Functional Adversarial Attacks

Cassidy Laidlaw
University of Maryland
claidlaw@umd.edu

Soheil Feizi
University of Maryland
sfeizi@cs.umd.edu

Abstract

We propose *functional adversarial attacks*, a novel class of threat models for crafting adversarial examples to fool machine learning models. Unlike a standard ℓ_p-ball threat model, a functional adversarial threat model allows only a *single* function to be used to perturb input features to produce an adversarial example. For example, a functional adversarial attack applied on colors of an image can change *all* red pixels simultaneously to light red. Such global uniform changes in images can be less perceptible than perturbing pixels of the image individually. For simplicity, we refer to functional adversarial attacks on image colors as ReColorAdv, which is the main focus of our experiments. We show that functional threat models can be combined with existing additive (ℓ_p) threat models to generate stronger threat models that allow both small, individual perturbations and large, uniform changes to an input. Moreover, we prove that such combinations encompass perturbations that would not be allowed in either constituent threat model. In practice, ReColorAdv can significantly reduce the accuracy of a ResNet-32 trained on CIFAR-10. Furthermore, to the best of our knowledge, combining ReColorAdv with other attacks leads to the strongest existing attack even after adversarial training.

1 Introduction

There is an extensive recent literature on *adversarial examples*, small perturbations to inputs of machine learning algorithms that cause the algorithms to report an erroneous output, e.g. the incorrect label for a classifier. Adversarial examples present serious security challenges for real-world systems like self-driving cars, since a change in the environment that is not noticeable to a human may cause unexpected, unwanted, or dangerous behavior. Many methods of generating adversarial examples (called *adversarial attacks*) have been proposed [19, 4, 13, 14, 2]. Defenses against such attacks have also been explored [15, 12, 26].

Most existing attack and defense methods consider a threat model of adversarial attacks where adversarial examples can differ from normal inputs by a small ℓ_p distance. However, using this threat model that encompasses a simple definition of "small perturbation" misses other types of perturbations that may also be imperceptible to humans. For instance, small spatial perturbations have been used to generate adversarial examples [3, 23, 22].

In this paper, we propose a new class of threat models for adversarial attacks, called *functional threat models*. Under a functional threat model, adversarial examples can be generated from a regular input to a classifier by applying a *single* function to all features of the input:

$$\begin{aligned} \text{Additive threat model:} \quad & (x_1, \ldots, x_n) \rightarrow (x_1 + \delta_1, \ldots, x_n + \delta_n) \\ \textbf{Functional threat model:} \quad & (x_1, \ldots, x_n) \rightarrow (f(x_1), \ldots, f(x_n)) \end{aligned}$$

For instance, the perturbation function $f(\cdot)$ could darken every red pixel in an image, or increase the volume of every timestep in an audio sample. Functional threat models are in some ways more restrictive because features cannot be perturbed individually. However, the uniformity of the

Preprint. Under review.

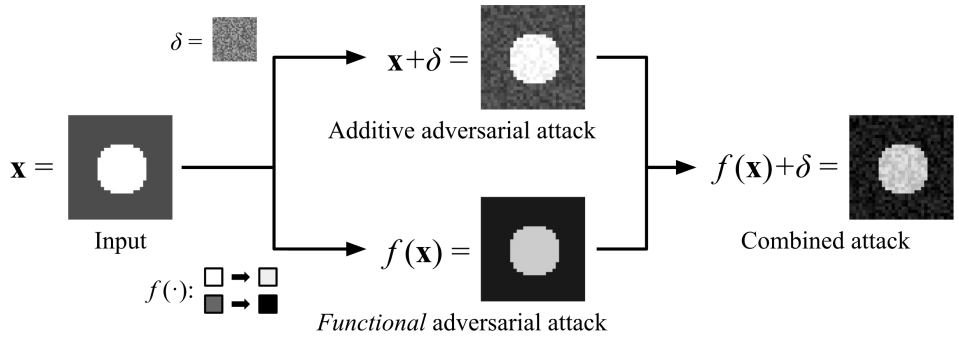


Figure 1: A visualization of an additive adversarial attack, a functional adversarial attack, and their combination. The additive attack perturbs each feature (pixel) separately, whereas the functional attack applies the same function $f(\cdot)$ to every feature.

perturbation in a functional threat model makes the change less perceptible, allowing for larger absolute modifications. For example, one could darken or lighten an entire image by quite a bit without the change becoming noticeable. This stands in contrast to separate changes to each pixel, which must be smaller to avoid becoming perceptible. We discuss various regularizations that can be applied to the perturbation function $f(\cdot)$ to ensure that even large changes are imperceptible.

The advantages and disadvantages of additive (ℓ_p) and functional threat models complement each other; additive threat models allow small, individual changes to every feature of an input while functional threat models allow large, uniform changes. Thus, we combine the threat models (see figure 1) and show that the combination encompasses more potential perturbations than either one separately, as we explain in the following theorem which is stated more precisely in section 3.2.

Theorem 1 (informal)**.** *Let* $\mathbf{x}$ *be a grayscale image with* $n \geq 2$ *pixels. Consider an additive threat model that allows changing each pixel by up to a certain amount, and a functional threat model that allows darkening or lightening the entire image by a greater amount. Then the combination of these threat models allows potential perturbations that are not allowed in either constituent threat model.*

Functional threat models can be used in a variety of domains such as images (e.g. by uniformly changing image colors), speech/audio (e.g. by changing the "accent" of an audio clip), text (e.g. by replacing a word in the entire document with its synonym), or fraud analysis (e.g. by uniformly modifying an actor's financial activities). Moreover, because functional perturbations are large and uniform, they may also be easier to use for physical adversarial examples, where the small pixel-level changes created in additive perturbations could be drowned out by environmental noise.

In this paper, we will focus on one such domain—images—and define ReColorAdv, a functional adversarial attack on pixel colors (see figure 2). In ReColorAdv, we use a flexibly parameterized function f to map each pixel color c in the input to a new pixel color $f(c)$ in an adversarial example. We regularize $f(\cdot)$ both to ensure that no color is perturbed by more than a certain amount, and to make sure that the mapping is smooth, i.e. similar colors are perturbed in a similar way. We show that ReColorAdv can use colors defined in the standard red, green, blue (RGB) color space and also in CIELUV color space, which results in less perceptually different adversarial examples (see figure 4).

We experiment by attacking defended and undefended classifiers with ReColorAdv, by itself and in combination with other attacks. We find that ReColorAdv is a strong attack, reducing the accuracy of a ResNet-32 trained on CIFAR-10 to 5.3%. Combinations of ReColorAdv and other attacks are yet more powerful; one such combination lowers a CIFAR-10 classifier's accuracy to 8.5%, even after adversarial training. This is lower than the previous strongest attack of Jordan et al. [9]. We also demonstrate the fragility of adversarial defenses based on an additive threat model by reducing the accuracy of a classifier trained with TRADES [26] to 10.1%. Although one might attempt to mitigate the ReColorAdv attack by converting images to grayscale before classification, which removes color information, we show that this simply decreases a classifier's accuracy (both natural and adversarial). Furthermore, we find that combining ReColorAdv with other attacks improves the strength of the attack without increasing the perceptual difference, as measured by LPIPS [27], of the generated adversarial example.

