# Adversarial robustness in discontinuous spaces via alternating sampling & descent

Rahul Venkatesh
Stanford University
Stanford, CA, USA
rmvenkat@stanford.edu

Eric Wong*
University of Pennsylvania
Philadelphia, PA, USA
exwong@cis.upenn.edu

Zico Kolter*
Carnegie Mellon University and
Bosch Center for AI
Pittsburgh, PA, USA
zkolter@cs.cmu.edu

## Abstract

*Several works have shown that deep learning models are vulnerable to adversarial attacks where seemingly simple label-preserving changes to the input image lead to incorrect predictions. To combat this, gradient based adversarial training is generally employed as a standard defense mechanism. However, in cases where the loss landscape is discontinuous with respect to a given perturbation set, first order methods get stuck in local optima, and fail to defend against threat. This is often a problem for many physically realizable perturbation sets such as 2D affine transformations and 3D scene parameters. To work in such settings, we introduce a new optimization framework that alternates between global zeroth order sampling and local gradient updates to compute strong adversaries that can be used to harden the model against attack. Further, we design a powerful optimization algorithm using this framework, called Alternating Evolutionary Sampling and Descent (ASD), which combines an evolutionary search strategy (viz. covariance matrix adaptation) with gradient descent. We consider two settings with discontinuous/discrete and non-convex loss landscapes to evaluate ASD: a) 3D scene parameters and b) 2D patch attacks, and find that it achieves state-of-the-art results on adversarial robustness.*

## 1. Introduction

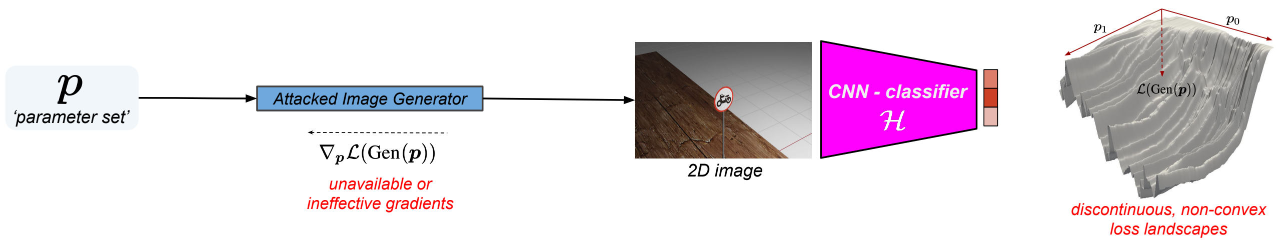
Deep learning models achieve state-of-the-art performance on various computer vision tasks, and have revolutionized multiple application areas [16, 3, 23]. However, several works have shown that these models are vulnerable to adversarial attacks where subtle modifications to the input image such as pixel perturbations [31], 2D [11] and 3D scene transformations [42, 2, 26] lead to incorrect predictions. This poses serious security concerns [15] for numerous vision systems which find ubiquitous use of deep learning (such as autonomous navigation [1], augmented/virtual reality [28], medical image analysis [27] etc.).

To harden deep learning systems against such threats, adversarial training [31] is a commonly used strategy, where robustness is achieved by augmenting the mini-batch with adversarial examples. For $l_p$ bounded pixel perturbation based attacks, gradient based adversarial training has been established as the gold standard over the past few years [31, 41, 38, 14]. The success of first order methods on this setting can be largely attributed to smooth loss landscapes which possess multiple critical points where the local and global minima are relatively close [8].

However, these loss surface properties do not hold in general for all perturbation sets. In particular, more natural, physically realizable settings such as 2D spatial transformations and 3D scene parameters often involve discontinuous, non-convex and rugged loss landscapes (Fig. 1a) where first order methods can easily get stuck in local optima [11]. Moreover, in many scenarios, gradients may be simply unavailable for a sub-set of parameters – for e.g. 2D patch attacks (see Fig. 1b) where the attack parameters are discrete variables and could be non-differentiable with respect to the loss function. Therefore, in the absence of conducive loss landscape properties, such as those in the $l_p$ pixel perturbation setting, purely gradient based adversarial training cannot be reliably used to procure robust models [11].

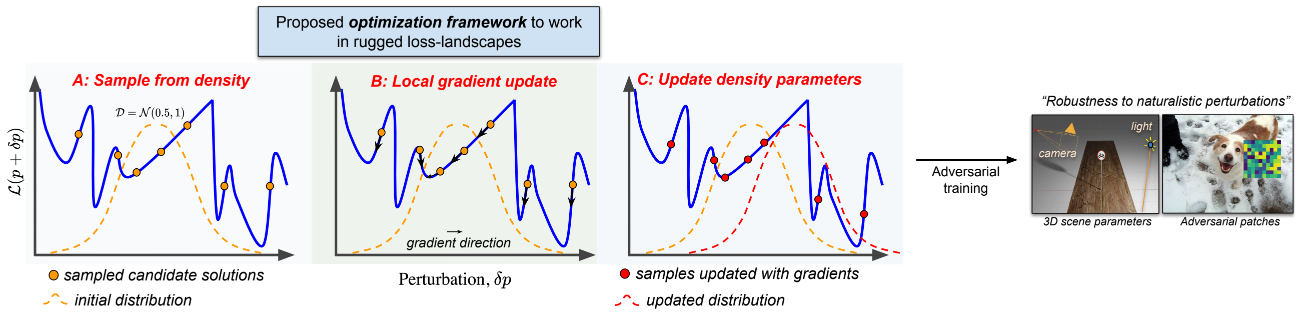
In this work, we introduce a novel optimization framework that addresses the need to design better optimization methods for working in the discontinuous loss landscape of naturalistic perturbations. Our framework is motivated by classic evolutionary-search based optimization algorithms that find the global optima of functions that are hard to optimize [9, 12, 13, 19, 36, 40], typically making no assumptions about the underlying loss surface. In a nutshell, these methods iteratively update a distribution defined on the feasible set using fitness scores of sampled candidate

* = Equal advising



$p$
'parameter set'
Attacked Image Generator
$\nabla_{\boldsymbol{p}}\mathcal{L}(\mathrm{Gen}(\boldsymbol{p}))$
unavailable or ineffective gradients
2D image
CNN - classifier $\mathcal{H}$
$p_1$
$p_0$
$\mathcal{L}(\mathrm{Gen}(\boldsymbol{p}))$
discontinuous, non-convex loss landscapes

(a) Adversarial image generation pipeline.



