A APPENDIX

A.1 CHANNEL ATTACKS

In contrast to the region attacks presented in the main paper, we also experiment 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, 4.1, 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 one random 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 fooling rates and examples from the top 5 attacks out of 16 are shown in Fig. 8. They to induce textural changes somewhat like adversaries crafted by Geirhos et al. (2018) and Bhattad et al. (2019)).

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

A.2 BLACK-BOX ATTACKS

Adversaries are often created using first order optimization on an input to a network which requires that the network's parameters are known. However, adversaries are often transferrable between models (Papernot et al., 2016), 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 (Krizhevsky et al., 2012), VGG19 (Simonyan & Zisserman, 2014), Inception-v3 (Szegedy et al., 2016), DenseNet121 (Huang et al., 2017), ViT(Dosovitskiy et al., 2020; Melas, 2020), and two robust ResNet50s Engstrom et al. (2019a), and then transferring to ResNet50 He et al. (2016). Otherwise, these attacks were identical to the ones we used in Section 4.1. These attacks were crafted with a random-source/target class and optimization for disguise. Many were unsuccessful, but a sizable fraction were able to fool 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. 9.

A.3 DISCOVERING FEATURE-CLASS ASSOCIATIONS

Fig. 11 shows two simple examples of using feature-fool 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 caucasian-colored skin. Notably though, the ability to identify feature-class associ-

Figure 9: Black-box adversarial patches (top) and generalized patches (bottom) created using transfer from an ensemble. Patches are displayed alongside their target class and mean fooling confidence.

ations such as this is not a unique capability of feature-fool attacks and could also be achieved with feature visualization (Olah et al., 2017).

A.4 Copy/Paste Attacks with Class Impressions

Mu & Andreas (2020) and Carter et al. (2019) both used interpretability methods to guide the development of copy/paste adversaries. Mu & Andreas (2020), used an interpretability method known as network dissection (Bau et al., 2017) to develop interpretations of neurons and fit a semantic description in compositional logic over the network using those interpretations and the network's 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 Carter et al. (2019) 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 use inspiration from this approach to create a baseline against which to compare our method of designing copy/paste attacks in Section 4.3. 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. Mopuri et al. (2018b) refers to these as *class impressions*. We train these inputs under transformations and with a decorrelated frequency-space parameterization of the input pixels using the Lucent (Kiat, 2019) package. We do this for the same three class pairs as in Fig. 7 and display 6 per class in Fig. 10. 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 and comparing to the ones from Fig. 10, we find no evidence of more blue coloration in the African elephant class impressions than the Indian ones. However, we find it plausible that some of the features in the fly class impression may resemble traffic lights and that those for the Lionfish may resemble an admiral Butterfly's wings. Nonetheless, these visualizations are certainly different in appearance from our adversarial ones.

These class impressions seem comparable but nonredundant with our adversarial method from Section 4.3. However, our adversarial 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 the rest of the image being of the source class. Contrastingly, a class impression is only meant to visualize features typical of the target class. It is possible that this is why our adversarial attacks were able to show that inserting a blue object into an image of an Indian elephant can cause a misclassification as an African elephant – two very similar classes – while the class impressions for the two appear very similar and suggest nothing of the sort.

