Adversarial Attacks Beyond the Image Space

Xiaohui Zeng[1], Chenxi Liu[2](✉), Yu-Siang Wang[3], Weichao Qiu[2],
Lingxi Xie[2,4], Yu-Wing Tai[5], Chi-Keung Tang[6], Alan L. Yuille[2]
[1]University of Toronto [2]The Johns Hopkins University [3]National Taiwan University
[4]Huawei Noah's Ark Lab [5]Tencent YouTu [6]Hong Kong University of Science and Technology
xiaohui@cs.toronto.edu cxliu@jhu.edu b03202047@ntu.edu.tw
{qiuwch, 198808xc, yuwing, alan.l.yuille}@gmail.com cktang@cs.ust.hk

Abstract

Generating adversarial examples is an intriguing problem and an important way of understanding the working mechanism of deep neural networks. Most existing approaches generated perturbations in the image space, i.e., each pixel can be modified independently. However, in this paper we pay special attention to the subset of adversarial examples that correspond to meaningful changes in 3D physical properties (like rotation and translation, illumination condition, etc.). These adversaries arguably pose a more serious concern, as they demonstrate the possibility of causing neural network failure by easy perturbations of real-world 3D objects and scenes.

In the contexts of object classification and visual question answering, we augment state-of-the-art deep neural networks that receive 2D input images with a rendering module (either differentiable or not) in front, so that a 3D scene (in the physical space) is rendered into a 2D image (in the image space), and then mapped to a prediction (in the output space). The adversarial perturbations can now go beyond the image space, and have clear meanings in the 3D physical world. Though image-space adversaries can be interpreted as per-pixel albedo change, we verify that they cannot be well explained along these physically meaningful dimensions, which often have a non-local effect. But it is still possible to successfully attack beyond the image space on the physical space, though this is more difficult than image-space attacks, reflected in lower success rates and heavier perturbations required.

1. Introduction

Recent years have witnessed a rapid development in the area of deep learning, in which deep neural networks have been applied to a wide range of computer vision tasks, such as image classification [17][13], object detection [32], semantic segmentation [35][8], visual question answering [2][14], *etc*. Despite the great success of deep learning, there still lacks an effective method to understand the working mechanism of deep neural networks. An interesting effort is to generate so-called *adversarial perturbations*. They are visually imperceptible noise [12] which, after being added to an input image, changes the prediction results completely, sometimes ridiculously. These examples can be constructed in a wide range of vision problems, including image classification [26], object detection and semantic segmentation [39]. Researchers believed that the existence of adversaries implies unknown properties in the feature space [37].

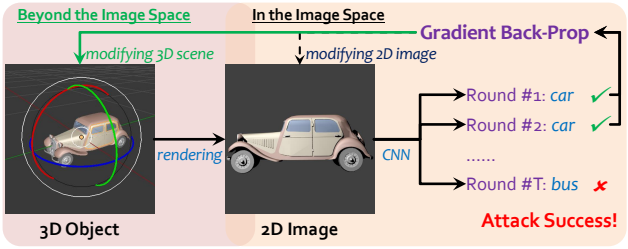


Figure 1. The vast majority of existing works on adversarial attacks focus on modifying pixel values in 2D images to cause wrong CNN predictions. In our work, we consider the more complete vision pipeline, where 2D images are in fact projections of the underlying 3D scene. This suggests that adversarial attacks can go *beyond* the image space, and directly change physically meaningful properties that define the 3D scene. We suspect that these adversarial examples are more physically plausible and thus pose more serious security concerns.

Our work is motivated by the fact that conventional 2D adversaries were often generated by modifying each image pixel individually. We instead consider perturbations of the 3D scene that are often non-local and correspond to physical properties of the object. We notice that previous work found adversarial examples "in the physical world" by taking photos on the printed perturbed images [18]. But

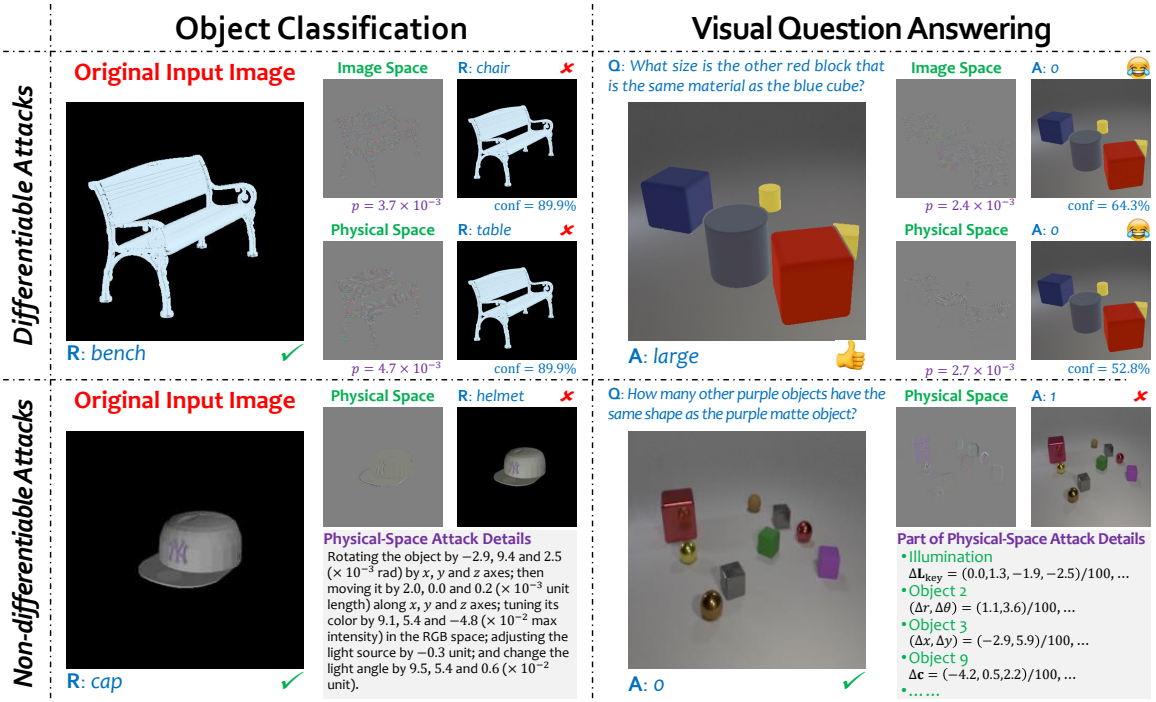


Figure 2. Adversarial examples for 3D object classification and visual question answering, under either a differentiable or a non-differentiable renderer. The top row shows that while it is of course possible to produce adversarial examples by attacking the image space, it is also possible to successfully attack on the physical space by changing factors such as surface normal, material, lighting condition (see Section 3.1). The bottom row demonstrates the same using a more realistic non-differentiable renderer, with descriptions of how to carry out the attack. p and conf are the perceptibility (see Section 3.2) and the confidence (post-softmax output) on the predicted class.

