

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](https://www.turbosquid.com), [free3d.com](https://www.free3d.com), and [cgtrader.com](https://www.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 $\sim 15\times$ 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 $\sim 15\times$ 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.

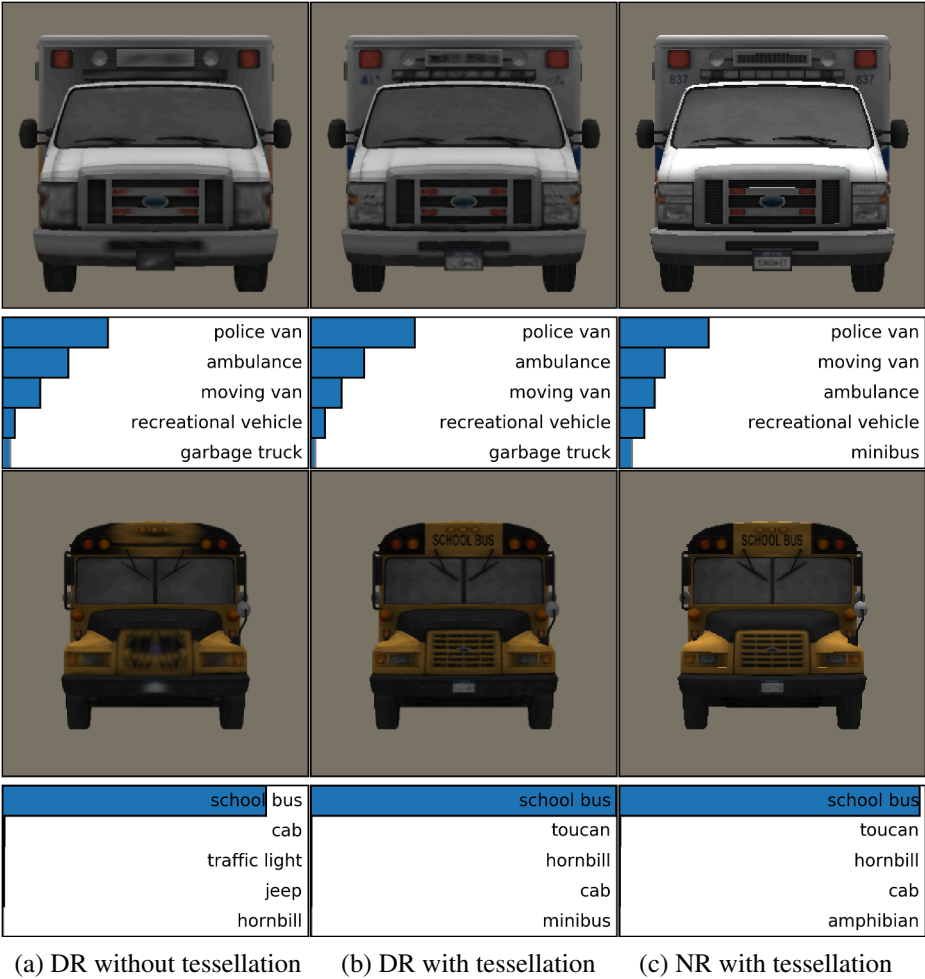


Figure S1: A comparison of 3D object renders (here, ambulance and school bus) before and after tessellation. (a) Original 3D models rendered by the differentiable renderer (DR) [19] without tessellation. (b) DR renderings of the same objects after manual tessellation. (c) The non-differentiable renderer (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 ~ 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 $30 \text{ objects} \times 2 \text{ pairs} = 60 \text{ pairs}$ are provided at <https://drive.google.com/drive/folders/1ti9zoldzU1e9b-mpqv0bhTrMeoeUBULm>. The pose alignment was done in our GUI tool³. 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 (%)	Δ acc (%)
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 (“ Δ acc (%)”).

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 detection⁴ of YOLO-v3 (see Table S3d for class-specific statistics). Our result shows adversarial poses transfer well across tasks (image classification \rightarrow object detection), models (Inception-v3 \rightarrow YOLO-v3), and datasets (ImageNet \rightarrow 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 $5 \text{ photos} \times 30 \text{ objects} = 150 \text{ photos}$ 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, $\sim 45\%$ 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,

⁴We 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).





























					
ambulance	motor scooter	sports car	fire engine	forklift	garbage truck
ambulance moving van minibus police van cash machine	motor scooter moped crash helmet ice cream cradle	sports car car wheel racer convertible pickup	fire engine jeep tow truck trolleybus ambulance	forklift golfcart vacuum printer harvester	garbage truck crane harvester sloth bear missile
					
golfcart	jeep	moving van	minibus	minivan	jean
golfcart forklift jeep Model T scale	jeep pickup car wheel beach wagon limousine	moving van trailer truck recreational vehicle garbage truck police van	minibus minivan moving van police van recreational vehicle	minivan beach wagon car wheel pickup minibus	jean swab balance beam cowboy hat vacuum
					
pickup	police van	recreational vehicle	school bus	street sign	cab
pickup car wheel beach wagon grille jeep	police van racer beach wagon sports car cab	recreational vehicle recreational vehicle minibus mobile home airship passenger car	school bus toucan hornbill amphibian crane	street sign parking meter spatula traffic light cash machine	cab pickup amphibian car wheel school bus
					
beach wagon	tow truck	trailer truck	backpack	bald eagle	tiger cat
beach wagon car wheel minivan grille cab	tow truck jeep snowplow plow forklift	trailer truck chain saw moving van recreational vehicle tow truck	backpack mailbag bulletproof vest gasmask oxygen mask	bald eagle kite vulture common newt eft	tiger cat Egyptian cat tabby doormat triceratops
					
cellular telephone	German shepherd	park bench	parking meter	traffic light	umbrella
cellular telephone modem hand-held computer hard disc desktop computer	German shepherd bloodhound Norwegian elkhound malinois English foxhound	park bench ashcan lakeside studio couch marimba	parking meter loudspeaker pole microphone tripod	traffic light walking stick maillot Rhodesian ridgeback loudspeaker	umbrella mountain tent table lamp parachute maillot

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.

