In-distribution adversarial attacks on object recognition models using gradient-free search.

Spandan Madan
Harvard University

Tomotake Sasaki
Fujitsu Limited*

Hanspeter Pfister
Harvard University

Tzu-Mao Li
UCSD

Xavier Boix
Fujitsu Research

Abstract

Neural networks are susceptible to small perturbations in the form of 2D rotations and shifts, image crops, and even changes in object colors. Past works attribute these errors to dataset bias, claiming that models fail on these perturbed samples as they do not belong to the training data distribution. Here, we challenge this claim and present evidence of the widespread existence of perturbed images within the training data distribution which networks fail to classify. We train models on data sampled from parametric distributions, and then search *inside* this data distribution to find such in-distribution adversarial examples. This is done using our gradient-free evolution strategies (ES) based approach which we call CMA-Search. Despite training with a large-scale (~ 0.5 million images), unbiased dataset of camera and light variations, CMA-Search can find a failure inside the data distribution in over 71% cases by perturbing the camera position. With lighting changes, CMA-Search finds misclassifications in 42% cases. These findings also extend to natural images from ImageNet and Co3D datasets. This phenomenon of in-distribution images presents a highly worrisome problem for artificial intelligence—they bypass the need for a malicious agent to add engineered noise to induce an adversarial attack. All code, datasets, and demos are available at `https://github.com/in-dist-adversarials/in_distribution_adversarial_examples`.

1 Introduction

Neural networks are highly susceptible to small perturbations—2D rotations and translations [1], image crops [2, 3], and even changes in the color space [4, 5, 6]. Building on works in the closely associated field of adversarial attacks, past works have claimed that these failures lie out of the training data distribution, attributing them to the dataset bias [7, 8, 9, 10, 11]. Here, we present evidence for the opposite—widespread presence of adversarial attacks that verifiably lie within the training data distribution. For this, we train and test classification models on data with parametrically controlled data distributions, and present a methodology to find in-distribution adversarial attacks. These experiments are enabled by our gradient-free, evolutionary strategies (ES) based approach for finding in-distribution adversarial examples, which we call CMA-Search.

We present results with CMA-Search across three levels of data complexity—(i) parametric data sampled from disjoint per-category uniform distributions, (ii) parametric and controlled data of rendered images, and (iii) natural image data from ImageNet and Co3D datasets. Across all datasets, models are highly susceptible to in-distribution adversarial attacks. CMA-Search can find in-distribution attacks for simplistic parametric data with a 100% attack rate—there existed a failure in the vicinity

*Tomotake Sasaki is currently affiliated with Tokyo Metropolitan, Chuo-Johoku Vocational Skills Development Center / Japan Electronics College.

AdvML-Frontiers'24: The 3nd Workshop on New Frontiers in Adversarial Machine Learning@NeurIPS'24, Vancouver, CA.

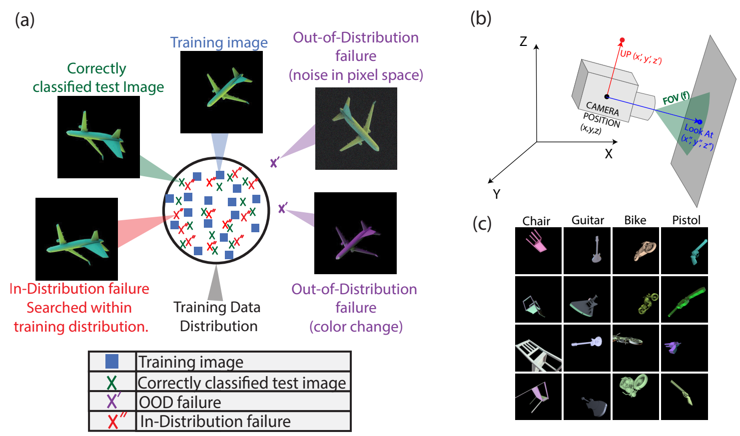

Figure 1: *In-distribution adversarial attacks.* (a) The data distribution (depicted in black) refers to the space of all camera and light variations. Typical adversarial examples are created by adding noise to the image, which may result in images out of the data distribution. CMA-Search finds failures inside the data distribution. (b) 3D scene setup for our rendered images with camera parameters illustrated. (c) Example images with camera and light variations.

of every single correctly classified test point. For rendered data, CMA-Search found failures in the vicinity of 71% correctly classified images by perturbing the camera position, and for 42% images by perturbing lighting parameters. With natural images from the Common Objects in 3D (Co3D) dataset [12], CMA-Search found in-distribution adversarial examples for over 51% images. Finally, we also employed CMA-Search in conjunction with a novel view synthesis pipeline [13] to find in-distribution adversarial examples in the vicinity of ImageNet [14].

