Strike (with) a Pose: Neural Networks Are Easily Fooled by Strange Poses of Familiar Objects

Michael A. Alcorn
alcorma@auburn.edu

Qi Li
qzl0019@auburn.edu

Zhitao Gong
gong@auburn.edu

Chengfei Wang
czw0078@auburn.edu

Long Mai
malong@adobe.com

Wei-Shinn Ku
weishinn@auburn.edu

Anh Nguyen
anhnguyen@auburn.edu

Auburn University Adobe Inc.

Abstract

Despite excellent performance on stationary test sets, deep neural networks (DNNs) can fail to generalize to out-of-distribution (OoD) inputs, including natural, non-adversarial ones, which are common in real-world settings. In this paper, we present a framework for discovering DNN failures that harnesses 3D renderers and 3D models. That is, we estimate the parameters of a 3D renderer that cause a target DNN to misbehave in response to the rendered image. Using our framework and a self-assembled dataset of 3D objects, we investigate the vulnerability of DNNs to OoD poses of well-known objects in ImageNet. For objects that are readily recognized by DNNs in their canonical poses, DNNs incorrectly classify 97% of their pose space. In addition, DNNs are highly sensitive to slight pose perturbations. Importantly, adversarial poses transfer across models and datasets. We find that 99.9% and 99.4% of the poses misclassified by Inception-v3 also transfer to the AlexNet and ResNet-50 image classifiers trained on the same ImageNet dataset, respectively, and 75.5% transfer to the YOLOv3 object detector trained on MS COCO.

1. Introduction

For real-world technologies, such as self-driving cars [10], autonomous drones [14], and search-and-rescue robots [37], the test distribution may be non-stationary, and new observations will often be out-of-distribution (OoD), *i.e.*, not from the training distribution [42]. However, machine learning (ML) models frequently assign wrong labels with high confidence to OoD examples, such as adversarial examples [46, 29]—inputs specially crafted by an adversary to cause a target model to misbehave. But ML models are also vulnerable to *natural* OoD examples [21, 2, 48, 3]. For example, when a Tesla autopilot car failed to recognize a white truck against a bright-lit sky—an unusual view that might be OoD—it crashed into the truck, killing the driver [3].

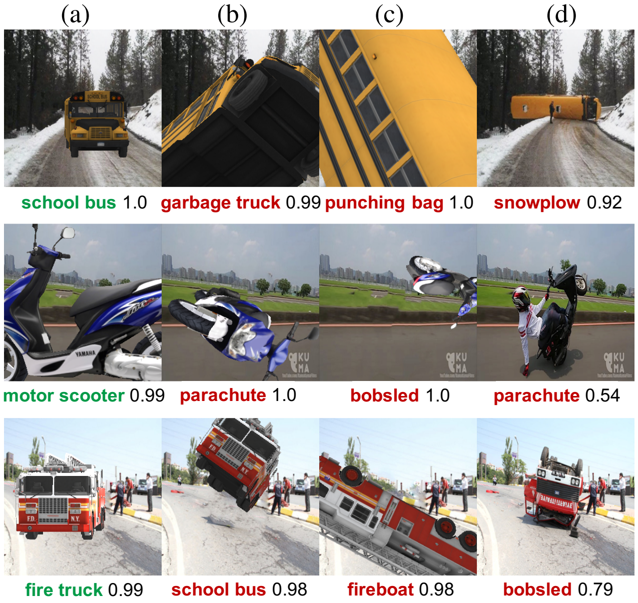


Figure 1: The Google Inception-v3 classifier [44] correctly labels the canonical poses of objects (a), but fails to recognize out-of-distribution images of objects in unusual poses (b–d), including real photographs retrieved from the Internet (d). The left 3×3 images (a–c) are found by our framework and rendered via a 3D renderer. Below each image are its top-1 predicted label and confidence score.

Previous research has successfully used 3D graphics as a diagnostic tool for computer vision systems [7, 31, 47, 32, 50]. To understand natural Type II classification errors in DNNs, we searched for misclassified 6D poses (*i.e.*, 3D translations and 3D rotations) of 3D objects. Our results reveal that state-of-the-art image classifiers and object detectors trained on large-scale image datasets [36, 22] misclassify most poses for many familiar training-set objects. For example, DNNs predict the front view of a school bus—an object in the ImageNet dataset [36]—extremely well (Fig. 1a) but fail to recognize the same object when it is too close or flipped over, *i.e.*, in poses that are OoD yet exist in the real world (Fig. 1d). However, a self-driving car needs to correctly estimate at least some attributes of an incoming, unknown object (instead of simply rejecting it [17, 38]) to handle the situation gracefully and minimize damage. Because road environments are highly variable [3, 2], addressing this type of OoD error is a non-trivial challenge.

In this paper, we propose a framework for finding OoD errors in computer vision models in which iterative optimization in the parameter space of a 3D renderer is used to estimate changes (*e.g.*, in object geometry and appearance, lighting, background, or camera settings) that cause a target DNN to misbehave (Fig. 2). With our framework, we generated unrestricted 6D poses of 3D objects and studied how DNNs respond to 3D translations and 3D rotations of objects. For our study, we built a dataset of 3D objects corresponding to 30 ImageNet classes relevant to the self-driving car application. The code for our framework is available at `https://github.com/airalcorn2/strike-with-a-pose`. In addition, we built a simple GUI tool that allows users to generate their own adversarial renders of an object. Our main findings are:

- ImageNet classifiers only correctly label 3.09% of the entire 6D pose space of a 3D object, and misclassify many generated adversarial examples (AXs) that are human-recognizable (Fig. 1b–c). A misclassification can be found via a change as small as 10.31°, 8.02°, and 9.17° to the yaw, pitch, and roll, respectively.

- 99.9% and 99.4% of AXs generated against Inception-v3 transfer to the AlexNet and ResNet-50 image classifiers, respectively, and 75.5% transfer to the YOLOv3 object detector.

- Training on adversarial poses generated by the 30 objects (in addition to the original ImageNet data) did not help DNNs generalize well to held-out objects in the same class.

In sum, our work shows that state-of-the-art DNNs perform *image classification* well but are still far from true *object recognition*. While it might be possible to improve DNN robustness through adversarial training with many more 3D objects, we hypothesize that future ML models capable of visual reasoning may instead benefit from better incorporation of 3D information.

2. Framework

2.1. Problem formulation

Let f be an image classifier that maps an image $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$ onto a softmax probability distribution over 1,000 output classes [44]. Let R be a 3D renderer that takes as input a set of parameters ϕ and outputs a render, *i.e.*, a 2D image $R(\phi) \in \mathbb{R}^{H \times W \times C}$ (see Fig. 2). Typically, ϕ is factored into mesh vertices V, texture images T, a background image B, camera parameters C, and lighting parameters L, *i.e.*, $\phi = \{V, T, B, C, L\}$ [19]. To change the 6D pose of a given 3D object, we apply a 3D rotation and 3D translation, parameterized by $\mathbf{w} \in \mathbb{R}^6$, to the original vertices V yielding a new set of vertices V^*.

Here, we wish to estimate only the pose transformation parameters $\mathbf{w}$ (while keeping all parameters in ϕ fixed) such that the rendered image $R(\mathbf{w}; \phi)$ causes the classifier f to assign the highest probability (among all outputs) to an incorrect target output at index t. Formally, we attempt to solve the below optimization problem:

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} (f_t(R(\mathbf{w}; \phi))) \tag{1}$$

In practice, we minimize the cross-entropy loss $\mathcal{L}$ for the target class. Eq. 1 may be solved efficiently via backpropagation if both f and R are differentiable, *i.e.*, we are able to compute $\partial \mathcal{L} / \partial \mathbf{w}$. However, standard 3D renderers, *e.g.*, OpenGL [51], typically include many non-differentiable operations and cannot be inverted [27]. Therefore, we attempted two approaches: (1) harnessing a recently proposed differentiable renderer and performing gradient descent using its analytical gradients; and (2) harnessing a non-differentiable renderer and approximating the gradient via finite differences.

We will next describe the target classifier (Sec. 2.2), the renderers (Sec. 2.3), and our dataset of 3D objects (Sec. 2.4) before discussing the optimization methods (Sec. 3).

2.2. Classification networks

We chose the well-known, pre-trained Google Inception-v3 [45] DNN from the PyTorch model zoo [33] as the main image classifier for our study (the default DNN if not otherwise stated). The DNN has a 77.45% top-1 accuracy on the ImageNet ILSVRC 2012 dataset [36] of 1.2 million images corresponding to 1,000 categories.

2.3. 3D renderers

Non-differentiable renderer. We chose ModernGL [1] as our non-differentiable renderer. ModernGL is a simple Python interface for the widely used OpenGL graphics engine. ModernGL supports fast, GPU-accelerated rendering.

Differentiable renderer. To enable backpropagation through the non-differentiable rasterization process, Kato et

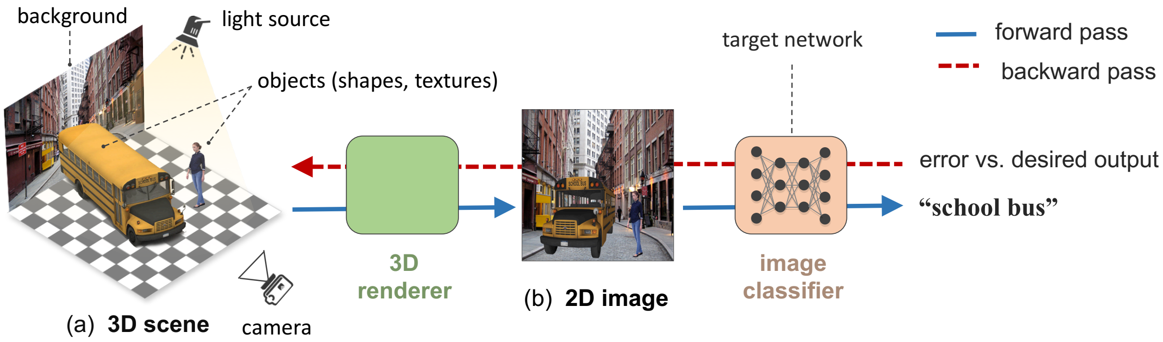


Figure 2: To test a target DNN, we build a 3D scene (a) that consists of 3D objects (here, a school bus and a pedestrian), lighting, a background scene, and camera parameters. Our 3D renderer renders the scene into a 2D image, which the image classifier labels `school bus`. We can estimate the pose changes of the school bus that cause the classifier to misclassify by (1) approximating gradients via finite differences; or (2) backpropagating (red dashed line) through a differentiable renderer.

al. [19] replaced the discrete pixel color sampling step with a linear interpolation sampling scheme that admits non-zero gradients. While the approximation enables gradients to flow from the output image back to the renderer parameters ϕ, the render quality is lower than that of our non-differentiable renderer (see Fig. S1 for a comparison). Hereafter, we refer to the two renderers as NR and DR.

2.4. 3D object dataset

Construction. Our main dataset consists of 30 unique 3D object models (purchased from many 3D model marketplaces) corresponding to 30 ImageNet classes relevant to a traffic environment (Fig. S2). The 30 classes include 20 vehicles (*e.g.*, `school bus` and `cab`) and 10 street-related items (*e.g.*, `traffic light`). See Sec. S1 for more details.

Each 3D object is represented as a mesh, *i.e.*, a list of triangular faces, each defined by three vertices [27]. The 30 meshes have on average 9,908 triangles (Table S1). To maximize the realism of the rendered images, we used only 3D models that have high-quality 2D image textures. We did not choose 3D models from public datasets, *e.g.*, ObjectNet3D [52], because most of them do not have high-quality image textures. That is, the renders of such models may be correctly classified by DNNs but still have poor realism.

Evaluation. We recognize that a reality gap will often exist between a render and a real photo. Therefore, we rigorously evaluated our renders to make sure the reality gap was acceptable for our study. From ~100 initially-purchased 3D models, we selected the 30 highest-quality models using the evaluation method below.

First, we quantitatively evaluated DNN predictions on the renders. For each object, we sampled 36 unique views (common in ImageNet) evenly divided into three sets. For each set, we set the object at the origin, the up direction to $(0, 1, 0)$, and the camera position to $(0, 0, -z)$ where $z = \{4, 6, 8\}$. We sampled 12 views per set by starting the object at a 10° yaw and generating a render at every 30° yaw-rotation. Across all objects and all renders, the Inception-v3 top-1 accuracy was 83.23% (compared to 77.45% on ImageNet images [44]) with a mean top-1 confidence score of 0.78 (Table S2). See Sec. S1 for more details.

Second, we qualitatively evaluated the renders by comparing them to real photos. We produced 116 (real photo, render) pairs via three steps: (1) we retrieved real photos of an object (*e.g.*, a car) from the Internet; (2) we replaced the object with matching background content in Adobe Photoshop; and (3) we manually rendered the 3D object on the background such that its pose closely matched that in the reference photo. Fig. S3 shows example (real photo, render) pairs. While discrepancies can be spotted in our side-by-side comparisons, we found that most of the renders passed our human visual Turing test if presented alone.

2.5. Background images

Previous studies have shown that image classifiers may be able to correctly label an image when foreground objects are removed (*i.e.*, based on only the background content) [57]. Because the purpose of our study was to understand how DNNs recognize an object itself, a non-empty background would have hindered our interpretation of the results. Therefore, we rendered all images against a plain background with RGB values of $(0.485, 0.456, 0.406)$, *i.e.*, the mean pixel of ImageNet images. Note that the presence of a non-empty background should not alter our main qualitative findings in this paper—adversarial poses can be easily found against real background photos (Fig. 1).

3. Methods

We will describe the common pose transformations (Sec. 3.1) used in the main experiments. We were able to experiment with non-gradient methods because: (1) the pose transformation space $\mathbb{R}^6$ that we optimize in is fairly low-dimensional; and (2) although the NR is non-differentiable, its rendering speed is several orders of magnitude faster than that of DR. In addition, our preliminary results showed that the objective function considered in Eq. 1 is highly non-convex (see Fig. 4), therefore, it is interesting to compare (1) random search vs. (2) gradient descent using finite-difference (FD) approximated gradients vs. (3) gradient descent using the DR gradients.

3.1. Pose transformations

We used standard computer graphics transformation matrices to change the pose of 3D objects [27]. Specifically, to rotate an object with geometry defined by a set of vertices $V = \{v_i\}$, we applied the linear transformations in Eq. 2 to each vertex $v_i \in \mathbb{R}^3$:

$$v_i^R = R_y R_p R_r v_i \tag{2}$$

where R_y, R_p, and R_r are the 3×3 rotation matrices for yaw, pitch, and roll, respectively (the matrices can be found in Sec. S6). We then translated the rotated object by adding a vector $T = \begin{bmatrix} x_\delta & y_\delta & z_\delta \end{bmatrix}^\top$ to each vertex:

$$v_i^{R,T} = T + v_i^R \tag{3}$$

In all experiments, the center $c \in \mathbb{R}^3$ of the object was constrained to be inside a sub-volume of the camera viewing frustum. That is, the x-, y-, and z-coordinates of c were within $[-s, s]$, $[-s, s]$, and $[-28, 0]$, respectively, with s being the maximum value that would keep c within the camera frame. Specifically, s is defined as:

$$s = d \cdot \tan(\theta_v) \tag{4}$$

where θ_v is one half the camera's angle of view (*i.e.*, $8.213°$ in our experiments) and d is the absolute value of the difference between the camera's z-coordinate and z_δ.

3.2. Random search

In reinforcement learning problems, random search (RS) can be surprisingly effective compared to more sophisticated methods [41]. For our RS procedure, instead of iteratively following some approximated gradient to solve the optimization problem in Eq. 1, we simply randomly selected a new pose in each iteration. The rotation angles for the matrices in Eq. 2 were uniformly sampled from $(0, 2\pi)$. x_δ, y_δ, and z_δ were also uniformly sampled from the ranges defined in Sec. 3.1.

3.3. z_δ-constrained random search

Our preliminary RS results suggest the value of z_δ (which is a proxy for the object's size in the rendered image) has a large influence on a DNN's predictions. Based on this observation, we used a z_δ-constrained random search (ZRS) procedure both as an initializer for our gradient-based methods and as a naive performance baseline (for comparisons in Sec. 4.4). The ZRS procedure consisted of generating 10 random samples of $(x_\delta, y_\delta, \theta_y, \theta_p, \theta_r)$ at each of 30 evenly spaced z_δ from -28 to 0.

When using ZRS for initialization, the parameter set with the maximum target probability was selected as the starting point. When using the procedure as an attack method, we first gathered the maximum target probabilities for each z_δ, and then selected the best two z_δ to serve as the new range for RS.

3.4. Gradient descent with finite-difference

We calculated the first-order derivatives via finite central differences and performed vanilla gradient descent to iteratively minimize the cross-entropy loss $\mathcal{L}$ for a target class. That is, for each parameter $\mathbf{w}_i$, the partial derivative is approximated by:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}_i} = \frac{\mathcal{L}(\mathbf{w}_i + \frac{h}{2}) - \mathcal{L}(\mathbf{w}_i - \frac{h}{2})}{h} \tag{5}$$

Although we used an h of 0.001 for all parameters, a different step size can be used per parameter. Because radians have a circular topology (*i.e.*, a rotation of 0 radians is the same as a rotation of 2π radians, 4π radians, etc.), we parameterized each rotation angle θ_i as $(\cos(\theta_i), \sin(\theta_i))$—a technique commonly used for pose estimation [30] and inverse kinematics [11]—which maps the Cartesian plane to angles via the $atan2$ function. Therefore, we optimized in a space of $3 + 2 \times 3 = 9$ parameters.