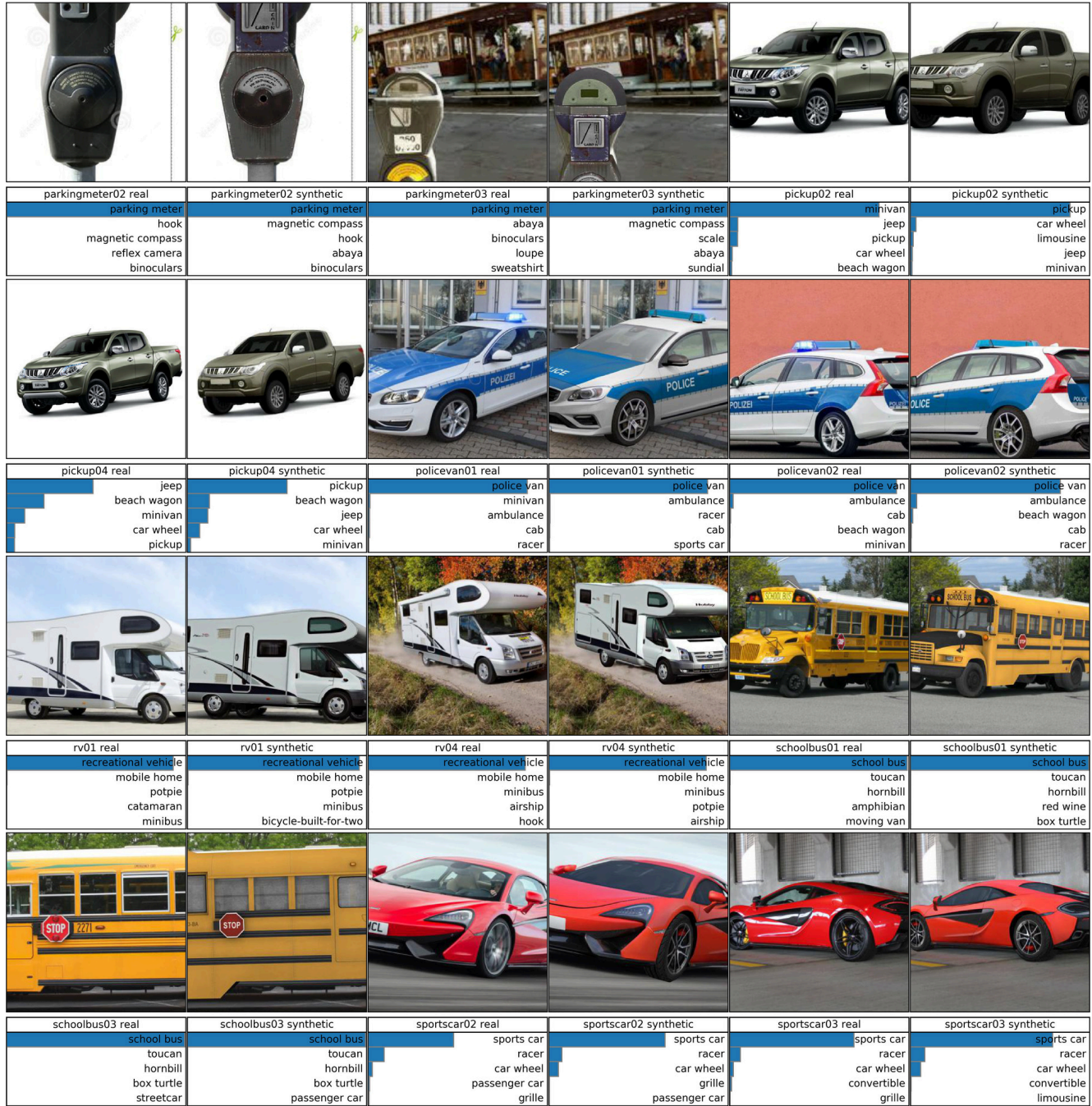


Figure S3: 12 random pairs of real photos (left) and renders (right) among 116 pairs produced in total for our 3D object rendering evaluation (Sec. S1.3.1). The renders are produced by ModernGL. More comparison images are available at <https://drive.google.com/drive/folders/1ti9zo1dzU1e9b-mpqv0bhTrMeoeUBULm>. 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.

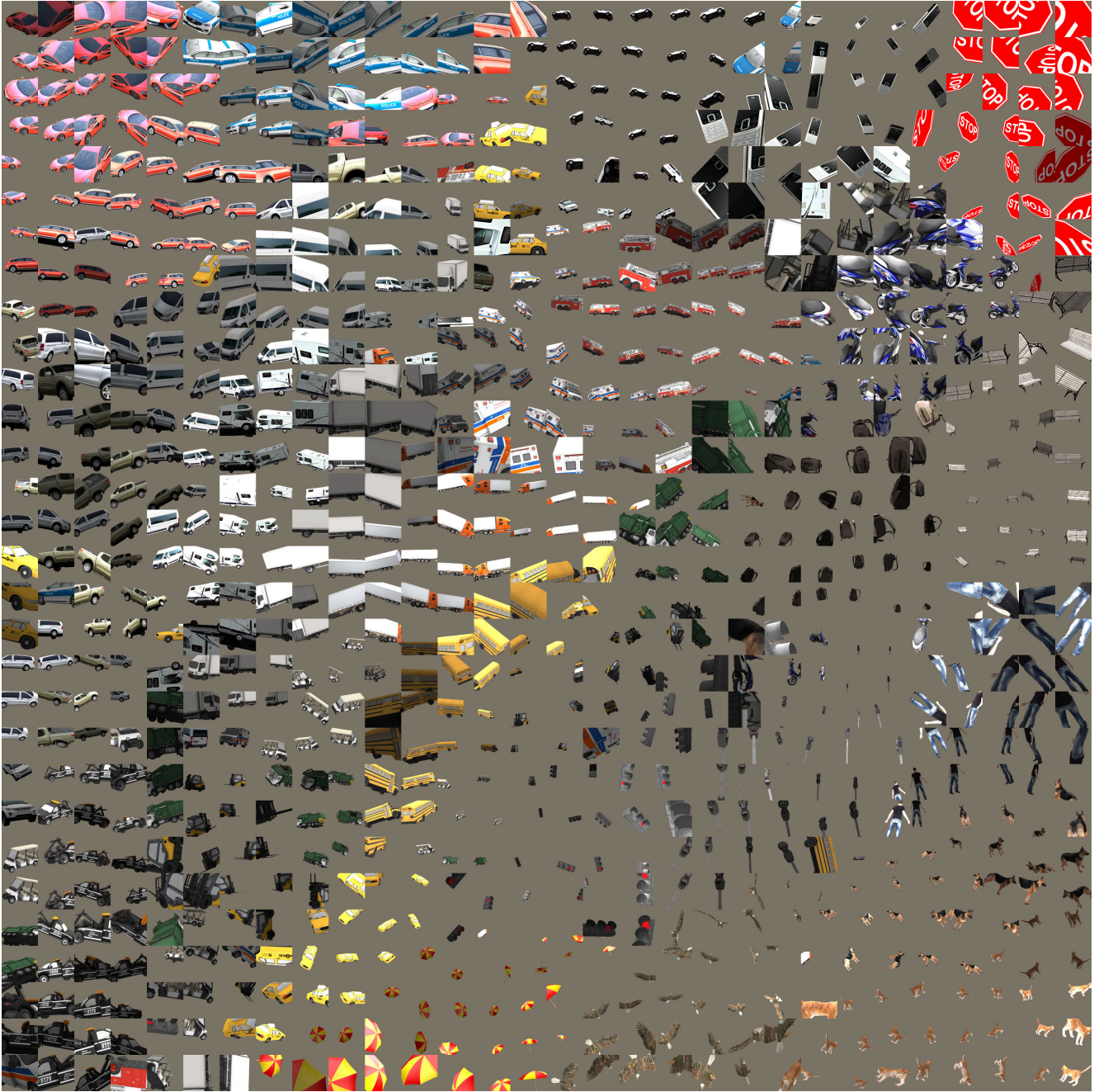


Figure S4: For each object, we collected 30 high-confidence ($p \geq 0.9$) correctly classified images by Inception-v3. The images were generated via the random search procedure. We show here a grid t-SNE of AlexNet [20] fc7 features for all 30 objects \times 30 images = 900 images. Correctly classified 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).