2 Related Work

In efforts to combat susceptibility to small transformations [1, 15], crops [2, 3], and 2D rotations and translations [16], alternative architectures have been proposed which are shift invariant. This includes anti-aliasing networks using the seminal signal processing trick of anti-aliasing [17], and recently proposed truly shift invariant networks which use a new sampling methodology to guarantee a 100% consistency in classification under 2D shifts [18]. Unlike our work, these works have focused only on 2D transformations. Recent work has also sought to generate adversarial perturbations which are human interpretable i.e. semantic adversarial examples. These works often rely on synthetic data, using differentiable rendering or other optimization methods to find adversarial images by modifying scene parameters [19, 20, 21, 22, 23, 24, 25]. These include a custom differentiable renderer to perturb the camera, lighting, or object mesh vertices, and using a neural renderer where light is represented by network activations. They key differences between these works and ours is that our adversarial attacks are guaranteed to lie within the training distribution. While in-distribution attacks have been shown in theoretical works and for toy data [26, 27, 28, 29], this work provides the first evidence of such failures with real-world data to the best of our knowledge.

3 Datasets with explicitly controlled data distributions

3.1 Generating simplistic parametrically controlled data

We created a binary classification task by sampling data from two N-dimensional uniform distributions confined to disjoint ranges (a, b) and (c, d), as described in the following:

$$x_i \sim \left\{ \begin{array}{ll} \text{Unif}(a, b, N); & y_i = 0 \\ \text{Unif}(c, d, N); & y_i = 1 \end{array} \right\}. \tag{1}$$

We set $a = -10, b = 10, c = 20, d = 40$ for experiments presented. However, we observed that the exact choice of these parameters does not impact our findings.

Algorithm 1 *CMA-Search* over camera parameters to find in-distribution adversarial examples.

```
Let x ∈ ℝ^10 denote the camera parameters.
Let Render and Network denote the rendering pipeline and classification network respectively.
function FITNESS(x, Render, Network)
    image = Render(x)
    predicted_category, probability = Network(image)
    return predicted_category, probability
end function

x_init: initial camera parameters, λ: number of offspring per generation, and y: image category.
procedure CMA-SEARCH(x_init, λ, y)
    initialize μ = x_init, C = I                                  ▷ I denotes identity matrix.
    while True do
        for j = 1, ..., λ do
            x_j = sample_multivariate_normal(μ, C)                ▷ Generate mutated offspring
            y_j, p_j = FITNESS(x_j, Render, Network)              ▷ Calculate fitness of offspring
            if y_j ≠ y then
                return x_j          ▷ Classification fails for image with camera parameters x_j
            end if
        end for
        x_{1...λ} ← x_{s(1)...s(λ)}, with s(j) = argsort(p_j)     ▷ Pick best offspring
        μ, C ← update_parameters(x_{1...λ}, μ, C)
    end while
end procedure
```

3.2 Generating an unbiased training dataset of camera and light variations

Large-scale datasets for computer vision have mostly been created by scraping pictures from the internet [14, 30, 31, 32, 33]. However, investigating in-distribution robustness requires sampling new points from regions of interest within the data distribution, which is not possible with these datasets. To address this issue, we use a computer graphics pipeline for generating and modifying images which ensures complete parametric control over the data distribution. We simply sample camera and lighting parameters from a fixed, uniform distribution, and render a subset of 3D models from ShapeNet [34] objects with the sampled camera and lighting parameters. Sample images are shown in Fig. 1(c) and Fig. S1. All models were trained on 0.5 million rendered images across 11 categories, with 1000 images for every 3D model. Additional details are provided in Sec. S1.