The approximate gradient $\nabla\mathcal{L}$ obtained from Equation (5) served as the gradient in our gradient descent. We used the vanilla gradient descent update rule:

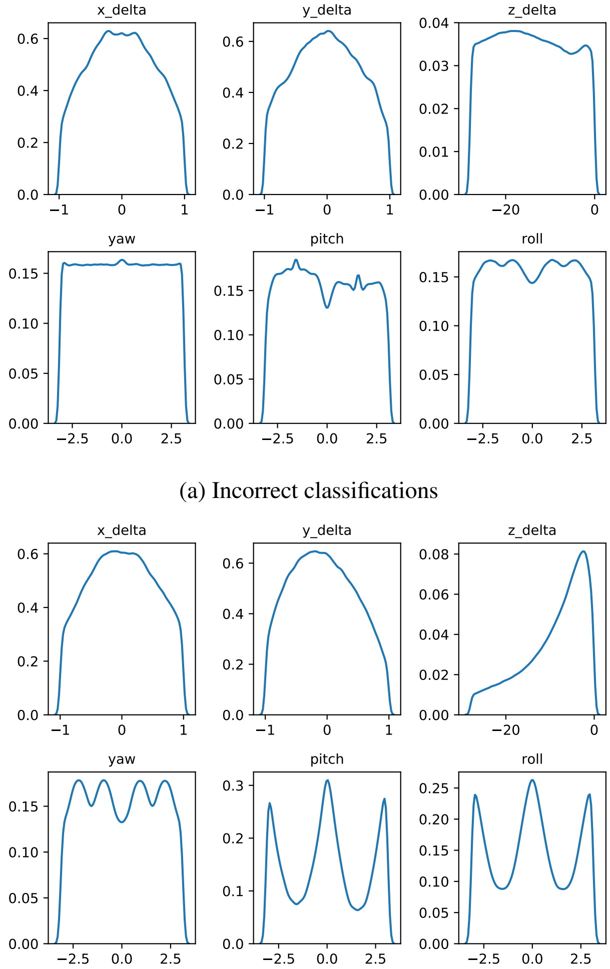
$$\mathbf{w} \coloneqq \mathbf{w} - \gamma \nabla \mathcal{L}(\mathbf{w}) \tag{6}$$

with a learning rate γ of 0.001 for all parameters and optimized for 100 steps (no other stopping criteria).

4. Experiments and results

4.1. Neural networks are easily confused by object rotations and translations

Experiment. To test DNN robustness to object rotations and translations, we used RS to generate samples for every 3D object in our dataset. In addition, to explore the impact of lighting on DNN performance, we considered three different lighting settings: bright, medium, and dark (example renders in Fig. S10). In all three settings, both the directional light and the ambient light were white in color, *i.e.*, had RGB values of $(1.0, 1.0, 1.0)$, and the directional light was oriented at $(0, -1, 0)$ (*i.e.*, pointing straight



(a) Incorrect classifications

(b) Correct classifications

Figure 3: The distributions of individual pose parameters for (a) high-confidence ($p \geq 0.7$) incorrect classifications and (b) correct classifications obtained from the random sampling procedure described in Sec. 3.2. x_δ and y_δ have been normalized w.r.t. their corresponding s from Eq. 4.

down). The directional light intensities and ambient light intensities were $(1.2, 1.6)$, $(0.4, 1.0)$, and $(0.2, 0.5)$ for the bright, medium, and dark settings, respectively. All other experiments used the medium lighting setting.

Misclassifications uniformly cover the pose space. For each object, we calculated the DNN accuracy (*i.e.*, percent of correctly classified samples) across all three lighting settings (Table S5). The DNN was wrong for the vast majority of samples, *i.e.*, the median percent of correct classifications for all 30 objects was only 3.09%. We verified the discovered adversarial poses transfer to the real world by using the 3D objects to reproduce natural, misclassified poses found on the Internet (see Sec. S3). High-confidence misclassifications ($p \geq 0.7$) are largely uniformly distributed across every pose parameter (Fig. 3a), *i.e.*, AXs can be found throughout the parameter landscape (see Fig. S15 for examples). In contrast, correctly classified examples are highly multimodal w.r.t. the rotation axis angles and heavily biased towards z_δ values that are closer to the camera (Fig. 3b; also compare Fig. S4 vs. Fig. S6). Intriguingly, for ball-like objects (not included in our main traffic dataset), the DNN was far more accurate across the pose space (see Sec. S8).

An object can be misclassified as many different labels. Previous research has shown that it is relatively easy to produce AXs corresponding to many different classes when optimizing input images [46] or 3D object textures [5], which are very high-dimensional. When finding adversarial poses, one might expect—because all renderer parameters, including the original object geometry and textures, are held constant—the success rate to depend largely on the similarities between a given 3D object and examples of the target in ImageNet. Interestingly, across our 30 objects, RS discovered $990/1000$ different ImageNet classes (132 of which were shared between all objects). When only considering high-confidence ($p \geq 0.7$) misclassifications, our 30 objects were still misclassified into 797 different classes with a median number of 240 incorrect labels found per object (see Fig. S16 and Fig. S6 for examples). Across all adversarial poses and objects, DNNs tend to be more confident when correct than when wrong (the median of median probabilities were 0.41 vs. 0.21, respectively).

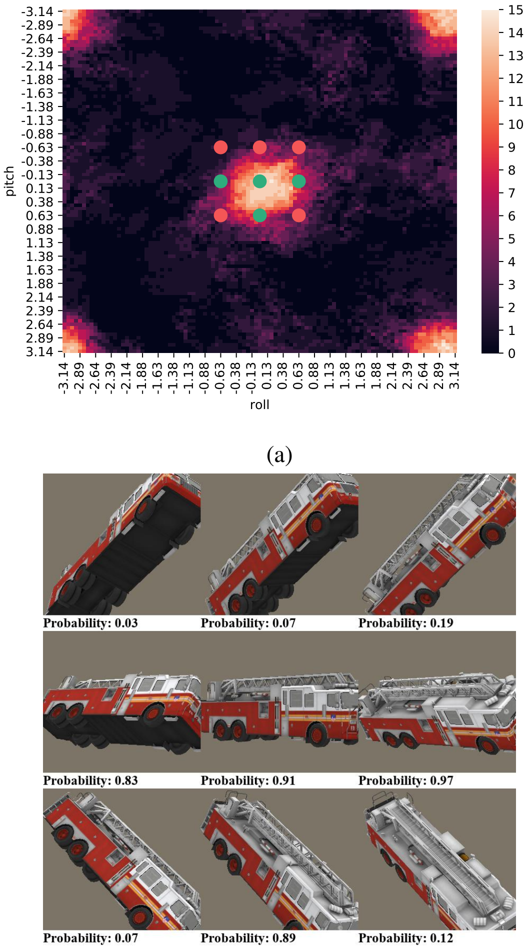
4.2. Common object classifications are shared across different lighting settings

Here, we analyze how our results generalize across different lighting conditions. From the data produced in Sec. 4.1, for each object, we calculated the DNN accuracy under each lighting setting. Then, for each object, we took the absolute difference of the accuracies for all three lighting combinations (*i.e.*, bright vs. medium, bright vs. dark, and medium vs. dark) and recorded the maximum of those values. The median "maximum absolute difference" of accuracies for all objects was 2.29% (compared to the median accuracy of 3.09% across all lighting settings). That is, DNN accuracy is consistently low across all lighting conditions. Lighting changes would not alter the fact that DNNs are vulnerable to adversarial poses.

We also recorded the 50 most frequent classes for each object under the different lighting settings (S_b, S_m, and S_d). Then, for each object, we computed the intersection over union score o_S for these sets:

$$o_S = 100 \cdot \frac{|S_b \cap S_m \cap S_d|}{|S_b \cup S_m \cup S_d|} \tag{7}$$

The median o_S for all objects was 47.10%. That is, for 15 out of 30 objects, 47.10% of the 50 most frequent classes were shared across lighting settings. While lighting does



(a)

(b)

Figure 4: Inception-v3's ability to correctly classify images is highly localized in the rotation and translation parameter space. (a) The classification landscape for 15 vehicle objects when altering θ_r and θ_p and holding $(x_\delta, y_\delta, z_\delta, \theta_y)$ at $(0, 0, -3, \frac{\pi}{4})$. Lighter regions correspond to poses with a greater number of correctly classified vehicle objects. Green and red circles indicate correct and incorrect classifications, respectively, corresponding to the fire truck object poses found in (b).

have an impact on DNN misclassifications (as expected), the large number of shared labels across lighting settings suggests ImageNet classes are strongly associated with certain adversarial poses regardless of lighting.

4.3. Correct classifications are highly localized in the rotation and translation landscape

To gain some intuition for how Inception-v3 responds to rotations and translations of an object, we plotted the probability and classification landscapes for paired parameters (*e.g.*, Fig. 4; pitch vs. roll) while holding the other parameters constant. We qualitatively observed that the DNN's ability to recognize an object (*e.g.*, a fire truck) in an image varies radically as the object is rotated in the world (Fig. 4). Further, adversarial poses often generalize across similar objects (*e.g.*, 83% of the sampled poses were misclassified for *all* 15 four-wheeled vehicle objects).

Experiment. To quantitatively evaluate the DNN's sensitivity to rotations and translations, we tested how it responded to single parameter disturbances. For each object, we randomly selected 100 distinct starting poses that the DNN had correctly classified in our random sampling runs. Then, for each parameter (*e.g.*, yaw rotation angle), we randomly sampled 100 new values[1] while holding the others constant. For each sample, we recorded whether or not the object remained correctly classified, and then computed the failure (*i.e.*, misclassification) rate for a given (object, parameter) pair. Plots of the failure rates for all (object, parameter) combinations can be found in Fig. S18.

Additionally, for each parameter, we calculated the median of the median failure rates. That is, for each parameter, we first calculated the median failure rate for all objects, and then calculated the median of those medians for each parameter. Further, for each (object, starting pose, parameter) triple, we recorded the magnitude of the smallest parameter change that resulted in a misclassification. Then, for each (object, parameter) pair, we recorded the median of these minimum values. Finally, we again calculated the median of these medians across objects (Table 1).

Results. As can be seen in Table 1, the DNN is highly sensitive to all single parameter disturbances, but it is especially sensitive to disturbances along the depth (z_δ), pitch (θ_p), and roll (θ_r). To aid in the interpretation of these results, we converted the raw disturbance values in Table 1 to image units. For x_δ and y_δ, the interpretable units are the number of pixels the object shifted in the x or y directions *of the image* (however, note that 3D translations are *not* equivalent to 2D translations due to the perspective projection).

We found that a change in rotation as small as $8.02°$ can cause an object to be misclassified (Table 1). Along the spatial dimensions, a translation resulting in the object moving as few as 2 px horizontally or 4.5 px vertically also caused the DNN to misclassify.[2] Lastly, along the z-axis, a change in "size" (*i.e.*, the area of the object's bounding box) of only 5.4% can cause an object to be misclassified.

4.4. Optimization methods can effectively generate targeted adversarial poses

Given a challenging, highly non-convex objective landscape (Fig. 4), we wish to evaluate the effectiveness of two

[1] using the random sampling procedure described in Sec. 3.2

[2] Note that the sensitivity of classifiers and object detectors to *2D* translations has been observed in concurrent work [35, 12, 56, 6].

Parameter	Fail Rate (%)	Min. Δ	Int. Δ
x_δ	42	0.09	2.0 px
y_δ	49	0.10	4.5 px
z_δ	81	0.77	5.4%
θ_y	69	0.18	10.31°
θ_p	83	0.14	8.02°
θ_r	81	0.16	9.17°

Table 1: The median of the median failure rates and the median of the median minimum disturbances (Min. Δ) for the single parameter sensitivity tests described in Section 4.3. Int. Δ converts the values in Min. Δ to more interpretable units. For x_δ and y_δ, the interpretable units are pixels. For z_δ, the interpretable unit is the percent change in the area of the bounding box containing the object. See main text and Fig. S18 for additional information.

different types of approximate gradients at targeted attacks, *i.e.*, finding adversarial examples misclassified as a target class [46]. Here, we compare (1) random search; (2) gradient descent with finite-difference gradients (FD-G); and (3) gradient descent with analytical, approximate gradients provided by a differentiable renderer (DR-G) [19].

Experiment. Because our adversarial pose attacks are inherently constrained by the fixed geometry and appearances of a given 3D object (see Sec. 4.1), we defined the targets to be the 50 most frequent incorrect classes found by our RS procedure for each object. For each (object, target) pair, we ran 50 optimization trials using ZRS, FD-G, and DR-G. All treatments were initialized with a pose found by the ZRS procedure and then allowed to optimize for 100 iterations.

Results. For each of the 50 optimization trials, we recorded both whether or not the target was hit and the maximum target probability obtained during the run. For each (object, target) pair, we calculated the percent of target hits and the median maximum confidence score of the target labels (see Table 2). As shown in Table 2, FD-G is substantially more effective than ZRS at generating targeted adversarial poses, having both higher median hit rates and confidence scores. In addition, we found the approximate gradients from DR to be surprisingly noisy, and DR-G largely underperformed even non-gradient methods (ZRS) (see Sec. S5).

4.5. Adversarial poses transfer to different image classifiers and object detectors

The most important property of previously documented AXs is that they transfer across ML models, enabling black-box attacks [55]. Here, we investigate the transferability of our adversarial poses to (a) two different image classifiers, AlexNet [20] and ResNet-50 [16], trained on the same ImageNet dataset; and (b) an object detector YOLOv3 [34] trained on the MS COCO dataset [22].

		Hit Rate (%)	Target Prob.
ZRS	random search	78	0.29
FD-G	gradient-based	**92**	**0.41**
DR-G[†]	gradient-based	32	0.22

Table 2: The median percent of target hits and the median of the median target probabilities for random search (ZRS), gradient descent with finite difference gradients (FD-G), and DR gradients (DR-G). All attacks are targeted and initialized with z_δ-constrained random search. [†]DR-G is not directly comparable to FD-G and ZRS (details in Sec. S4).

For each object, we randomly selected 1,350 AXs that were misclassified by Inception-v3 with high confidence ($p \geq 0.9$) from our untargeted RS experiments in Sec. 4.1. We exposed the AXs to AlexNet and ResNet-50 and calculated their misclassification rates. We found that almost all AXs transfer with median misclassification rates of 99.9% and 99.4% for AlexNet and ResNet-50, respectively. In addition, 10.1% of AlexNet misclassifications and 27.7% of ResNet-50 misclassifications were identical to the Inception-v3 predicted labels.

There are two orthogonal hypotheses for this result. First, the ImageNet training-set images themselves may contain a strong bias towards common poses, omitting uncommon poses (Sec. S7 shows supporting evidence from a nearest-neighbor test). Second, the models themselves may not be robust to even slight disturbances of the known, in-distribution poses.

Object detectors. Previous research has shown that object detectors can be more robust to adversarial attacks than image classifiers [25]. Here, we investigate how well our AXs transfer to a state-of-the-art object detector—YOLOv3. YOLOv3 was trained on MS COCO, a dataset of bounding boxes corresponding to 80 different object classes. We only considered the 13 objects that belong to classes present in both the ImageNet and MS COCO datasets. We found that 75.5% of adversarial poses generated for Inception-v3 are also misclassified by YOLOv3 (see Sec. S2 for more details). These results suggest the adversarial pose problem transfers across datasets, models, and tasks.

4.6. Adversarial training

One of the most effective methods for defending against OoD examples has been adversarial training [15], *i.e.* augmenting the training set with AXs—also a common approach in anomaly detection [9]. We tested whether adversarial training can improve DNN robustness to new poses generated for (1) our 30 training-set 3D objects; and (2) seven held-out 3D objects (see Sec. S9 for details). Following adversarial training, the accuracy of the DNN substantially increased for *known* objects (Table 3; 99.67% vs. 6.7%). However, the model (AT) still misclassified the adversarial poses of held-out objects at an 89.2% error rate.

	PT	AT
Error (T)	99.67	6.7
Error (H)	99.81	89.2
High-confidence Error (T)	87.8	1.9
High-confidence Error (H)	48.2	33.3

Table 3: The median percent of misclassifications (Error) and high-confidence (*i.e.*, $p > 0.7$) misclassifications by the pre-trained AlexNet (PT) and our AlexNet trained with adversarial examples (AT) on random poses of training-set objects (T) and held-out objects (H).

5. Related work

Out-of-distribution detection. OoD classes, *i.e.*, classes not found in the training set, present a significant challenge for computer vision technologies in real-world settings [38]. Here, we study an orthogonal problem—correctly classifying OoD poses of objects from *known* classes. While rejecting to classify is a common approach for handling OoD examples [17, 38], the OoD poses in our work come from known classes and thus *should be* assigned correct labels.

2D adversarial examples. Numerous techniques for crafting AXs that fool image classifiers have been discovered [55]. However, previous work has typically optimized in the 2D input space [55], *e.g.*, by synthesizing an entire image [29], a small patch [18, 13], a few pixels [8], or only a single pixel [40]. But pixel-wise changes are uncorrelated [28], so pixel-based attacks may not transfer well to the real world [24, 26] because there is an infinitesimal chance that such specifically crafted, uncorrelated pixels will be encountered in the vast physical space of camera, lighting, traffic, and weather configurations. [54] generated spatially transformed adversarial examples that are perceptually realistic and more difficult to defend against, but the technique still directly operates on pixels.

3D adversarial examples. Athalye et al. [5] used a 3D renderer to synthesize textures for a 3D object such that, under a wide range of camera views, the object was still rendered into an effective AX. We also used 3D renderers, but instead of optimizing textures, we optimized the poses of known objects to cause DNNs to misclassify (*i.e.*, we kept the textures, lighting, camera settings, and background image constant).

Concurrent work. We describe below two concurrent attempts that are closely related to ours. First, Liu et al. [23] proposed a differentiable 3D renderer and used it to perturb both an object's geometry and the scene's lighting to cause a DNN to misbehave. However, their geometry perturbations were constrained to be infinitesimal so that the visibility of the vertices would not change. Therefore, their result of minutely perturbing the geometry is effectively similar to that of perturbing textures [5]. In contrast, we performed 3D rotations and 3D translations to move an object inside a 3D space (*i.e.*, the viewing frustum of the camera).

Second, Engstrom et al. [12] showed how simple 2D image rotations and translations can cause DNNs to misclassify. However, these 2D transformations still do not reveal the type of adversarial poses discovered by rotating 3D objects (*e.g.*, a flipped-over school bus; Fig. 1d).

To the best of our knowledge, our work is the first attempt to harness 3D objects to study the OoD poses of well-known training-set objects that cause state-of-the-art ImageNet classifiers and MS COCO detectors to misclassify.

6. Discussion and conclusion

In this paper, we revealed how DNNs' understanding of objects like "school bus" and "fire truck" is quite naive—they can correctly label only a small subset of the entire pose space for 3D objects. Note that we can also find real-world OoD poses by simply taking photos of real objects (Sec. S3). We believe classifying an arbitrary pose into one of the object classes is an ill-posed task, and that the adversarial pose problem might be alleviated via multiple orthogonal approaches. The first is addressing biased data [49]. Because ImageNet and MS COCO datasets are constructed from photographs taken by people, the datasets reflect the aesthetic tendencies of their captors. Such biases can be somewhat alleviated through data augmentation, specifically, by harnessing images generated from 3D renderers [39, 4]. From the modeling view, we believe DNNs would benefit from the incorporation of 3D information, *e.g.*, [4].

Finally, our work introduced a new promising method (Fig. 2) for testing computer vision DNNs by harnessing 3D renderers and 3D models. While we only optimize a single object here, the framework could be extended to jointly optimize lighting, background image, and multiple objects, all in one "adversarial world". Not only does our framework enable us to enumerate test cases for DNNs, but it also serves as an interpretability tool for extracting useful insights about these black-box models' inner functions.