Proposed optimization framework to work in rugged loss-landscapes
A: Sample from density
$\mathcal{D} = \mathcal{N}(0.5, 1)$
B: Local gradient update
gradient direction
C: Update density parameters
$\mathcal{L}(p + \delta p)$
Perturbation, $\delta p$
sampled candidate solutions
initial distribution
samples updated with gradients
updated distribution
Adversarial training
"Robustness to naturalistic perturbations"
light
camera
3D scene parameters
Adversarial patches

(b) Outline of proposed optimization framework.

Figure 1: ***Top:*** Loss landscape of a deep neural network can be discontinuous and non-convex with respect to certain parameter sets that generate an image such as 3D scene parameters, 2D affine transformations, adversarial patch location, etc. ***Bottom:*** Sequence of steps in our proposed optimization algorithm (Alternating Evolutionary Sampling and Descent or *ASD*) which iteratively modifies the parameters of a probability density and accelerates search by capturing the global properties of the loss landscape. On the right, we list the utility of *ASD* in procuring models robust to naturalistic perturbations.

solutions. However, these gradient-free, function sampling based methods are still impractical in high dimensional settings due to exponential increase in search volume.

How can we accelerate the search process in discontinuous spaces? We hypothesize that while gradients might not get us to the global optima, the update directions are *locally* meaningful. They can be used to update the candidate solutions that are sampled from the search distribution to help accelerate the optimization in high dimensional settings.

Leveraging this observation, we introduce a new optimization framework (Sec. 3) that switches between *global* zeroth order sampling and *local* gradient updates to compute adversarial examples in discontinuous loss landscapes. Concretely, the framework consists of a sequence of three steps, a) sampling a probability density defined in the perturbation space, b) first-order sample updates and c) modification of the density function's parameters using updated samples (see Fig. 1b). Further, we perform adversarial training using this framework to procure robust models.

Under this framework, we design a powerful optimization algorithm for robust model training – Alternating Evolutionary Sampling and Descent (or *ASD*) which combines an evolutionary search strategy viz. covariance matrix adaptation (or CMA [4]) with projected gradient descent (PGD) (Sec. 4) and outperforms existing baselines (empirical evaluation in Sec. 6) on physically realizable perturbation sets such as a) 3D scene parameters viz. pose, lighting and materials b) patch attacks i.e. adversarial images crafted by overlaying a textured patch. For evaluating robustness to 3D scene parameter changes, we found an absence of a standard setting in the literature – most datasets are focused on evaluating attack methods [2, 42]. Therefore we collect a simple dataset of 3D traffic sign models to serve as a benchmark for evaluating both our model and methods developed in future. We provide more background on this in Sec. 2 and describe specific details of our new setting in Sec. 6.1. To summarize, our contributions are as follows:

- A novel framework for computing adversarial examples in discontinuous loss landscapes.
- A new benchmark dataset for evaluating robustness with respect to 3D scene parameters.
- A specific instantiation of the framework (*ASD*) that combines evolutionary search with PGD, achieving state-of-the art robustness results on 3D scene parameters as well as 2D patch attack settings.

## 2. Background/Related work

In this section, we briefly introduce various forms of adversarial attacks, defense mechanisms, and datasets currently used to evaluate robust models. Additionally, we also provide some background on existing optimization methods for working in discontinuous loss landscapes.

## 2.1. $l_p$ bounded adversarial attacks

A standard method for computing pixel-space adversarial examples for an image, $x$ with class label, $y$ is by projected gradient descent on the negative loss function [31],

$$\boldsymbol{x}^{(k+1)} = \Pi_{\boldsymbol{x}+\mathcal{S}}(x^{(k)} + \alpha(\nabla_{\boldsymbol{x}}\mathcal{L}(\boldsymbol{x}^{(k)}, y)) \qquad (1)$$

Here $\alpha$ is the learning rate, $y$ is the class label, and $\mathcal{S} \subseteq \mathbb{R}^d$ is the set of allowable perturbations. $\mathcal{L}$ refers to a loss function applied to predictions from a classifier, $\mathcal{H}$. $\Pi_{\boldsymbol{x}+\mathcal{S}}$ is a projection onto the feasible set (an $l_\infty$ ball in this case).

## 2.2. Physically realizable adversarial attacks

The $l_p$ bounded adversarial attacks do not contain changes that one would expect in the real world, making them less important from a practical standpoint. In recent years, the community has therefore researched more natural/physically realizable class of perturbations which occur in natural settings such as robotics, autonomous driving, etc. Robustness to such classes of perturbations forms the primary focus of our work – we discuss some of them below.

**2D Spatial Attacks.** In general, these refer to perturbations such as 2D spatial translations and rotations. Past work [11] has shown that standard first order techniques such as projected gradient descent [31] cannot be used to reliably find strong adversaries in this setting owing to the loss landscape being highly non-convex and discontinuous. Given the low dimensional parameter space, attacks can be crafted by selecting the best adversary among 10 randomly sampled transformations from the feasible set. However, this simple random search strategy is not guaranteed to work well in higher dimensional spaces such as 3D scene parameters owing to the *curse-of-dimensionality* [5]. Moreover, these 2D transformations do not sufficiently capture the full set of camera changes that might occur in the real world.

**3D scene parameters.** To match real-world conditions more closely, a number of methods have recently been proposed to craft adversarial attacks by perturbing 3D scene parameters such as camera position [2, 44], mesh geometry and light position [42]. However, most experimental setups consider unconstrained ranges for each parameter, and use unrealistic synthetic objects which results in largely unnatural adversarial examples [2]. Therefore, these settings are unsuitable for training and evaluation of robust models. In this work, we consider a simplified scenario of 3D traffic signs whose geometry is closer to the real world in comparison to generic object classes. Further, we constrain the scene parameters to match natural conditions of an autonomous vehicle, and employ a differentiable renderer [35] to simulate the real world image generation process. Since traffic sign recognition has clear practical applications in tasks like autonomous driving, this is a well motivated setting for evaluating adversarial robustness. Note that the loss landscape with respect to 3D scene parameters is often discontinuous and non-convex[1] – the main goal of our work is to design an adversarial optimization framework that can operate in such rugged landscapes to procure robust models.