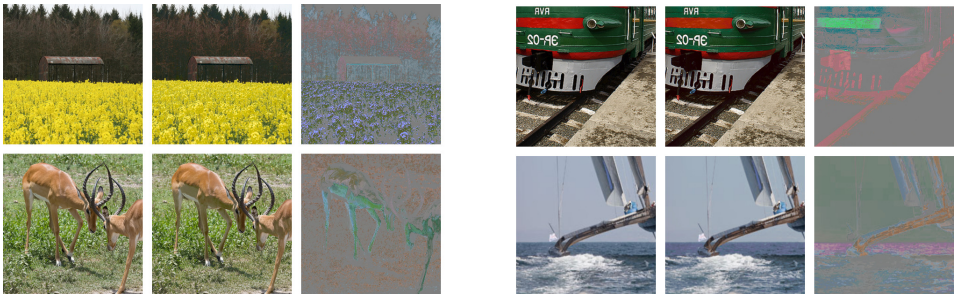


Figure 2: Four ImageNet adversarial examples generated by ReColorAdv against an Inception-v4 classifier. From left to right in each group: original image, adversarial example, magnified difference.

Our contributions are summarized as follows:

- We **introduce** a novel class of threat models, functional adversarial threat models, and combine them with existing threat models. We also describe ways of regularizing functional threat models to ensure that generated adversarial examples are imperceptible.
- **Theoretically**, we prove that additive and functional threat models combine to create a threat model that encompasses more potential perturbations than either threat model alone.
- **Experimentally**, we show that ReColorAdv, which uses a functional threat model on images, is a strong adversarial attack against image classifiers. To the best of our knowledge, combining ReColorAdv with other attacks leads to the strongest existing attack even after adversarial training.

2 Review of Existing Threat Models

In this section, we define the problem of generating an adversarial example and review existing adversarial threat models and attacks.

Problem Definition Consider a classifier $g : \mathcal{X}^n \to \mathcal{Y}$ from a feature space $\mathcal{X}^n$ to a set of labels $\mathcal{Y}$. Given an input $\mathbf{x} \in \mathcal{X}^n$, an adversarial example is a slight perturbation $\widetilde{\mathbf{x}}$ of $\mathbf{x}$ such that $g(\widetilde{\mathbf{x}}) \neq g(\mathbf{x})$; that is, $\widetilde{\mathbf{x}}$ is given a different label than $\mathbf{x}$ by the classifier. Since the aim of an adversarial example is to be perceptually indistinguishable from a normal input, $\widetilde{\mathbf{x}}$ is usually constrained to be close to $\mathbf{x}$ by some *threat model*. Formally, Jordan et al. [9] define a threat model as a function $t : \mathcal{P}(\mathcal{X}^n) \to \mathcal{P}(\mathcal{X}^n)$, where $\mathcal{P}$ denotes the power set. The function $t(\cdot)$ maps a set of classifier inputs $\mathcal{S}$ to a set of perturbed inputs $t(\mathcal{S})$ that are imperceptibly different. With this definition, we can formalize the problem of generating an adversarial example from an input:

$$\text{find } \widetilde{\mathbf{x}} \quad \text{such that } \; g(\widetilde{\mathbf{x}}) \neq g(\mathbf{x}) \;\; \text{and} \;\; \widetilde{\mathbf{x}} \in t(\{\mathbf{x}\})$$

Additive Threat Model The most common threat model used when generating adversarial examples is the additive threat model. Let $\mathbf{x} = (x_1, \ldots, x_n)$, where each $x_i \in \mathcal{X}$ is a feature of $\mathbf{x}$. For instance, x_i could correspond to a pixel in an image or the filterbank energies for a timestep in an audio sample. In an additive threat model, we assume $\widetilde{\mathbf{x}} = (x_1 + \delta_1, \ldots, x_n + \delta_n)$; that is, a value δ_i is added to each feature of $\mathbf{x}$ to generate the adversarial example $\widetilde{\mathbf{x}}$. Under this threat model, perceptual similarity is usually enforced by a bound on the norm of $\delta = (\delta_1, \ldots, \delta_n)$. Thus, the additive threat model is defined as

$$t_{\text{add}}(\mathcal{S}) \triangleq \{(x_1 + \delta_1, \ldots, x_n + \delta_n) \mid (x_1, \ldots, x_n) \in \mathcal{S}, \|\delta\| \leq \epsilon\}\,.$$

Commonly used norms include $\|\cdot\|_2$ (Euclidean distance), which constrains the sum of squares of the δ_i, $\|\cdot\|_0$, which constrains the number of features can be changed, and $\|\cdot\|_\infty$, which allows changing each feature by up to a certain amount. Note that all of the δ_i can be modified individually to generate a misclassification, as long as the norm constraint is met. Thus, a small ϵ is usually necessary because otherwise the input could be made incomprehensible by noise.

Most previous work on generating adversarial examples has employed the additive threat model. This includes gradient-based methods like FGSM [4], DeepFool [13], and Carlini & Wagner [2], and gradient-free methods like SPSA [21] and the Boundary Attack [1].

Other Threat Models Some recent work has focused on *spatial threat models*, which allow for slight perturbations of the locations of features in an input rather than perturbations of the features themselves [23, 22, 3]. Others have proposed threat models based on properties of a 3D renderer [25], modification of an image's hue and saturation [6], and inverting images [7]. See appendix D for discussion of non-additive threat models and comparison to our proposed functional threat model.

3 Functional Threat Model

In this section, we define *functional threat model* and explore its combinations with existing threat models. Recall that in the additive threat model, each feature of an input can only be perturbed by a small amount. Because all the features are changed separately, larger changes could make the input unrecognizable. Our key insight is that larger perturbations to an input should be possible if the dependencies between features are considered.

Unlike the additive threat model, in the functional threat model the features x_i are transformed by a single function $f : \mathcal{X} \to \mathcal{X}$, called the perturbation function. That is,

$$\widetilde{\mathbf{x}} = f(\mathbf{x}) = (f(x_1), \ldots, f(x_n))$$

Under this threat model, features which have the same value in the input must be mapped to the same value in the adversarial example. Even large perturbations allowed by a functional threat model may be imperceptible to human eyes because they preserve dependencies between features (for example, shape boundaries and shading in images, see figure 1). Note that the features x_i which are modified by the perturbation function $f(\cdot)$ need not be scalars; depending on the application, vector-valued features may be useful.

3.1 Regularizing Functional Threat Models

In the functional threat model, various regularizations can be used to ensure that the change remains imperceptible. In general, we can enforce that $f \in \mathcal{F}$; $\mathcal{F}$ is a family of allowed perturbation functions. For instance, we may want to bound by some small ϵ the maximum difference between the input and output of the perturbation function. In that case, we will have:

$$\mathcal{F}_{\text{diff}} \triangleq \{f : \mathcal{X} \to \mathcal{X} \mid \forall x_i \in \mathcal{X} \; \|f(x_i) - x_i\| \leq \epsilon\} \tag{1}$$

$\mathcal{F}_{\text{diff}}$ prevents absolute changes of more than a certain amount. Note that the ϵ bound may be higher than that of an additive model, since uniform changes are less perceptible. However, this regularization may not be enough to prevent noticeable changes. $\mathcal{F}_{\text{diff}}$ still includes functions that map similar (but not identical) features very differently. Therefore, a second constraint could be used that forces similar features to be perturbed similarly:

$$\mathcal{F}_{\text{smooth}} \triangleq \{f \mid \forall x_i, x_j \in \mathcal{X} \; \|x_i - x_j\| \leq r \Rightarrow \|(f(x_i) - x_i) - (f(x_j) - x_j)\| \leq \epsilon_{\text{smooth}}\} \tag{2}$$

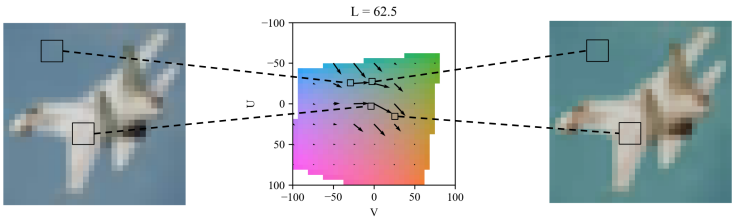
$\mathcal{F}_{\text{smooth}}$ requires that similar features are perturbed in the same "direction". For instance, if green pixels in an image are lightened, then yellow-green pixels should be as well.