3.3 Natural image datasets—ImageNet and Common Objects in 3D

As a real litmus test, we also ensure that our findings hold true for natural images. We present results on two popular natural image datasets—ImageNet [35] and the Common Objects in 3D (Co3D) [12] dataset. Co3D was created by capturing short videos of fixed objects placed on a surface by a user moving a mobile phone around the object. Thus, nearby frames in the video represent views in the vicinity of an image (See Sec. S2.2 for details). CMA-Search checks within $1-5$ frames of the correctly classified image to find a failure. For ImageNet, we used Novel View Synthesis (NVS) [13] to generate views in the vicinity of ImageNet images (See Sec. S2.1 for details). Thus, CMA-Search optimizes the camera parameters of the NVS model to find a perturbed image which is misclassified.

4 CMA-Search: Finding in-distribution failures by searching the vicinity

Most adversarial attack methods predominantly rely on on gradient-based approaches, such as Fast Gradient Sign Method (FGSM)[36] and Projected Gradient Descent (PGD)[37]. However, we observed that classification models were significantly robust against gradient based attacks in the camera and light parameter space. In particular, these gradient-based adversarial attacks only became effective at exceptionally large step sizes, at which point the approximate gradient no longer accurately represented the true gradient. Such irregularities made it challenging to construct adversarial examples through standard gradient-based methods, motivating a shift towards gradient-free optimization techniques.

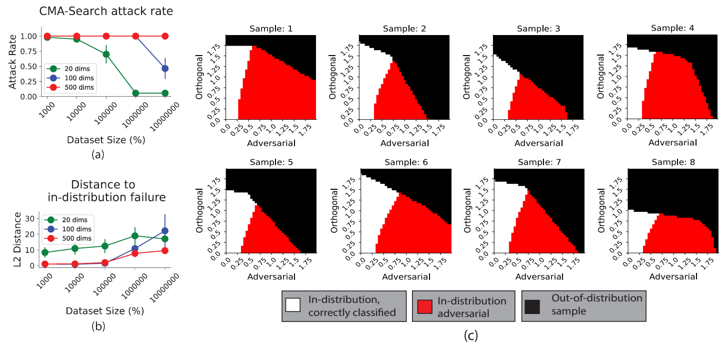

Figure 2: *In-distribution adversarial attacks on parametric data sampled from high-dimensional, disjoint uniform distributions.* (a) Attack rate measured using CMA-Search is 100% for all models—there exists an in-distribution failure in the vicinity of every correctly classified sample. Models become robust beyond a critical dataset size, but the data needed scales poorly with dimensionality. (b) Average Euclidean distance between the starting point and the identified in-distribution adversarial sample increases as dataset size increases. (c) Church window plots depicting adversarial examples (red) located contiguously and in between the learned and ground-truth boundaries.

To address this concern, We propose a gradient-free search method using Covariance Matrix Adaptation-Evolution Strategy (CMA-ES) to find in-distribution adversarial examples. We explain our methodology with an example of finding in-distribution adversarial attacks by searching within the distribution of camera parameters. The algorithm for searching adversarial attacks in light space, and for all other datasets is analogous. Algorithm 1 provides an outline for the method which was implemented using pycma [38, 39].

Starting from the initial camera parameters of the scene, CMA-ES generates offspring by sampling from a multivariate normal (MVN) distribution i.e. mutating the original parameters. These offspring are sorted based on the fitness function ($1 - p$, where p denotes classification probability). The best offspring are used to modify the mean and covariance matrix of the MVN for the next generation. The mean represents the current best estimate of the solution i.e. the maximum likelihood solution, while the covariance matrix dictates the direction in which the population should be directed in the next generation. The search is stopped either when a misclassification occurs, or after 15 iterations.

5 Results

5.1 In-distribution adversarial attacks on uniformly distributed data

Fig. 2(a) reports the attack rate for models—the percentage of correctly classified points for which we successfully found an in-distribution failure using CMA-Search. Despite a near perfect accuracy on a held-out test set, in-distribution adversarial examples can be identified in the vicinity of all correctly classified test points—the attack rate is 100% for models trained with 20, 100 and 500 dimensional data. Note that this simplistic dataset is easily separable by the simplest of models including a decision tree. However, DNNs trained on this dataset are plagued by in-distribution failures.

Impact of dataset size: The attack rate start dips once a critical dataset size is reached (Fig. 2(a)). However, data complexity scales poorly with number of dimensions. As dimensionality grows from 20 to 100, the number of points required for robustness scales almost 100-fold. For 500 dimensions even 10 million training points were not sufficient.