**Patch Attacks.** As opposed to the $l_p$ setting, where pixels of the entire image are manipulated in a visually imperceptible manner (i.e within $l_p$ bounds), patch attacks [6, 22, 25, 30] craft adversarial images by inpainting a rectangular patch of texture onto an image. Note that, the patch need not be within a predefined $l_p$ bound, thus making it perceptibly different from the benign image. However, these attacks are physically realizable, as the optimized patched can be printed and potentially overlayed on a real 3D object to fool a recognition system. Note that it's relatively simple to use gradients to search for the optimal patch content that fools the network. However, finding the optimal patch location is hard given that 2D pixel locations are discrete and hence not differentiable w.r.t the loss. To alleviate this prior work [34] has used a method to jointly optimize both the patch content and location by employing a numerical gradient on the patch location in each step of the optimization process. However, this strategy is prone to get stuck in local optima given that the search space is inherently discrete. Our proposed optimization framework, on the other hand, finds better solutions by considering global statistics of the loss landscape to avoid falling in local optima.

**Methods for optimization in discontinuous spaces.** In many scenarios, gradient information may be either unavailable (e.g. patch location discussed above) or simply cannot be used to find global optima (discontinuous loss landscapes). Zeroth order optimization techniques are generally used to work in such settings. The simplest method is random search which of course does not work well in high dimensional settings. Evolutionary strategies [9, 36] compute better optima with a significantly lower computational footprint. Among these, covariance matrix adaptation (CMA-ES) [17] is widely regarded as the standard method for working in rugged, non-linear, non-convex and discontinuous spaces [18]. Put simply, the method uses evolutionary-search to iteratively adapt the covariance matrix of a distribution defined in the space of parameters that we desire to optimzie. We find that an instantiation of our optimization framework that uses CMA-ES in the sampling step yields state-of-the-art results (see Sec. 6).

**Datasets.** While there are well established benchmarking datasets for patch attacks [25, 6], little work has been

[1]This is primarily due to the fact that a small change in the parameter space (i.e. camera rotation/translation) leads to a large change in pixel space. e.g. out of plane object rotations exposing some part of the background leads to a sharp change in image-pixels

done to curate datasets for evaluating robustness w.r.t 3D scene parameters. We note that prior work in adversarial machine learning has investigated 3D datasets such as Shapenet [42, 2], KITTI [21] and PASCAL3D+ [42]. However, these are mostly limited to evaluating attacks and do not deal with robustness benchmarking. Therefore, in this work, we collect a simple dataset of 3D traffic signs to create a new benchmark for evaluating adversarial robustness to 3D scene parameters (see Sec. 6.1).

## 3. Framework for optimization

**Setup.** Given a parameter set, $\boldsymbol{p}$ that generates an image, $\texttt{Gen}(\boldsymbol{p})$, our goal is to find a perturbation, $\boldsymbol{\delta p}$ that maximizes the loss function ($\mathcal{L}(\texttt{Gen}(\boldsymbol{p} + \boldsymbol{\delta p}))$); where the loss landscape w.r.t $\boldsymbol{p}$ can be arbitrarily discontinuous and non-convex[2]. Here, $y_p$ refers to the class label, and the generation function $\texttt{Gen}$ can be specific to the chosen parameter set. For e.g. it can be a differentiable renderer if $\boldsymbol{p}$ defines a set of 3D scene parameters or it could be a simple in-painting transformation in case of a patch attack where $\boldsymbol{p}$ would denote the patch content and patch location.

**Background.** Our framework is inspired by classic black box methods which use evolutionary search for optimization in discontinuous and non-convex spaces [9, 36, 4]. These methods work well in relatively low dimensional settings, but by virtue of being gradient-free, they rely on function sampling to perform optimization – this does not scale well with increase in dimensionality. A more general scenario is when the parameter set consists of a sub-set of parameters for which gradients are reliable (e.g. patch content) and a mutually exclusive subset for which gradients are either unavailable (e.g. patch location) or are unreliable given the discontinuous loss landscape (e.g. 3D scene parameters). It is desirable to jointly optimize the full set of parameters, and use gradient-based updates when they are useful – standard black box gradient-free optimization methods do not account for this generic case. Therefore, we propose a novel optimization framework to address it.

**Motivation.** We begin with the observation that while gradients might not suffice to find the global optima for discontinuous loss landscapes, the update directions do make sense *locally*. Therefore, in the evolutionary search process, we simply perform local gradient updates to sampled candidate solutions to accelerate convergence. We describe the framework more formally in Alg. 1 and discuss it below.

**Optimization Steps.** Our framework broadly comprises of the following. First, we define a density $\mathcal{D}$ in the space of allowable perturbations (i.e. $\mathcal{S} = [-\boldsymbol{\epsilon}, \boldsymbol{\epsilon}]$) to keep track of our estimate of the loss landscape's underlying structure. We then iteratively update the parameters of the distribution over the course of the optimization process. Each iteration involves sampling a set of candidate solutions (**Step A**); followed by a gradient based update to the subset of parameters for which gradients are available (**Step B**), and a rank/score assignment to each sample using the loss function (**Step C**). These weighted samples are then used to update the parameters of the density with samples that have higher rank influencing the update more significantly (**Step D**). The process is repeated for a predefined number of steps, $n_{iter}$, and the final solution is the best sample over the course of the full optimization process (i.e. $\boldsymbol{\delta p}_{best}$ in **Step E** of Alg. 1).