Depending on the application, these constraints or others may be needed to maintain an imperceptible change. We may want to choose $\mathcal{F}$ to be $\mathcal{F}_{\text{diff}}$, $\mathcal{F}_{\text{smooth}}$, $\mathcal{F}_{\text{diff}} \cap \mathcal{F}_{\text{smooth}}$, or an entirely different family of functions. Once we have chosen an $\mathcal{F}$, we can define a corresponding functional threat model as

$$t_{\text{func}}(\mathcal{S}) \triangleq \{(f(x_1), \ldots, f(x_n)) \mid (x_1, \ldots, x_n) \in \mathcal{S}, f \in \mathcal{F}\}$$

3.2 Combining Threat Models

Jordan et al. [9] argue that combining multiple threat models allows better approximation of the complete set of adversarial perturbations which are imperceptible. Here, we show that combining the additive threat model with a simple functional threat model can allow adversarial examples which are not allowable by either model on its own. The following theorem (proved in appendix A) demonstrates this on images for a combination of an additive threat model which allows changing each pixel by a small, bounded amount and a functional threat model which allows darkening or lightening the entire image by up to a larger amount, both of which are arguably imperceptible transformations.



"airplane" $f(\cdot)$ "dog"

Figure 3: ReColorAdv transforms each pixel in the input image $\mathbf{x}$ (left) by the same function $f(\cdot)$ (center) to produce an adversarial example $\widetilde{\mathbf{x}}$ (right). The perturbation function f is shown as a vector field in CIELUV color space.

Theorem 1. *Let $\mathbf{x}$ be a grayscale image with $n \geq 2$ pixels, i.e. $\mathbf{x} \in [0,1]^n = \mathcal{X}^n$. Let t_{add} be an additive threat model where the ℓ_∞ distance between input and adversarial example is bounded by ϵ_1, i.e. $\|(\delta_1, \ldots, \delta_n)\|_\infty \leq \epsilon_1$. Let t_{func} be a functional threat model where $f(x) = c\,x$ for some $c \in [1-\epsilon_2, 1+\epsilon_2]$ such that $\epsilon_2 > \epsilon_1 > 0$. Let $t_{combined} = t_{add} \circ t_{func}$. Then the combined threat model allows adversarial perturbations which are not allowed by either constituent threat model. Formally, if $\mathcal{S} \subseteq \mathcal{X}^n$ contains an image $\mathbf{x}$ that is not dark, that is $\exists\, x_i$ s.t. $x_i > \epsilon_1/\epsilon_2$, then*

$$t_{combined}(\mathcal{S}) \supsetneq t_{add}(\mathcal{S}) \cup t_{func}(\mathcal{S}) \qquad \text{or equivalently} \qquad \exists\, \widetilde{\mathbf{x}} \text{ s.t. } \begin{array}{l} \widetilde{\mathbf{x}} \in t_{combined}(\mathcal{S}) \\ \widetilde{\mathbf{x}} \notin t_{add}(\mathcal{S}) \cup t_{func}(\mathcal{S}) \end{array}$$

4 ReColorAdv: Functional Adversarial Attacks on Image Colors

In this section, we define ReColorAdv, a novel adversarial attack against image classifiers that leverages a functional threat model. ReColorAdv generates adversarial examples to fool image classifiers by uniformly changing colors of an input image. We treat each pixel x_i in the input image $\mathbf{x}$ as a point in a 3-dimensional color space $\mathcal{C} \subseteq [0,1]^3$. For instance, $\mathcal{C}$ could be the normal RGB color space. In section 4.1, we discuss our use of alternative color spaces. We leverage a perturbation function $f : \mathcal{C} \to \mathcal{C}$ to produce the adversarial example. Specifically, each pixel in the output $\widetilde{\mathbf{x}}$ is perturbed from the input $\mathbf{x}$ by applying $f(\cdot)$ to the color in that pixel:

$$x_i = (c_{i,1}, c_{i,2}, c_{i,3}) \in \mathcal{C} \subseteq [0,1]^3 \quad \rightarrow \quad \widetilde{x}_i = (\widetilde{c}_{i,1}, \widetilde{c}_{i,2}, \widetilde{c}_{i,3}) = f(c_{i,1}, c_{i,2}, c_{i,3})$$

For the purposes of finding an $f(\cdot)$ that generates a successful adversarial example, we need a parameterization of the function that allows both flexibility and ease of computation. To accomplish this, we let $\mathcal{G} = g_1, \ldots, g_m \subseteq [0,1]^3$ be a discrete grid of points (or point lattice) where f is explicitly defined. That is, we define parameters $\theta_1, \ldots, \theta_m$ and let $f(g_i) = \theta_i$. For points not on the grid, i.e. $x_i \notin \mathcal{G}$, we define $f(x_i)$ using trilinear interpolation. Trilinear interpolation considers the "cube" of the lattice points g_j surrounding the argument x_i and linearly interpolates the explicitly defined θ_j values at the 8 corners of this cube to calculate $f(x_i)$.

Constraints on the perturbation function We enforce two constraints on $f(\cdot)$ to ensure that the crafted adversarial example is indistinguishable from the original image. These constraints are based on slight modifications of $\mathcal{F}_{\text{diff}}$ and $\mathcal{F}_{\text{smooth}}$ defined in section 3.1. First, we ensure that no pixel can be perturbed by more than a certain amount along each dimension in color space:

$$\mathcal{F}_{\text{diff-col}} \triangleq \{f : \mathcal{C} \to \mathcal{C} \mid \forall (c_1, c_2, c_3) \in \mathcal{G} \quad |c_i - \widetilde{c}_i| < \epsilon_i \quad i = 1, 2, 3\}$$

This particular formulation allows us to set different bounds $(\epsilon_1, \epsilon_2, \epsilon_3)$ on the maximum perturbation along each dimension in color space. We also define a constraint based on $\mathcal{F}_{\text{smooth}}$, but instead of using a radius parameter r as in (2) we consider the neighbors $\mathcal{N}(g_j)$ of each lattice point g_j in the grid $\mathcal{G}$:

$$\mathcal{F}_{\text{smooth-col}} \triangleq \{f : \mathcal{C} \to \mathcal{C} \mid \forall g_j \in \mathcal{X}, g_k \in \mathcal{N}(g_j) \quad \|(f(g_j) - g_j) - (f(g_k) - g_k)\|_2 \leq \epsilon_{\text{smooth}}\}$$

In the above, $\|\cdot\|_2$ is the ℓ_2 (Euclidean) norm in the color space $\mathcal{C}$. We define our set of allowable perturbation functions as $\mathcal{F}_{\text{col}} = \mathcal{F}_{\text{diff-col}} \cap \mathcal{F}_{\text{smooth-col}}$ with parameters $(\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_{\text{smooth}})$.