Acknowledgements

We thank Hiroharu Kato and Nikos Kolotouros for their valuable discussions and help with the differentiable renderer. We also thank Rodrigo Sardinas for his help with some GPU servers used in the project. AN is supported by multiple funds from Auburn University, a donation from Adobe Inc., and computing credits from Amazon AWS.

References

[1] Moderngl — moderngl 5.4.1 documentation. https://moderngl.readthedocs.io/en/stable/index.html. (Accessed on 11/14/2018).

[2] The self-driving uber that killed a pedestrian didn't brake. here's why. https://slate.com/technology/2018/05/uber-car-in-fatal-arizona-crash-perceived-pedestrian-1-3-seconds-before-impact.html. (Accessed on 07/13/2018).

[3] Tesla car on autopilot crashes, killing driver, united states news & top stories - the straits times. https://www.straitstimes.com/world/united-states/tesla-car-on-autopilot-crashes-killing-driver. (Accessed on 06/14/2018).

[4] H. A. Alhaija, S. K. Mustikovela, A. Geiger, and C. Rother. Geometric image synthesis. *arXiv preprint arXiv:1809.04696*, 2018.

[5] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok. Synthesizing robust adversarial examples. In *2018 Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 284–293, 2018.

[6] A. Azulay and Y. Weiss. Why do deep convolutional networks generalize so poorly to small image transformations? *arXiv preprint arXiv:1805.12177*, 2018.

[7] A. Borji, S. Izadi, and L. Itti. ilab-20m: A large-scale controlled object dataset to investigate deep learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2221–2230, 2016.

[8] N. Carlini and D. Wagner. Towards Evaluating the Robustness of Neural Networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, 2017.

[9] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009.

[10] C. Chen, A. Seff, A. Kornhauser, and J. Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2722–2730, 2015.

[11] B. B. Choi and C. Lawrence. Inverse Kinematics Problem in Robotics Using Neural Networks. *NASA Technical Memorandum*, 105869:1–23, 1992.

[12] L. Engstrom, B. Tran, D. Tsipras, L. Schmidt, and A. Madry. A rotation and a translation suffice: Fooling CNNs with simple transformations, 2019.

[13] I. Evtimov, K. Eykholt, E. Fernandes, T. Kohno, B. Li, A. Prakash, A. Rahmati, and D. Song. Robust physical-world attacks on machine learning models. *arXiv preprint arXiv:1707.08945*, 2017.

[14] D. Gandhi, L. Pinto, and A. Gupta. Learning to fly by crashing. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 3948–3955. IEEE, 2017.

[15] I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

[16] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[17] D. Hendrycks and K. Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *Proceedings of International Conference on Learning Representations*, 2017.

[18] D. Karmon, D. Zoran, and Y. Goldberg. Lavan: Localized and visible adversarial noise. *arXiv preprint arXiv:1801.02608*, 2018.

[19] H. Kato, Y. Ushiku, and T. Harada. Neural 3D Mesh Renderer. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems (NIPS 2012)*, pages 1097–1105, 2012.

[21] F. Lambert. Understanding the fatal tesla accident on autopilot and the nhtsa probe. *Electrek, July*, 2016.

[22] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[23] H.-T. D. Liu, M. Tao, C.-L. Li, D. Nowrouzezahrai, and A. Jacobson. Adversarial Geometry and Lighting using a Differentiable Renderer. *arXiv preprint*, 8 2018.

[24] J. Lu, H. Sibai, E. Fabry, and D. Forsyth. NO Need to Worry about Adversarial Examples in Object Detection in Autonomous Vehicles. *arXiv preprint*, 7 2017.

[25] J. Lu, H. Sibai, E. Fabry, and D. A. Forsyth. Standard detectors aren't (currently) fooled by physical adversarial stop signs. *CoRR*, abs/1710.03337, 2017.

[26] Y. Luo, X. Boix, G. Roig, T. Poggio, and Q. Zhao. Foveation-based Mechanisms Alleviate Adversarial Examples. *arXiv preprint*, 11 2015.

[27] S. Marschner and P. Shirley. *Fundamentals of computer graphics*. CRC Press, 2015.

[28] A. Nguyen, J. Clune, Y. Bengio, A. Dosovitskiy, and J. Yosinski. Plug & play generative networks: Conditional iterative generation of images in latent space. In *CVPR*, volume 2, page 7, 2017.

[29] A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 427–436, 2015.

[30] M. Osadchy, M. L. Miller, and Y. LeCun. Synergistic Face Detection and Pose Estimation with Energy-Based Models. In *Advances in Neural Information Processing Systems*, pages 1017–1024, 2005.

[31] M. Ozuysal, V. Lepetit, and P. Fua. Pose estimation for category specific multiview object localization. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 778–785. IEEE, 2009.

[32] N. Pinto, J. J. DiCarlo, and D. D. Cox. How far can you get with a modern face recognition test set using only simple

features? In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2591–2598. IEEE, 2009.

[33] PyTorch. torchvision.models — pytorch master documentation. https://pytorch.org/docs/stable/torchvision/models.html. (Accessed on 11/14/2018).

[34] J. Redmon and A. Farhadi. YOLOv3: An Incremental Improvement. 2018.

[35] A. Rosenfeld, R. Zemel, and J. K. Tsotsos. The elephant in the room. *arXiv preprint arXiv:1808.03305*, 2018.

[36] 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. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[37] C. Sampedro, A. Rodriguez-Ramos, H. Bavle, A. Carrio, P. de la Puente, and P. Campoy. A fully-autonomous aerial robot for search and rescue applications in indoor environments using learning-based techniques. *Journal of Intelligent & Robotic Systems*, pages 1–27, 2018.

[38] W. J. Scheirer, A. de Rezende Rocha, A. Sapkota, and T. E. Boult. Toward open set recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(7):1757–1772, 2013.

[39] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb. Learning from simulated and unsupervised images through adversarial training. In *CVPR*, volume 2, page 5, 2017.

[40] J. Su, D. V. Vargas, and S. Kouichi. One Pixel Attack for Fooling Deep Neural Networks. *arXiv preprint*, 2017.

[41] F. P. Such, V. Madhavan, E. Conti, J. Lehman, K. O. Stanley, and J. Clune. Deep neuroevolution: genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *arXiv preprint arXiv:1712.06567*, 2017.

[42] M. Sugiyama, N. D. Lawrence, A. Schwaighofer, et al. *Dataset shift in machine learning*. The MIT Press, 2017.

[43] X. Sun, J. Wu, X. Zhang, Z. Zhang, C. Zhang, T. Xue, J. B. Tenenbaum, and W. T. Freeman. Pix3d: Dataset and methods for single-image 3d shape modeling. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[44] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[45] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the Inception Architecture for Computer Vision. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 12 2016.

[46] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

[47] G. R. Taylor, A. J. Chosak, and P. C. Brewer. Ovvv: Using virtual worlds to design and evaluate surveillance systems. In *2007 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2007.

[48] Y. Tian, K. Pei, S. Jana, and B. Ray. Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In *Proceedings of the 40th International Conference on Software Engineering*, pages 303–314. ACM, 2018.

[49] A. Torralba and A. A. Efros. Unbiased look at dataset bias. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1521–1528. IEEE, 2011.

[50] Y. Z. S. Q. Z. X. T. S. K. Y. W. A. Y. Weichao Qiu, Fangwei Zhong. Unrealcv: Virtual worlds for computer vision. *ACM Multimedia Open Source Software Competition*, 2017.

[51] M. Woo, J. Neider, T. Davis, and D. Shreiner. *OpenGL programming guide: the official guide to learning OpenGL, version 1.2*. Addison-Wesley Longman Publishing Co., Inc., 1999.

[52] Y. Xiang, W. Kim, W. Chen, J. Ji, C. Choy, H. Su, R. Mottaghi, L. Guibas, and S. Savarese. Objectnet3d: A large scale database for 3d object recognition. In *European Conference on Computer Vision*, pages 160–176. Springer, 2016.

[53] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017.

[54] C. Xiao, J.-Y. Zhu, B. Li, W. He, M. Liu, and D. Song. Spatially transformed adversarial examples. In *International Conference on Learning Representations*, 2018.

[55] X. Yuan, P. He, Q. Zhu, and X. Li. Adversarial Examples: Attacks and Defenses for Deep Learning. *arXiv preprint*, 2017.

[56] R. Zhang. Making convolutional networks shift-invariant again, 2019.

[57] Z. Zhu, L. Xie, and A. L. Yuille. Object recognition with and without objects. *arXiv preprint arXiv:1611.06596*, 2016.

Supplementary materials for:
Strike (with) a Pose: Neural Networks Are Easily Fooled by Strange Poses of Familiar Objects

S1. Extended description of the 3D object dataset and its evaluation

S1.1. Dataset construction

Classes. Our main dataset consists of 30 unique 3D object models corresponding to 30 ImageNet classes relevant to a traffic environment. The 30 classes include 20 vehicles (*e.g.*, `school bus` and `cab`) and 10 street-related items (*e.g.*, `traffic light`). See Fig. S2 for example renders of each object.

Acquisition. We collected 3D objects and constructed our own datasets for the study. 3D models with high-quality image textures were purchased from `turbosquid.com`, `free3d.com`, and `cgtrader.com`.

To make sure the renders were as close to real ImageNet photos as possible, we used only 3D models that had high-quality 2D image textures. We did not choose 3D models from public datasets, *e.g.*, ObjectNet3D [52], because most of them do not have high-quality image textures. While the renders of such models may be correctly classified by DNNs, we excluded them from our study because of their poor realism. We also examined the ImageNet images to ensure they contained real-world examples qualitatively similar to each 3D object in our 3D dataset.

3D objects. Each 3D object is represented as a mesh, *i.e.*, a list of triangular faces, each defined by three vertices [27]. The 30 meshes have on average $9,908$ triangles (see Table S1 for specific numbers).

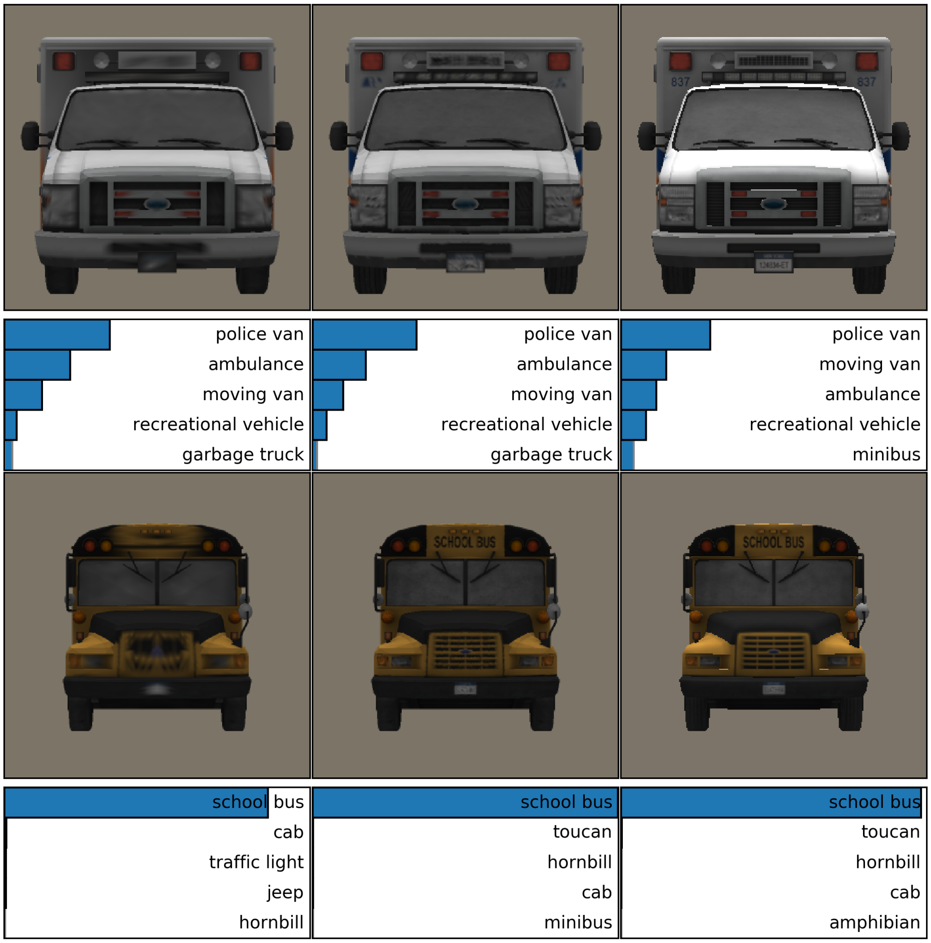
3D object	Tessellated N_T	Original N_O	3D object	Tessellated N_T	Original N_O
`ambulance`	70,228	5,348	`motor scooter`	96,638	2,356
`backpack`	48,251	1,689	`moving van`	83,712	5,055
`bald eagle`	63,212	2,950	`park bench`	134,162	1,972
`beach wagon`	220,956	2,024	`parking meter`	37,246	1,086
`cab`	53,776	4,743	`pickup`	191,580	2,058
`cellphone`	59,910	502	`police van`	243,132	1,984
`fire engine`	93,105	8,996	`recreational vehicle`	191,532	1,870
`forklift`	130,455	5,223	`school bus`	229,584	6,244
`garbage truck`	97,482	5,778	`sports car`	194,406	2,406
`German shepherd`	88,496	88,496	`street sign`	17,458	17,458
`golf cart`	98,007	5,153	`tiger cat`	107,431	3,954
`jean`	17,920	17,920	`tow truck`	221,272	5,764
`jeep`	191,144	2,282	`traffic light`	392,001	13,840
`minibus`	193,772	1,910	`trailer truck`	526,002	5,224
`minivan`	271,178	1,548	`umbrella`	71,410	71,410

Table S1: The triangle number for the 30 objects used in our study. N_O shows the number of triangles for the original 3D objects, and N_T shows the same number after tessellation. Across 30 objects, the average triangle count increases $\sim$ 15x from $\overline{N_O} = 9,908$ to $\overline{N_T} = 147,849$.

S1.2. Manual object tessellation for experiments using the Differentiable Renderer

In contrast to ModernGL [1]—the non-differentiable renderer (NR) in our paper—the differentiable renderer (DR) by Kato et. al [19] does not perform tessellation, a standard process to increase the resolution of renders. Therefore, the render quality of the DR is lower than that of the NR. To minimize this gap and make results from the NR more comparable with those from the DR, we manually tessellated each 3D object as a pre-processing step for rendering with the DR. Using the manually tessellated objects, we then (1) evaluated the render quality of the DR (Sec. S1.3); and (2) performed research experiments with the DR (*i.e.*, the DR-G method in Sec. 4.4).

Tessellation. We used the *Quadify Mesh Modifier* feature (quad size of 2%) in 3ds Max 2018 to tessellate objects, increasing the average number of faces $\sim$15x from $9,908$ to $147,849$ (see Table S1). The render quality after tessellation is sharper and of a higher resolution (see Fig. S1a vs. b). Note that the NR pipeline already performs tessellation for every input 3D object. Therefore, we did not perform manual tessellation for 3D objects rendered by the NR.



(a) DR without tessellation (b) DR with tessellation (c) NR with tessellation

Figure S1: A comparison of 3D object renders (here, `ambulance` and `school bus`) before and after tessellation.
(a) Original 3D models rendered by the differentiable renderer (DR) [19] without tessellation.
(b) DR renderings of the same objects after manual tessellation.
(c) The non-differentiable renderer (NR), *i.e.*, ModernGL [1], renderings of the original objects.
After manual tessellation, the render quality of the DR appears to be sharper (a vs. b) and closely matches that of the NR, which also internally tessellates objects (b vs. c).

S1.3. Evaluation

We recognize that a reality gap will often exist between a render and a real photo. Therefore, we rigorously evaluated our renders to make sure the reality gap was acceptable for our study. From $\sim$100 initially-purchased 3D object models, we selected the 30 highest-quality objects that both (1) passed a visual human Turing test; and (2) were correctly recognized with high confidence by the Inception-v3 classifier [44].

S1.3.1 Qualitative evaluation

Here, we attempt to provide a qualitative "apples to apples" comparison between renders of our high-quality 3D objects and photos of their real-world counterparts by generating (real photo, render) pairs. The entire process follows the standard pose-annotation procedure (*e.g.*, for the Pix3D [43] or YCB-Video [53] datasets) and is described below:

1. We retrieved ∼3 real photos for each 3D object (*e.g.*, a car) from the Internet using descriptive information (*e.g.*, a car's make, model, and year).

2. For each real photo, we replaced the object with matching background content via Adobe Photoshop's Context-Aware Fill-In feature to obtain a background-only (*i.e.*, no foreground objects) photo B.

3. We then rendered the 3D object on the background B obtained in Step 2 and manually aligned the pose of the 3D object so that it closely matched the reference photo.

4. Finally, we evaluated the (photo, render) pairs in a side-by-side comparison.

In total, we generated 116 (photo, render) pairs for two sets of 3D objects:

- Set 1: The 30 objects used as the 3D dataset in our main experiments (see Fig. S3). All 30 objects × 2 pairs = 60 pairs are provided at https://drive.google.com/drive/folders/1ti9zo1dzU1e9b-mpqv0bhTrMeoeUBULm. The pose alignment was done in our GUI tool[3]. The scenes were rendered via the NR (*i.e.*, ModernGL).

- Set 2: 17 objects gathered separately from Set 1 only for evaluation. We collected the 17 extra objects because we were able to find Internet photos of their exact real-world counterparts (*e.g.*, photos of the 2014 Mercedes-Benz E-Klasse Coupe). These 17 objects are of the same high quality as the 30 main objects. The pose alignment was done in Blender, and the scenes were rendered with the DR. All 56 pairs generated from these 17 objects are provided at https://goo.gl/8z42zL.

While discrepancies can be visually spotted in our side-by-side comparisons, we found most of the renders passed our human visual Turing test if presented alone. That is, it is not easy for humans to tell whether a render is a real photo or not (if they are not primed with the reference photos).

S1.3.2 Quantitative evaluation

In addition to the qualitative evaluation, we also quantitatively evaluated the Google Inception-v3 [44]'s top-1 accuracy on renders that use either (a) an empty background or (b) real background images.

a. Evaluation of the renders of 30 objects on an empty background

Because the experiments in the main text used our self-assembled 30-object dataset (Sec. S1.1), we describe the process and the results of our quantitative evaluation for only those objects.