**Algorithm 1:** Our optimization framework (Sec. 3)

```
require: Parameter set $\boldsymbol{p}$, a loss function, $\mathcal{L}(\boldsymbol{p})$,
range of allowable perturbations, $\mathcal{S}$, $n_{iters}$.
objective: Find the optimal perturbation $\boldsymbol{\delta p}$ that
  maximizes $\mathcal{L}(\boldsymbol{p} + \boldsymbol{\delta p})$
init: $\mathcal{L}_{best} = -inf$, $\boldsymbol{\delta p_{best}}$ = None
/* Define a multivariate probability density, $\mathcal{D}$ in
  the range of allowable perturbations, $\mathcal{S}$ */
for iter in $n_{iters}$ do
    A: Sample
    /* Sample a set of perturbations,
      $\mathcal{P} = \{\boldsymbol{\delta p_i} \sim \mathcal{D} | i = 1 : n_s\}$ */
    B: Sample updates using gradient
    /* Update each $\boldsymbol{\delta p} \in \mathcal{P}$ using gradients,
      $\nabla_{\boldsymbol{\delta p}}\mathcal{L}(\boldsymbol{p} + \boldsymbol{\delta p})$ (if available) */
    C: Sample scoring
    /* Assign an attack score/weight, $w_i$ to each
      $\boldsymbol{\delta p_i}$'s using the loss, $\mathcal{L}$ as a scoring function.
      $\mathcal{W} = \{\mathcal{L}(\boldsymbol{p} + \boldsymbol{\delta p_i}) \, | i = 1 : n_s\}$ */
    D: Probability density update
    /* Update the parameters of the density,
      $\mathcal{D}$ using $\mathcal{W}$ and $\mathcal{P}$*/
    E: Track best sample
    /* Best pertubation in this iter */
    $\boldsymbol{\delta p}^{iter}_{best} = \arg\max_{\boldsymbol{\delta p_i}} \mathcal{W}$
    /* Best loss in this iter */
    $\mathcal{L}^{iter}_{best} = \mathcal{L}(\boldsymbol{p} + \boldsymbol{\delta p}^{iter}_{best})$
    /* Update overall best */
    if $\mathcal{L}^{iter}_{best} > \mathcal{L}_{best}$ then
        $\mathcal{L}_{best} = \mathcal{L}^{iter}_{best}$
        $\boldsymbol{\delta p}_{best} = \boldsymbol{\delta p}^{iter}_{best}$
```

[2]notation: vectors indicated in bold

## 4. Alternating Evolutionary Sampling and Descent (*ASD*) – proposed attack method

Having setup our framework, we now ask: what is the best optimization algorithm that we can design using this framework? There are multiple options that we can choose from for each step – i.e. the distribution, sampling mechanism, gradient update procedure, etc. However, we propose a specific instantiation called *ASD*, which combines evolutionary search with PGD and is tuned to operate in the discontinuous space of adversarial perturbations. The underlying reasons for our design choices and the algorithm specifics are outlined below.

In the literature, we find covaraince matrix adaptation (i.e. CMA-ES) to be a strong algorithm that achieves good performance in discontinuous and non-convex loss landscapes. Studies have shown its dominant performance over other methods on multiple evaluation benchmarks [18]. Therefore, we use a similar strategy as CMA for defining the probability density (**Step A**) and updating it's parameters (**Step C, D**) in Alg. 1. For **Step B**, since we are concerned specifically with adversarial robustness settings, we employ projected gradient descent (or PGD) which is the gold standard technique for gradient-based adversarial training. We describe these steps more concretely below. [3]

Let $\boldsymbol{p}$ be the initial parameter set which we want to perturb with an optimal $\boldsymbol{\delta p}$ to maximize $\mathcal{L}(\boldsymbol{p}+\boldsymbol{\delta p})$. We first define a multivariate normal density i.e. $\mathcal{D} = \mathcal{N}(\boldsymbol{\mu}^{(0)}, \boldsymbol{\Sigma}^{(0)})$. Here, $\boldsymbol{\mu}$ is a vector defining the mean of the density and $\boldsymbol{\Sigma}$ is the covariance matrix which we initialize to be identity before starting the optimization. Next, using this density we sample a set of candidate solutions ($i = 1 : n_s$) for a generation $g$ (i.e **Step A** in Alg. 1),

$$\boldsymbol{\delta p_i}^{(g)} \sim \mathcal{N}(\boldsymbol{\mu}^{(g-1)}, \boldsymbol{C}^{(g-1)}) \tag{2}$$

Next, we improve these samples using gradients from the network (i.e only updating parameters that we can differentiated with respect to the loss function). This constitutes **Step B** in Alg. 1. In practice we repeat the following step for a set of $n_{pgd}$ steps, using a learning rate $\alpha_{pgd}$ as a gradient multiplier,

$$\boldsymbol{\delta p_i}^{(g)} \leftarrow \prod_{\mathcal{S}} (\boldsymbol{\delta p_i}^{(g)} + \alpha_{pgd} \cdot \nabla_{\boldsymbol{\delta p}} \mathcal{L}(\texttt{Gen}(\boldsymbol{p} + \boldsymbol{\delta p_i}^{(g)}))) \tag{3}$$

We then assign a score, $w_i$ to each updated sample using the loss as the scoring function. We use a variant of CMA where only the top $k$ samples in each generation are retained for updation of the density's parameters – in the eqn. below, $\chi$ denotes the $k$'th highest loss in $\mathcal{W}$ (**Step C**).

$$w_i = \begin{cases} \mathcal{L}(\boldsymbol{p} + \boldsymbol{\delta p}_i^{(g)}) & \mathcal{L}(\boldsymbol{p} + \boldsymbol{\delta p_i}^{(g)}) >= \chi \\ 0 & \text{otherwise} \end{cases}$$

Finally, the density parameters, which in this case are the mean and covariance are updated using the following equations (**Step D**). Note that we normalize the sample scores before updating the parameters viz. $\hat{w}_i = w_i / (\sum_{i=1}^{n_s} w_i)$.

$$\begin{aligned} \boldsymbol{\Sigma}^{(g)} &= \sum_{i=1}^{n_s} \hat{w}_i (\boldsymbol{\delta p_i}^{(g)} - \boldsymbol{\mu}^{(g-1)})(\boldsymbol{\delta p_i}^{(g)} - \boldsymbol{\mu}^{(g-1)})^T \\ \boldsymbol{\mu}^{(g)} &= \boldsymbol{\mu}^{(g-1)} + \alpha_{cma} \cdot \sum_{i=1}^{n_s} w_i (\boldsymbol{\delta p_i}^{(g)} - \boldsymbol{\mu}^{(g-1)}) \end{aligned} \tag{4}$$

To train robust models we optimize for strong adversarial examples using this framework, and use them for adversarial training.

## 5. Attack method variants

We describe different threat models constructed using the framework defined in Alg. 1 below.