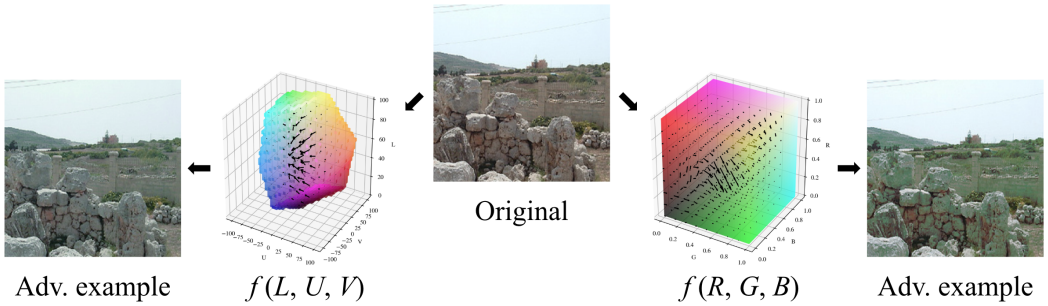


Figure 4: The color space used affects the adversarial example produced by ReColorAdv. The original image at center is attacked by ReColorAdv with the CIELUV color space (left) and RGB color space (right). The RGB color space results in noticeable bright green artifacts in the adversarial example, while the perceptually accurate CIELUV color space produces a more realistic perturbation.

Optimization To generate an adversarial example with ReColorAdv, we wish to minimize $\mathcal{L}_{\text{adv}}(f, x)$ subject to $f \in \mathcal{F}_{\text{col}}$, where $\mathcal{L}_{\text{adv}}$ enforces the goal of generating an adversarial example that is misclassified and is defined as the f_6 loss from Carlini and Wagner [2]. When solving this constrained minimization problem, it is easy to constrain $f \in \mathcal{F}_{\text{diff-col}}$ by clipping the perturbation of each color to be within the ϵ_i bounds. However, it is difficult to enforce $f \in \mathcal{F}_{\text{smooth-col}}$ directly. Thus, we instead solve a Lagrangian relaxation where the smoothness constraint is replaced by an additional regularization term:

$$\arg\min_{f \in \mathcal{F}_{\text{diff-col}}} \mathcal{L}_{\text{adv}}(f, \mathbf{x}) + \lambda \mathcal{L}_{\text{smooth}}(f) \qquad (*)$$

$$\mathcal{L}_{\text{smooth}}(f) \triangleq \sum_{g_j \in \mathcal{G}} \sum_{g_k \in \mathcal{N}(g_j)} \|(f(g_j) - g_j) - (f(g_k) - g_k)\|_2$$

Our $\mathcal{L}_{\text{smooth}}$ is similar to the loss function used by Xiao et al. [23] to ensure a smooth flow field. We use the projected gradient descent (PGD) optimization algorithm to solve $(*)$.

4.1 RGB vs. LUV Color Space

Most image classifiers take as input an array of pixels specified in RGB color space, but the RGB color space has two disadvantages. The ℓ_p distance between points in RGB color space is weakly correlated with the perceptual difference between the colors they represent. Also, RGB gives no separation between the luma (brightness) and chroma (hue/colorfulness) of a color.

In contrast, the CIELUV color space separates luma from chroma and places colors such that the Euclidean distance between them is roughly equivalent to the perceptual difference [18]. CIELUV presents a color by three components (L, U, V); L is the luma while U and V together define the chroma. We run experiments using both RGB and CIELUV color spaces. CIELUV allows us to regularize the perturbation function $f(\cdot)$ perceptually accurately (see figure 4 and appendix B.1).

5 Experiments

We evaluate ReColorAdv against defended and undefended neural networks on CIFAR-10 [11] and ImageNet [17]. For CIFAR-10 we evaluate the attack against a ResNet-32 [5] and for ImageNet we evaluate against an Inception-v4 network [20]. We also consider all combinations of ReColorAdv with *delta* attacks, which use an ℓ_∞ additive threat model with bound $\epsilon = 8/255$, and the *StAdv* attack of Xiao et al. [23] that perturbs images spatially through a flow field. See appendix B for a full discussion of the hyperparameters and computing infrastructure used in our experiments.

5.1 Adversarial Training

We first experiment by attacking adversarially trained models with ReColorAdv and other attacks. For each combination of attacks, we adversarially train a ResNet-32 on CIFAR-10 against that particular

Table 1: Accuracy of adversarially trained models against various combinations of attacks on CIFAR-10. Columns correspond to attacks and rows correspond to models trained against a particular attack. C(-RGB) is ReColorAdv using CIELUV (RGB) color space, D is delta attack, and S is StAdv attack. TRADES is the method of Zhang et al. [26]. For classifiers marked (B&W), the images are converted to black-and-white before classification.

	Attack								
Defense ↓	**None**	**C-RGB**	**C**	**D**	**S**	**C+S**	**C+D**	**S+D**	**C+S+D**
Undefended	92.3	8.3	5.3	**0.0**	2.2	1.8	**0.0**	**0.0**	**0.0**
C	89.2	47.4	50.3	5.9	4.6	4.6	1.7	**0.5**	0.9
D	84.7	77.3	61.9	32.8	18.6	17.2	17.3	4.3	**4.2**
S	82.7	20.3	15.7	0.8	29.9	10.7	**0.2**	**0.2**	**0.2**
C+S	82.3	47.2	44.6	7.5	26.2	20.2	3.5	2.2	**2.0**
C+D	84.3	74.7	63.5	35.2	14.0	13.4	22.2	4.5	**4.2**
S+D	82.0	69.3	53.9	36.5	26.4	21.1	21.7	9.6	**8.0**
C+S+D	82.3	70.1	56.4	35.5	25.5	21.4	23.4	10.0	**8.5**
TRADES	84.2	81.6	72.8	53.7	31.2	27.5	39.3	11.1	**10.1**
Undefended (B&W)	87.9	7.3	6.1	**0.0**	1.6	1.6	**0.0**	**0.0**	**0.0**
C (B&W)	85.6	46.7	43.8	5.0	3.5	3.8	1.3	**0.4**	0.7

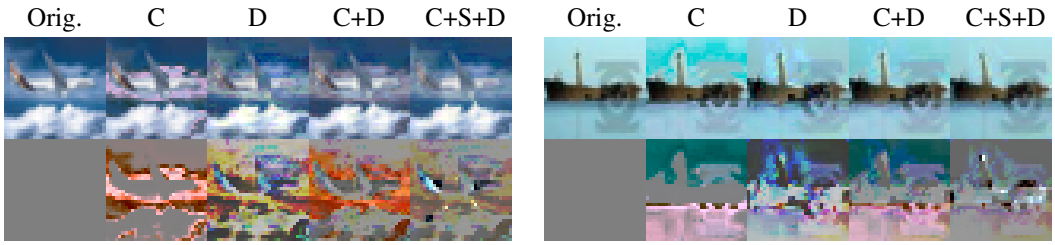


Figure 5: Adversarial examples generated with combinations of attacks against a CIFAR-10 WideResNet [24] trained using TRADES; the difference from the original is shown below each example. Combinations of attacks tend to produce less perceptible changes than the attacks do separately.

combination. We attack each of these adversarially trained models with all combinations of attacks. The results of this experiment are shown in the first part of table 1.