our work is different and more essential, as we are attacking the intrinsic parameters that define the 3D scene/object, whereas [18] is still limited to attacking 2D image pixels. For this respect, we plug 3D rendering as a network module into the state-of-the-art neural networks for object classification and visual question answering. In this way, we build a mapping function from the *physical space* (a set of physical parameters, including surface normals, illumination and material), via the *image space* (a rendered 2D image), to the *output space* (the object class or the answer to a question). See Figure 1 which illustrates this framework.

The per-pixel image-space attack can be explained in terms of per-pixel changes of albedo, but it is highly unlikely that these individual perturbations happen to correspond to, e.g., a simple rotation of the object in 3D. Using our pipeline with rendering, we indeed found it almost impossible to approximate the 2D image adversaries using the 3D physically meaningful perturbations. At the same time, this suggests a natural mechanism for defending adversaries – finding an approximate solution in the physical space and re-rendering will make most image-space adversaries fail. This analysis-by-synthesis process offers new direction in dealing with adversarial examples and occlusion cases.

Our paper mainly tries to answer the following question: **can neural networks still be fooled if we do not perturb 2D image pixels, but instead perturb 3D physical properties?** This is about directly generating perturbations in the physical space (*i.e.*, modifying basic physical parameters) that cause the neural network predictions to fail. Specifically, we compute the difference between the current output and the desired output, and use gradient descent to update parameters in the physical space (*i.e.*, *beyond* the image space, which contains physical parameters such as surface normals and illumination conditions). This attack is implemented by either iterative Fast Gradient Sign Method (FGSM) [12] (for differentiable rendering) or the Zeroth-Order Optimization approach [9] (for non-differentiable rendering). We constrain the change in the image intensities to guarantee the perturbations to be visually imperceptible. Our major finding is that attacking the physical space is more difficult than attacking the image space. Although it is possible to find adversaries in this way (see Figure 2 for a few of these examples), the success rate is lower and the perceptibility of perturbations becomes much larger than required in the image space. This is expected, as the rendering process couples changes in pixel values, *i.e.*, modifying one physical parameter (*e.g.*, illumination) may cause many pixels to be changed at the same time.

2. Related Work

Deep learning is the state-of-the-art machine learning technique to learn visual representations from labeled data. Yet despite the success of deep learning, it remains challenging to explain what is learned by these complicated models. One of the most interesting evidence is *adversaries* [12]: small noise that is (i) imperceptible to humans, and (ii) able to cause deep neural networks make wrong predictions after being added to the input image. Early studies were mainly focused on image classification [26][25]. But soon, researchers were able to attack deep networks for detection and segmentation [39], and also visual question answering [40]. Efforts were also made in finding universal perturbations which can transfer across images [24], as well as adversarial examples in the physical world produced by taking photos on the printed perturbed images [18].

Attacking a known network (both network architecture and weights are given, *a.k.a*, a white box) started with setting a goal. There were generally two types of goals. The first one (a non-targeted attack) aimed at reducing the probability of the true class [26], and the second one (a targeted attack) defined a specific class that the network should predict [21]. After that, the error between the current and the target predictions was computed, and gradients back-propagated to the image layer. This idea was developed into a set of algorithms, including the Steepest Gradient Descent Method (SGDM) [25] and the Fast Gradient Sign Method (FGSM) [12]. The difference lies in that SGDM computed accurate gradients, while FGSM merely kept the sign in every dimension. The iterative version of these two algorithms were also studied [18]. In comparison, attacking an unknown network (*a.k.a.*, a black box) is much more challenging [21], and an effective way is to sum up perturbations from a set of white-box attacks [39]. In opposite, there exist efforts in protecting deep networks from adversarial attacks [29][19][38]. People also designed algorithms to hack these defenders [6] as well as to detect whether adversarial attacks are present [23]. This competition has boosted both attackers and defenders to a higher level [3].

More recently, there is increasing interest in adversarial attacks other than modifying pixel values. [18] showed that the adversarial effect still exists if we print the digitally-perturbed 2D image on paper. [10][30] fooled vision systems by rotating the 2D image or changing its brightness. [11][4] created real-world 3D objects, either by 3D printing or applying stickers, that consistently cause perception failure. However, these adversaries have high perceptibility and must involve sophisticated change in object appearance. To find adversarial examples in 3D, we use a renderer, either differentiable or non-differentiable, to map a 3D scene to a 2D image and then to the output. In this way it is possible, though challenging, to generate interpretable and physically plausible adversarial perturbations in the 3D scene.

3. Approach

3.1. From Physical Parameters to Prediction

As the basis of this work, we extend deep neural networks to receive the physical parameters of a 3D scene, render them into a 2D image, and output prediction, *e.g.*, the class of an object, or the answer to a visual question. Note that our research involves 3D to 2D rendering as part of the pipeline, which stands out from previous work which either worked on rendered 2D images [36][15], or directly processed 3D data without rendering them into 2D images [31][34].

We denote the physical space, image space and output space by $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$, respectively. Given a 3D scene $\mathbf{X} \in \mathcal{X}$, the first step is to render it into a 2D image $\mathbf{Y} \in \mathcal{Y}$, and the second step is to predict the output of $\mathbf{Y}$, denoted by $\mathbf{Z} \in \mathcal{Z}$. The overall framework is denoted by $\mathbf{Z} = \mathbf{f}[\mathbf{r}(\mathbf{X}); \boldsymbol{\theta}]$, where $\mathbf{r}(\cdot)$ is the renderer, $\mathbf{f}[\cdot; \boldsymbol{\theta}]$ is the target deep network with $\boldsymbol{\theta}$ being parameters.