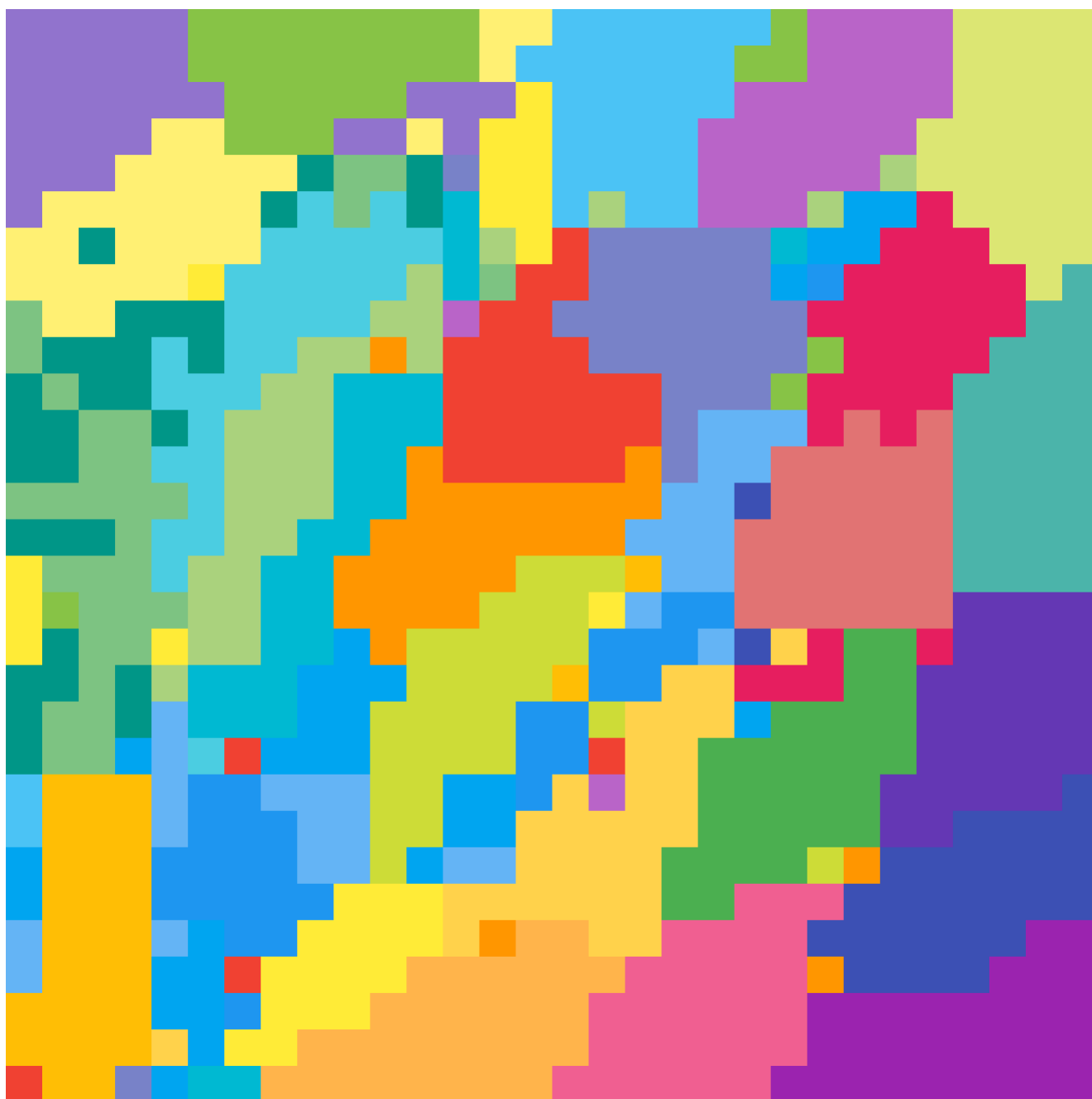


Figure S5: The same t-SNE found in Fig. S4 but using a unique color to denote the 3D object found in each rendered image. Here, each color also corresponds to a unique Inception-v3 label. Compare and contrast this plot with the t-SNE of only misclassified poses (Fig. S7). The original, high-resolution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

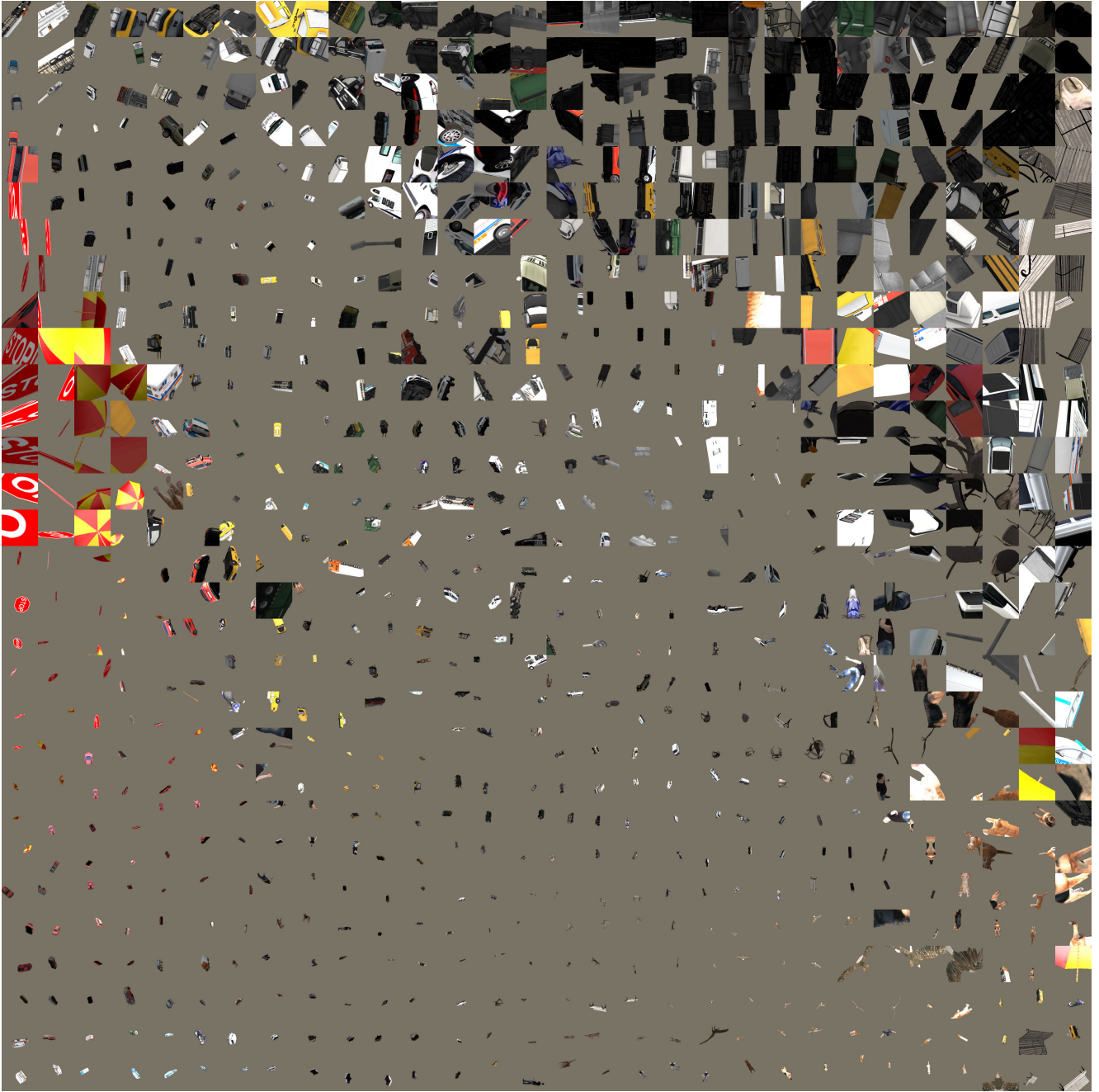


Figure S6: Following the same process as described in Fig. S4, we show here a grid t-SNE of generated adversarial poses. For each object, we assembled 30 high-confidence ($p \geq 0.9$) adversarial examples generated via a random search against Inception-v3 [44]. The t-SNE was generated from the AlexNet [20] fc7 features for $30 \text{ objects} \times 30 \text{ images} = 900 \text{ images}$. 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).