Combination attacks are most powerful As expected, combinations of attacks are the strongest against defended and undefended classifiers. In particular, the ReColorAdv + StAdv + delta attack often resulted in the lowest classifier accuracy. The accuracy of only 8.5% after adversarially training against the ReColorAdv + StAdv + delta attack is the lowest we know of.

Transferability of defenses To some degree, the attacks investigated are orthogonal; that is, a model trained against a particular attack is less effective against others. StAdv is especially separate from the other two attacks; models trained against StAdv attacks are still very vulnerable to ReColorAdv and delta attacks. However, the ReColorAdv and delta attacks allow more transferable defenses between each other. These results are likely due to the fact that both the delta and ReColorAdv attacks operate on a per-pixel basis, whereas the StAdv attack allows spatial movement of features across pixels.

Effect of color space The ReColorAdv attack using CIELUV color space is stronger than that using RGB color space. In addition, the CIELUV color space produces less perceptible perturbations (see figure 4). This highlights the need for using perceptually accurate models of color when designing and defending against adversarial examples.

5.2 Other Defenses

TRADES TRADES is a training algorithm for deep neural networks that aims to improve robustness to adversarial examples by optimizing a surrogate loss [26]. The algorithm is designed around

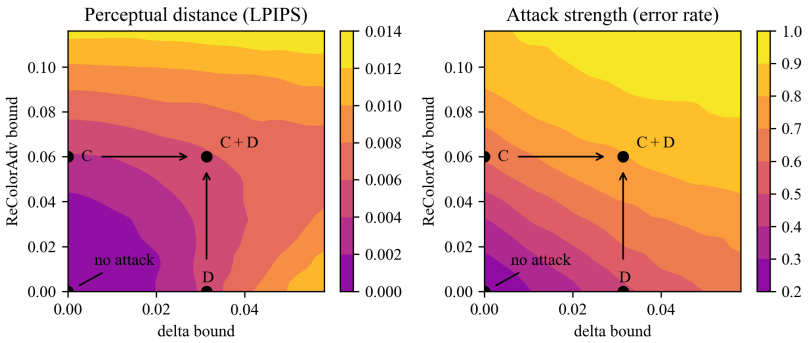


Figure 6: The perceptual distortion (LPIPS) and strength (error rate) of combinations of ReColorAdv and delta attacks with various bounds. The annotated points mark the bounds used in other experiments: C is ReColorAdv, D is a delta attack, and C+D is their combination. Combining the attacks does not increase perceptable change by much (left), but it greatly increases attack strength (right).

an additive threat model, but we evaluate a TRADES-trained classifier on all combinations of attacks (see the second part of table 1). This is the best defense method against almost all attacks, despite having been trained based on just an additive threat model. However, the combined ReColorAdv + StAdv + delta attack still reduces the accuracy of the classifier to just 10.1%.

Grayscale conversion Since ReColorAdv attacks input images by changing their colors, one might attempt to mitigate the attack by converting all images to grayscale before classification. This could reduce the potential perturbations available to ReColorAdv since altering the chroma of a color would not affect the grayscale image; only changes in luma would. We train models on CIFAR-10 that convert all images to grayscale as a preprocessing step both with and without adversarial training against ReColorAdv. The results of this experiment (see the third part of table 1) show that conversion to grayscale is not a viable defense against ReColorAdv. In fact, the natural accuracy and robustness against almost all attacks decreases when applying grayscale conversion.

5.3 Perceptual Distance

We quantify the perceptual distortion caused by ReColorAdv attacks using the Learned Perceptual Image Patch Similarity (LPIPS) metric, a distance measure between images based on deep network activations which has been shown to correlate with human perception [27]. We combine ReColorAdv and delta attacks and vary the bound of each attack (see figure 6). We find that the attacks can be combined without much increase, or with even sometimes a *decrease*, in perceptual difference. As Jordan et al. [9] find for combinations of StAdv and delta attacks, the lowest perceptual difference at a particular attack strength is achieved by a combination of ReColorAdv and delta attacks.

6 Conclusion

We have presented *functional threat models* for adversarial examples, which allow large, uniform changes to an input. They can be combined with additive threat models to provably increase the potential perturbations allowed in an adversarial attack. In practice, the ReColorAdv attack, which leverages a functional threat model against image pixel colors, is a strong adversarial attack on image classifiers. It can also be combined with other attacks to produce yet more powerful attacks—even after adversarial training—without a significant increase in perceptual distortion. Besides images, functional adversarial attacks could be designed for audio, text, and other domains. It will be crucial to develop defense methods against these attacks, which encompass a more complete threat model of which potential adversarial examples are imperceptible to humans.

References

[1] Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models. *International Conference on*

Learning Representations, February 2018.

[2] Nicholas Carlini and David Wagner. Towards Evaluating the Robustness of Neural Networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57. IEEE, 2017.

[3] Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. A Rotation and a Translation Suffice: Fooling CNNs with Simple Transformations. *arXiv preprint arXiv:1712.02779*, 2017.

[4] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. In *International Conference on Learning Representations*, 2015.

[5] 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.

[6] Hossein Hosseini and Radha Poovendran. Semantic Adversarial Examples. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1614–1619, 2018.

[7] Hossein Hosseini, Baicen Xiao, Mayoore Jaiswal, and Radha Poovendran. On the Limitation of Convolutional Neural Networks in Recognizing Negative Images. In *16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 352–358. IEEE, 2017.

[8] Yerlan Idelbayev. Proper ResNet Implementation for CIFAR10/CIFAR100 in pytorch, 2018. URL `https://github.com/akamaster/pytorch_resnet_cifar10`. original-date: 2018-01-15T09:50:56Z.

[9] Matt Jordan, Naren Manoj, Surbhi Goel, and Alexandros G. Dimakis. Quantifying Perceptual Distortion of Adversarial Examples. *arXiv preprint arXiv:1902.08265*, February 2019. arXiv: 1902.08265.

[10] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[11] Alex Krizhevsky and Geoffrey Hinton. Learning Multiple Layers of Features from Tiny Images. Technical report, Citeseer, 2009.

[12] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. 2018.

[13] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a Simple and Accurate Method to Fool Deep Neural Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2574–2582, 2016.

[14] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The Limitations of Deep Learning in Adversarial Settings. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 372–387. IEEE, 2016.

[15] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks. In *IEEE Symposium on Security and Privacy (SP)*, pages 582–597. IEEE, 2016.

[16] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic Differentiation in PyTorch. In *NIPS-W*, 2017.

[17] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, and Michael Bernstein. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[18] Jim Schwiegerling. *Field Guide to Visual and Ophthalmic Optics*. SPIE Publications, Bellingham, Wash, November 2004. ISBN 978-0-8194-5629-8.

[19] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing Properties of Neural Networks. In *International Conference on Learning Representations*, 2014.