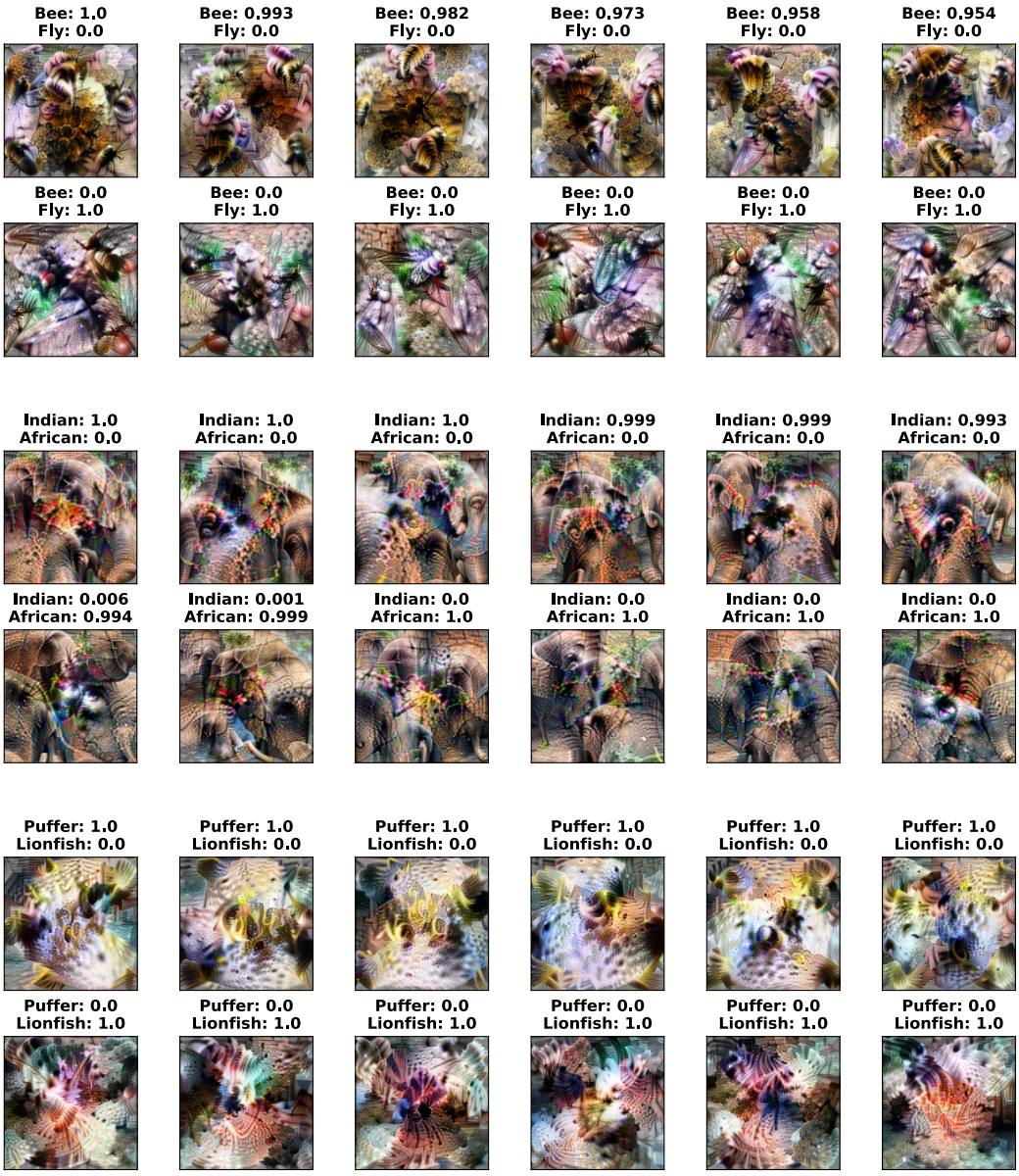

Figure 10: Class impressions for three pairs of classes. These could be used for providing insight about copy/paste attacks in the same way as the examples from Fig. 7. Each subfigure is labeled with the network's output confidence for both classes.

A.5 Defense via Adversarial Training

Adversarial training is a common and broadly 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 images with/without adversarial perturbations. 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. A.5 and find that across the class pairs and attack methods, the adversarial training improves binary classification accuracy by a mean of 42%.

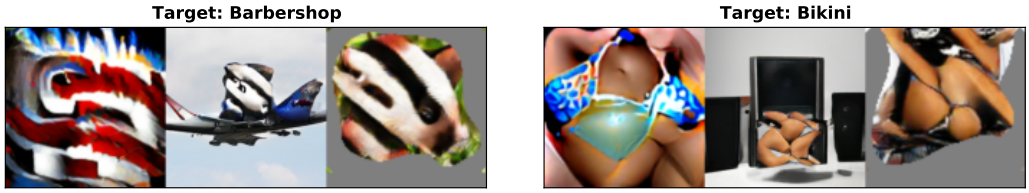

Figure 11: Examples of good and bad features class associations in which barber pole stripes are associated with a barbershop and caucasian-colored skin is associated with a bikini. Patch adversaries are on the left, region adversaries in the middle, and generalized patch adversaries on the right.