There are different models for the **3D rendering function** $\mathbf{r}(\cdot)$. One of them is *differentiable* [20], which considers three sets of physical parameters, *i.e.*, surface normals $\mathbf{N}$, illumination $\mathbf{L}$, and material $\mathbf{m}$[1]. By giving these parameters, we assume that the camera geometries, *e.g.*, position, rotation, field-of-view, *etc.*, are known beforehand and will remain unchanged in each case. The rendering module is denoted by $\mathbf{Y} = \mathbf{r}(\mathbf{N}, \mathbf{L}, \mathbf{m})$. In practice, the rendering process is implemented as a network layer, which is differentiable to input parameters $\mathbf{N}$, $\mathbf{L}$ and $\mathbf{m}$. Another option is to use a *non-differentiable* renderer which often provides much higher quality [5][22]. In practice we choose an open-source software named Blender [5]. Not assuming differentiability makes it possible to work on a wider range of parameters, such as color ($\mathbf{C}$), translation ($\mathbf{T}$), rotation ($\mathbf{R}$) and lighting ($\mathbf{L}$) considered in this work, in which translation and rotation cannot be implemented by a differentiable renderer[2].

[1] In this model, $\mathbf{N}$ is a 2-channel image of spatial size $W_\mathrm{N} \times H_\mathrm{N}$, where each pixel is encoded by the azimuth and polar angles of the normal vector at this position; $\mathbf{L}$ is defined by an HDR environment map of dimension $W_\mathrm{L} \times H_\mathrm{L}$, with each pixel storing the intensity of the light coming from this direction (a spherical coordinate system is used); and $\mathbf{m}$ impacts image rendering with a set of bidirectional reflectance distribution functions (BRDFs) which describe the point-wise light reflection for both diffuse and specular surfaces [27]. The material parameters used in this paper come from the directional statistics BRDF model [28], which represents a BRDF as a combination of D_m distributions with P_m parameters in each. Mathematically, we have $\mathbf{N} \in \mathbb{R}^{W_\mathrm{N} \times H_\mathrm{N} \times 2}$, $\mathbf{L} \in \mathbb{R}^{W_\mathrm{L} \times H_\mathrm{L}}$ and $\mathbf{m} \in \mathbb{R}^{D_\mathrm{m} \times P_\mathrm{m}}$.

[2] For 3D object classification, we follow [36] to configure the 3D scene. $\mathbf{L}$ is a 5-dimensional vector, where the first two dimensions indicate the magnitudes of the environment and point light sources, and the last three the position of the point light source. $\mathbf{C}$, $\mathbf{T}$, $\mathbf{R}$ are all 3-dimensional properties of the single object. For 3D visual question answering we follow [14]. $\mathbf{L}$ is a 12-dimensional vector that represents the energy and position of 3 point light sources. For every object in the scene, $\mathbf{C}$ is 3-dimensional, corresponding to RGB; $\mathbf{T}$ is 2-dimensional which is the

We consider two popular **object understanding tasks**, namely, 3D object classification and 3D visual question answering, both of which are straightforward based on the rendered 2D images. Object classification is built upon standard deep networks, and visual question answering, when both the input image $\mathbf{Y}$ and question $\mathbf{q}$ are given, is also a variant of image classification (the goal is to choose the correct answer from a pre-defined set of choices).

In the adversary generation stage, given pre-trained networks, the goal is to attack a model $\mathbf{Z} = \mathbf{f}[\mathbf{r}(\mathbf{X}); \boldsymbol{\theta}] = \mathbf{f} \circ \mathbf{r}(\mathbf{X}; \boldsymbol{\theta})$. For object classification, $\boldsymbol{\theta}$ is fixed network weights, denoted by $\boldsymbol{\theta}^{\mathrm{C}}$. For visual question answering, it is weights from an assembled network determined by the question $\mathbf{q}$, denoted by $\boldsymbol{\theta}^{\mathrm{V}}(\mathbf{q})$. $\mathbf{Z} \in [0,1]^K$ is the output, with K being the number of object classes or choices.

3.2. Attacks Beyond the Image Space

Attacking the physical parameters starts with setting a *goal*, which is what we hope the network to predict. This is done by minimizing a loss function $\mathcal{L}(\mathbf{Z})$, which determines how far the current output is from the desired status. An adversarial attack may either be targeted or non-targeted, and in this work we focus on the non-targeted attack, which specifies a class c' (usually the original true class) as which the image should *not* be classified, and the goal is to minimize the c'-th dimension of the output $\mathbf{Z}$: $\mathcal{L}(\mathbf{Z}) \doteq \mathcal{L}(\mathbf{Z}; c') = Z_{c'}$.

An obvious way to attack the physical space works by expanding the loss function $\mathcal{L}(\mathbf{Z})$, *i.e.*, $\mathcal{L}(\mathbf{Z}) = \mathcal{L} \circ \mathbf{f} \circ \mathbf{r}(\mathbf{X}; \boldsymbol{\theta})$, and minimizing this function with respect to the physical parameters $\mathbf{X}$. The optimization starts with an initial (unperturbed) state $\mathbf{X}_0 \doteq \mathbf{X}$. A total of $T_{\max}$ iterations are performed. In the t-th round, we compute the gradient vectors with respect to $\mathbf{X}_{t-1}$, *i.e.*, $\Delta\mathbf{X}_t = \nabla_{\mathbf{X}_{t-1}} \mathcal{L} \circ \mathbf{f} \circ \mathbf{r}(\mathbf{X}_{t-1}, \boldsymbol{\theta})$, and update $\mathbf{X}_{t-1}$ along this direction: $\mathbf{X}_t = \mathbf{X}_{t-1} + \eta \cdot \Delta\mathbf{X}_{t-1}$, where η is the *learning rate*. This iterative process is terminated if the goal of attacking is achieved or the maximal number of iterations $T_{\max}$ is reached. The accumulated perturbation over all T iterations is denoted by $\Delta\mathbf{X} = \eta \cdot \sum_{t=1}^{T} \Delta\mathbf{X}_t$.