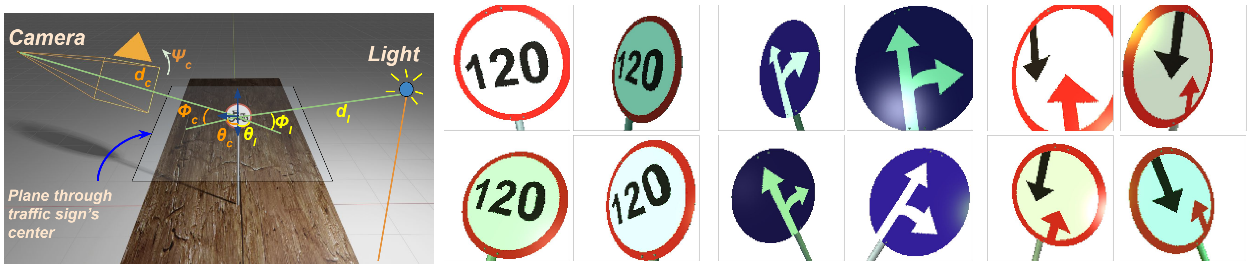
**RS (Random Search) + PGD:** In case of 2D pixel perturbation based attacks, this method has shown impressive results [31]. Therefore, to investigate whether it can be extended to more complex scenarios, we consider a variant where we use PGD in Step B. But, as opposed to the pixel perturbation setting, we find the best solution over many random starts to account for the discontinuous loss landscape – i.e $\mathcal{D} = \mathcal{U}(-\boldsymbol{\epsilon}, +\boldsymbol{\epsilon})$ is a uniform, immutable distribution in the range of allowable perturbations. Also, we set $n_{iter} = 1$, and do a multi-step PGD update (i.e. for $n_{pgd}$ steps) according to Eqn. 3 in Step B. Performance of RS+PGD in comparison to *ASD* will tell us whether the evolutionary search mechanism and density updates are helpful in finding better adversaries.

**Random search (RS):** This is our simplest baseline where we discard both the gradient based sample update and the improvement to the probability density. Similar to the previous setting we consider $\mathcal{D}$ to be uniform and use $n_{iter} = 1$. We simply omit step B (i.e. $n_{pgd} = 0$).

**CMA:** This is essentially a variant of *ASD* where gradient based updates to the sampled perturbations are omitted, reducing the framework to the original CMA-ES algorithm defined in [17]. Relative performance to this baseline will tell us whether the additional gradient update that we propose is adding any benefit.

**Grid search (GS):** Without sufficient samples, random search does not ensure good coverage all regions of the search-space. Therefore, we also consider a grid-search variant where we use a uniformly placed grid of values in

[3]Note that, the outline above is a simplified version of the full CMA-ES algorithm [17], which involves additional mechanisms for updating the learning rate $\alpha_{cma}$ on the fly, and other tricks for selecting the optimal sub-sample of a population for the parameter updates. For a complete description of the algorithm we refer the reader to [17].



(a) 3D scene with scene parameters.

(b) Exemplar rendered images from the dataset.

Figure 2: ***Left:*** An illustration of 3D scene parameters viz. camera and light. The classifier takes in as input a rendered image of the 3D traffic sign captured by a camera at an initial distance $d_c$ from the traffic sign. Additionally, the scene is illuminated by a directional light source. The relative rotations of the camera and light can be uniquely described using a combination of three angles: azimuth ($\theta$), elevation ($\phi$) and tilt/in-plane rotation ($\psi$). ***Right:*** Rendered images.

the feasible set (i.e. $[-\boldsymbol{\epsilon}, +\boldsymbol{\epsilon}]$) which guarantees coverage of all regions of the search-space.

**GS + PGD:** To further improve search-space coverage over GS, in this variant, we update each grid sample with gradients (i.e. using Eqn. 3 in Step B).

## 6. Experiments

In this section, we define the various settings we use for evaluating, followed by a description of the results.

### 6.1. Evaluation settings and metrics

To recap: the main goal of our work is to train models robust to naturalistic perturbations. Here, we describe two such settings and the metrics we use to evaluate robustness.

**3D scene parameters.** We use a traffic sign 3D mesh model pack (56 different traffic signs from an open source website[4]) to create 3D scenes consisting of a single traffic sign per scene. See Fig. 2a for a view of the setup. We use *Blender* [10] – a 3D modeling software to simulate various scene changes such as lighting effects (both color and direction), camera viewpoint, traffic sign material, etc. (See Fig. 2b). Our dataset is formally defined as,

$$\mathcal{X} = \{\texttt{Render}(\boldsymbol{p_j}, t) : \boldsymbol{p_j} \sim \mathcal{P}, t \in \mathcal{T}, 1 \leq j \leq N\} \quad (5)$$

Here, $\boldsymbol{p_j}$ refers to 3D scene parameter that is randomly sampled from a range of values ($\mathcal{P}$) defined for the dataset, and $\mathcal{T}$ is the set of all traffic signs. We use pytorch3D [35] to render the traffic signs and *Blender* to set up the scene.

**Patch attacks.** This setting involves crafting adversarial images by inpainting a patch of texture onto an image (See Sec. 2 for some background). Note that the content of the patch is differentiable with respect to the loss, however, the patch location isn't. This introduces discreteness in the loss landscape that makes it hard to use purely gradient based search for joint optimization of patch content location [34]. However, as discussed above, our framework is well equipped to work in such settings.

Following standard practice for patch attacks [34, 29, 7], we use CIFAR-10 as the training and evaluation dataset.

**Metrics.** We use the robust test accuracy (*RTA*) metric for evaluation, which measures the accuracy on test-set adversarial samples generated using a given attack method (e.g. *ASD*, CMA, Grid Search, etc.)

### 6.2. Robustness to 3D scene parameter changes

**Scenarios.** We consider two scenarios for training and evaluation – a) $|\boldsymbol{p}| = 7$, a low dimensional setting where the parameter set comprises of camera pose and light position, and b) $|\boldsymbol{p}| = 17$, a high dimensional setting consisting of light color (ambient, specular and diffuse components – see phong reflection model [32]) and traffic sign material (shininess) in addition to the parameters chosen in a). Additionally, we constrain the camera angles to match real world conditions and simulate the illumination of the sun by using a simple directional light source with the Hue component of the color (parameterized in HSV color space [37]) tuned to be in the yellow-green spectrum – more details and illustrations in supplementary.