We rendered the objects on a white background with RGB values of (1.0, 1.0, 1.0), an ambient light intensity of 0.9, and a directional light intensity of 0.5. For each object, we sampled 36 unique views (common in ImageNet) evenly divided into three sets. For each set, we set the object at the origin, the up direction to $(0, 1, 0)$, and the camera position to $(0, 0, -z)$ where $z = \{4, 6, 8\}$. We sampled 12 views per set by starting the object at a 10° yaw and generating a render at every 30° yaw-rotation. Across all objects and all renders, the Inception-v3 top-1 accuracy is 83.23% (comparable to 77.45% on ImageNet images [44]) with a mean top-1 confidence score of 0.78. The top-1 and top-5 average accuracy and confidence scores are shown in Table S2.

[3] https://github.com/airalcorn2/strike-with-a-pose

Distance	4	6	8	Average
top-1 mean accuracy	84.2%	84.4%	81.1%	83.2%
top-5 mean accuracy	95.3%	98.6%	96.7%	96.9%
top-1 mean confidence score	0.77	0.80	0.76	0.78

Table S2: The top-1 and top-5 average accuracy and confidence scores for Inception-v3 [44] on the renders of the 30 objects in our dataset.

b. Evaluation of the renders of test objects on real backgrounds

In addition to our qualitative side-by-side (real photo, render) comparisons (Fig. S3), we quantitatively compared Inception-v3's predictions for our renders to those for real photos. We found a high similarity between real photos and renders for DNN predictions. That is, across all 56 pairs (Sec. S1.3.1), the top-1 predictions match 71.43% of the time. Across all pairs, 76.06% of the top-5 labels for real photos match those for renders.

S2. Transferability from the Inception-v3 classifier to the YOLO-v3 detector

Previous research has shown that object detectors can be more robust to adversarial attacks than image classifiers [25]. Here, we investigate how well our AXs generated for an Inception-v3 classifier trained to perform 1,000-way image classification on ImageNet [36] transfer to YOLO-v3, a state-of-the-art object detector trained on MS COCO [22].

Note that while ImageNet has 1,000 classes, MS COCO has bounding boxes classified into only 80 classes. Therefore, among 30 objects, we only selected the 13 objects that (1) belong to classes found in both the ImageNet and MS COCO datasets; and (2) are also well recognized by the YOLO-v3 detector in common poses.

S2.1. Class mappings from ImageNet to MS COCO

See Table S3a for 13 mappings from ImageNet labels to MS COCO labels.

S2.2. Selecting 13 objects for the transferability test

For the transferability test (Sec. S2.3), we identified the 13 objects (out of 30) that are well detected by the YOLO-v3 detector via the two tests described below.

S2.2.1 YOLO-v3 correctly classifies 93.80% of poses generated via yaw-rotation

We rendered 36 unique views for each object by generating a render at every 30° yaw-rotation (see Sec. S1.3.2). Note that, across all objects, these yaw-rotation views have an average accuracy of 83.2% by the Inception-v3 classifier. We tested them against YOLO-v3 to see whether the detector was able to correctly find one single object per image and label it correctly. Among 30 objects, we removed those that YOLO-v3 had an accuracy $\leq 70\%$, leaving 13 for the transferability test. Across the remaining 13 objects, YOLO-v3 has an accuracy of 93.80% on average (with an NMS threshold of 0.4 and a confidence threshold of 0.5). Note that the accuracy was computed as the total number of correct labels over the total number of bounding boxes detected (*i.e.*, we did not measure bounding-box IoU errors). See class-specific statistics in Table S3. This result shows that YOLO-v3 is substantially more accurate than Inception-v3 on the standard object poses generated by yaw-rotation (93.80% vs. 83.2%).

S2.2.2 YOLO-v3 correctly classifies 81.03% of poses correctly classified by Inception-v3

Additionally, as a sanity check, we tested whether poses *correctly classified* by Inception-v3 transfer well to YOLO-v3. For each object, we randomly selected 30 poses that were 100% correctly classified by Inception-v3 with high confidence ($p \geq 0.9$). The images were generated via the random search procedure in the main text experiment (Sec. 3.2). Across the final 13 objects, YOLO-v3 was able to correctly detect one single object per image and label it correctly at a 81.03% accuracy (see Table S3c).

	(a) Label mapping		(b) Accuracy on yaw-rotation poses		(c) Accuracy on random poses		(d) Accuracy on adversarial poses		
	ImageNet	MS COCO	#/36	acc (%)	#/30	acc (%)	#/1350	acc (%)	Δacc (%)
1	park bench	bench	31	86.11	22	73.33	211	15.63	57.70
2	bald eagle	bird	34	94.11	24	80.00	597	44.22	35.78
3	school bus	bus	36	100.00	18	60.00	4	0.30	69.70
4	beach wagon	car	34	94.44	30	100.00	232	17.19	82.81
5	tiger cat	cat	26	72.22	25	83.33	181	13.41	69.93
6	German shepherd	dog	32	88.89	28	93.33	406	30.07	63.26
7	motor scooter	motorcycle	36	100.00	18	60.00	384	28.44	31.56
8	jean	person	36	100.00	29	96.67	943	69.85	26.81
9	street sign	stop sign	31	86.11	26	86.67	338	25.04	61.15
10	moving van	truck	36	100.00	24	80.00	15	1.11	78.89
11	umbrella	umbrella	35	97.22	25	83.33	907	67.19	16.15
12	police van	car	36	100.00	25	83.33	55	4.07	79.26
13	trailer truck	truck	36	100.00	22	73.33	26	1.93	71.41
		Average		93.80		81.03		24.50	56.53

Table S3: Adversarial poses generated for a state-of-the-art ImageNet image classifier (here, Inception-v3) transfer well to an MS COCO detector (here, YOLO-v3). The table shows the YOLO-v3 detector's accuracy on: (b) object poses generated by a standard process of yaw-rotating the object; (c) random poses that are 100% correctly classified by Inception-v3 with high confidence ($p \geq 0.9$); and (d) adversarial poses, *i.e.*, 100% misclassified by Inception-v3.

(a) The mappings of 13 ImageNet classes onto 12 MS COCO classes.
(b) The accuracy ("acc (%)") of the YOLO-v3 detector on 36 yaw-rotation poses per object.
(c) The accuracy of YOLO-v3 on 30 random poses per object that were correctly classified by Inception-v3.
(d) The accuracy of YOLO-v3 on 1,350 adversarial poses ("acc (%)") and the differences between c and d ("Δacc (%)").

S2.3. Transferability test: YOLO-v3 fails on 75.5% of adversarial poses misclassified by Inception-v3

For each object, we collected 1,350 random adversarial poses (*i.e.*, incorrectly classified by Inception-v3) generated via the random search procedure (Sec. 3.2). Across all 13 objects and all adversarial poses, YOLO-v3 obtained an accuracy of only 24.50% (compared to 81.03% when tested on images correctly classified by Inception-v3). In other words, 75.5% of adversarial poses generated for Inception-v3 also escaped the detection[4] of YOLO-v3 (see Table S3d for class-specific statistics). Our result shows adversarial poses transfer well across tasks (image classification → object detection), models (Inception-v3 → YOLO-v3), and datasets (ImageNet → MS COCO).

S3. Adversarial poses do exist in the real world

Our main experiments showed that adversarial poses exist in 3D simulation. Here, we provide evidence that adversarial poses also transfer to and exist in the real world.

First, we collected 5 photos × 30 objects = 150 photos from the Internet that were *misclassified* by the Inception-v3 classifier and repeated the same experiment as described in Sec. S1.3.1 to produce (real photo, render) pairs (see Fig. S20). We found that when the real photos appear out-of-distribution, 98.3% of the renders are also misclassified. However, when the real failure photos appear ImageNet-like, ∼45% of the renders are correctly classified (*i.e.*, our 3D objects are *easier* to recognize than their real-world counterparts). This transferability result confirms the high realism of our 3D object renders and suggests that the adversarial poses do exist in the real world.

Second, we found that real-world, high-confidence adversarial poses can be found by simply taking photos from strange angles of a familiar object. We took real-world videos of four example objects (`cellular phone`, `jeans`, `street sign`,

[4] We were not able to check how many misclassification labels by YOLO-v3 were the same as those by Inception-v3 because only a small set of 80 the MS COCO classes overlap with the 1,000 ImageNet classes.

and umbrella) and extracted the misclassified frames from the videos. While Inception-v3 [44] correctly recognized these objects in canonical poses, the model misclassified the same objects in unusual poses (Fig. S17).

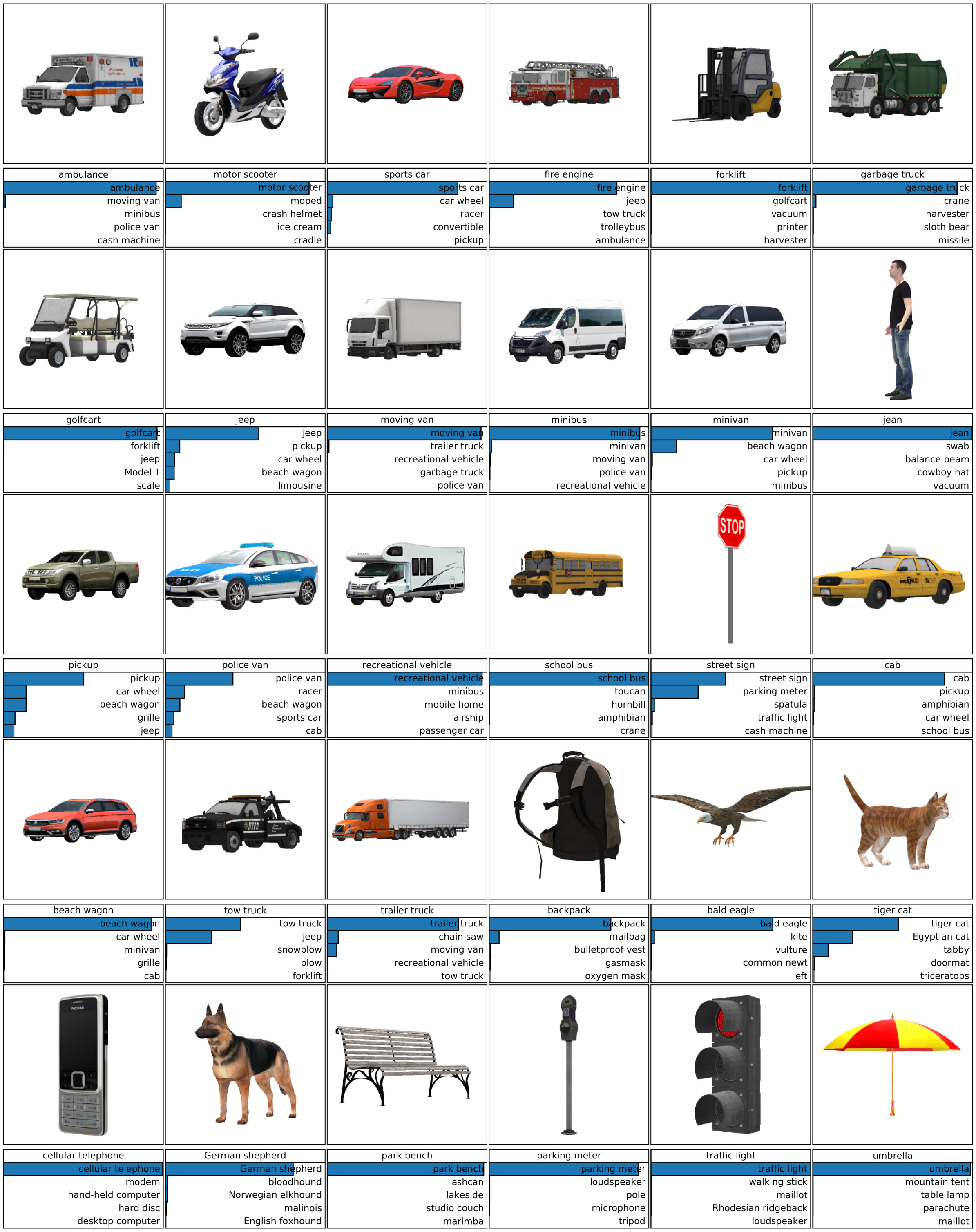


Figure S2: We tested Inception-v3's predictions on the renders generated by the differentiable renderer (DR). We show here the top-5 predictions for one random pose per object. However, in total, we generated 36 poses for each object by (1) varying the object distance to the camera; and (2) rotating the object around the yaw axis. See https://goo.gl/7LG3Cy for all the renders and DNN top-5 predictions. Across all 30 objects, on average, Inception-v3 correctly recognizes 83.2% of the renders. See Sec. S1.3.2 for more details.

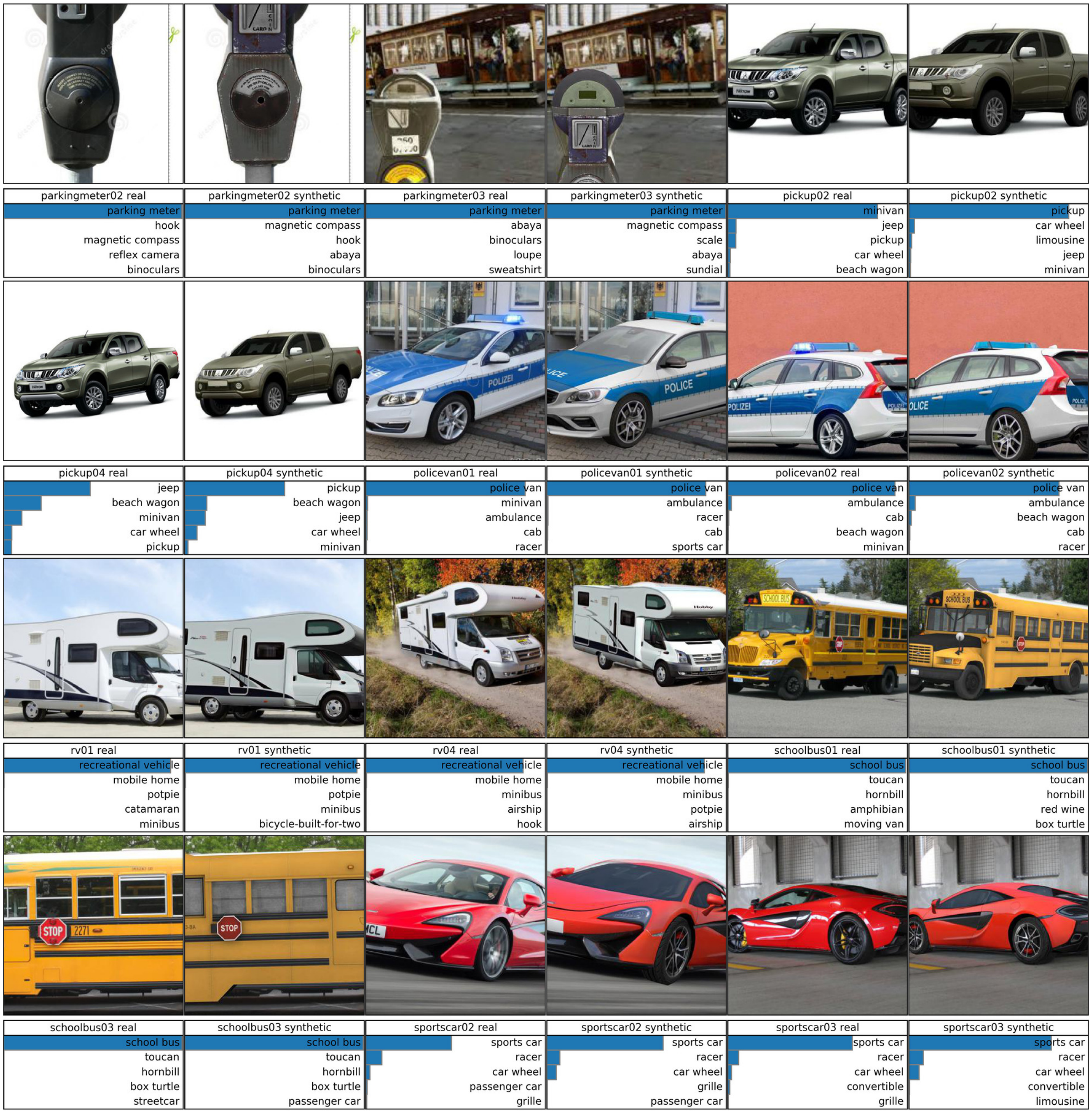


Figure S3: 12 random pairs of real photos (left) and renders (right) among 116 pairs produced in total for our 3D object rendering evaluation (Sec. S1.3.1). The renders are produced by ModernGL. More comparison images are available at `https://drive.google.com/drive/folders/1ti9zo1dzU1e9b-mpqv0bhTrMeoeUBULm`. While discrepancies can be spotted in our side-by-side comparisons, we found that most of the renders passed our human visual Turing test if presented alone.