The way of computing gradients $\Delta\mathbf{X}_t$ depends on whether $\mathbf{r}(\cdot)$ is differentiable. If so, this can be simply back-propagate gradients from the output space to the physical space. We follow the Fast Gradient Sign Method (FGSM) [12] to only preserve the sign in each dimension of the gradient vector. Otherwise, we apply zeroth-order optimization. To attack the d-th dimension in $\mathbf{X}$, we set a small value δ and approximate the gradient of $\mathbf{Z}$ by $\frac{\partial \mathcal{L}(\mathbf{Z})}{\partial X_d} \approx \frac{\mathcal{L} \circ \mathbf{f} \circ \mathbf{r}(\mathbf{X} + \delta \cdot \mathbf{e}_d) - \mathcal{L} \circ \mathbf{f} \circ \mathbf{r}(\mathbf{X} - \delta \cdot \mathbf{e}_d)}{2 \times \delta}$, where $\mathbf{e}_d$ is a D-dimensional vector with the d-th dimension set to be 1 and all the others to be 0. In general, every step of such update may randomly select a subset of all D dimensions for efficiency considerations, so our optimization algorithm is a form of stochastic coordinate descent. This is reminiscent of [9], where each step updates the values of a random subset of pixel values. Also following [9], we use the Adam optimizer [16] instead of standard gradient descent for its faster convergence.

3.3. Perceptibility

The goal of an adversarial attack is to produce a visually imperceptible perturbation, so that the network makes incorrect predictions after it is added to the original image. Given a rendering model $\mathbf{Y} = \mathbf{r}(\mathbf{X})$ and an added perturbation $\Delta\mathbf{X}$, the perturbation added to the rendered image is: $\Delta\mathbf{Y} = \mathbf{r}(\mathbf{X} + \Delta\mathbf{X}) - \mathbf{r}(\mathbf{X})$.

There are in general two ways of computing perceptibility. One of them works directly on the rendered image, which is similar to the definition in [37][25]: $p \doteq p(\Delta\mathbf{Y}) = \left(\frac{1}{W_{\mathrm{N}} \times H_{\mathrm{N}}} \sum_{w=1}^{W_{\mathrm{N}}} \sum_{h=1}^{H_{\mathrm{N}}} \|\Delta\mathbf{y}_{w,h}\|_2^2 \right)^{1/2}$, where $\mathbf{y}_{w,h}$ is a 3-dimensional vector representing the RGB intensities (normalized in $[0,1]$) of a pixel. Similarly, we can also define the perceptibility values for each set of physical parameters, *e.g.*, $p(\Delta\mathbf{N}) = \left(\frac{1}{W_{\mathrm{N}} \times H_{\mathrm{N}}} \sum_{w=1}^{W_{\mathrm{N}}} \sum_{h=1}^{H_{\mathrm{N}}} \|\Delta\mathbf{n}_{w,h}\|_2^2 \right)^{1/2}$.

We take $p(\Delta\mathbf{Y})$ as the major criterion of *visual* imperceptibility. Because of continuity, this can guarantee that all physical perturbations are sufficiently small as well. An advantage of placing the perceptibility constraint on pixels is that it allows a fair comparison of the attack success rates between image space attacks and physical space attacks. It also allows a direct comparison between attacks on different physical parameters. One potential disadvantage of placing the perceptibility constraint on physical parameters is that different physical parameters have different units and ranges. For example, the value range of RGB is $[0, 255]$, whereas that of spatial translation is $(-\infty, \infty)$. It is not directly obvious how to find a common threshold for different physical parameters.

When using the differentiable renderer, in order to guarantee imperceptibility, we constrain the RGB intensity changes on the image layer. In each iteration, after a new set of physical perturbations are generated, we check all pixels on the re-rendered image, and any perturbations exceeding a fixed threshold $U = 18$ from the original image is *truncated*. Truncations cause the inconsistency between the physical parameters and the rendered image and risk failures in attacking. To avoid frequent truncations, we set the learning rate η to be small, which consequently increases the number of iterations needed to attack the network.

When using the non-differentiable renderer, we pursue an alternative approach by adding another term $\|\Delta\mathbf{Y}\|_2^2$ into the loss function (weighted by λ) [9, 6], such that optimization can balance between attack success and perceptibility.

object's 2D location on the plane; $\mathbf{R}$ is a scalar rotation angle.

Attacking Perturbations	Image		Surface N.		Illumination		Material		Combined	
	Succ.	p	Succ.	p	Succ.	p	Succ.	p	Succ.	p
On AlexNet	100.00	5.7	89.27	10.8	29.61	25.8	18.88	25.8	94.42	18.1
On ResNet-34	99.57	5.1	88.41	9.3	14.16	29.3	3.43	55.2	94.85	16.4

Table 1. Effect of white-box adversarial attacks on ShapeNet object classification. By *combined*, we allow the three sets of physical parameters to be perturbed jointly. **Succ.** denotes the success rate of attacks (%, higher is better), and p is the perceptibility value (unit: 10^{-3}, lower is better). All p values are measured in the image space, *i.e.*, they are directly comparable.

3.4. Interpreting Image Space Adversaries in Physical Space

We do a reality check to confirm that image-space adversaries are almost never consistent with the non-local physical perturbations according to our (admittedly imperfect) rendering model. They are, of course, consistent with per-pixel changes of albedo.

We first find a perturbation $\Delta\mathbf{Y}$ in the image space, and then compute a perturbation in the physical space, $\Delta\mathbf{X}$, that corresponds to $\Delta\mathbf{Y}$. This is to set the optimization goal in the image space instead of the output space, though the optimization process is barely changed. Note that we are indeed pursuing interpreting $\Delta\mathbf{Y}$ in the physical space. Not surprisingly, as we will show in experiments, the reconstruction loss $\|\mathbf{Y}+\Delta\mathbf{Y}-\mathbf{r}(\mathbf{X}+\Delta\mathbf{X})\|_1$ does not go down, suggesting that approximations of $\Delta\mathbf{Y}$ in the physical space either do not exist, or cannot be found by the currently available optimization methods such as FGSM.