**Data creation.** For both scenarios a.) and b.) described above, we render a dataset of 200 images per traffic sign using PyTorch3D, and set aside one scene configuration per traffic sign for creating the test set. The remaining data is split into a train set (90%), and a validation set (10%). Please refer to the supp. for the set of allowed scene configurations for each setting.

**Network architecture and training details.** The architecture of the classifier $\mathcal{H}$, comprises of the first 6 layers of VGG16 [39], followed by a 3x3 convolution and two fully connected layers of dimensions 256 and 128 with ReLU activations. We train with an Adam optimizer using a learning rate $1e-3$ and batch size 128, and set the perturbation strength, $\epsilon$ to 20% of the total range for each parameter. For

[4]turbosquid.com

| | RS [11] | RS+PGD | CMA | *ASD* | GS | GS+PGD |
|---|---|---|---|---|---|---|
| $\lvert\boldsymbol{p}\rvert$ | Attack Method *RTA* ↓ | | | | | |
| 7 | 0.60 | 0.51 | 0.53 | **0.46** | 0.35 | **0.27** |
| 17 | 0.38 | 0.23 | 0.32 | **0.20** | - | - |
| | Hyperparameters | | | | | |
| $\#fevals$ | 225 | | | | 16.3k | 15.3k |
| $n_s$ | 225 | 45 | 9 | 7 | $4^7$ | $3^7$ |
| $n_{pgd}$ | 0 | 4 | 0 | 3 | 0 | 6 |
| $n_{iter}$ | 1 | 1 | 25 | 8 | 1 | 1 |

(a) *RTA* of various attacks on an undefended model.

| | *RTA* ↑ | | | | |
|---|---|---|---|---|---|
| Model | $\lvert\boldsymbol{p}\rvert = 7$ | | $\lvert\boldsymbol{p}\rvert = 17$ | | |
| | $\#fevals$ | GS + PGD | $\#fevals$ | *ASD* | RS + PGD |
| TRADES [43] | | 0.28 | | 0.21 | 0.25 |
| LL [33] | | 0.30 | | 0.22 | 0.26 |
| RS [11] | | 0.73 | | 0.31 | 0.52 |
| Data aug. | | 0.45 | | 0.20 | 0.35 |
| CMA | 15.3k | **0.89** | 50k | 0.38 | 0.64 |
| RS+PGD | | 0.80 | | 0.55 | 0.73 |
| *ASD* | | **0.89** | | **0.63** | **0.73** |

(b) Adversarial training using different attack models.

Table 1: Results on the 3D scene parameter setting. ***Left:*** @ 225 $\#fevals$ our proposed method, *ASD* is the best performing threat model on both low ($|\boldsymbol{p}| = 7$) and high dim. ($|\boldsymbol{p}| = 7$) settings. Here, we also evaluate GS+PGD @ $15.3k$ fevals to compute the upper bound on attack efficacy. Note that while GS+PGD serves as a good evaluation strategy during testing, the high $\#feval$ count makes it infeasible during training. ***Right:*** Adversarial training using *ASD* outperforms other methods.

the variants which use gradients as part of the optimization procedure, we set $\alpha_{pgd} = \epsilon/n_{pgd}$ (See Eqn. 3). Note that we use pytorch3D's differentiable renderer to compute the gradient of the loss w.r.t 3D scene parameters.

**Evaluating various attacks on an undefended model.** To validate whether our proposed threat model (i.e. *ASD*) is stronger than other baselines we first evaluate attack efficacy on an undefended model (i.e. a model that hasn't been hardened with adversarial training).

On the low dim. (i.e. $|\boldsymbol{p}| = 7$) we find GS+PGD to be stronger than GS for a similar number of function evaluations ($\#fevals$). This clearly suggests that gradients are locally meaningful in parts of the search space. [5]

While GS+PGD is our strongest test time attack, it is too expensive to use in training. Therefore, we find the best attack method at a more feasible $\#fevals$ count for adv. training, by comparing different attack methods (Sec. 3) with a budget of 225 $\#fevals$ (results in Table. 1a).

Our proposed attack method (i.e *ASD*) outperforms the other attacks on both the $|\boldsymbol{p}| = 7$ and $|\boldsymbol{p}| = 17$ scenarios. Additionally, we find that pure CMA performs marginally worse than RS+PGD suggesting that gradients might be useful in this setting. Nevertheless, the evolutionary sampling process does add value given that CMA performs better that simple random search (RS).

We also find the undefended model to be more vulnerable to attack in the high dim. setting (even though it has high benign test-set accuracy 95%). For e.g., given the same number of $\#fevals$, *ASD* has a much higher threat efficacy on the high dim. setting (0.20 *RTA* vs 0.46 *RTA*) – this is to be expected since the added degrees of freedom make it easier for an attack method to find adversaries.

Please refer to the supplementary for comparisons to some alternative black box optimization methods.

[5]Note: grid-search is infeasible in the high dim. (i.e. $|p| = 17$) setting, so we omit those results here

**Adversarial Training.** Having established *ASD* as the best attack, we aim to analyze it's effectiveness in procuring robust models (via adversarial training), by comparing it to other variants we evaluated in Table. 1a.

We initialize with a model trained on our traffic sign dataset (benign test accuracy of 95%), and train till convergence with adversaries generated using a particular attack method (e.g. *ASD*/CMA, etc.). For each attack, we use the hyperparameters considered for evaluating *RTA* in Table. 1a. As standard practice, we verify that benign test accuracy is unaffected (93%) at convergence.

Additionally, we also include a data augmentation (Data aug.) baseline where instead of adversarial images we simply train with images rendered on-the-fly using scene parameters randomly sampled within the dataset's limits – akin to having infinite data in the real world. Performance on this baseline will help us know whether robustness can be achieved just by training on a large amount of data.

On the low dim. setting, we find that both CMA and *ASD* perform on par (0.89 *RTA*) and better than the rest of the models – even though we found *ASD* to be a stronger threat model (Table. 1a). The second best model here is RS+PGD (0.80 *RTA*) which performs better than simple random search (0.73 *RTA*). Note that, we use GS+PGD as our attack model during test time.