[20] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A. Alemi. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[21] Jonathan Uesato, Brendan O'Donoghue, Pushmeet Kohli, and Aaron Oord. Adversarial Risk and the Dangers of Evaluating Against Weak Attacks. In *International Conference on Machine Learning*, pages 5025–5034, July 2018. URL `http://proceedings.mlr.press/v80/uesato18a.html`.

[22] Eric Wong, Frank R. Schmidt, and J. Zico Kolter. Wasserstein Adversarial Examples via Projected Sinkhorn Iterations. *arXiv preprint arXiv:1902.07906*, February 2019. arXiv: 1902.07906.

[23] Chaowei Xiao, Jun-Yan Zhu, Bo Li, Warren He, Mingyan Liu, and Dawn Song. Spatially Transformed Adversarial Examples. *arXiv preprint arXiv:1801.02612*, 2018.

[24] Sergey Zagoruyko and Nikos Komodakis. Wide Residual Networks. In *British Machine Vision Conference*. British Machine Vision Association, 2016.

[25] Xiaohui Zeng, Chenxi Liu, Yu-Siang Wang, Weichao Qiu, Lingxi Xie, Yu-Wing Tai, Chi Keung Tang, and Alan L. Yuille. Adversarial Attacks Beyond the Image Space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. arXiv: 1711.07183.

[26] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P. Xing, Laurent El Ghaoui, and Michael I. Jordan. Theoretically Principled Trade-off Between Robustness and Accuracy. *arXiv preprint arXiv:1901.08573*, 2019.

[27] 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.

A Combining the Additive and Functional Threat Models

Here we provide a proof of Theorem 1.

Threat model Let $\mathbf{x}$ be a grayscale image with $n \geq 2$ pixels, i.e. $\mathbf{x} \in [0,1]^n = \mathcal{X}^n$. Let t_{add} be an additive threat model where the ℓ_∞ distance between input and adversarial example is bounded by ϵ_1, i.e. $\|(\delta_1, \ldots, \delta_n)\|_\infty \leq \epsilon_1$. Let t_{func} be a functional threat model where $f(x) = c\,x$ for some $c \in [1-\epsilon_2, 1+\epsilon_2]$ and let $\epsilon_2 > \epsilon_1 > 0$. The additive threat model allows individually changing each pixel's value by up to ϵ_1; the functional threat model allows darkening or lightening the entire image by up to a proportion of ϵ_2. Both of these are arguably imperceptible perturbations for small enough ϵ_1 and ϵ_2. We also consider $t_{\text{combined}} = t_{\text{add}} \circ t_{\text{func}}$:

$$t_{\text{combined}}(\mathcal{S}) \triangleq \left\{ (c\,x_1 + \delta_1, \ldots, c\,x_n + \delta_n) \;\middle|\; \begin{array}{l} (x_1, \ldots, x_n) \in \mathcal{S} \\ |\delta_i| \leq \epsilon_1 \\ c \in [1-\epsilon_2, 1+\epsilon_2] \end{array} \right\} \tag{3}$$

This combined threat model allows darkening or lightening the image, followed by changing each pixel value individually by a small amount.

Theorem 1 (restated)**.** *Let $\mathcal{S} \in \mathcal{P}(\mathcal{X}^n)$ be a set of inputs such that $\mathcal{S}$ contains an image that is not too dark; that is, $\exists\, \mathbf{x} \in \mathcal{S}$ for which $\exists\, x_i$ s.t. $x_i > \epsilon_1/\epsilon_2$. Then*

$$t_{combined}(\mathcal{S}) \;\supsetneq\; t_{add}(\mathcal{S}) \cup t_{func}(\mathcal{S}) \qquad \text{or equivalently} \qquad \exists\, \widetilde{\mathbf{x}} \text{ s.t. } \begin{array}{l} \widetilde{\mathbf{x}} \in t_{combined}(\mathcal{S}) \\ \widetilde{\mathbf{x}} \notin t_{add}(\mathcal{S}) \cup t_{func}(\mathcal{S}) \end{array}$$

Proof. The above two statements are equivalent, so we focus on the formulation on the right. We calculate $\widetilde{\mathbf{x}}$ and show that it satisfies the given criteria. Let $\mathbf{x} \in \mathcal{S}$ such that $\exists\, x_i$ s.t. $x_i > \epsilon_1/\epsilon_2$. Without loss of generality, assume that in particular $x_2 > \epsilon_1/\epsilon_2$. Then let

$$\widetilde{\mathbf{x}} = ((1-\epsilon_2)\,x_1 + \epsilon_1, (1-\epsilon_2)\,x_2, \ldots, (1-\epsilon_2)\,x_n)$$

First, we show that $\widetilde{\mathbf{x}} \in t_{\text{combined}}(\mathcal{S})$. Using the definition of t_{combined} in (3), we set $c = 1-\epsilon_2$, $\delta_1 = \epsilon_1$, and $\delta_2 = \cdots = \delta_n = 0$, which generates $\widetilde{\mathbf{x}}$. These values clearly satisfy the constraints in (3).

Second, we prove that $\widetilde{\mathbf{x}} \notin t_{\text{add}}(\mathcal{S})$ by contradiction. Say that $\widetilde{\mathbf{x}} \in t_{\text{add}}(\mathcal{S})$. Then $\exists\, \delta_1, \delta_2, \ldots, \delta_n$ such that $\widetilde{x}_i = x_i + \delta_i$ and $\|\delta_i\| \leq \epsilon_1$. Consider δ_2, which must satisfy $\widetilde{x}_2 = (1-\epsilon_2)\,x_2 = x_2 + \delta_2$, or alternatively $\delta_2 = x_2 - (1-\epsilon_2)\,x_2 = \epsilon_2\,x_2$. However, $x_2 > \epsilon_1/\epsilon_2$ implies that $\delta_2 > \epsilon_1$, which is a contradiction since the constraints on t_{add} specify that $|\delta_2| \leq \epsilon_1$. Thus, $\widetilde{\mathbf{x}} \notin t_{\text{add}}(\mathcal{S})$.

Third, we prove that $\widetilde{\mathbf{x}} \notin t_{\text{func}}(\mathcal{S})$, again by contradiction. Say that $\widetilde{\mathbf{x}} \in t_{\text{func}}(\mathcal{S})$. Then $\exists\, c \in [1-\epsilon_2, 1+\epsilon_2]$ such that $\widetilde{x}_i = c\,x_i$ for all i. Considering $i = 1, 2$, we have the following system of equations:

$$\begin{aligned} \widetilde{x}_1 &= cx_1 = (1-\epsilon_2)\,x_1 + \epsilon_1 \\ \widetilde{x}_2 &= cx_2 = (1-\epsilon_2)\,x_2 \end{aligned}$$

From the second equation, we have $c = 1-\epsilon_2$. However, using this in the first equation gives $(1-\epsilon_2)\,x_1 = (1-\epsilon_2)\,x_1 + \epsilon_1$, which implies $0 = \epsilon_1$. This is a contradiction since $\epsilon_1 > 0$, showing that $\widetilde{\mathbf{x}} \notin t_{\text{func}}(\mathcal{S})$. ■

B Experimental Setup

We implement ReColorAdv using the `mister_ed` library [9] and PyTorch [16]. Adversarial examples are generated by 100 iterations of PGD using the Adam optimizer [10] with learning rate 0.001. After all iterations have completed, we choose the result of the iteration with the lowest loss as the adversarial example.

When combining attacks, we apply multiple attacks sequentially to the input example and optimize over the parameters of all attacks simultaneously, similarly to Jordan et al. [9].