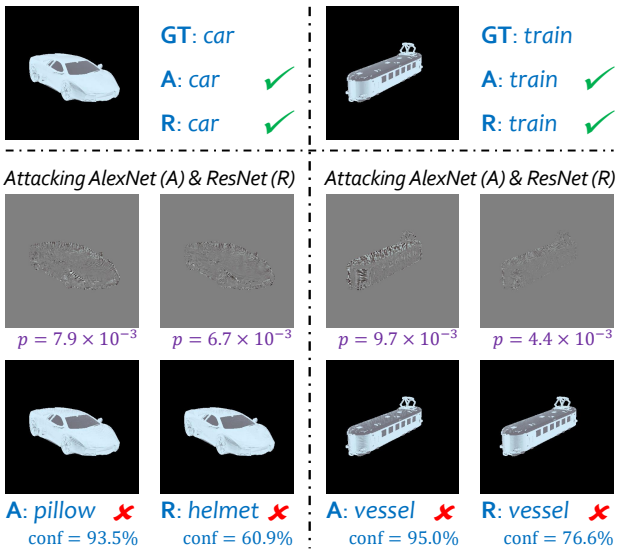


Figure 3. Examples of physical-space adversaries in 3D object classification on ShapeNet (using a differentiable renderer). In each example, the top row shows the original testing image, which is correctly classified by both AlexNet (A) and ResNet (R). The following two rows show the perturbations and the attacked image, respectively. All perturbations are magnified by a factor of **5** and shifted by **128**. p is the perceptibility value, and conf is the confidence (post-softmax output) of the prediction.

4. Experiments

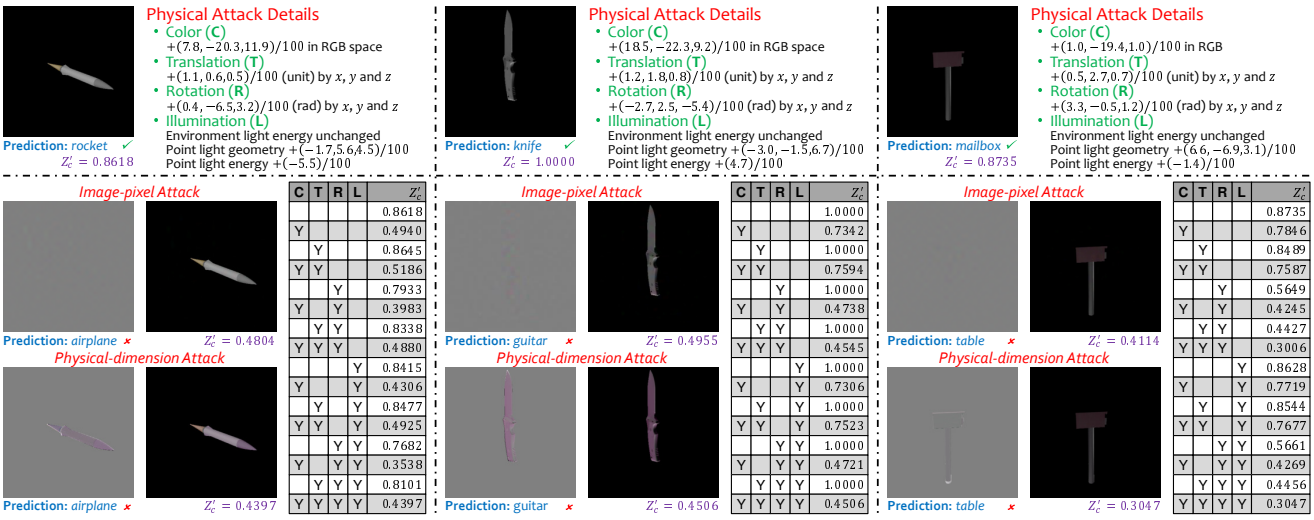
4.1. 3D Object Classification

3D object recognition experiments are conducted on the ShapeNetCore-v2 dataset [7], which contains 55 rigid object categories, each with various 3D models. Two popular deep neural networks are used: an 8-layer AlexNet [17] and a 34-layer deep residual network [13]. Both networks are pre-trained on the ILSVRC2012 dataset [33], and fine-tuned in our training set for 40 epochs using batch size 256. The learning rate is 0.001 for AlexNet and 0.005 for ResNet-34.

We experiment with both a differentiable renderer [20] and a non-differentiable renderer [5], and as a result there are some small differences in the experimental setup, despite the shared settings described above.

For the **differentiable renderer**, we randomly sample 125 3D models from each class, and select 4 fixed viewpoints for each object, so that each category has 500 training images. Similarly, another randomly chosen 50×4 images for each class are used for testing. AlexNet and ResNet-34 achieve 73.59% and 79.35% top-1 classification accuracies, respectively. These numbers are comparable to the single-view baseline accuracy reported in [36]. For each class, from the correctly classified testing samples, we choose 5 images with the highest classification probabilities on ResNet-34, and filter out 22 of them which are incorrectly classified by AlexNet, resulting in a target set of 233 images. The attack algorithm is the iterative version of FGSM [12]. We use the SGD optimizer with momentum 0.9 and weight decay 10^{-4}, and the maximal number of iterations is 120. Learning rate is 0.002 for attacking image space, 0.003 for attacking illumination and material, and 0.004 for attacking surface normal.

For the **non-differentiable renderer**, we render images with an azimuth angle uniformly sampled from $[0, \pi)$, a fixed elevation angle of $\pi/9$ and a fixed distance of 1.8. AlexNet gives a 65.89% top-1 testing set classification accuracy, and ResNet-34 achieves an even higher number of 68.88%. Among 55 classes, we find 51 with at least two images correctly classified. From each of them, we choose the two correct testing cases with the highest confidence score and thus compose a target set with 102 images. The



C	T	R	L	Z_c'
				0.8618
Y				0.4940
	Y			0.8645
Y	Y			0.5186
		Y		0.7933
Y		Y		0.3983
	Y	Y		0.8338
Y	Y	Y		0.4880
			Y	0.8415
Y			Y	0.4306
	Y		Y	0.8477
Y	Y		Y	0.4925
		Y	Y	0.7682
Y		Y	Y	0.3538
	Y	Y	Y	0.8101
Y	Y	Y	Y	0.4397

C	T	R	L	Z_c'
				1.0000
Y				0.7342
	Y			1.0000
Y	Y			0.7594
		Y		1.0000
Y		Y		0.4738
	Y	Y		1.0000
Y	Y	Y		0.4545
			Y	1.0000
Y			Y	0.7306
	Y		Y	1.0000
Y	Y		Y	0.7523
		Y	Y	1.0000
Y		Y	Y	0.4721
	Y	Y	Y	1.0000
Y	Y	Y	Y	0.4506