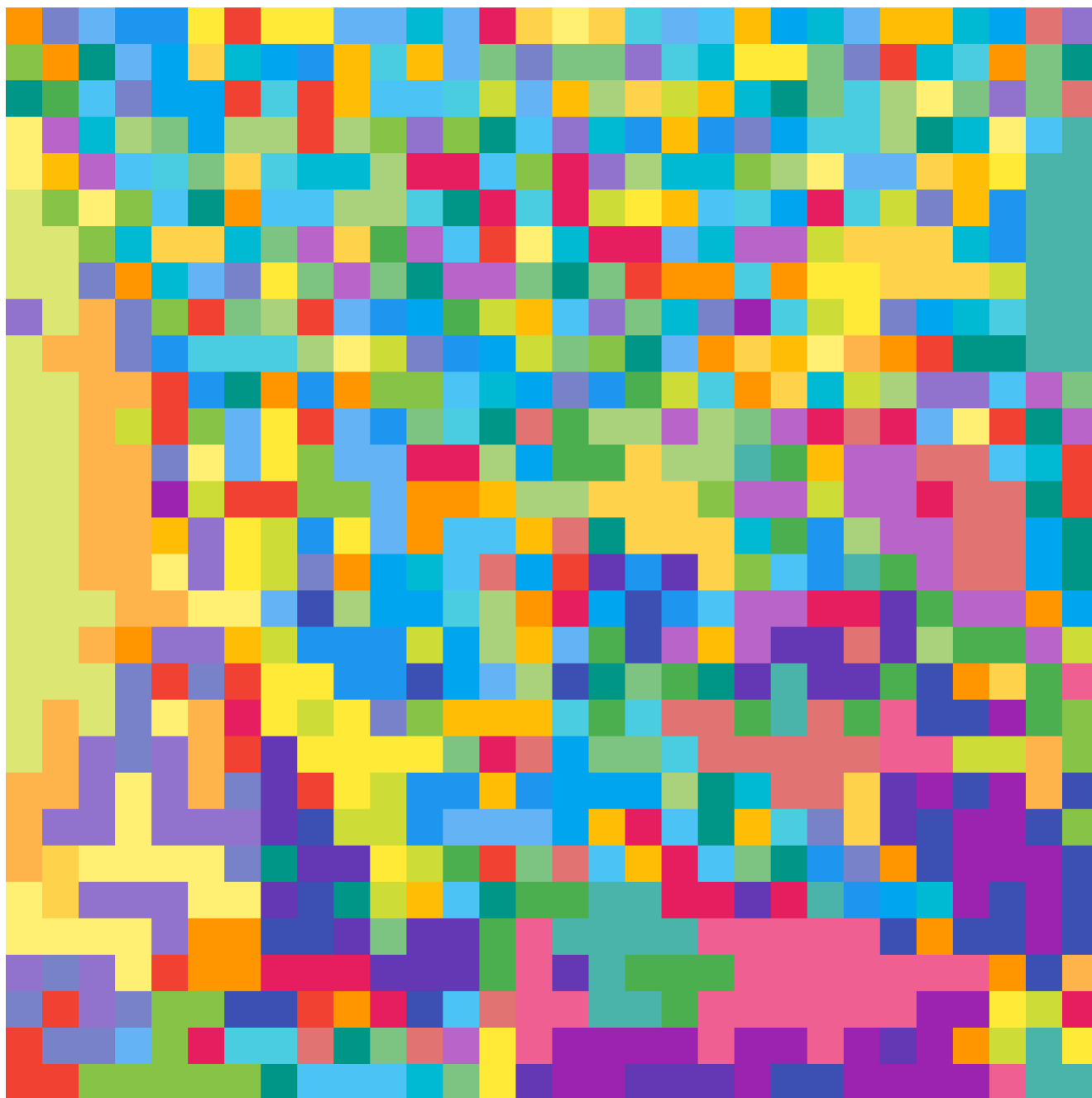


Figure S7: The same t-SNE as that in Fig. S6 but using a unique color to denote the 3D object used to render the adversarial image (*i.e.*, Inception-v3's misclassification labels are not shown here). The original, high-resolution figure is available at <https://goo.gl/TGgPgB>. Compare and contrast this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

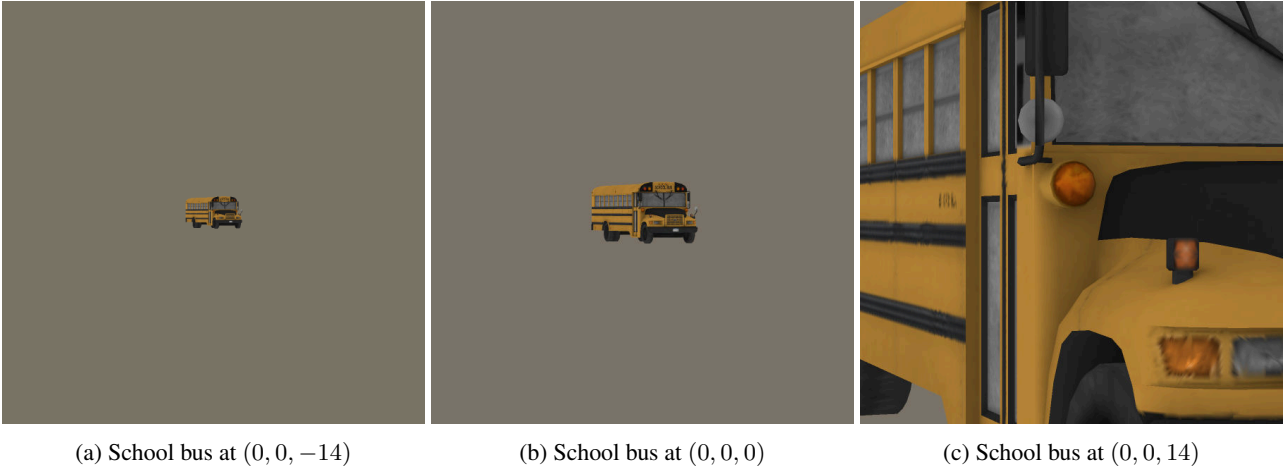


Figure S8: School bus rendered by the DR at different distances.

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

⁵In 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.

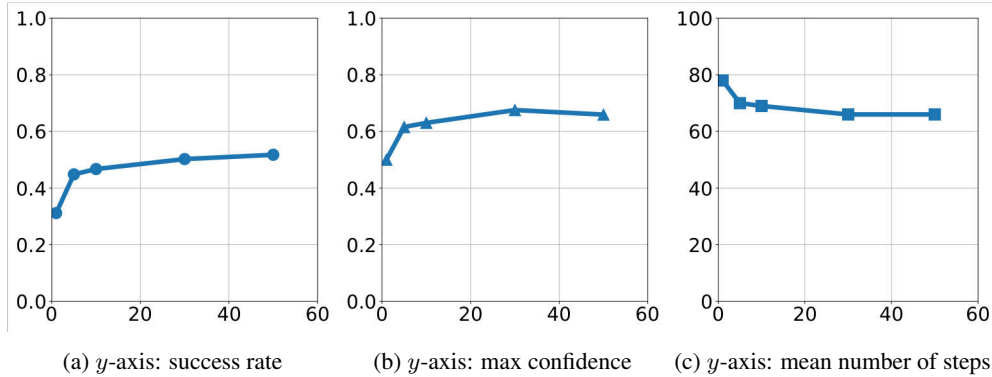


Figure S9: We found that increasing the number of rotations (displayed in x -axes) per step helped:

- (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.

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

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.

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

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 $\{\text{bright, medium, dark}\}$, we generated 10^6 samples. See Sec. 3.2 for details on the sampling procedure.