In all adversarial training experiments on CIFAR-10, we begin with a trained ResNet32 [8] and then train it further on batches which are half original training data and half adversarial examples. We adversarially train with a batch size of 500 for 50 epochs. We preprocess images after adversarial perturbation, but before classification, by standardizing them based on the mean and standard deviation of each channel for all images in the dataset. The CIFAR-10 dataset can be obtained from `https://www.cs.toronto.edu/~kriz/cifar.html`.

In CIELUV color space (see section 4.1), we define

$$(c_1, c_2, c_3) = \left(\frac{L}{100}, \frac{U + 100}{200}, \frac{V + 100}{200} \right) \tag{4}$$

so that $(c_1, c_2, c_3) \in [0, 1]^3$.

For the experiments described in section 5.3, we use LPIPS v0.1 with AlexNet.

B.1 Regularization Parameters

The objective function and constraints described in section 4 include a number of constants that can be used to regularize the outputs of the ReColorAdv attack. Changing these constants alters the strength of the attack and the perceptual similarity of a generated adversarial example to the input.

First, ϵ_1, ϵ_2, and ϵ_3 control the maximum amount by which a color in $\mathbf{x}$ can be changed to produce $\widetilde{\mathbf{x}}$. For RGB color space, we set $\epsilon_1 = \epsilon_2 = \epsilon_3 = 0.1$; that is, each channel of a color can change by up to $\sim 25/255$. This is greater than the usual $\epsilon = 8/255$ allowed for adversarial examples, but we find that the uniform perturbation used by the functional threat model allows each pixel to change by a greater amount while remaining almost indistinguishable. For the CIELUV color space, we let $\epsilon_1 = \epsilon_2 = \epsilon_3 = 0.06$. This corresponds to a maximum change of 6 in L and a maximum change of 3 in U and V, since we find that changes in luma are usually less noticeable than changes in chroma. The ϵ_i values for RGB and CIELUV color spaces result in similar total amounts of perturbation, but the CIELUV color space allows the perturbation to be greater in areas where it is less noticeable.

Second, we can control the resolution of the grid $\mathcal{G}$ over which the perturbation function $f(\cdot)$ is parameterized. Let $R_1 \times R_2 \times R_3$ be the resolution of $\mathcal{G}$. Lowering the resolution in a particular dimension acts as a regularizer because it allows less variation in how colors are transformed along that dimension. For RGB color space, we use $R_1 = R_2 = R_3 = 25$. However, for CIELUV color space, we use $R_1 = 16$ and $R_2 = R_3 = 32$. With a high R_1 value, we find that the attack sometimes recolors different values of a particular hue very differently. For instance, the attack might make the light parts of a white car green and the dark parts purple. Lowering R_1 forces the attack to alter these colors more similarly.

Finally, λ controls the importance of the smoothness optimization term $\mathcal{L}_{\text{smooth}}$. We always set $\lambda = 0.05$.

C Learning Rate Experiments

We consistently use Adam with a learning rate of 0.001 throughout the main paper to craft adversarial examples. However, we also experimented with a learning rate of 0.01. The results of these experiments are shown below similarly to table 1. All numbers reported are accuracy over the CIFAR-10 test set. Each column corresponds to an attack and each row corresponds to a model trained against a particular attack. C(-RGB) is ReColorAdv using CIELUV (RGB) color space, D is delta attack, and S is StAdv attack. TRADES is the method of Zhang et al. [26]. For classifiers marked (B&W), the images are converted to black-and-white before classification. The learning rate used in an attack is marked above that attack or to the right when the attack is used in adversarial training. There are a couple interesting conclusions that can be drawn from this experiment:

- The higher learning rate (0.01) is stronger against TRADES and undefended networks. A ReColorAdv + StAdv + delta (C+S+D) attack with learning rate 0.01 against a TRADES-trained classifier reduces its accuracy to just 6.0%, compared with 10.1% at learning rate 0.001.

- Adversarial training against an attack at the higher learning rate (0.01) increases robustness against that attack but lowers it against other attacks. For instance, consider the network defended against C+S+D with learning rate 0.01. This network achieves 15.0% accuracy against attacks of the same type, but the accuracy decreases to 7.1% against some other attacks. In contrast, adversarial training against attacks at the lower learning rate (0.001) leads to more robustness across different attacks.

		Attack (learning rate = 0.01)								
Defense	**LR**	**None**	**C-RGB**	**C**	**D**	**S**	**C+S**	**C+D**	**S+D**	**C+S+D**
Undefended		92.3	5.1	3.8	**0.0**	1.5	1.5	**0.0**	**0.0**	**0.0**
C	0.01	87.8	37.4	45.5	4.7	3.2	2.9	1.2	**0.2**	0.4
D	0.01	88.8	40.4	22.7	32.7	4.2	4.3	15.0	5.0	**4.0**
S	0.01	89.3	11.5	9.8	0.3	29.0	9.5	0.4	0.4	**0.3**
C+S	0.01	90.5	27.0	24.3	2.8	31.4	23.0	**2.1**	2.8	**2.1**
C+D	0.01	88.3	46.3	32.7	34.4	6.0	5.4	22.6	**4.7**	4.8
S+D	0.01	88.0	25.3	17.4	28.4	9.3	**8.1**	22.6	17.0	13.9
C+S+D	0.01	89.0	32.3	23.8	29.8	13.4	**11.4**	26.1	17.4	15.0
C	0.001	89.2	37.2	46.6	5.1	3.4	3.0	1.1	**0.3**	**0.3**
D	0.001	84.7	72.9	57.4	30.8	12.2	11.2	12.6	2.4	**1.8**
S	0.001	82.7	14.9	11.9	0.5	22.2	6.7	**0.1**	0.2	**0.1**
C+S	0.001	82.3	37.5	40.4	5.9	18.5	13.1	1.9	0.9	**0.8**
C+D	0.001	84.3	70.8	60.0	33.8	9.4	8.7	18.1	**1.8**	1.9
S+D	0.001	82.0	65.2	49.9	35.0	18.5	14.0	16.5	5.5	**4.5**
C+S+D	0.001	82.3	65.8	53.0	34.8	16.8	14.7	18.3	5.1	**5.0**
TRADES		84.2	79.7	69.2	53.5	21.0	17.8	33.8	6.6	**6.0**
Undefended (B&W)		87.9	4.7	4.8	**0.0**	1.6	1.5	**0.0**	0.1	**0.0**
C (B&W)	0.01	84.7	40.4	41.7	4.5	2.4	2.4	1.0	**0.2**	0.3
C (B&W)	0.001	85.6	37.8	40.7	4.0	2.5	2.5	0.7	**0.3**	**0.3**