However, on the high dim. setting (i.e. $|\boldsymbol{p}| = 17$) *ASD* gets an *RTA* of 0.63, outperforming CMA (*RTA* 0.38) – suggesting that a purely evolutionary sampling based attack does not function as well in high dim. settings due to exponential increase in search volume [5]. Interestingly, even RS+PGD (*RTA* 0.55), which doesn't do any evolutionary sampling outperforms CMA suggesting the benefit of using gradients. Nevertheless, we do find in both settings that CMA outperforms random search (RS) – which speaks to the benefit of evolutionary-search on this setting. [6]

[6]Here we use *ASD* as a test-time attack, given 17 dim. GS is infeasible

|  | LO [34] | CMA | RS+PGD | *ASD* |
|---|---|---|---|---|
| *RTA* ↓ | 0.38 | 0.52 | 0.28 | **0.22** |
| Hyperparameters |  |  |  |  |
| $\#fevals$ | 50 |  |  |  |
| $n_s$ | 5 | 8 | 5 | 7 |
| $n_{pgd}$ | 1 | 0 | 10 | 10 |
| $n_{iter}$ | 10 | 6 | 1 | 3 |

(a) *RTA* on an undefended model.

| Model | *ASD RTA* ↑ |
|---|---|
| LO [34] | 0.30 |
| CMA | 0.05 |
| PGD | 0.56 |
| ASD | **0.60** |

(b) Adversarial training.

Table 2: Patch attack results on CIFAR-10. Attack methods here jointly optimize both patch content and patch location. *ASD* achieves *SOTA* attack efficacy (***Left***) and is the best optimization method for robust training (***Right***).

Also, it is interesting to note that simple Data aug. is quite vulnerable to attack and has a very poor *RTA* of 0.45 which further deteriorates in the high dim. setting (*RTA* 0.20). This suggest that simply training on more data samples does not lead to robustness – even if one were to hypothetically collect infinite amount of data to train a classification system, it would still be seriously vulnerable.

We also evaluate against TRADES [43] – a robust training method that is marginally better than PGD, and Local Linearization [33] – a gradient free defense method. As these methods are not guaranteed to work in discontinuous loss landscapes, *ASD* significantly outperforms them.

## 6.3. Robustness to patch attacks

To demonstrate *ASD*'s potential general utility in other discontinuous search settings beyond the 3D scene parameter framework presented above, we consider evaluation on a second setting – patch attacks.

**Attack method specifics.** We compare *ASD* against RS+PGD, CMA and location optimized patches (LO) – a prior method that jointly optimizes patch content and location [34]. For *ASD*, we only search for the patch location using CMA, and use PGD for patch content optimization. Concretely, for a given population of size $n_s$, containing patch locations, $\{(x_i, y_i), i = 1 : n_s\}$, we first optimize for the patch content, $\mathcal{C} \in \mathbb{R}^{H^2}$ by fixing the patch at $(x_0, y_0)$. We use the loss at the end of the patch content optimization as the sample score for $(x_0, y_0)$. Further, to compute the scores for the other locations, we simply move the optimized patch, $\mathcal{C}$, to the other locations specified in the population. In this manner, the total number of function evaluations can be computed as $(n_{pgd} + n_s - 1) \times n_{iter}$[7]. For CMA we optimize both the patch location $\in \mathbb{R}^2$, and content $\in \mathbb{R}^{H^2}$ together using a $H^2 + 2$ dim. space for the evolutionary search process.

***ASD* is a *SOTA* patch attack model.** We first evaluate *ASD*'s effectiveness in attacking an undefended classifier trained on CIFAR-10. We find that *ASD* (*RTA* 0.22) has superior performance in comparison to other methods (Table. 2a), with RS+PGD performing second best (*RTA* 0.28). Here, CMA performs significantly worse, given that it doesn't leverage useful gradient information to optimize for patch content. On the other hand, LO (*RTA* 0.38) performs better than CMA but not as good as RS+PGD.[8]. Note that, we use a patch size of 4x4 since an 8x8 patch can trivially attack an undefended model with an *RTA* of ∼0.0 [34].

**Adversarial training and architecture details.** Following a similar experimental procedure for adversarial training as LO, we train on the 50k training images of CIFAR-10, and evaluate on a held out test set of 1k images. We use a ResNet-20 [20] architecture, and train with the Adam [24] optimizer using a learning rate of $1e-4$. To maintain performance on clean samples, we construct mini batches with 50% benign images and 50% adversarial images. For all attack models we cap the $\#fevals$ budget at 25, and use a patch size of $8 \times 8$. Additionally, for those methods which use gradients, we train with $n_{pgd} = 5$, and $\alpha_{pgd} = 0.1$.

***ASD* yields the most robust defense to patch attacks.** We perform adversarial patch training using different attack methods (Table. 2b) and compute *RTA* using our most powerful attack, i.e *ASD*, that uses a population size of $n_s = 14$, generation length $n_{iter} = 13$, and a strong multi-step PGD with $n_{pgd} = 100$ and $\alpha_{pgd} = 0.05$.

*ASD* outperforms all other methods, achieving *SOTA* performance. Interestingly, we discover that adversarial patch training using CMA performs very poorly (0.05 *RTA*) and cannot reliably find strong adversaries to harden the model. This suggests that using gradient information while optimizing for patch content is critical. However, as our method leverages both gradient information and evolutionary sampling, it proves to be the most effective.

## 7. Conclusion

In this work, we introduced a novel attack method (*ASD*) that combines evolutionary sampling with first order methods for effective optimization in discontinuous and non-convex loss landscapes. To demonstrate its power, we successfully used *ASD* in adversarial training pipelines to procure models that are robust to physically realizable attacks such as adversarial patches and 3D scene parameter perturbations. Further, we introduced a new benchmark to test robustness to 3D scene parameter changes which we believe will help advance future research in this direction. Another exciting avenue is the study of the applications of this optimization framework in problems involving parameter search in discrete loss landscapes – such as neural architecture search, meta learning hyperparameters, etc.

[7] $(6 + 10 - 1) \times 3 = 48\ \#fevals$ in Table. 2a, *ASD* column

[8] Note that PGD with random starts being superior to LO (for the same $\#fevals$) was also the finding reported in the author's paper [34]. We use the code provided by the authors for the evaluation in our paper (Table. 2), and our results reconfirm this finding.

## References

[1] Mohammed Al-Qizwini, Iman Barjasteh, Hothaifa Al-Qassab, and Hayder Radha. Deep learning algorithm for autonomous driving using googlenet. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 89–96. IEEE, 2017.