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

A.6 Resizable, Printable Patches

See Figs. 12 and 13 for feature-fool and control adversarial images respectively. We encourage readers to experiment with these images which were optimized to fool a ResNet50. In doing so, one might find a mobile app to be convenient. We used Photo Classifier.[2]

A.7 Jargon-Free Summary

AI and related fields are making rapid progress, but there exists a communication gap between researchers and the public which too-often serves as a barrier to the spread of information outside the field. For readers who may not know all of the field's technical concepts and jargon, we provide a more readable summary here.

Historically, it has proved difficult to write conventional computer programs that accurately classify real-world images. But this task has seen revolutionary success by neural networks which can now classify images into hundreds or thousands of categories sometimes with higher accuracy than humans. Despite the impressive performance, we still don't fully understand the features that these networks use to classify images, and we cannot be confident that they will always do so correctly. In fact, past research has demonstrated that it is usually very easy to take an image that the network classifies correctly and perturb its pixel values by a tiny amount – often imperceptibly to a human – but in such a way that the network will misclassify it with high confidence as whatever target class the attacker desires. For example, we can take an elephant, make minute changes in a few pixels, and make the network believe that it is a dog. Researchers have also discovered perturbations that can be added onto a wide range of images to cause them to be misclassified, making those perturbations "universal". In general, this process of designing an image that the network will misclassify is called an "adversarial attack" on the network.

Unfortunately, conventional adversarial attacks tend to produce perturbations that are not interpretable. To a human, they usually just appear as pixelated noise. As a result, they do not help us to understand how networks will process sensible inputs, and they do not reveal weaknesses that could be exploited by adversarial features in the real world. To make progress toward solving these problems, we focus on developing interpretable adversarial features. In one sense, this is not a new idea. Quite the opposite, in fact – there are already examples in the animal kingdom. Figure 2 shows

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

examples of adversarial eyespots on a peacock and butterfly and adversarial patterns on a mimic octopus.

To generate interpretable adversarial features, we introduce a method that uses "generative" modeling. In addition to classifying images, networks can also be used to great effect for learning to generate them. These networks are often trained with the goal of producing images that are so realistic that one cannot tell whether they came from the training set or not, and modern generation methods are moving closer to this goal. Typically, these networks take in a random input and form it into an image. Inside this process are intermediate "latent" representations of the image at each "layer" inside the generator that gradually shift from abstract, low-dimensional representations of high-level features of the image (e.g. the original input) to the actual pixel values of the final image.

In order to create interpretable adversarial images, we take images created by the generator and use them for adversarial attacks. In the simplest possible procedure, one can generate an image and then modify the generator's representations to change the generation of the image in such a way that the classifier is fooled by it. We also found that optimizing under transformations to our images (like blurring, cropping, flipping, rotating, etc.) and adding in some additional terms into our optimization objective to encourage more interpretable and better disguised images greatly improved results. Ultimately, we produce adversarial images that differ from normal ones in higher-level, more intelligible ways than conventional adversaries.

These adversaries are useful and informative in two main ways. First, they allow us to create patches that are simultaneously "disguised", "physically-realizable", and "universal" at the same time. By "disguised", we mean that they look like one thing to a human but cause an unrelated misclassification. By "physically-realizable", we mean that these images can be printed and physically placed in a scene with some other object, causing a photo of the scene to be misclassified. And by "universal", we mean that these images can cause photos of a wide range of objects to be misclassified as the target class. As an example, Fig. 1 shows a patch of a crane that can be physically inserted next to any real world object (such as sunglasses) in order to cause a misclassification as a pufferfish.