C	T	R	L	Z_c'
				0.8735
Y				0.7846
	Y			0.8489
Y	Y			0.7587
		Y		0.5649
Y		Y		0.4245
	Y	Y		0.4427
Y	Y	Y		0.3006
			Y	0.8628
Y			Y	0.7719
	Y		Y	0.8544
Y	Y		Y	0.7677
		Y	Y	0.5661
Y		Y	Y	0.4269
	Y	Y	Y	0.4456
Y	Y	Y	Y	0.3047

Figure 4. Examples of image-space and physical-space adversaries in 3D object classification on ShapeNet (using a non-differentiable renderer). In each example, the top row contains the original testing image and the detailed description of mid-level physical operations that can cause classification to fail. In the bottom row, we show the perturbations and attacked images in both attacks. Z_c' is the confidence (post-softmax output) of the true class. For each case, we also show results with different combinations of physical attacks in a table (a Y indicates the corresponding attack is on).

attack algorithm is ZOO [9] with $\delta = 10^{-4}$, $\eta = 2 \times 10^{-3}$ and $\lambda = 0.1$. The maximal number of iterations is 500 for AlexNet and 200 for ResNet-34.

4.1.1 Differentiable Renderer Results

First, we demonstrate in Table 1 that adversaries widely exist in the image space – as researchers have explored before [37][25], it is easy to confuse the network with small perturbations. In our case, the success rate is at or close to 100% and the perceptibility does not exceed 10^{-2}.

The next study is to find the correspondence of these image-space perturbations in the physical space. We tried the combination of 3 learning rates ($10^{-3}, 10^{-4}, 10^{-5}$) and 2 optimizers (SGD, Adam). However, for AlexNet, the objective (ℓ_1-distance) remains mostly constant; the malicious label after image-space attack is kept in only 8 cases, and in the vast majority cases, the original true label of the object is recovered. Therefore, using the current optimization method and rendering model, it is very difficult to find physical parameters that are approximately rendered into these image-space adversaries. This is expected, as physical parameters often have a non-local effect on the image.

Finally we turn to directly generating adversaries in the physical space. As shown in Table 1, this is much more difficult than in the image space – the success rate becomes lower and large perceptibility values are often observed on the successful cases. Typical adversarial examples generated in the physical space are shown in Figure 3. Allowing all physical parameters to be jointly optimized (*i.e.*, the *combined* strategy) produces the highest success rate. Among the three sets of physical parameters, attacking surface normals is more effective than the other two. This is expected, as using local perturbations is often easier in attacking deep neural networks [12]. The surface normal matrix shares the same dimensionality with the image lattice, and changing an element in the matrix only has very local impact on the rendered image. In comparison, illumination and material are both global properties of the 3D scene or the object, so tuning each parameter will cause a number of pixels to be modified, hence less effective in adversarial attacks.

We also examined truncation during the attack. For ResNet-34, on average, only 6.3, 1.6, 0 pixels were ever truncated for normal, illumination, material throughout the 120 iterations of attack. This number of truncation is relatively small comparing to the size of the rendered image (448×448). Therefore, the truncation is unlikely to contribute much to the attack.

4.1.2 Non-differentiable Renderer Results

We first report quantitative results with two settings, *i.e.*, attacking the image space and the physical space. Similarly, image-space adversaries are relatively easy to find. Among all 102 cases, 99 of them are successfully attacked within 500 steps on AlexNet, and all of them within 200 steps on ResNet-34. On the other hand, physical-space adversaries are much more difficult to construct. Using the same numbers of steps (500 on AlexNet and 200 on ResNet-34), the numbers of success attacks are merely 14 and 6 respectively.

We show several successful cases of image-space and

physical-space attacks in Figure 4. One can see quite different perturbation patterns from these two scenarios. An image-space perturbation is the sum of pixel-level differences, *e.g.*, even the intensities of two adjacent pixels can be modified individually, thus it is unclear if these images can really appear in the real world, nor can we diagnose the reason of failure. On the other hand, a physical-space perturbation is generated using a few mid-level operations such as slight rotation, translation and minor lighting changes. In theory, these adversaries can be instantiated in the physical world using a fine-level robotic controlling system.

Another benefit of generating physical-dimension adversaries lies in the ability of diagnosing vision algorithms. We use the cases shown in Figure 4 as examples. There are 14 changeable physical parameters, and we partition them into 4 groups, *i.e.*, the environment illumination (5 parameters), object rotation, position and color (3 parameters each). We enumerate all 2^4 subsets of these parameters, and thus generate 2^4 perturbations by only applying the perturbations in the subsets. It is interesting to see that in the first case, the effects of different perturbations are almost additive, *e.g.*, the joint attack on color and rotation has roughly the same effect as the sum of individual attacks. However, this is not always guaranteed. In the second case, for example, we find that attacking rotation alone produces little effect, but adding it to color attack causes a dramatic accuracy drop of 26%. On the other hand, the second case is especially sensitive to color, and the third one to rotation, suggesting that different images are susceptible to attacks in different subspaces. It is the interpretability of the physical-dimension attacks that provides the possibility to diagnose these cases at a finer level.

4.2. Visual Question Answering

We extend our experiments to a more challenging vision task – visual question answering. Experiments are performed on the recently released CLEVR dataset [14]. This is an engine that can generate an arbitrary number of 3D scenes with meta-information (object configuration). Each scene is also equipped with multiple generated questions, *e.g.*, asking for the number of specified objects in the scene, or if the object has a specified property.

The baseline algorithm is named Inferring and Executing Programs (IEP) [15]. It applies an LSTM to parse each question into a tree-structure program, which is then converted into a neural module network [1] that queries the visual features. We use the released model without training it by ourselves. We randomly pick up 100 testing images, on which all associated questions are correctly answered, as the target images.