[2] Michael A Alcorn, Qi Li, Zhitao Gong, Chengfei Wang, Long Mai, Wei-Shinn Ku, and Anh Nguyen. Strike (with) a pose: Neural networks are easily fooled by strange poses of familiar objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4845–4854, 2019.

[3] Hassan Abu Alhaija, Siva Karthik Mustikovela, Lars Mescheder, Andreas Geiger, and Carsten Rother. Augmented reality meets deep learning for car instance segmentation in urban scenes. In *British machine vision conference*, volume 1, page 2, 2017.

[4] Anne Auger and Nikolaus Hansen. Tutorial cma-es: evolution strategies and covariance matrix adaptation. In *Proceedings of the 14th annual conference companion on Genetic and evolutionary computation*, pages 827–848, 2012.

[5] Stefan Berchtold, Christian Böhm, and Hans-Peter Kriegal. The pyramid-technique: Towards breaking the curse of dimensionality. In *Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, pages 142–153, 1998.

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

[7] Ping-yeh Chiang, Renkun Ni, Ahmed Abdelkader, Chen Zhu, Christoph Studer, and Tom Goldstein. Certified defenses for adversarial patches. *arXiv preprint arXiv:2003.06693*, 2020.

[8] Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. In *Artificial intelligence and statistics*, pages 192–204. PMLR, 2015.

[9] Krzysztof Choromanski, Mark Rowland, Vikas Sindhwani, Richard Turner, and Adrian Weller. Structured evolution with compact architectures for scalable policy optimization. In *International Conference on Machine Learning*, pages 970–978. PMLR, 2018.

[10] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018.

[11] Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. Exploring the landscape of spatial robustness. In *International Conference on Machine Learning*, pages 1802–1811, 2019.

[12] David B Fogel. An introduction to simulated evolutionary optimization. *IEEE transactions on neural networks*, 5(1):3–14, 1994.

[13] Carlos M Fonseca and Peter J Fleming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary computation*, 3(1):1–16, 1995.

[14] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016.

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

[16] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, 37(3):362–386, 2020.

[17] Nikolaus Hansen. The cma evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*, 2016.

[18] Nikolaus Hansen, Anne Auger, Raymond Ros, Steffen Finck, and Petr Pošík. Comparing results of 31 algorithms from the black-box optimization benchmarking bbob-2009. In *Proceedings of the 12th annual conference companion on Genetic and evolutionary computation*, pages 1689–1696, 2010.

[19] Nikolaus Hansen, Sibylle D Müller, and Petros Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evolutionary computation*, 11(1):1–18, 2003.

[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] Lakshya Jain, Wilson Wu, Steven Chen, Uyeong Jang, Varun Chandrasekaran, Sanjit Seshia, and Somesh Jha. Generating semantic adversarial examples with differentiable rendering. *arXiv preprint arXiv:1910.00727*, 2019.

[22] Danny Karmon, Daniel Zoran, and Yoav Goldberg. Lavan: Localized and visible adversarial noise. In *International Conference on Machine Learning*, pages 2507–2515. PMLR, 2018.

[23] Justin Ker, Lipo Wang, Jai Rao, and Tchoyoson Lim. Deep learning applications in medical image analysis. *Ieee Access*, 6:9375–9389, 2017.

[24] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[25] Mark Lee and Zico Kolter. On physical adversarial patches for object detection. *arXiv preprint arXiv:1906.11897*, 2019.

[26] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. Differentiable monte carlo ray tracing through edge sampling. *ACM Transactions on Graphics (TOG)*, 37(6):1–11, 2018.

[27] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen Awm Van Der Laak, Bram Van Ginneken, and Clara I Sánchez. A survey on deep learning in medical image analysis. *Medical image analysis*, 42:60–88, 2017.

[28] Chen Liu, Kihwan Kim, Jinwei Gu, Yasutaka Furukawa, and Jan Kautz. Planercnn: 3d plane detection and reconstruction from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4450–4459, 2019.

[29] Xin Liu, Huanrui Yang, Ziwei Liu, Linghao Song, Hai Li, and Yiran Chen. Dpatch: An adversarial patch attack on object detectors. *arXiv preprint arXiv:1806.02299*, 2018.

[30] Xin Liu, Huanrui Yang, Linghao Song, Hai Li, and Yiran Chen. Dpatch: Attacking object detectors with adversarial patches. *CoRR, abs/1806.02299*, 2, 2018.

[31] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[32] Bui Tuong Phong. Illumination for computer generated pictures. *Communications of the ACM*, 18(6):311–317, 1975.

[33] Chongli Qin, James Martens, Sven Gowal, Dilip Krishnan, Krishnamurthy Dvijotham, Alhussein Fawzi, Soham De, Robert Stanforth, and Pushmeet Kohli. Adversarial robustness through local linearization. *Advances in Neural Information Processing Systems*, 32, 2019.

[34] Sukrut Rao, David Stutz, and Bernt Schiele. Adversarial training against location-optimized adversarial patches. In *European Conference on Computer Vision*, pages 429–448. Springer, 2020.

[35] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv:2007.08501*, 2020.

[36] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

[37] Michael W Schwarz, William B Cowan, and John C Beatty. An experimental comparison of rgb, yiq, lab, hsv, and opponent color models. *ACM Transactions on Graphics (TOG)*, 6(2):123–158, 1987.

[38] Ali Shafahi, Mahyar Najibi, Zheng Xu, John Dickerson, Larry S Davis, and Tom Goldstein. Universal adversarial training. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5636–5643, 2020.

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

[40] Keigo Watanabe and MMA Hashem. Evolutionary optimization of constrained problems. In *Evolutionary computations*, pages 53–64. Springer, 2004.

[41] Eric Wong, Leslie Rice, and J Zico Kolter. Fast is better than free: Revisiting adversarial training. *arXiv preprint arXiv:2001.03994*, 2020.

[42] Chaowei Xiao, Dawei Yang, Bo Li, Jia Deng, and Mingyan Liu. Meshadv: Adversarial meshes for visual recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6898–6907, 2019.

[43] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In *International conference on machine learning*, pages 7472–7482. PMLR, 2019.

[44] Yue Zhao, Yuwei Wu, Caihua Chen, and Andrew Lim. On isometry robustness of deep 3d point cloud models under adversarial attacks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1201–1210, 2020.