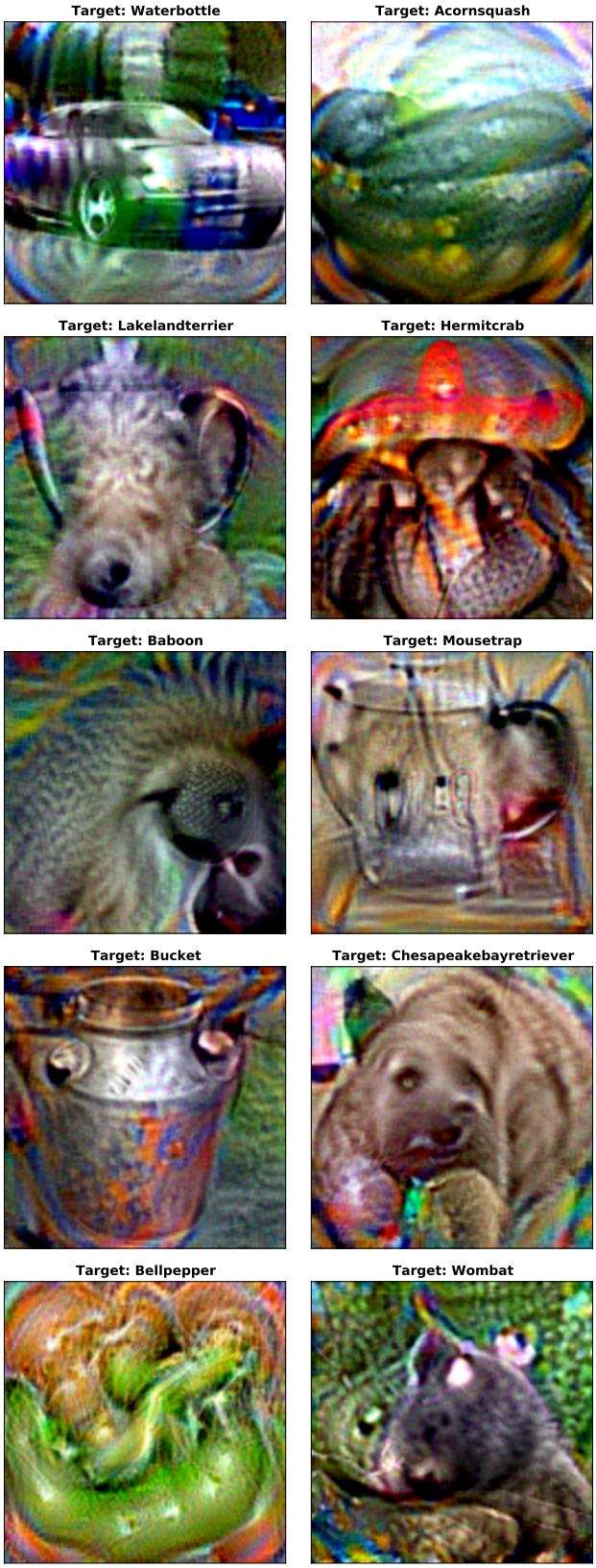


Figure 15: Examples of pixel-space attacks [6]. See section 4.1 and the "Brown et al. '17" datapoints from Fig. 4.