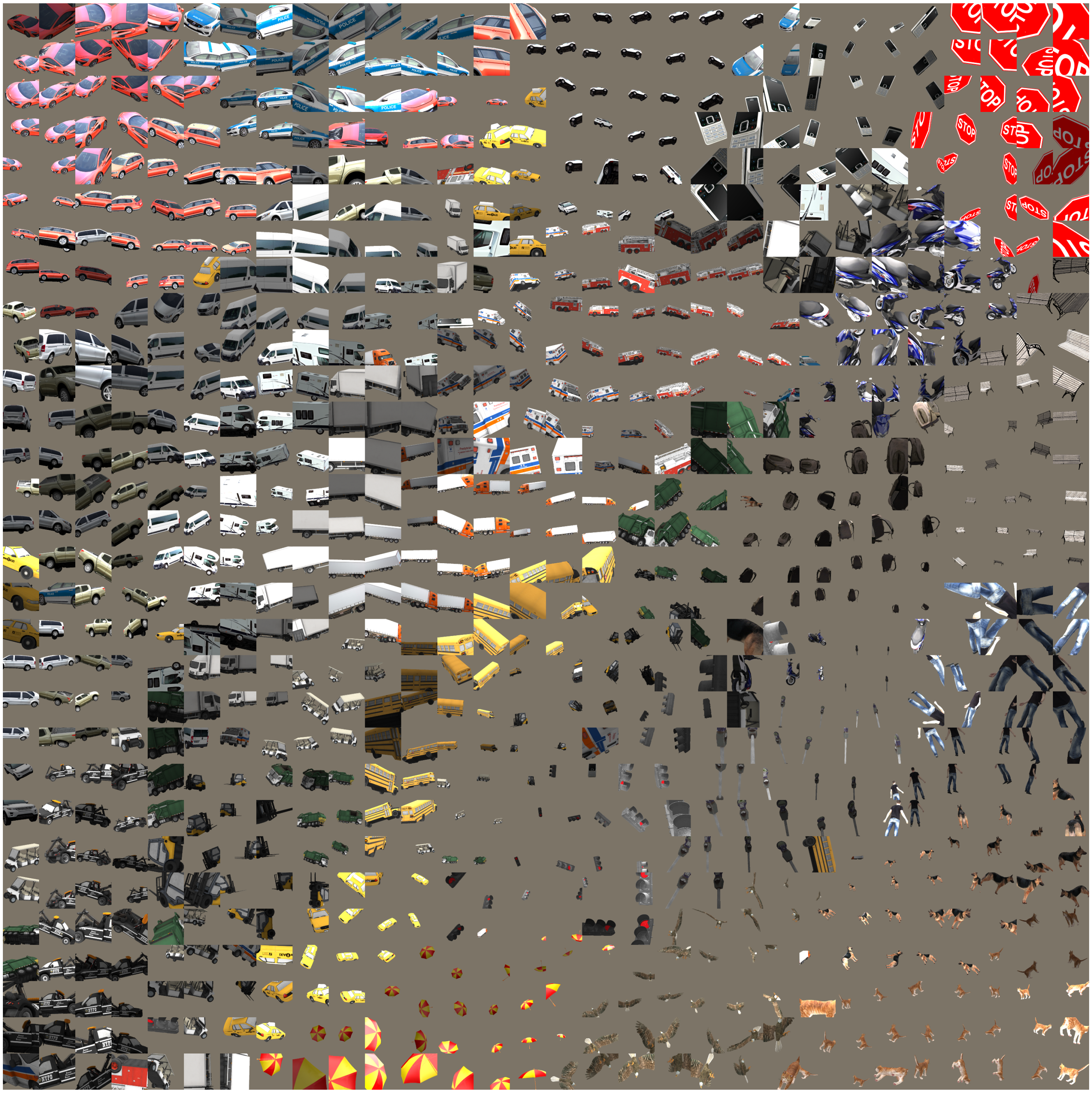


Figure S4: For each object, we collected 30 high-confidence ($p \geq 0.9$) correctly classified images by Inception-v3. The images were generated via the random search procedure. We show here a grid t-SNE of AlexNet [20] fc7 features for all 30 objects × 30 images = 900 images. Correctly classified images for each object tend to be similar and clustered together. The original, high-resolution figure is available at `https://goo.gl/TGgPgB`.

To better visualize the clusters, we plotted the same t-SNE but used unique colors to denote the different 3D objects in the renders (Fig. S5). Compare and contrast this plot with the t-SNE of only misclassified poses (Figs. S6 & S7).

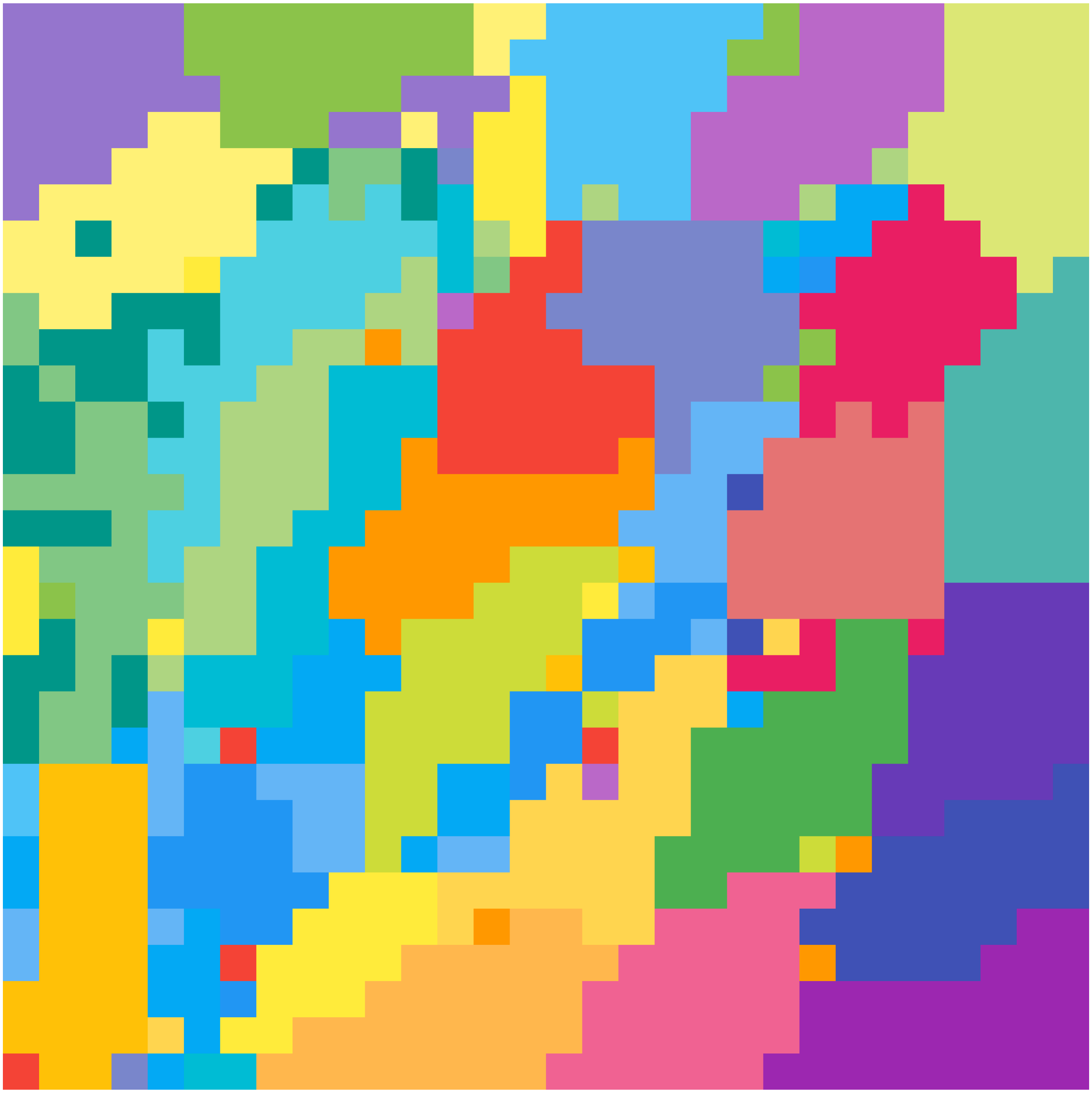


Figure S5: The same t-SNE found in Fig. S4 but using a unique color to denote the 3D object found in each rendered image. Here, each color also corresponds to a unique Inception-v3 label. Compare and contrast this plot with the t-SNE of only misclassified poses (Fig. S7). The original, high-resolution figure is available at https://goo.gl/TGgPgB.

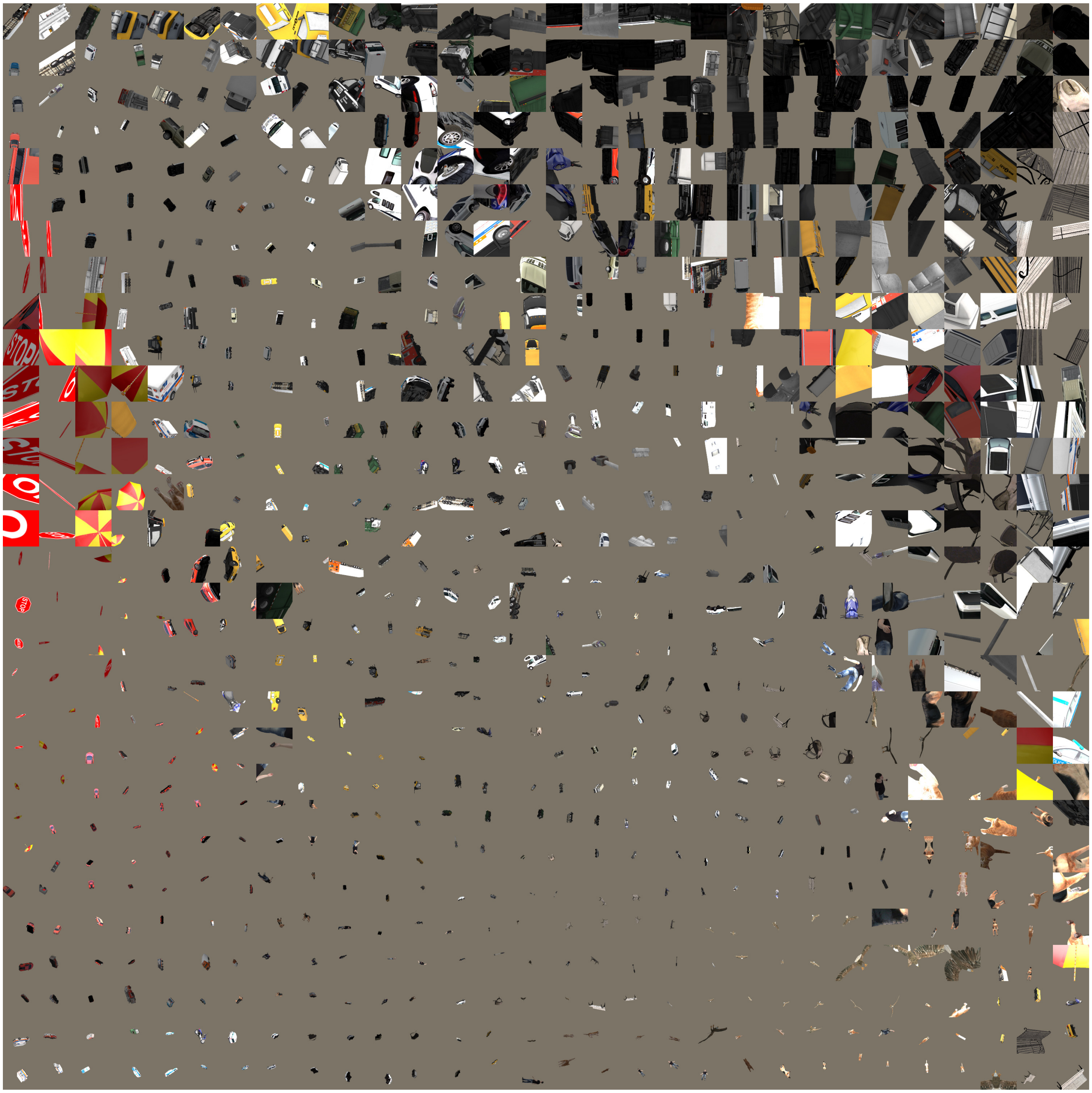


Figure S6: Following the same process as described in Fig. S4, we show here a grid t-SNE of generated adversarial poses. For each object, we assembled 30 high-confidence ($p \geq 0.9$) adversarial examples generated via a random search against Inception-v3 [44]. The t-SNE was generated from the AlexNet [20] fc7 features for 30 objects × 30 images = 900 images. The original, high-resolution figure is available at `https://goo.gl/TGgPgB`. Adversarial poses were found to be both common across different objects (*e.g.*, the top-right corner) and unique to specific objects (*e.g.*, the `traffic sign` and `umbrella` objects in the middle left).

To better understand how similar misclassified poses can be found across many objects, see Fig. S7. Compare and contrast this plot with the t-SNE of correctly classified poses (Figs. S4 & S5).

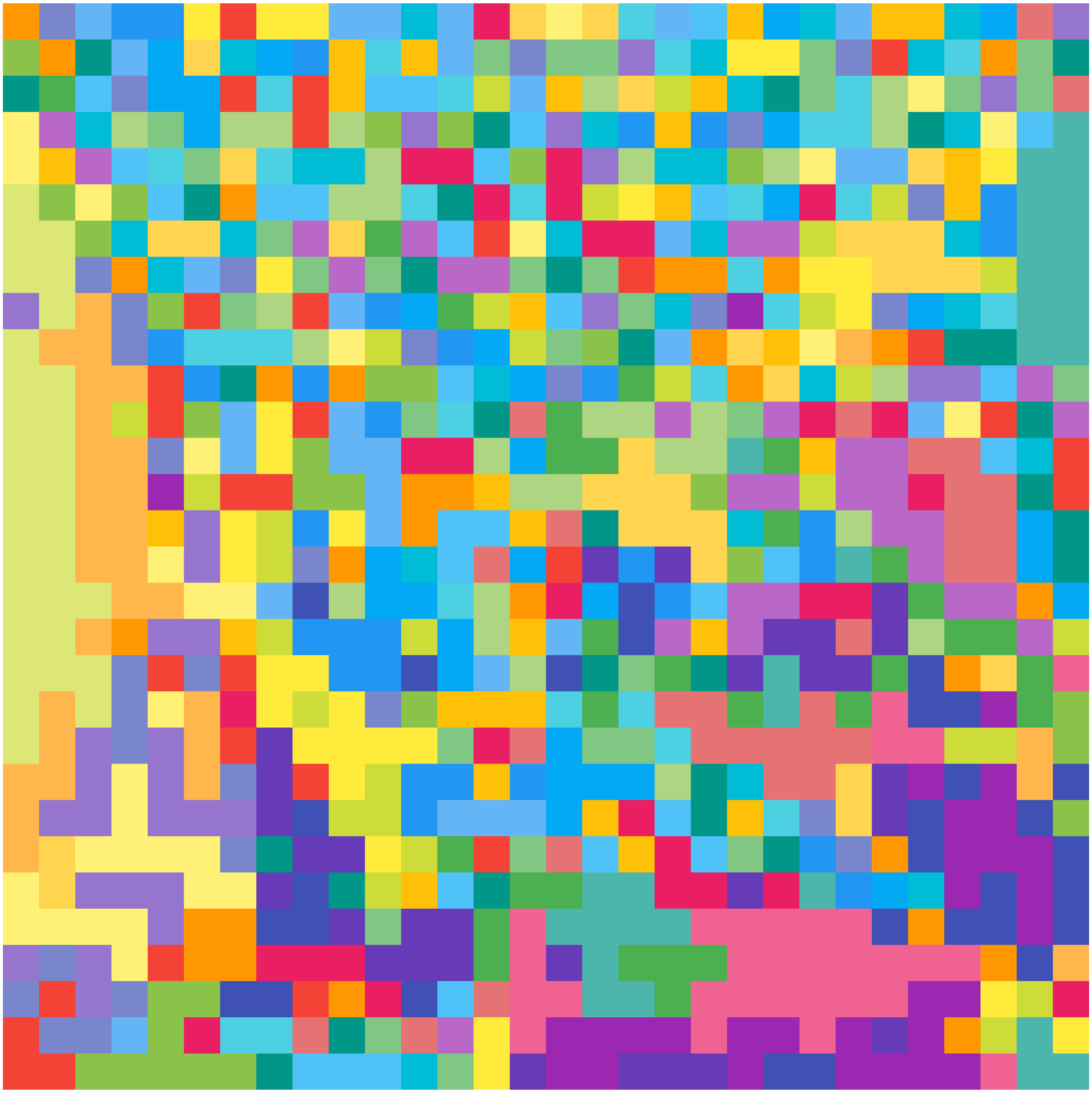
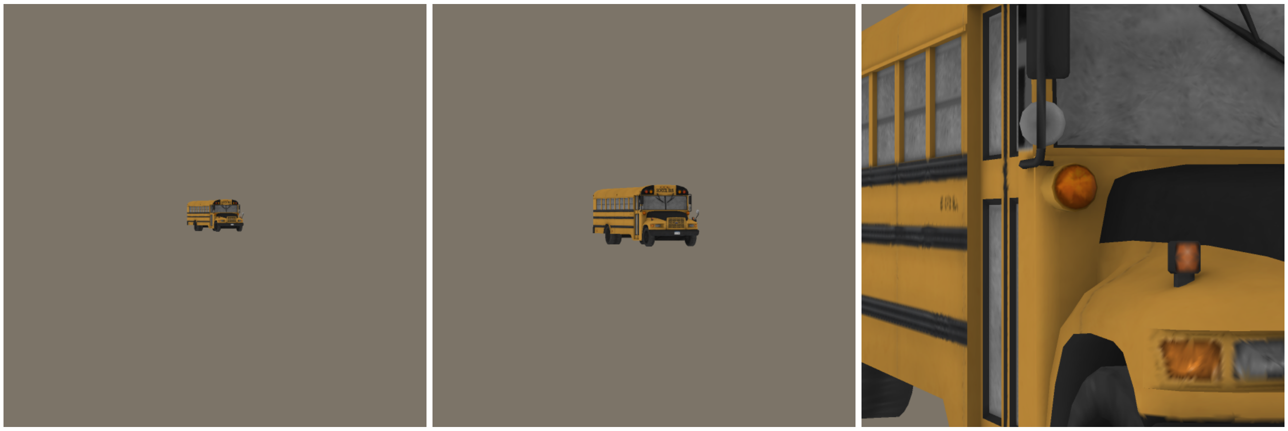


Figure S7: The same t-SNE as that in Fig. S6 but using a unique color to denote the 3D object used to render the adversarial image (*i.e.*, Inception-v3's misclassification labels are not shown here). The original, high-resolution figure is available at `https://goo.gl/TGgPgB`.
Compare and contrast this plot with the t-SNE of correctly classified poses (Fig. S5).

S4. Experimental setup for the differentiable renderer

For the gradient descent method (DR-G) that uses the approximate gradients provided by the differentiable renderer [19] (DR), we set up the rendering parameters in the DR to closely match those in the NR. However, there were still subtle discrepancies between the DR and the NR that made the results (DR-G vs. FD-G in Sec. 4.4) not directly comparable. Despite these discrepancies (described below), we still believe the FD gradients are more stable and informative than the DR gradients (*i.e.*, FD-G outperformed DR-G).[5]

DR setup. For all experiments with the DR, the camera was centered at $(0, 0, 16)$ with an up direction $(0, 1, 0)$. The object's spatial location was constrained such that the object center was always within the frame. The depth values were constrained to be within $[-14, 14]$. Similar to experiments with the NR, we used the medium lighting setting. The ambient light color was set to white with an intensity 1.0, while the directional light was set to white with an intensity 0.4. Fig. S8 shows an example school bus rendered under this medium lighting at different distances.



(a) School bus at $(0, 0, -14)$

(b) School bus at $(0, 0, 0)$

(c) School bus at $(0, 0, 14)$

Figure S8: School bus rendered by the DR at different distances.

The known discrepancies between the experimental setups of FD-G (with the NR) vs. DR-G (with the DR) are:

1. The exact medium lighting parameters for the NR described in the main text (Sec. 4.1) did not produce similar lighting effects in the DR. Therefore, the DR lighting parameters described above were the result of manually tuning to qualitatively match the effect produced by the NR medium lighting parameters.