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 $\sim 1,300$ 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 $\sim 1,300$ 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 accuracy⁶ 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

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

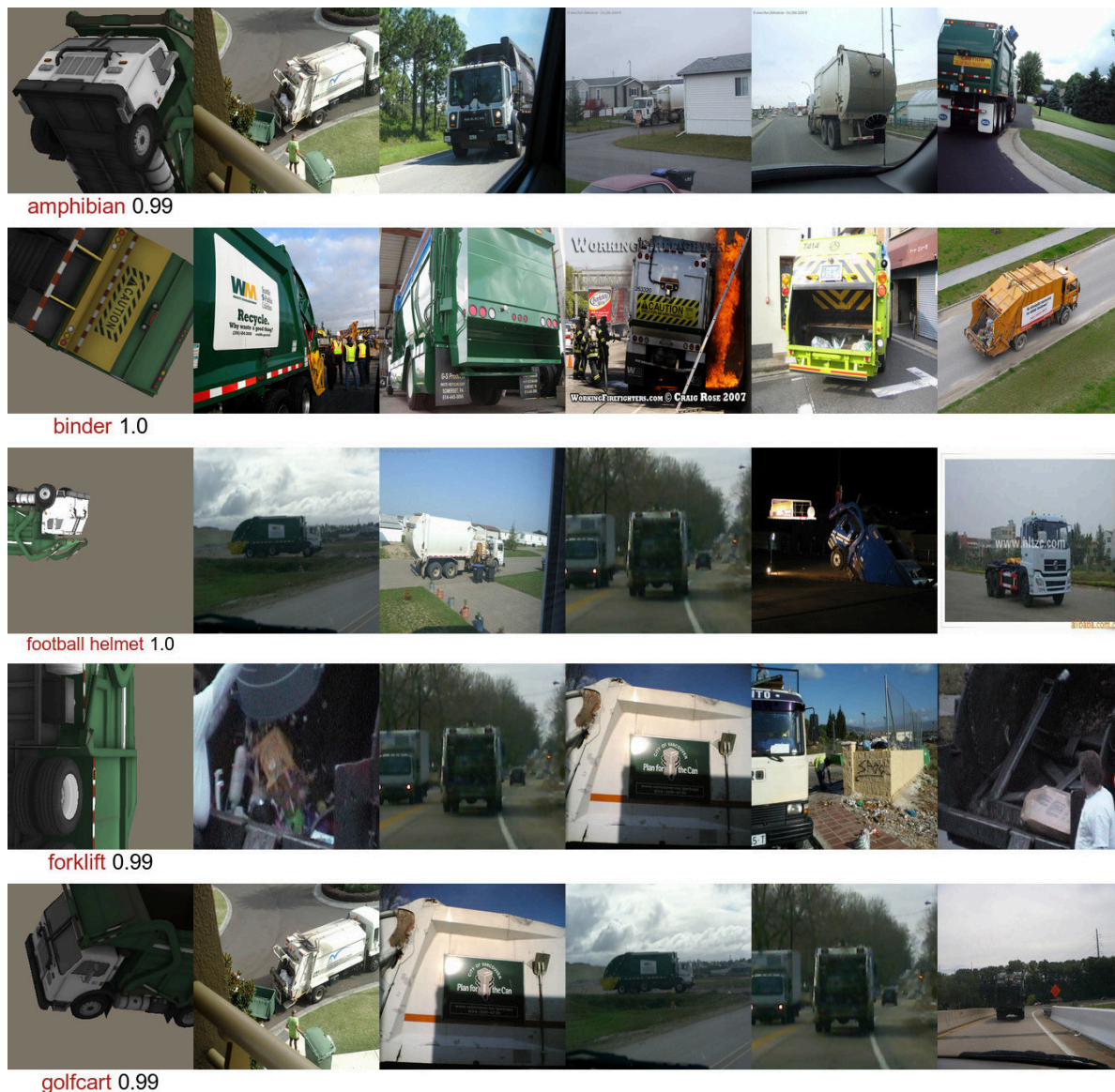


Figure S13: For each adversarial example (leftmost), we retrieved the five nearest neighbors (five rightmost photos) from all $\sim 1,300$ images in the `garbage truck` class. The Euclidean distance between a pair of images was computed in the `fc7` feature space of a pre-trained AlexNet [20]. The nearest 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.