Second, these adversaries allow us to interpret networks by revealing feature-class associations. We even find that this can be used to create an additional type of attack. We show that this process can guide the creation of "copy/paste" attacks in which one natural image is pasted as a patch into another in order to cause a particular misclassification. Some of these are unexpected. For example, in Fig. 7, we find that a traffic light can make a bee look like a fly. These copy/paste attacks also have implications for physically-realizable attacks because they suggest combinations of real objects that could yield unexpected classifications.

Together, our findings offer potential for better understanding network representations and better predicting the ways that they may fail. We join others in the AI community in calling for caution and adversarial robustness when deploying networks in the real world.

A.8 EPITAPH

One thing to fool them all,
One class to assign them,
One thing to see it all,
And in the real world find them.[3]

[3] Adapted from J.R.R. Tolkein's *Ring Verse*.

Figure 12: Printable examples of the disguised, transformation-robust, physically-realizable feature-fool adversarial patches from Section 4.2. Patches can be resized before printing.

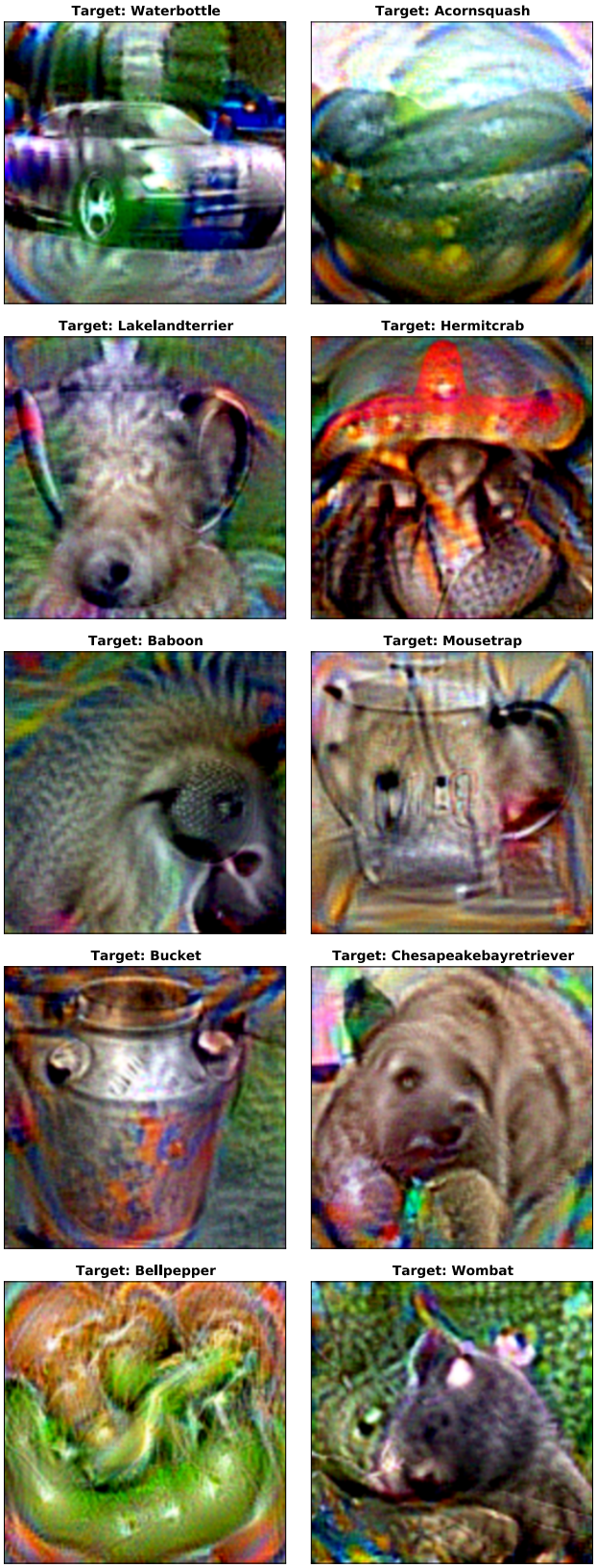

Figure 13: Printable examples of transformation-robust, physically-realizable pixel-space adversarial patches from Section 4.2. Patches can be resized before printing.