The settings for generating adversarial perturbations are the same as in the object classification experiments: when using the differentiable renderer, the iterative FGSM

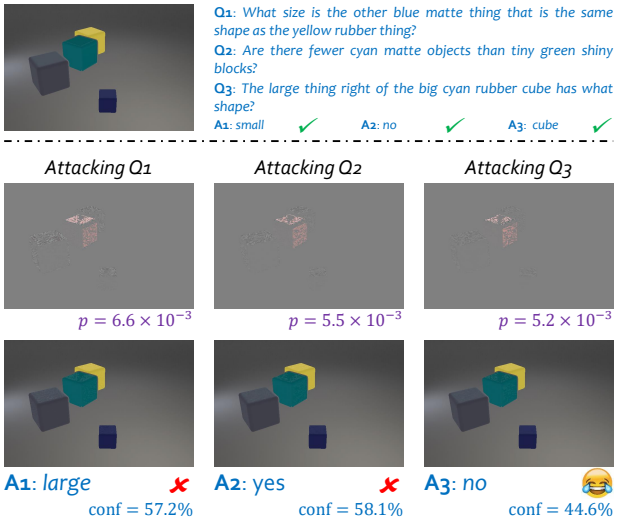


Figure 5. An example of physical-space adversaries in 3D visual question answering on CLEVR (using a differentiable renderer). In each example, the top row shows a testing image and three questions, all of which are correctly answered. The following two rows show the perturbations and the attacked image, respectively. All perturbations are magnified by a factor of **5** and shifted by **128**. p is the perceptibility value, and conf is the confidence (post-softmax output) of choosing this answer.

is used, and three sets of physical parameters are attacked either individually or jointly; when using the non-differentiable renderer, the ZOO algorithm [9] is used with $\delta = 10^{-3}$, $\eta = 10^{-2}$, $\lambda = 0.5$.

4.2.1 Differentiable Renderer Results

Results are shown in Table 2. We observe similar phenomena as in the classification experiments. This is expected, since after the question is parsed and a neural module network is generated, attacking either the image or the physical space is essentially equivalent to that in the classification task. Some typical examples are shown in Figure 5.

A side note comes from perturbing the material parameters. Although some visual questions are asking about the material (*e.g.*, *metal* or *rubber*) of an object, the success rate of this type of questions does not differ from that in attacking other questions significantly. This is because we are constraining perceptibility, which does not allow the material parameters to be modified by a large value.

A significant difference of visual question answering comes from the so called *language prior*. With a language parser, the network is able to clinch a small subset of answers without looking at the image, *e.g.*, when asked about the *color* of an object, it is very unlikely for the network to answer *yes* or *three*. Yet we find that sometimes the network can make such ridiculous errors. For instance, in

Attacking Perturbations	Image		Surface N.		Illumination		Material		Combined	
	Succ.	p	Succ.	p	Succ.	p	Succ.	p	Succ.	p
On IEP [15]	96.33	2.1	83.67	6.8	48.67	9.5	8.33	12.3	90.67	8.8

Table 2. Effect of white-box adversarial attacks on CLEVR visual question answering. By *combined*, we allow the three sets of physical parameters to be perturbed jointly. **Succ.** denotes the success rate of attacks (%, higher is better) of giving a correct answer, and p is the perceptibility value (unit: 10^{-3}, lower is better). All p values are measured in the image space, *i.e.*, they are directly comparable.

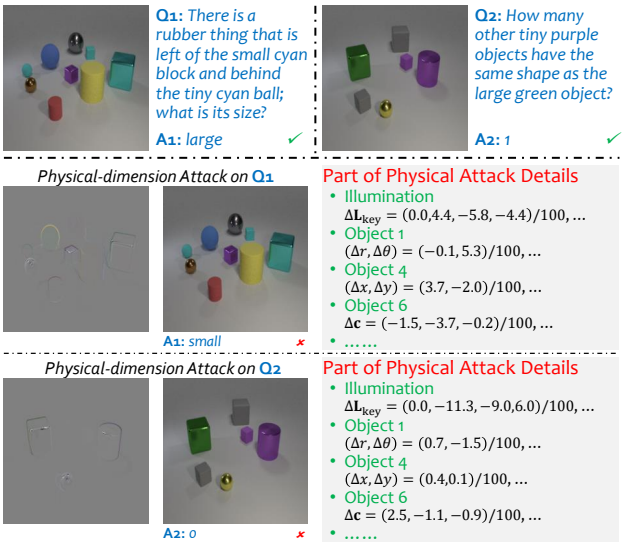


Figure 6. Examples of physical-space adversaries in 3D visual question answering on CLEVR (using a non-differentiable renderer). In each example, the top row contains a testing image and three questions. In the bottom row, we show the perturbations and attacked images. Detailed description of physical attacks on selective dimensions are also provided. All units of physical parameters follow the default setting in Blender.

the rightmost column of Figure 5, when asked about the *shape* of an object, the network answers *no* after a *non-targeted* attack.

4.2.2 Non-differentiable Renderer Results

We observe quite similar results as in ShapeNet experiments. It is relatively easy to find image-space adversaries, as our baseline successfully attacks 66 out of 100 targets within 500 steps, and 93 within 1,200 steps. Due to computational considerations, we set 500 to be the maximal step in our attack experiment, but only find 22 physical-space adversaries. This is expected, since visual question answering becomes quite similar to classification after the question is fixed.

We show two successfully attacked examples in Figure 6. Unlike ShapeNet experiments, color plays an important role in CLEVR, as many questions are related to filtering/counting objects with specified colors. We find that in many cases, our algorithm achieves success by mainly attacking the color of the key object (*i.e.* that asked in the question). This could seem problematic, as generated adversaries may threaten the original correct answer. But according to our inspection, the relatively big λ we chose ensured otherwise. Nevertheless, this observation is interesting because our algorithm does not know the question (*i.e.*, IEP is a black-box) or the answer (*i.e.*, each answer is simply a class ID), but it automatically tries to attack the weakness (*e.g.*, color) of the vision system.

5. Conclusions

In this paper, we generalize adversarial examples beyond the 2D image pixel intensities to 3D physical parameters. We are mainly interested to know: are neural networks vulnerable to perturbation on these intrinsic parameters that define a 3D scene, just like they are vulnerable to artificial noise added to the image pixels?

To study this, we plug a rendering module in front of the state-of-the-art deep networks, in order to connect the underlying 3D scene with the perceived 2D image. We are then able to conduct gradient based attacks on this more complete vision pipeline. Extensive experiments in object classification and visual question answering show that directly constructing adversaries in the physical space is effective, but the success rate is lower than that in the image space, and much heavier perturbations are required for successful attacks. To the best of our knowledge, ours is the first work to study imperceptible adversarial examples in 3D, where each dimension of the adversarial perturbation has clear meaning in the physical world.