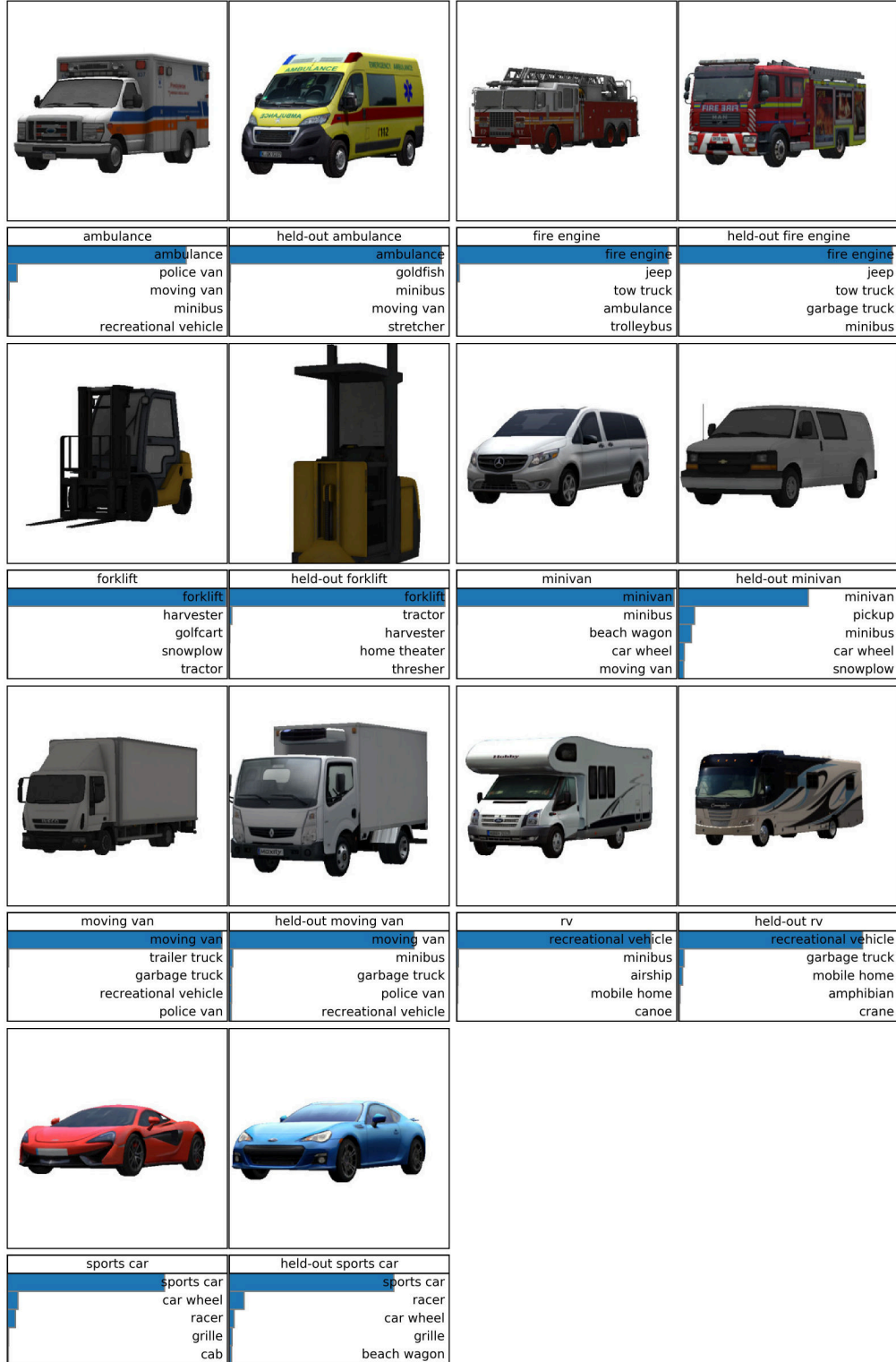
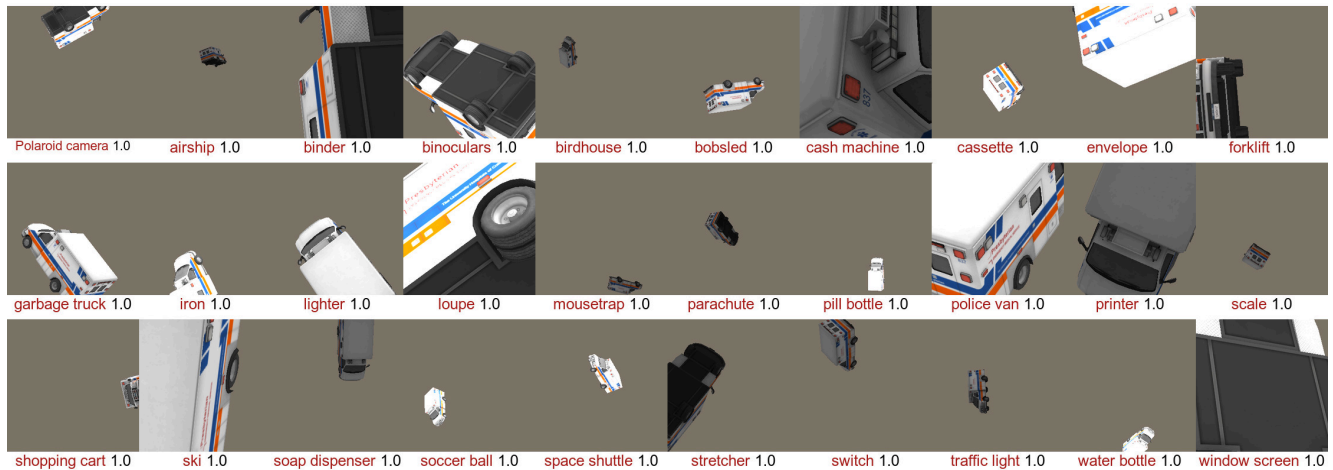
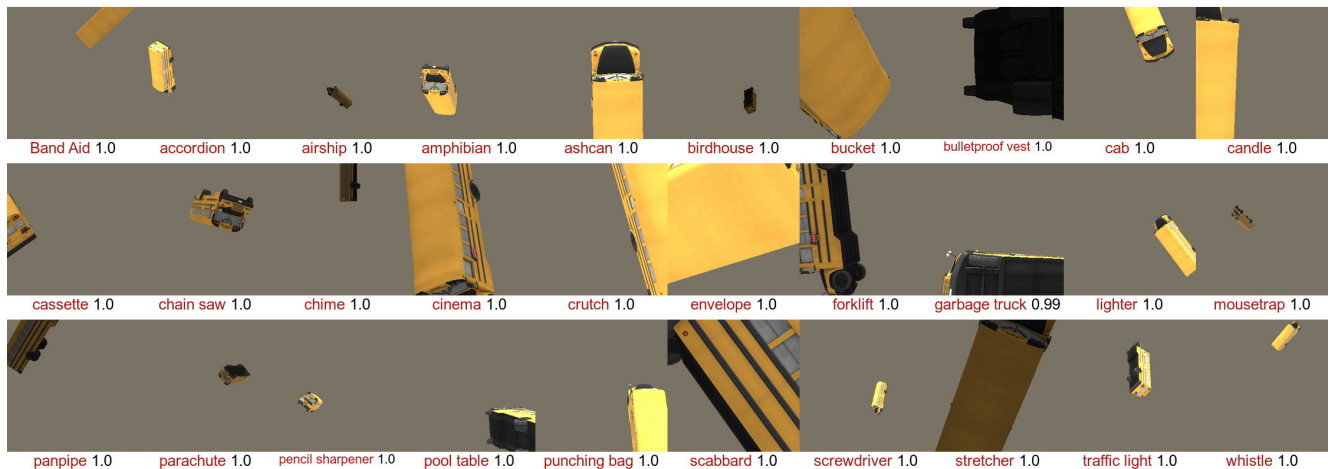


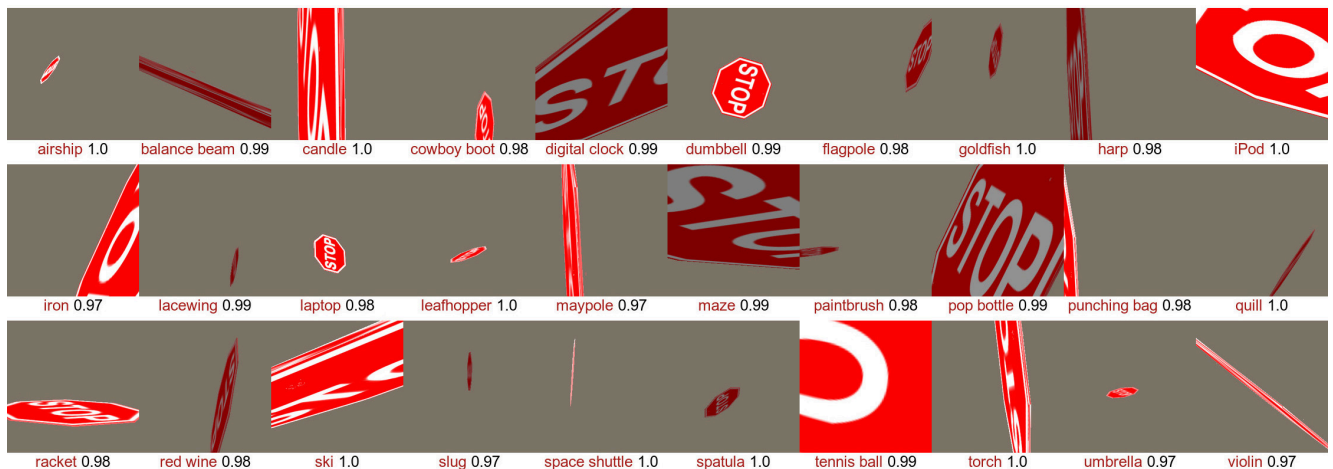
Figure S14: In Sec. S9, we trained an AlexNet classifier on the 1000-class ImageNet dataset augmented with 30 additional classes that contain adversarial poses corresponding to the 30 *known* objects used in the main experiments. We also tested this model on 7 *held-out* objects. Here, we show the renders 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/LileKU>.