2. While the NR uses a built-in tessellation procedure that automatically tessellates input objects before rendering, we had to perform an extra pre-processing step of manually tessellating each object for the DR. While small, a discrepancy still exists between the two rendering results (Fig. S1b vs. c).

S5. Gradient descent with the DR gradients

In preliminary experiments (data not shown), we found the DR gradients to be relatively noisy when using gradient descent to find targeted adversarial poses (*i.e.*, DR-G experiments). To mitigate this problem, we experimented with (1) parameter augmentation (Sec. S5.1); and (2) multi-view optimization (Sec. S5.2). In short, we found parameter augmentation helped and used it in DR-G. However, when using the DR, we did not find multiple cameras improved optimization performance and thus only performed regular single-view optimization for DR-G.

S5.1. Parameter augmentation

We performed gradient descent using the DR gradients (DR-G) in an augmented parameter space corresponding to 50 rotations and one translation to be applied to the original object vertices. That is, we backpropagated the DR gradients into

[5] In preliminary experiments with only the DR (not the NR), we also empirically found FD-G to be more stable and effective than DR-G (data not shown).

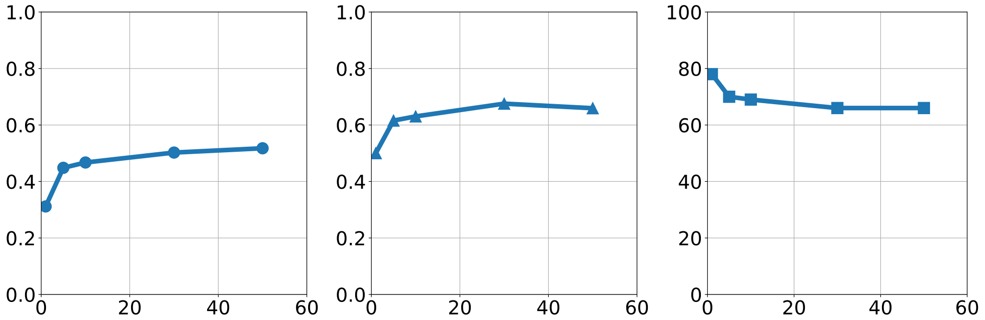
the parameters of these pre-defined transformation matrices. Note that DR-G is given the same budget of 100 steps per optimization run as FD-G and ZRS for comparison in Sec. 4.4.

The final transformation matrix is constructed by a series of rotations followed by one translation, i.e.,

$$M = T \cdot R_{n-1} R_{n-2} \cdots R_0$$

where M is the final transformation matrix, R_i the rotation matrices, and T the translation matrix.

We empirically found that increasing the number of rotations per step helped (a) improve the success rate of hitting the target labels; (b) increase the maximum confidence score of the found AXs; and (c) reduce the number of steps, *i.e.*, led to faster convergence (see Fig. S9). Therefore, we empirically chose $n = 50$ for all DR-G experiments reported in the main text.



(a) y-axis: success rate (b) y-axis: max confidence (c) y-axis: mean number of steps

Figure S9: We found that increasing the number of rotations (displayed in x-axes) per step helped:
(a) improve the success rate of hitting the target labels;
(b) increase the maximum confidence score of the found adversarial examples;
(c) reduce the average number of steps required to find an AX, *i.e.*, led to faster convergence.

S5.2. Multi-view optimization

Additionally, we attempted to harness multiple views (from multiple cameras) to increase the chance of finding a target adversarial pose. Multi-view optimization did not outperform single-view optimization using the DR in our experiments. Therefore, we only performed regular single-view optimization for DR-G. We briefly document our negative results below.

Instead of backpropagating the DR gradient to a single camera looking at the object in the 3D scene, one may set up multiple cameras, each looking at the object from a different angle. This strategy intuitively allows gradients to still be backpropagated into the vertices that may be occluded in one view but visible in some other view. We experimented with six cameras and backpropagating to all cameras in each step. However, we only updated the object following the gradient from the view that yielded the lowest loss among all views. One hypothesis is that having multiple cameras might improve the chance of hitting the target.

In our experiments with the DR using 100 steps per optimization run, multi-view optimization performed worse than single-view in terms of both the success rate and the number of steps to converge. We did not compare all 30 objects due to the expensive computational cost, and only report the results from optimizing two objects `bald eagle` and `tiger cat` in Table S4. Intuitively, multi-view optimization might outperform single-view optimization given a large enough number of steps.

	bald eagle		tiger cat	
	Steps	Success rate	Steps	Success rate
Single-view	71.80	0.44	90.70	0.15
Multi-view	81.28	0.23	96.84	0.04

Table S4: Multi-view optimization performed worse than single-view optimization in both (a) the number of steps to converge and (b) success rates. We show here the results of two runs of optimizing with the `bald eagle` and `tiger cat` objects. The results are averaged over 50 target labels $\times 50$ trials = $2,500$ trials. Each optimization trial for both single- and multi-view settings is given the budget of 100 steps.

S6. 3D transformation matrix

A rotation of θ around an arbitrary axis (x, y, z) is given by the following homogeneous transformation matrix.

$$R = \begin{vmatrix} xx(1-c)+c & xy(1-c)-zs & xz(1-c)+ys & 0 \\ xy(1-c)+zs & yy(1-c)+c & yz(1-c)-xs & 0 \\ xz(1-c)-ys & yz(1-c)+xs & yz(1-c)+c & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \tag{8}$$

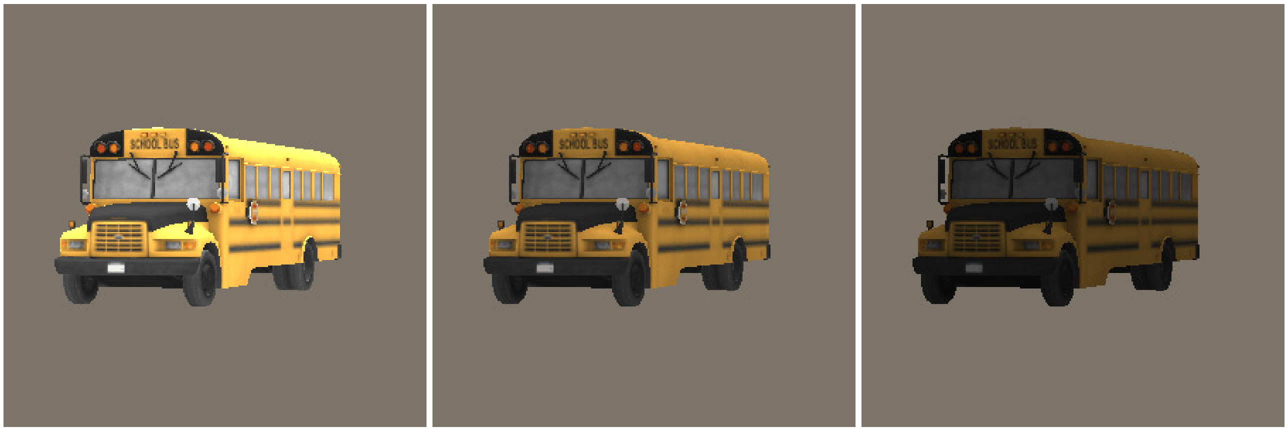
where $s = \sin\theta$, $c = \cos\theta$, and the axis is normalized, *i.e.*, $x^2 + y^2 + z^2 = 1$. Translation by a vector (x, y, z) is given by the following homogeneous transformation matrix.

$$T = \begin{vmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{vmatrix} \tag{9}$$

Note that in the optimization experiments with random search (RS) and finite-difference gradients (FD-G), we dropped the homogeneous component for simplicity, *i.e.*, the rotation matrices of yaw, pitch, and roll are all 3×3. The homogeneous component is only necessary for translation, which can be achieved via simple vector addition. However, in DR-G, we used the homogeneous component because we had some experiments interweaving translation and rotation. The matrix representation was more convenient for the DR-G experiments. As they are mathematically equivalent, this arbitrary implementation choice should not alter our results.

Object	Accuracy (%)	Object	Accuracy (%)	Object	Accuracy (%)
ambulance	3.64	golfcart	2.14	police van	0.95
backpack	8.63	jean	2.71	recreational vehicle	2.05
bald eagle	13.26	jeep	0.29	school bus	3.48
beach wagon	0.60	minibus	0.83	sports car	2.50
cab	2.64	minivan	0.66	street sign	26.32
cell phone	14.97	motor scooter	20.49	tiger cat	7.36
fire engine	4.31	moving van	0.45	tow truck	0.87
forklift	5.20	park bench	5.72	traffic light	14.95
garbage truck	4.88	parking meter	1.27	trailer truck	1.27
German shepherd	9.61	pickup	0.86	umbrella	49.88

Table S5: The percent of three million random samples that were correctly classified by Inception-v3 [44] for each object. That is, for each lighting setting in $\{\mathsf{bright}, \mathsf{medium}, \mathsf{dark}\}$, we generated 10^6 samples. See Sec. 3.2 for details on the sampling procedure.



(a) bright (b) medium (c) dark

Figure S10: Renders of the `school bus` object using the NR [1] at three different lighting settings. The directional light intensities and ambient light intensities were $(1.2, 1.6)$, $(0.4, 1.0)$, and $(0.2, 0.5)$ for the bright, medium, and dark settings, respectively.

S7. Adversarial poses were not found in ImageNet classes via a nearest-neighbor search

We performed a nearest-neighbor search to check whether adversarial poses generated (in Sec. 4.1) can be found in the ImageNet dataset.

Retrieving nearest neighbors from a single class corresponding to the 3D object. We retrieved the five nearest training-set images for each adversarial pose (taken from a random selection of adversarial poses) using the fc7 feature space from a pre-trained AlexNet [20]. The Euclidean distance was used to measure the distance between two fc7 feature vectors. We did not find qualitatively similar images despite comparing all ∼1,300 class images corresponding to the 3D object used to generate the adversarial poses (*e.g.*, `cellphone`, `school bus`, and `garbage truck` in Figs. S11, S12, and S13). This result supports the hypothesis that the generated adversarial poses are out-of-distribution.

Searching from the validation set. We also searched the entire 50,000-image validation set of ImageNet. Interestingly, we found the top-5 nearest images were sometimes from the same class as the *targeted* misclassification label (see Fig. S19).

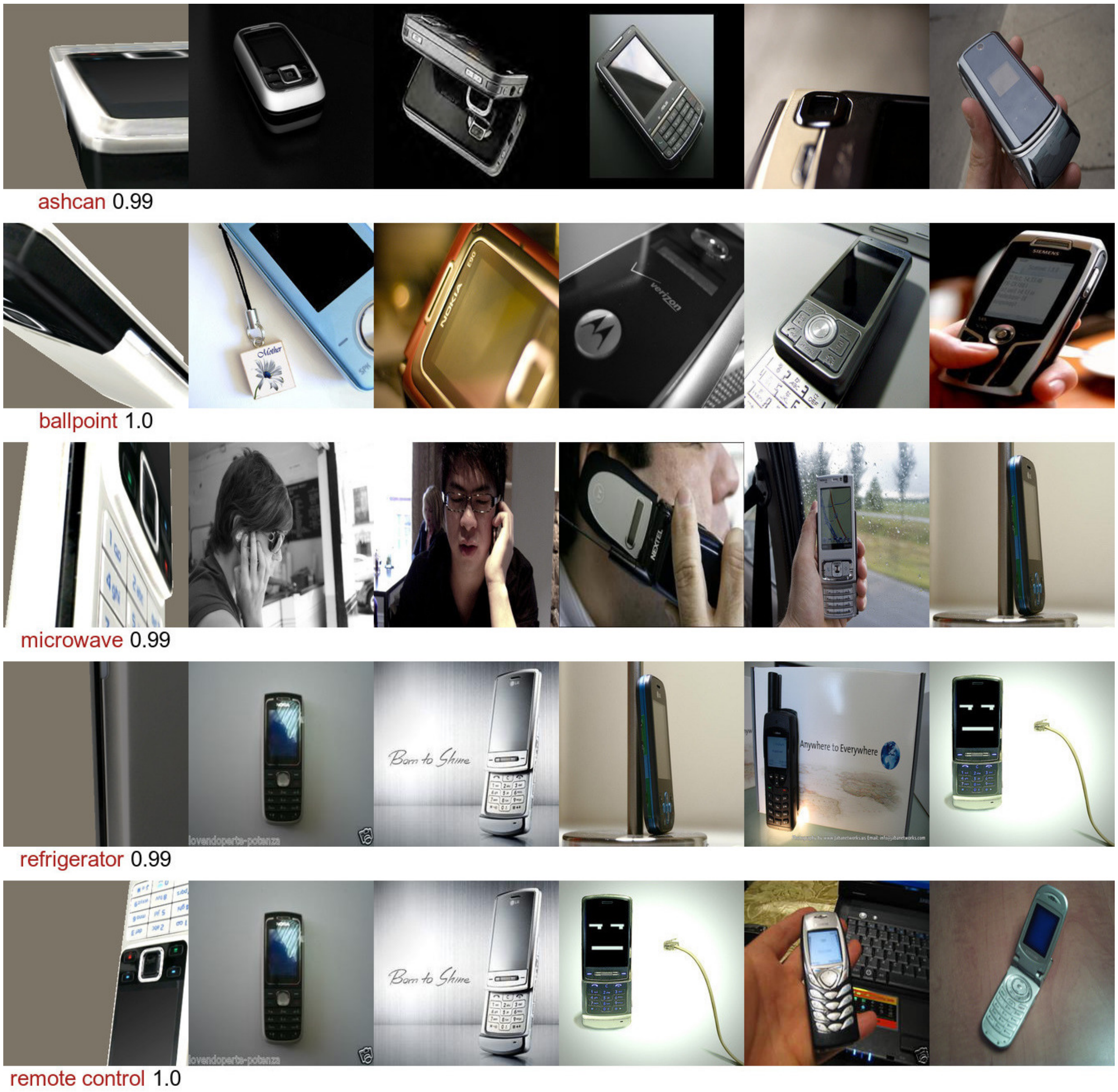


Figure S11: For each adversarial example (leftmost), we retrieved the five nearest neighbors (five rightmost photos) from all ∼1,300 images in the `cellular phone` class. The Euclidean distance between a pair of images was computed in the fc7 feature space of a pre-trained AlexNet [20]. The nearest photos from the class are mostly different from the adversarial poses. This result supports the hypothesis that the generated adversarial poses are out-of-distribution. The original, high-resolution figure is available at `https://goo.gl/X31VXh`.

S8. DNN failure rates for ball-like objects

To investigate the role of object geometry in DNN pose failures, we re-ran our random search procedure on five purchased ball objects: three different soccer balls, a volleyball, and a ping-pong ball. The error rates were 4%, 4%, 5%, 47%, and 48%, respectively; however, the incorrect labels for the volleyball and ping-pong ball objects were qualitatively often reasonable (*e.g.*, `golf ball` and `billiard ball` for the ping-pong ball). Clearly, the DNN can gracefully handle much of the pose space for these "easy" objects, but whether this robustness is due to the specific features of the classes (*e.g.*, black and white corners where hexagons meet on a soccer ball) or data variability in the training set requires further research.

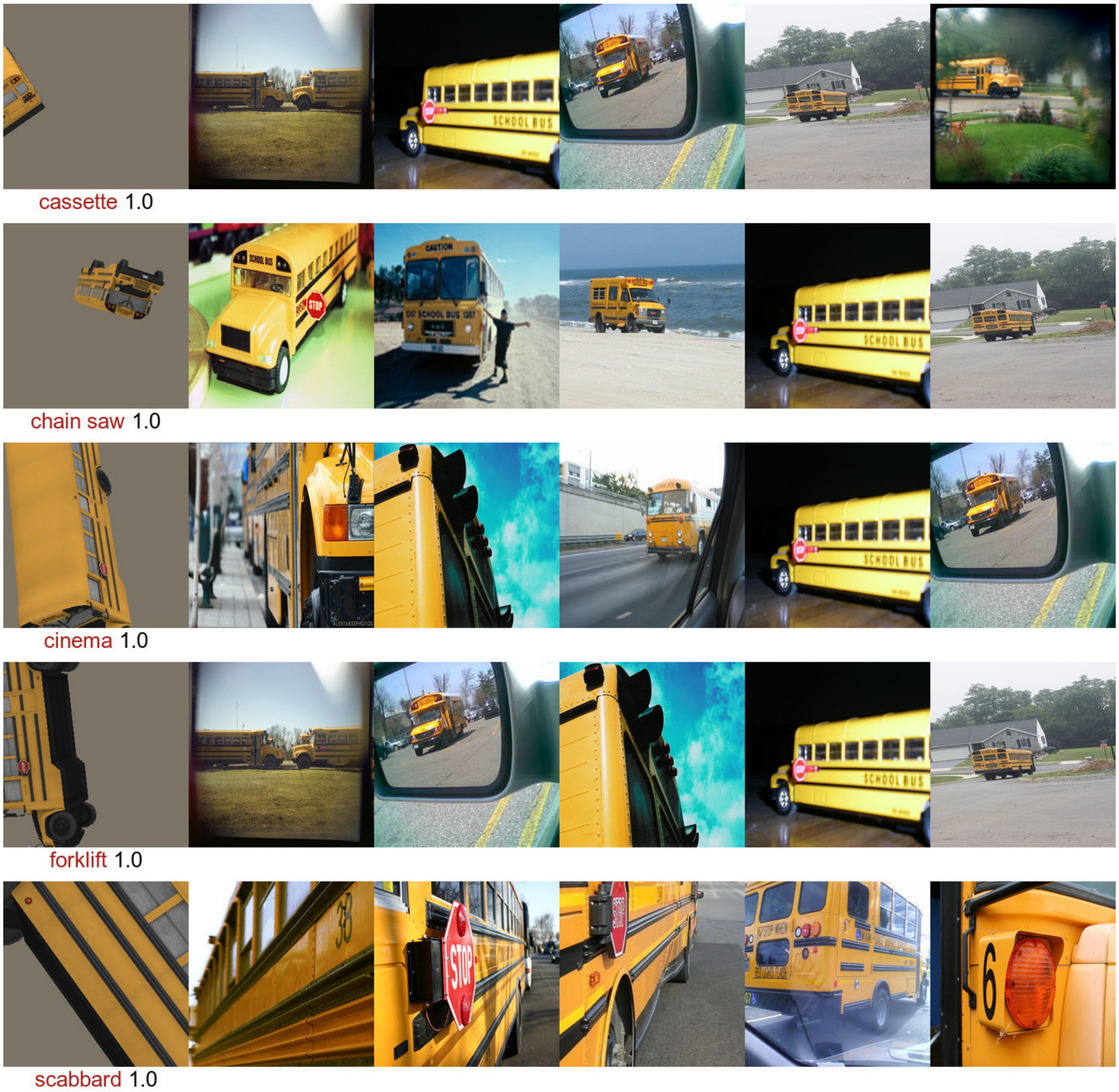


Figure S12: For each adversarial example (leftmost), we retrieved the five nearest neighbors (five rightmost photos) from all $\sim$1,300 images in the `school bus` class. The Euclidean distance between a pair of images was computed in the fc7 feature space of a pre-trained AlexNet [20]. The nearest photos from the class are mostly different from the adversarial poses. This result supports the hypothesis that the generated adversarial poses are out-of-distribution. The original, high-resolution figure is available at `https://goo.gl/X31VXh`.