		Attack (learning rate = 0.001)								
Defense	**LR**	**None**	**C-RGB**	**C**	**D**	**S**	**C+S**	**C+D**	**S+D**	**C+S+D**
Undefended		92.3	8.3	5.3	**0.0**	2.2	1.8	**0.0**	**0.0**	**0.0**
C	0.01	87.8	46.2	48.4	5.9	4.5	4.4	1.6	**0.3**	0.7
D	0.01	88.8	43.7	25.4	26.4	4.1	**3.8**	15.7	8.3	7.9
S	0.01	89.3	18.7	13.9	0.4	13.8	8.4	0.8	**0.6**	0.9
C+S	0.01	90.5	39.1	32.3	4.3	22.7	17.5	**2.8**	3.5	3.3
C+D	0.01	88.3	49.1	35.6	29.2	**5.3**	5.4	20.3	8.7	8.3
S+D	0.01	88.0	25.8	17.7	10.7	4.5	**4.1**	9.3	6.1	5.9
C+S+D	0.01	89.0	33.9	24.9	15.7	7.5	**7.1**	13.0	8.4	8.5
C	0.001	89.2	47.4	50.3	5.9	4.6	4.6	1.7	**0.5**	0.9
D	0.001	84.7	77.3	61.9	32.8	18.6	17.2	17.3	4.3	**4.2**
S	0.001	82.7	20.3	15.7	0.8	29.9	10.7	**0.2**	**0.2**	**0.2**
C+S	0.001	82.3	47.2	44.6	7.5	26.2	20.2	3.5	2.2	**2.0**
C+D	0.001	84.3	74.7	63.5	35.2	14.0	13.4	22.2	4.5	**4.2**
S+D	0.001	82.0	69.3	53.9	36.5	26.4	21.1	21.7	9.6	**8.0**
C+S+D	0.001	82.3	70.1	56.4	35.5	25.5	21.4	23.4	10.0	**8.5**
TRADES		84.2	81.6	72.8	53.7	31.2	27.5	39.3	11.1	**10.1**
Undefended (B&W)		87.9	7.3	6.1	**0.0**	1.6	1.6	**0.0**	**0.0**	**0.0**
C (B&W)	0.01	84.7	49.3	44.9	5.4	4.1	3.8	1.6	0.5	0.7
C (B&W)	0.001	85.6	46.7	43.8	5.0	3.5	3.8	1.3	**0.4**	0.7

D Non-Additive Threat Models

Here, we discuss some other non-additive adversarial threat models that have been explored in the literature and how our work differs from them.

Spatial Threat Models Some recent work has focused on *spatial threat models*, which allow for slight perturbations of the locations of features in an input rather than perturbations of the features themselves. Xiao et al. [23] propose StAdv, which optimizes the parameters of a smooth flow field that moves each pixel of an input image by a small, bounded distance to generate an example that fools the classifier. Wong et al. [22] bound the Wasserstein distance between the original input and the adversarial example. Engstrom et al. [3] apply an small rotation and translation to an input image to generate a misclassification.

Other Threat Models A few papers have focused on threat models that are neither additive or spatial. Zeng et al. [25] perturb the properties of a 3D renderer to render an image of an object which is unrecognizable to a classifier or other machine learning algorithm. Hosseini and Poovendran [6] propose "Semantic Adversarial Examples," which allow modifications of the input image's hue and saturation. Hosseini et al. [7] also explore inverting images to cause misclassification. These latter two papers can be considered as special examples of functional threat models. In the first, each pixel's hue and saturation is shifted by the same amount; that is, each pixel is transformed by the function $f(h, s, v) = (h + \delta_h, s + \delta_s, v)$. In the second, each pixel is inverted, i.e. each pixel channel is transformed by the function $f(x_i) = 1 - x_i$. However, the authors do not propose a general framework for these types of attacks, as we do. Furthermore, the adversarial examples generated by these attacks are often not realistic and not imperceptible. For example, their crafted adversarial examples include green skies, purple fields of grass, and inverted street signs—unlike our proposed ReColorAdv attack, which results in imperceptible changes.

E Additional Images

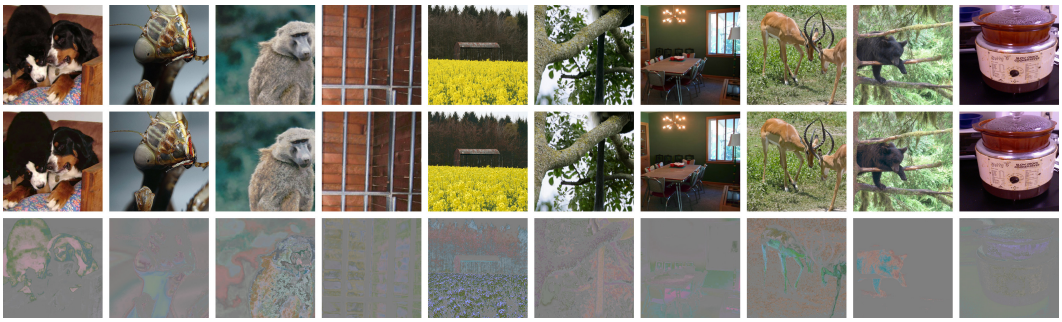


Figure 7: More adversarial examples like those in figure 2, generated by ReColorAdv against an Inception-v4 classifier on ImageNet. Top row: original images; middle row: adversarial examples; bottom row: magnified difference.

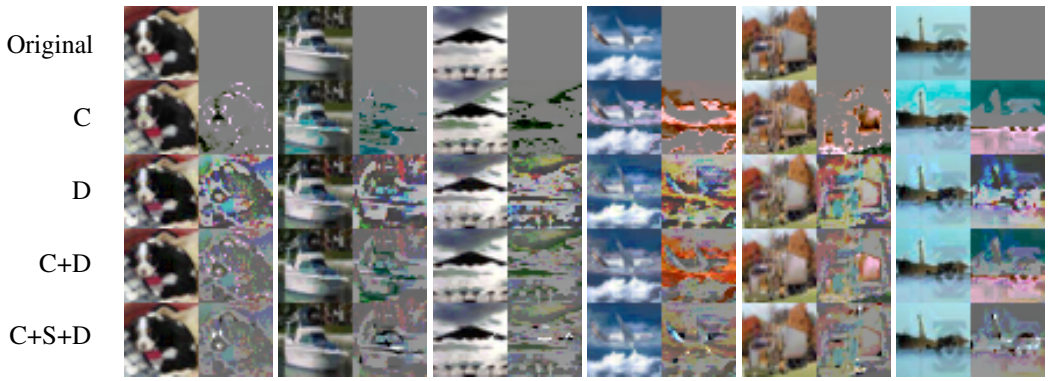


Figure 8: More adversarial examples like those in figure 5, generated with combinations of attacks against a CIFAR-10 WideResNet trained using TRADES. C is ReColorAdv, D is delta attack, and S is StAdv attack [23]. The difference from the original is shown to the right of each example. Combinations of attacks tend to produce less perceptible changes than the attacks do separately.

F Lipschitz Regularization

In addition to the regularizations defined in section 3.1, we can also enforce that the perturbation function $f(\cdot)$ in a functional threat model is Lipschitz for some suitably small κ:

$$\mathcal{F}_{\text{lips}} \triangleq \{f : \mathcal{X} \to \mathcal{X} \mid \forall x_1, x_2 \in \mathcal{X} \; \|f(x_1) - f(x_2)\| \leq \kappa \|x_1 - x_2\|\} \tag{5}$$

$\mathcal{F}_{\text{lips}}$ requires some smoothness in the perturbation function $f(\cdot)$, ensuring that similar features in the input are mapped to similar features in the adversarial example. However, one disadvantage of $\mathcal{F}_{\text{lips}}$ is that it includes constant functions $f(x) = c$, i.e. functions which map every feature to a single value, removing salient features from the input. Thus, we ultimately use $\mathcal{F}_{\text{smooth}}$ instead.