Going forward, we see three potential directions for further research. First, as a side benefit, our study may provide practical tools to diagnose vision algorithms, especially evaluating the robustness in some interpretable dimensions such as color, lighting and object movements. Second, in 3D vision scenarios, we show the promise to defend the deep neural networks against 2D adversaries by interpreting an image in the physical space, so that the adversarial effects are weakened or removed after re-rendering. Third, while our pipeline will continue to benefit from higher quality rendering, we also acknowledge the necessity to test out our findings in real-world scenarios.

Acknowledgments We thank Guilin Liu, Cihang Xie, Zhishuai Zhang and Yi Zhang for discussions. This research is supported by IARPA D17PC00342 and a gift from YiTu.

References

[1] J. Andreas, M. Rohrbach, T. Darrell, and D. Klein. Neural Module Networks. *CVPR*, 2016. 7

[2] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. Lawrence Zitnick, and D. Parikh. VQA: Visual Question Answering. *ICCV*, 2015. 1

[3] A. Athalye, N. Carlini, and D. Wagner. Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. *ICML*, 2018. 3

[4] A. Athalye and I. Sutskever. Synthesizing Robust Adversarial Examples. *ICML*, 2018. 3

[5] Blender Online Community. Blender – a 3D modelling and rendering package. https://www.blender.org/, 2017. Blender Foundation, Blender Institute, Amsterdam. 3, 5

[6] N. Carlini and D. Wagner. Towards Evaluating the Robustness of Neural Networks. *IEEE Symposium on SP*, 2017. 3, 4

[7] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. ShapeNet: An Information-Rich 3D Model Repository. *arXiv preprint arXiv:1512.03012*, 2015. 5

[8] L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *TPAMI*, 2017. 1

[9] P. Chen, H. Zhang, Y. Sharma, J. Yi, and C. Hsieh. ZOO: Zeroth Order Optimization based Black-box Attacks to Deep Neural Networks without Training Substitute Models. *ACM Workshop on AI and Security*, 2017. 2, 4, 6, 7

[10] L. Engstrom, D. Tsipras, L. Schmidt, and A. Madry. A Rotation and a Translation Suffice: Fooling CNNs with Simple Transformations. *arXiv preprint arXiv:1712.02779*, 2017. 3

[11] I. Evtimov, K. Eykholt, E. Fernandes, T. Kohno, B. Li, A. Prakash, A. Rahmati, and D. Song. Robust Physical-World Attacks on Deep Learning Models. *arXiv preprint arXiv:1707.08945*, 2017. 3

[12] I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and Harnessing Adversarial Examples. *ICLR*, 2015. 1, 2, 3, 4, 5, 6

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. *CVPR*, 2016. 1, 5

[14] J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. L. Zitnick, and R. Girshick. CLEVR: A Diagnostic Dataset for Compositional Language and Elementary Visual Reasoning. *CVPR*, 2017. 1, 3, 7

[15] J. Johnson, B. Hariharan, L. van der Maaten, J. Hoffman, L. Fei-Fei, C. L. Zitnick, and R. Girshick. Inferring and Executing Programs for Visual Reasoning. *ICCV*, 2017. 3, 7, 8

[16] D. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *ICLR*, 2015. 4

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *NIPS*, 2012. 1, 5

[18] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial Examples in the Physical World. *ICLR Workshop*, 2017. 1, 2, 3

[19] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial Machine Learning at Scale. *ICLR*, 2017. 3

[20] G. Liu, D. Ceylan, E. Yumer, J. Yang, and J. M. Lien. Material Editing Using a Physically Based Rendering Network. *ICCV*, 2017. 3, 5

[21] Y. Liu, X. Chen, C. Liu, and D. Song. Delving into Transferable Adversarial Examples and Black-Box Attacks. *ICLR*, 2017. 3

[22] J. McCormac, A. Handa, S. Leutenegger, and A. Davison. SceneNet RGB-D: 5M Photorealistic Images of Synthetic Indoor Trajectories with Ground Truth. *ICCV*, 2017. 3

[23] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff. On Detecting Adversarial Perturbations. *ICLR*, 2017. 3

[24] S. M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard. Universal Adversarial Perturbations. *CVPR*, 2017. 3

[25] S. M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. *CVPR*, 2016. 3, 4, 6

[26] A. Nguyen, J. Yosinski, and J. Clune. Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images. *CVPR*, 2015. 1, 3

[27] F. E. Nicodemus, J. C. Richmond, J. J. Hsia, I. W. Ginsberg, and T. Limperis. Geometrical Considerations and Nomenclature for Reflectance. *Radiometry*, pages 94–145, 1992. 3

[28] K. Nishino. Directional Statistics BRDF Model. *ICCV*, 2009. 3

[29] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami. Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks. *IEEE Symposium on SP*, 2016. 3

[30] K. Pei, Y. Cao, J. Yang, and S. Jana. Towards Practical Verification of Machine Learning: The Case of Computer Vision Systems. *arXiv preprint arXiv:1712.01785*, 2017. 3

[31] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *CVPR*, 2017. 3

[32] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *TPAMI*, 39(6):1137–1149, 2017. 1

[33] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, pages 1–42, 2015. 5

[34] K. Sfikas, T. Theoharis, and I. Pratikakis. Exploiting the PANORAMA Representation for Convolutional Neural Network Classification and Retrieval. *Eurographics Workshop on 3D Object Retrieval*, 2017. 3

[35] E. Shelhamer, J. Long, and T. Darrell. Fully Convolutional Networks for Semantic Segmentation. *TPAMI*, 39(4):640–651, 2017. 1

[36] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. Multi-view Convolutional Neural Networks for 3D Shape Recognition. *ICCV*, 2015. 3, 5

[37] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing Properties of Neural Networks. In *ICLR*, 2014. 1, 4, 6

[38] F. Tramer, A. Kurakin, N. Papernot, D. Boneh, and P. McDaniel. Ensemble Adversarial Training: Attacks and Defenses. *arXiv preprint arXiv:1705.07204*, 2017. 3

[39] C. Xie, J. Wang, Z. Zhang, Y. Zhou, L. Xie, and A. L. Yuille. Adversarial Examples for Semantic Segmentation and Object Detection. *ICCV*, 2017. 1, 3

[40] X. Xu, X. Chen, C. Liu, A. Rohrbach, T. Darell, and D. Song. Can You Fool AI with Adversarial Examples on a Visual Turing Test? *arXiv preprint arXiv:1709.08693*, 2017. 3