S9. Adversarial training

Training. We augmented the original 1,000-class ImageNet dataset with an additional 30 AX classes. Each AX class included 1,350 randomly selected high-confidence ($p \geq 0.9$) misclassified images split 1,300/50 into training/validation sets. Our AlexNet trained on the augmented dataset (AT) achieved a top-1 accuracy of 0.565 for the original ImageNet validation set and a top-1 accuracy[6] of 0.967 for the AX validation set.

Evaluation. To evaluate our AT model vs. a pre-trained AlexNet (PT), we used RS to generate 10^6 samples for each of our 3D training objects. In addition, we collected seven held-out 3D objects not included in the training set that belong to the

[6] In this case, a classification was "correct" if it matched *either* the original ImageNet positive label *or* the negative, object label.

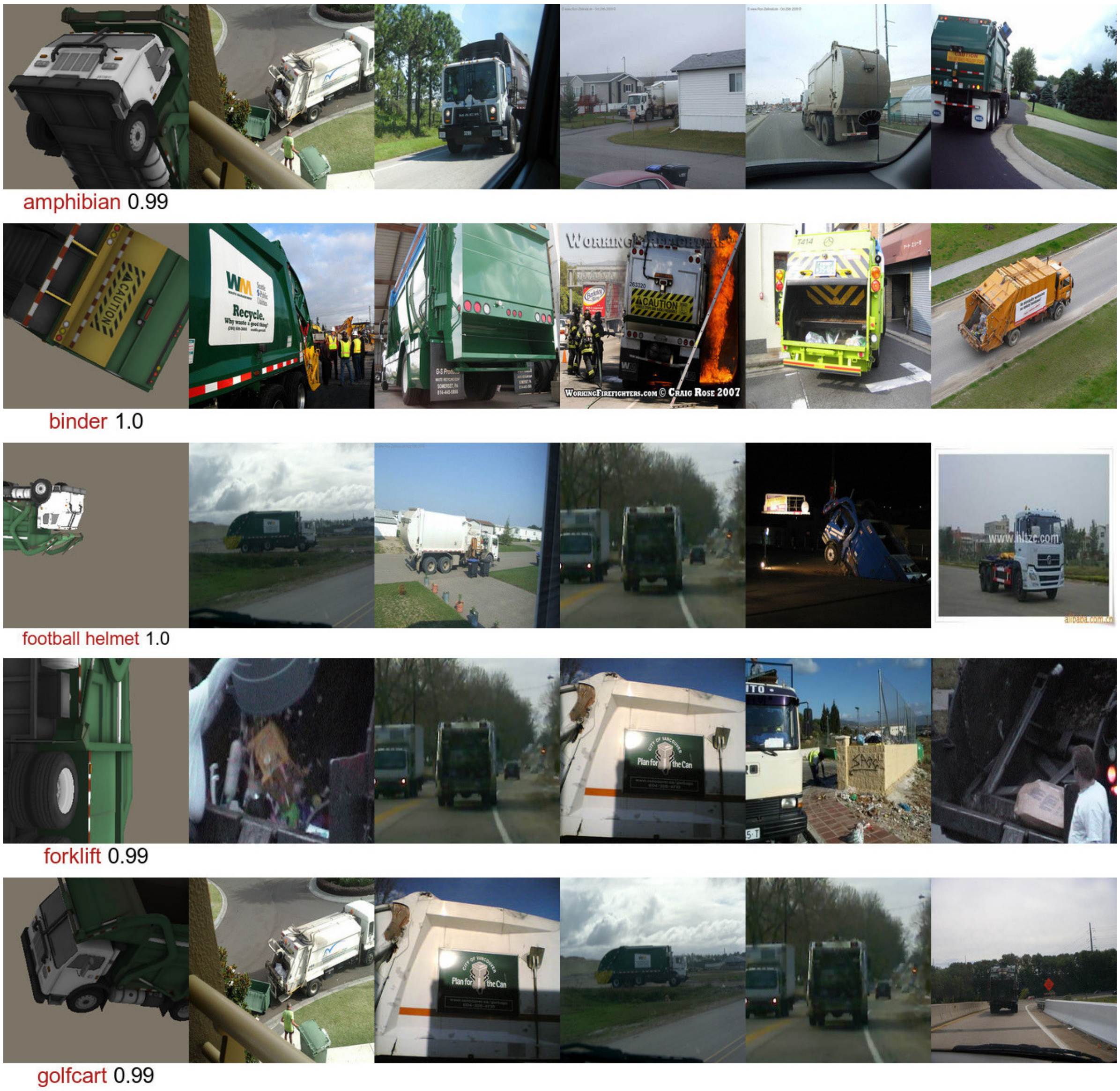


Figure S13: For each adversarial example (leftmost), we retrieved the five nearest neighbors (five rightmost photos) from all ∼1,300 images in the `garbage truck` class. The Euclidean distance between a pair of images was computed in the fc7 feature space of a pre-trained AlexNet [20]. The nearest photos from the class are mostly different from the adversarial poses. This result supports the hypothesis that the generated adversarial poses are out-of-distribution. The original, high-resolution image is available at `https://goo.gl/X31VXh`.

same classes as seven training-set objects (example renders in Fig. S14). We followed the same sampling procedure for the held-out objects to evaluate whether our AT generalizes to unseen objects.

For each of these $30 + 7 = 37$ objects and for both the PT and our AT, we recorded two statistics: (1) the percent of misclassifications, *i.e.* errors; and (2) the percent of high-confidence (*i.e.*, $p \geq 0.7$) misclassifications (Table 3).

We hypothesize that augmenting the dataset with many more 3D objects may improve DNN generalization on held-out objects. Here, AT might have used (1) the grey background to separate the 1,000 original ImageNet classes from the 30 AX classes; and (2) some non-geometric features sufficient to discriminate among only 30 objects. However, as suggested by our work (Sec. 2.4), acquiring a large-scale, high-quality 3D object dataset is costly and labor-intensive. Currently, no such public dataset exists, and thus we could not test this hypothesis.

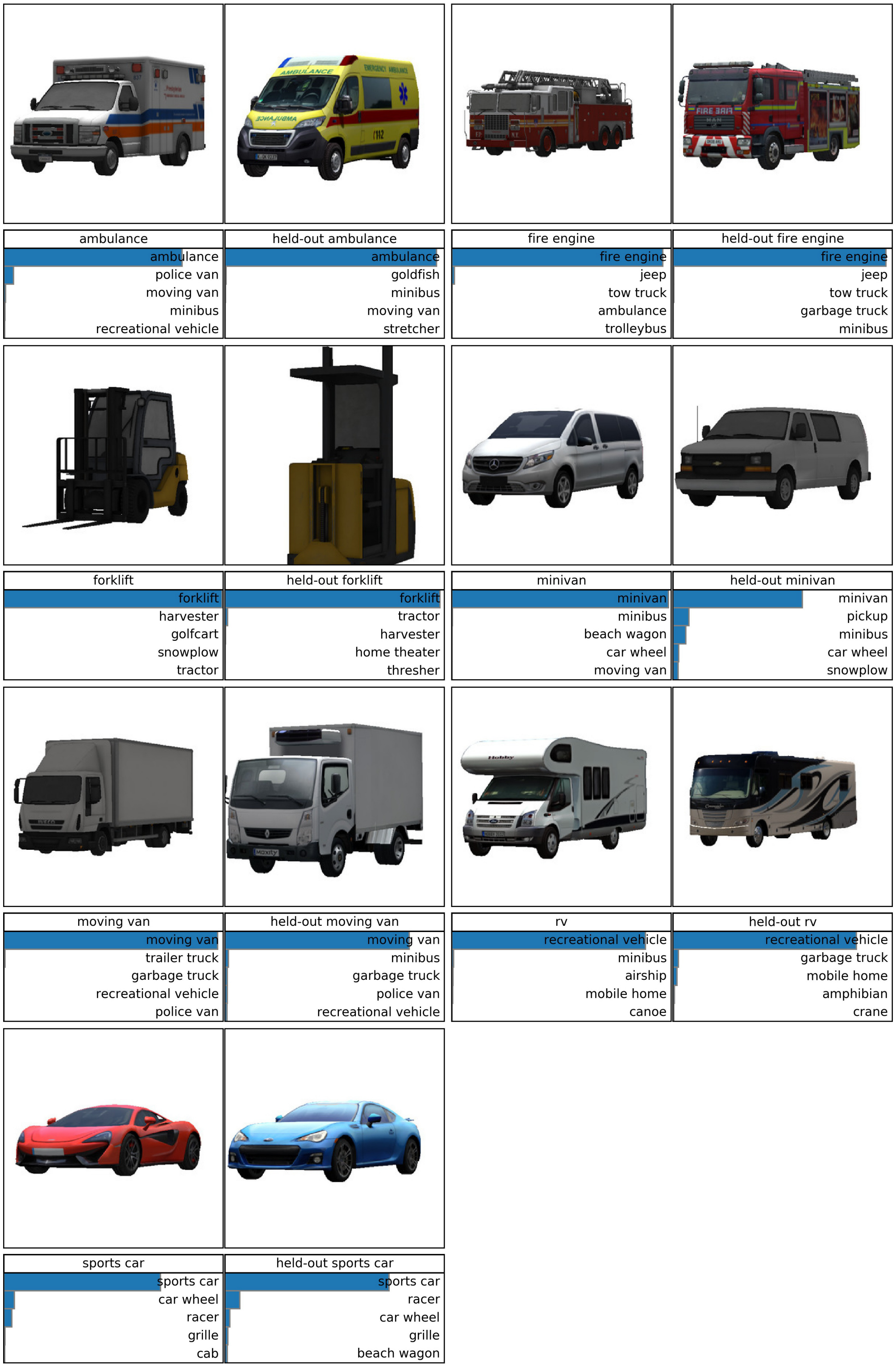
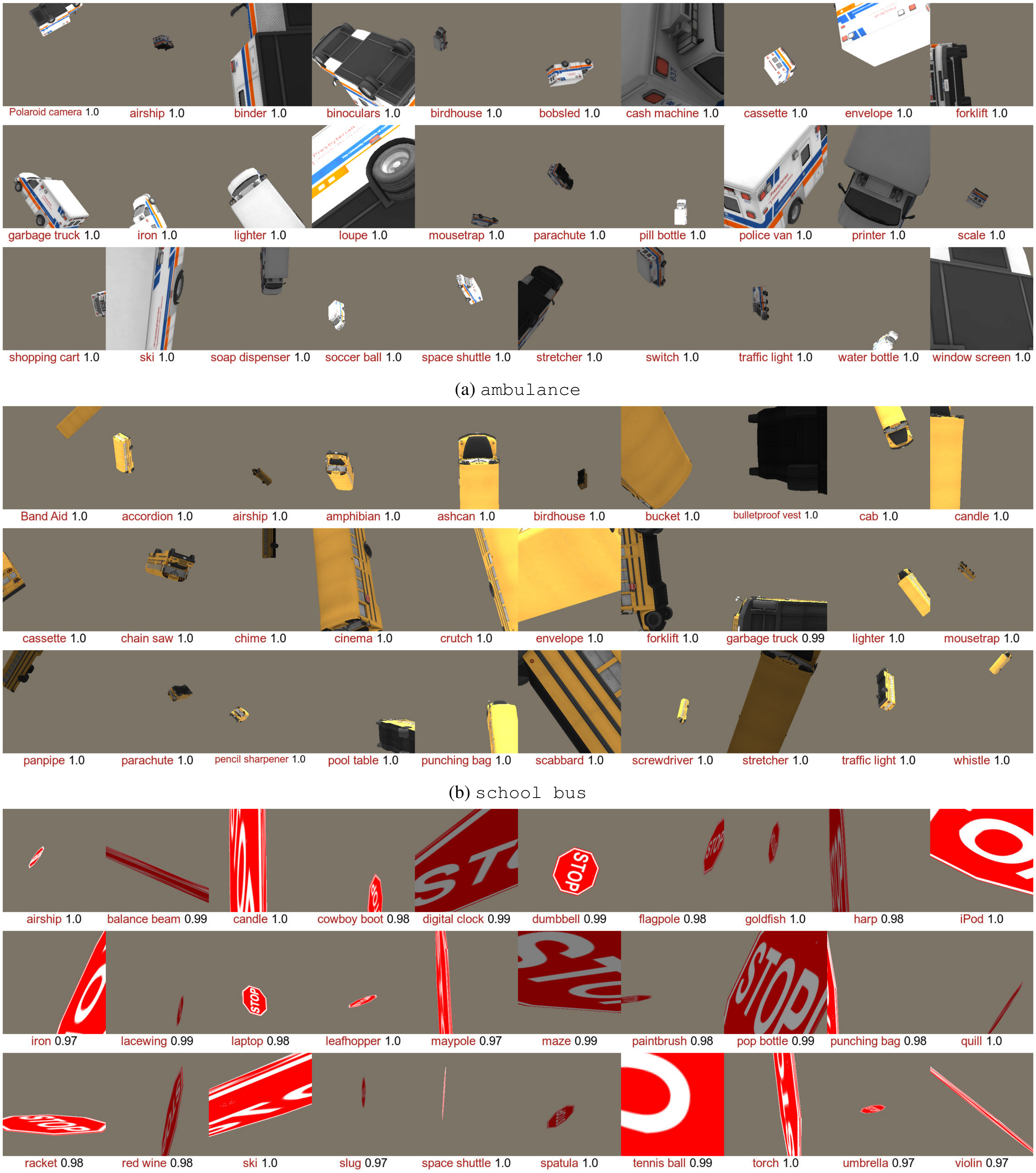


Figure S14: In Sec. S9, we trained an AlexNet classifier on the 1000-class ImageNet dataset augmented with 30 additional classes that contain adversarial poses corresponding to the 30 *known* objects used in the main experiments. We also tested this model on 7 *held-out* objects. Here, we show the renders of 7 pairs of (training-set object, held-out object). The 3D objects are rendered by the NR [1] at a distance of $(0, 0, 4)$. Below each image is its top-5 predictions by Inception-v3 [44]. The original, high-resolution figure is available at https://goo.gl/Li1eKU.



(a) `ambulance`

(b) `school bus`

(c) `street sign`

Figure S15: 30 random adversarial examples misclassified by Inception-v3 [44] with high confidence ($p \geq 0.9$) generated from 3 objects: `ambulance`, `school bus`, and `street sign`. Below each image is the top-1 prediction label and confidence score. The original, high-resolution figures for all 30 objects are available at `https://goo.gl/rvDzjy`.