(a) ambulance



(b) school bus

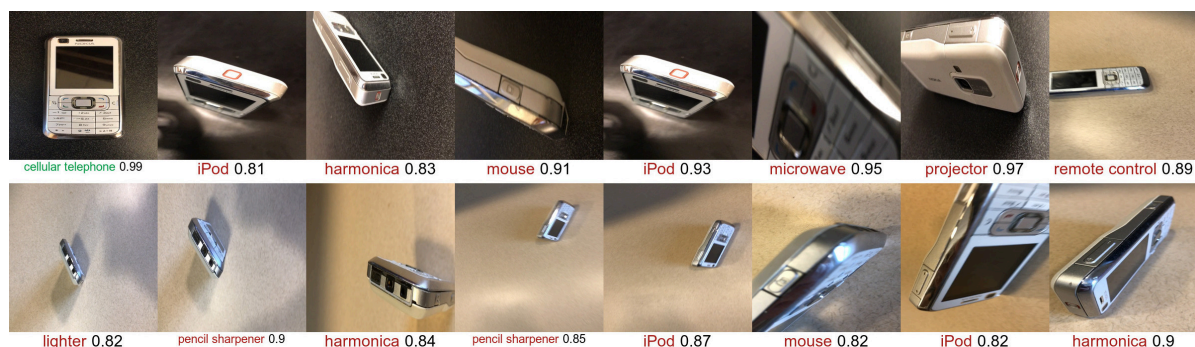


(c) street sign

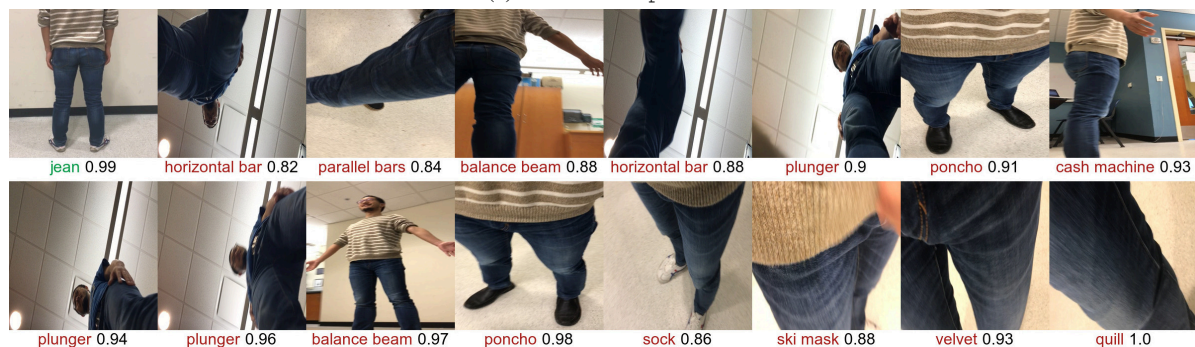
Figure S15: 30 random adversarial examples misclassified by Inception-v3 [44] with high confidence ($p \geq 0.9$) generated from 3 objects: ambulance, school bus, and street sign. Below each image is the top-1 prediction label and confidence score. The original, high-resolution figures for all 30 objects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



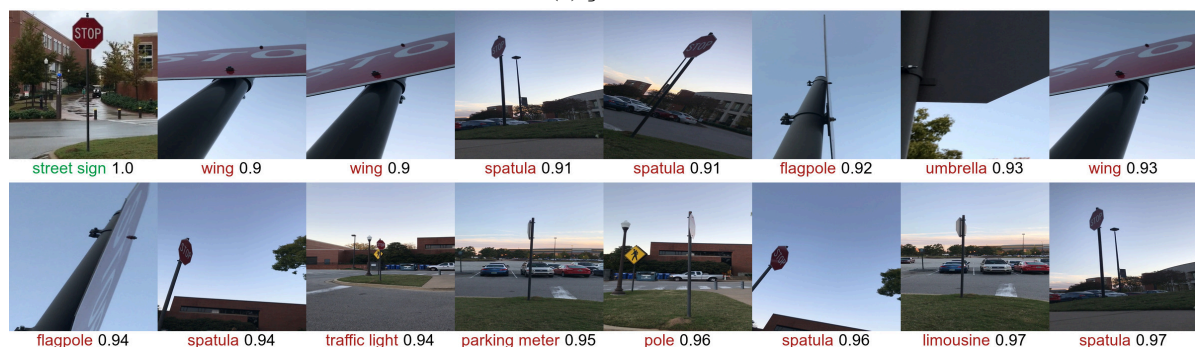
Figure S16: For each target class (e.g., accordion piano), we show five adversarial poses generated from five unique 3D objects. Adversarial poses are interestingly found to be homogeneous for some classes, e.g., safety pin. However, for most classes, the failure modes are heterogeneous. The original,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



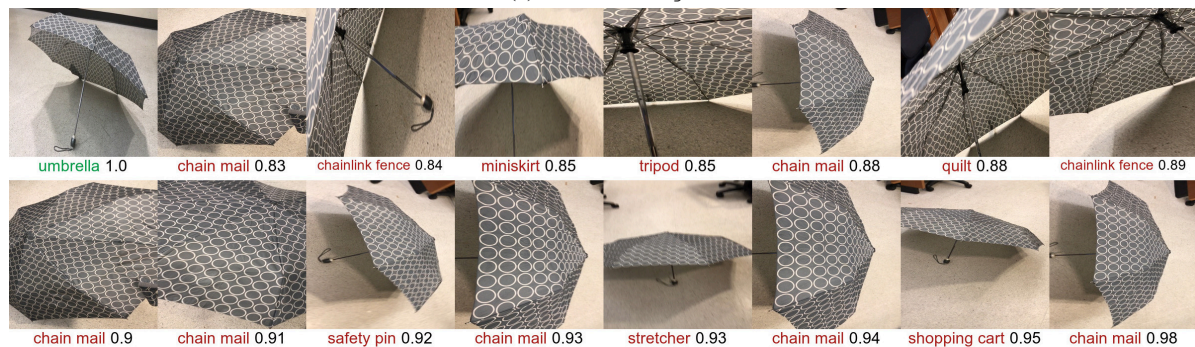
(a) cellular phone



(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>.