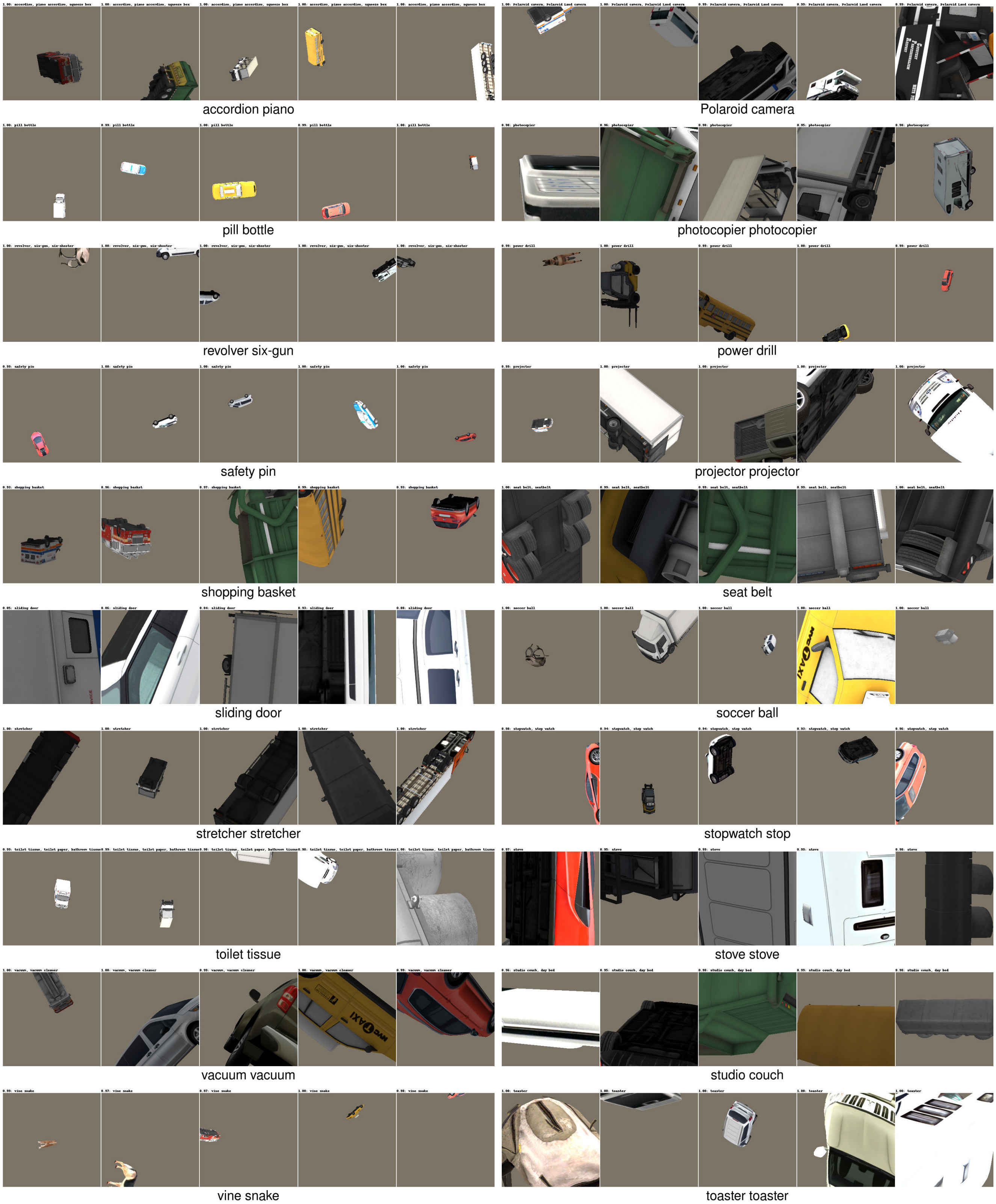
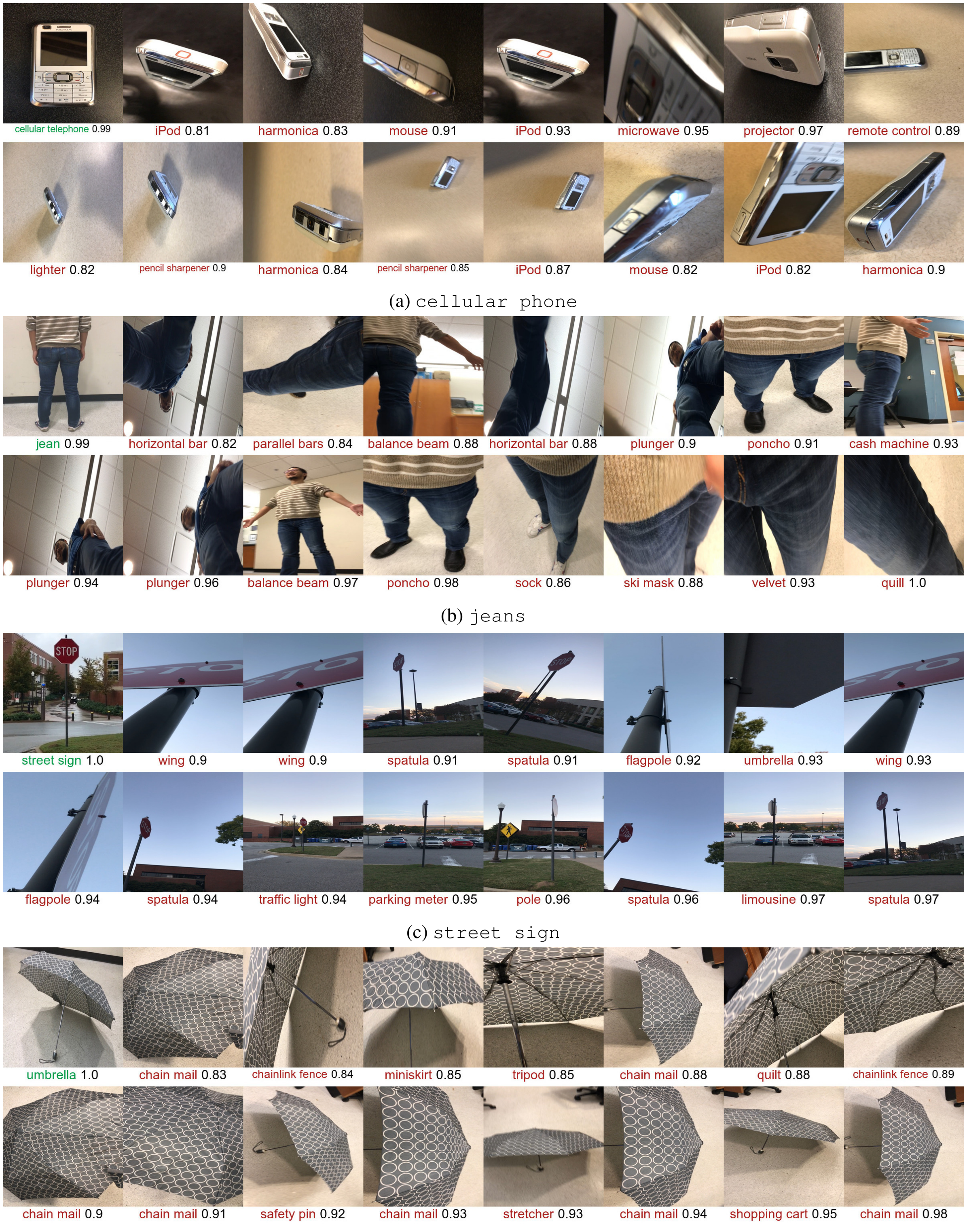


Figure S16: For each target class (*e.g.*, `accordion piano`), we show five adversarial poses generated from five unique 3D objects. Adversarial poses are interestingly found to be homogeneous for some classes, *e.g.*, `safety pin`. However, for most classes, the failure modes are heterogeneous. The original, high-resolution figure is available at `https://goo.gl/37HYcE`.



(a) `cellular phone`

(b) `jeans`

(c) `street sign`

(d) `umbrella`

Figure S17: Real-world, high-confidence adversarial poses can be found by taking photos from strange angles of a familiar object, here, `cellular phone`, `jeans`, `street sign`, and `umbrella`. While Inception-v3 [44] can correctly predict the object in canonical poses (the top-left image in each panel), the model misclassified the same objects in unusual poses. Below each image is its top-1 prediction label and confidence score. We took real-world videos of these four objects and extracted these misclassified poses from the videos. The original, high-resolution figures are available at `https://goo.gl/zDWcjG`.

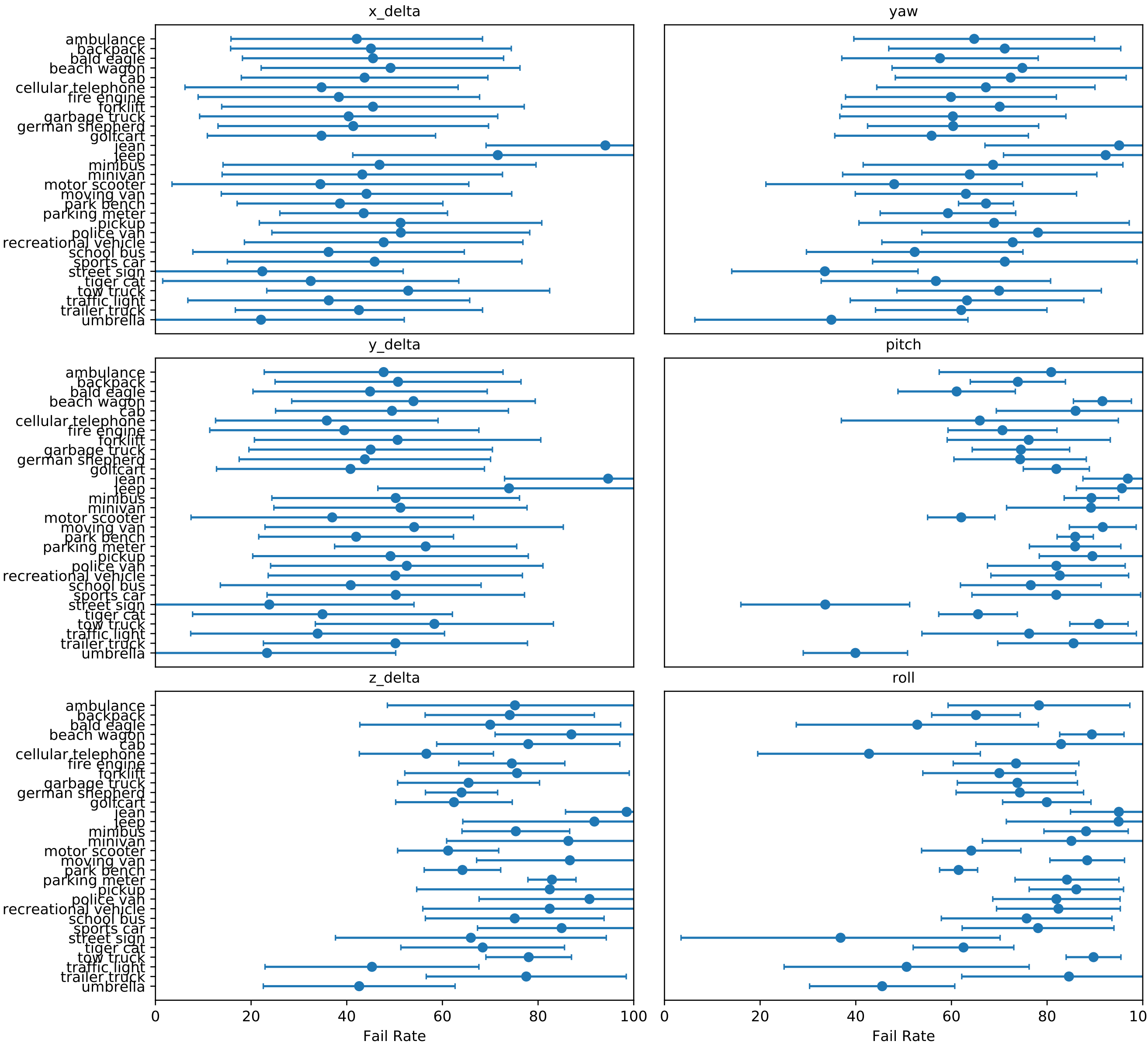


Figure S18: Inception-v3 [44] is sensitive to single parameter disturbances of object poses that had originally been correctly classified. For each object, we found 100 correctly classified 6D poses via a random sampling procedure (Sec. 4.3). Given each such pose, we re-sampled one parameter (shown on top of each panel, *e.g.*, yaw) 100 times, yielding 100 classifications, while holding the other five pose parameters constant. In each panel, for each object (*e.g.*, `ambulance`), we show an error plot for all resultant $100 \times 100 = 10,000$ classifications. Each circle denotes the mean misclassification rate ("Fail Rate") for each object, while the bars enclose one standard deviation. Across all objects, Inception-v3 is more sensitive to changes in yaw, pitch, roll, and depth ("z_delta") than spatial changes ("x_delta" and "y_delta").

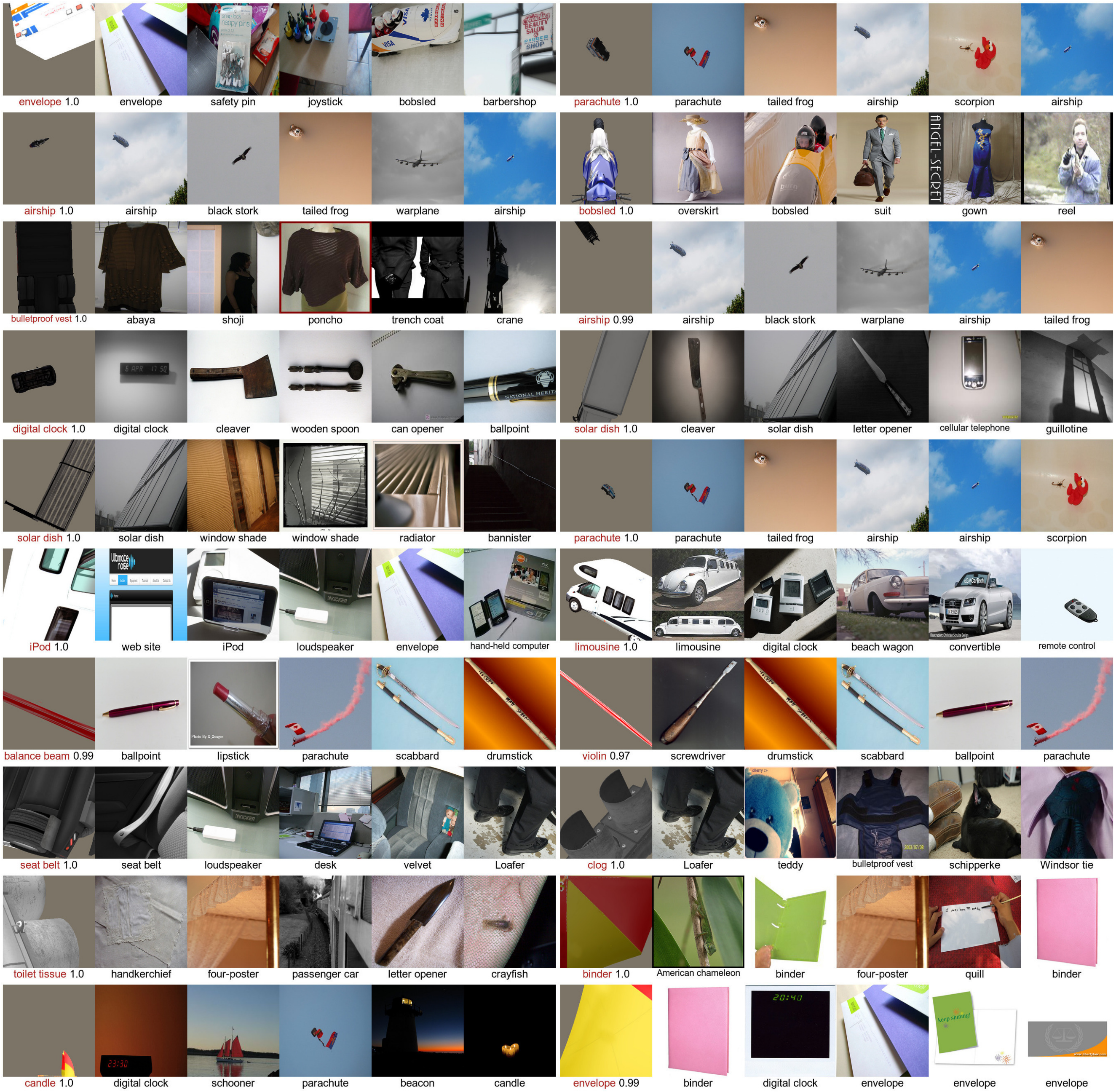
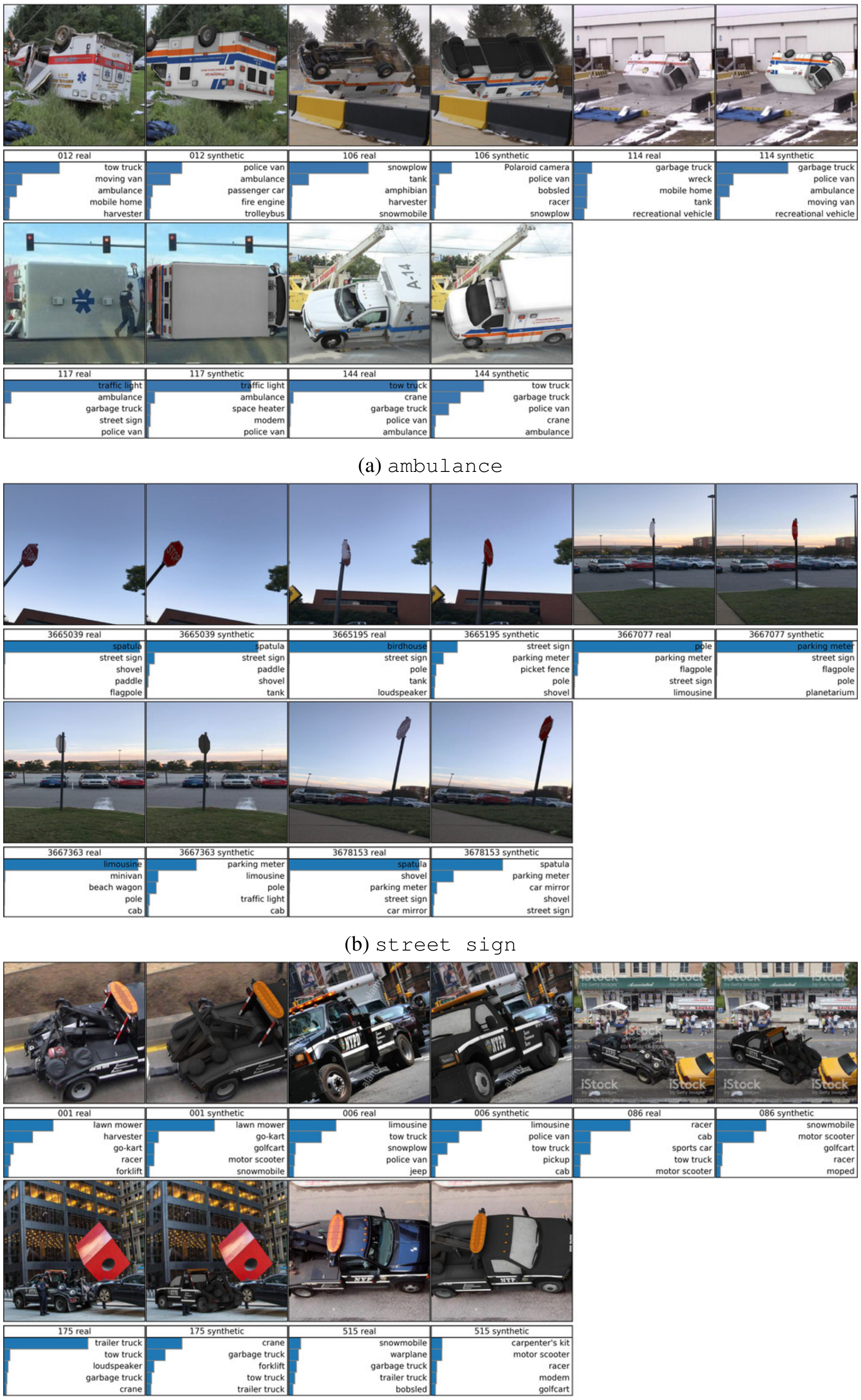


Figure S19: For each adversarial example (leftmost), we retrieved the five nearest neighbors (five rightmost photos) from the 50,000-image ImageNet validation set. The Euclidean distance between a pair of images was computed in the fc7 feature space of a pre-trained AlexNet [20]. Below each adversarial example (AX) is its Inception-v3 [44] top-1 prediction label and confidence score. The associated ground-truth ImageNet label is beneath each retrieved photo. Here, we show an interesting, cherry-picked collection of cases where the nearest photos (in the fc7 feature space) are also qualitatively similar to the reference AX and sometimes come from the exact same class as the AX's predicted label. More examples are available at https://goo.gl/8ib2PR.



(a) `ambulance`

(b) `street sign`

(c) `tow truck`

Figure S20: Adversarial poses do transfer to the real world. We collected a set of 150 real photos (5 photos × 30 objects) from the Internet that caused the Inception-v3 classifier to misclassify. For each pair, given the real, misclassified photo (left), we produced a render of the corresponding object (right) and gathered its top-5 predictions. We found that when the real photos appear out-of-distribution, 98.3% of the renders are also misclassified, sometimes with the same top-1 label *e.g.*, `spatula` in (b) or `lawn mower` in (c). Here, we show 5 pairs for each of the three example objects: (a) `ambulance`, (b) `street sign`, and (c) `tow truck`. The original, high-resolution figures for all 30 objects are available at https://drive.google.com/open?id=18p-S9qO4dhE9toJbRRlAIWRcVqVH6Zsd.