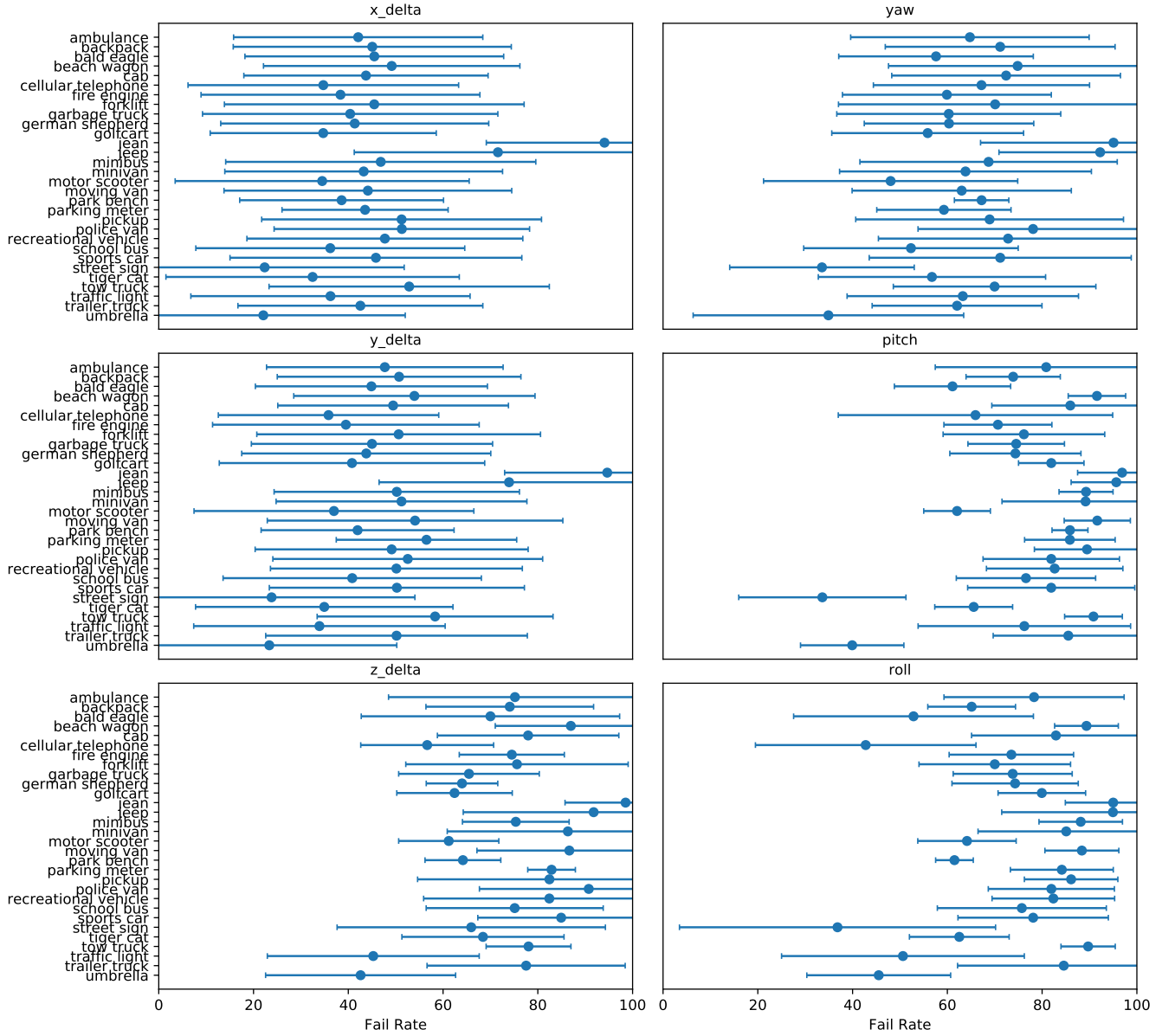


Figure S18: Inception-v3 [44] is sensitive to single parameter disturbances of object poses that had originally been correctly classified. For each object, we found 100 correctly classified 6D poses via a random sampling procedure (Sec. 4.3). Given each such pose, we re-sampled one parameter (shown on 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”).

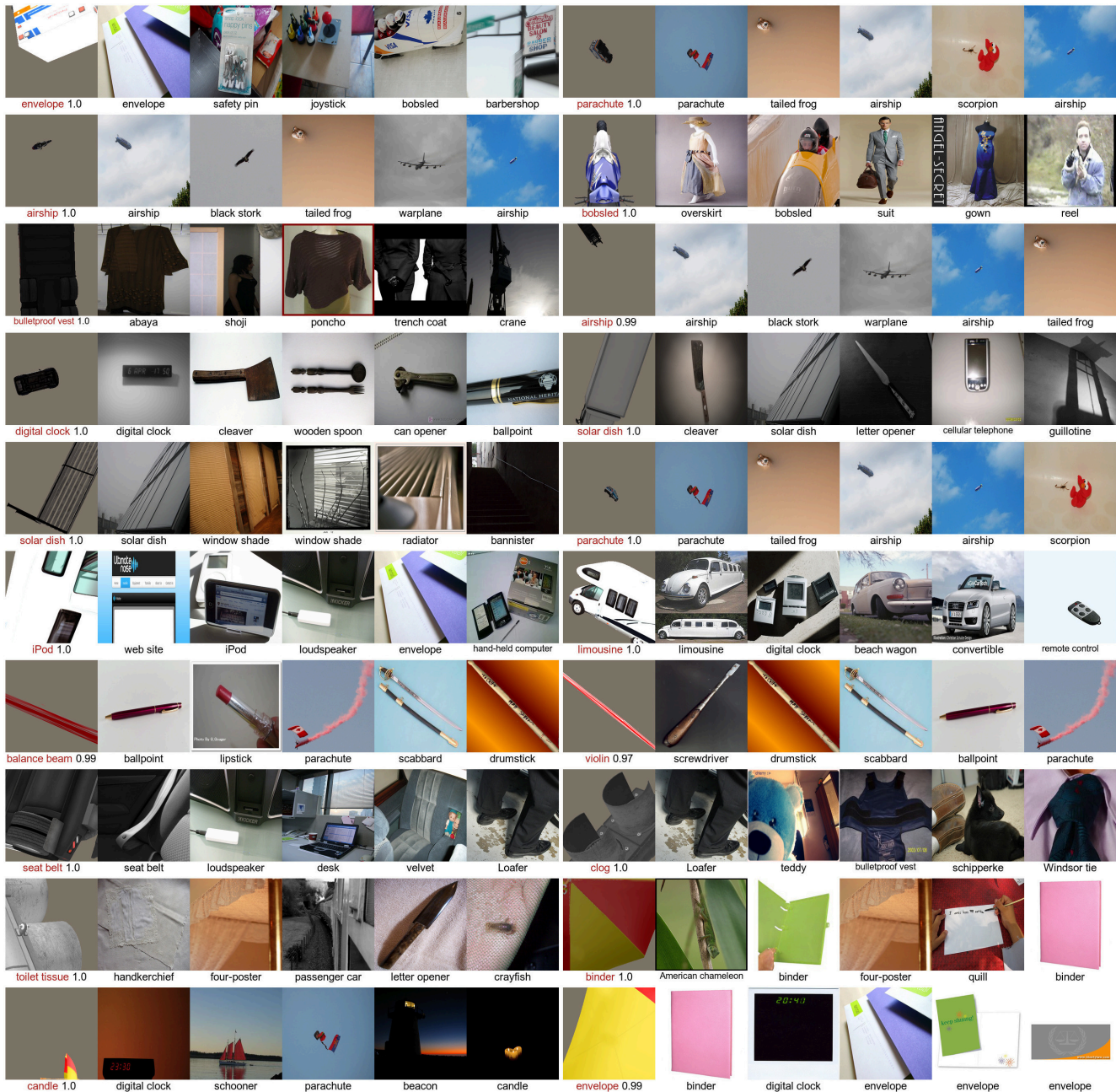
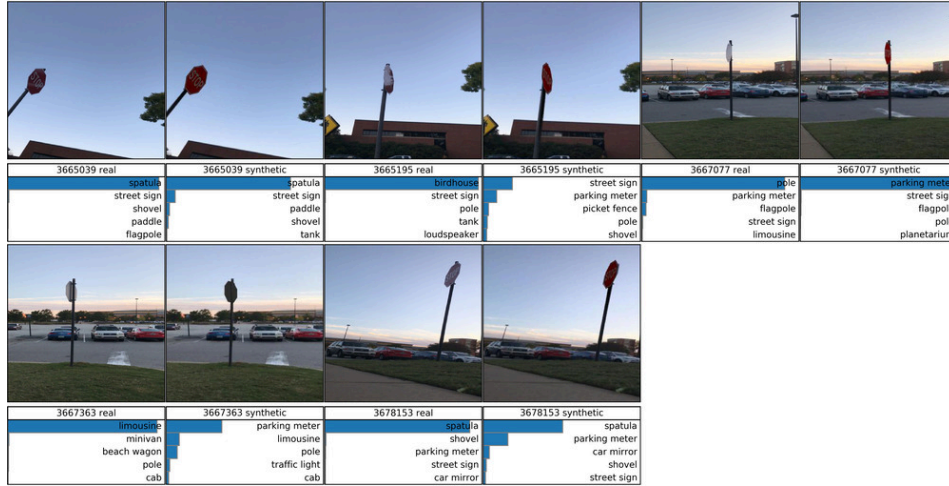


Figure S19: For each adversarial example (leftmost), we retrieved the five nearest neighbors (five rightmost photos) from the 50,000-image ImageNet validation set. The Euclidean distance between a pair of images was computed in the fc7 feature space of a pre-trained AlexNet [20]. Below each adversarial 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 \times 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-S9qO4dhE9toJbRR1AIWRcVqVH6Zsd>.