Impact of robust training: We fine-tuned models on $20,000$ in-distribution adversarial examples found using CMA-Search for 100 dimensional data. The attack rate stayed at 100%, with no improvement in model robustness against CMA-Search. This is expected, as our identified adversarial examples lie within the training distribution. Thus, robust training in this case essentially amounts to a marginal increase in the training dataset size which is already discussed above.

Fig. 2(b) reports the average distance between the (correctly classified) start point and the closest in-distribution adversarial example identified using CMA-Search. This distance increased with

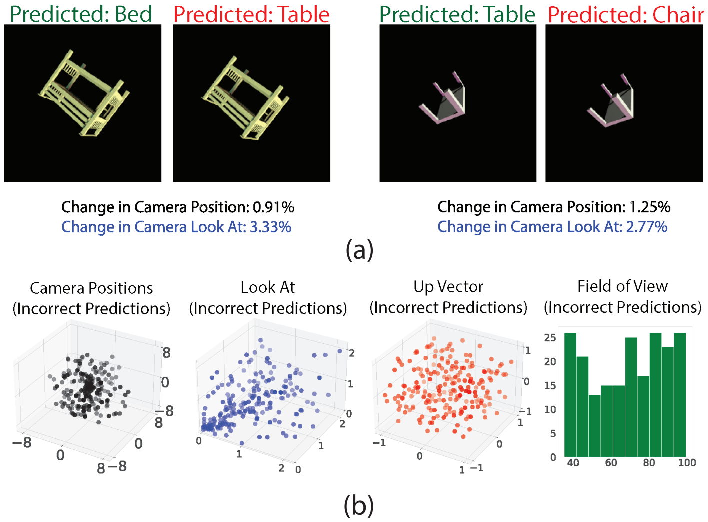

Figure 3: *In-distribution adversarial attacks in the camera parameter space.* a) Sample in-distribution adversarial examples. Percentage of change in Camera Position and Camera Look At parameters needed to induce the misclassification are also reported. Attack rates are reported in Table 1. (b) Distribution of camera parameters for in-distribution adversarial images. Unlike human vision, there were no clear patterns characterizing the camera and light conditions of misclassified images.

Model Architecture	CMA Cam		CMA Light	
	Attack Rate (%)	Distance (mean ± std)	Attack Rate (%)	Distance (mean ± std)
ResNet18 [41]	71	1.83 ± 1.33	42	6.52 ± 5.68
Anti-Aliased Networks [17]	45	2.32 ± 2.09	40	7.03 ± 5.10
Truly Shift Invariant Network [18]	53	2.22 ± 2.16	25	6.72 ± 5.41
ViT [42]	85	1.34 ± 1.16	65	4.63 ± 3.49
DeIT [43]	85	1.27 ± 0.81	51	4.54 ± 2.75
DeIT Distilled [43]	86	1.22 ± 0.87	55	4.49 ± 2.27

Table 1: *Attack Rates for models attacked with CMA-Search over camera and light parameters.* CMA-Search starts with correctly classified images, and searches the space of camera and light parameters to find an in-distribution misclassification. The attack rate reports percentage of correctly classified images for which CMA-Search found a failure. The change in parameter space (mean distance) required to induce an error is extremely small, highlighting the brittleness of these models.

dataset size. At critical dataset sizes, adversarial examples are far enough from starting points that they are now not in-distribution. This results in the dip in the attack rate shown in Fig. 2(a).

Visualizing failures: Fig. 2(c) shows the learned decision boundary using church window plots [40] (see S3.3 for details). Intriguingly, there is a clean transition from correctly classified points (white) to in-distribution adversarial examples near the decision boundary (red), beyond which points become out of the distribution (black). Thus, in-distribution adversarial examples are isolated to a region close to the category boundary, and in a contiguous fashion. This finding has been theorized [27, 28, 26, 29], but to the best of our knowledge this is the first empirical evidence for this phenomenon.

5.2 Networks struggle to generalize across camera and light variations

Table 1 reports in-distribution adversarial attacks identified by CMA-Search. For 71% images correctly classified by a ResNet, there lies an in-distribution failure within a 1.83% change in the camera position. For transformers, the impact is far worse with an Attack Rate of 85%. We hypothesize that transformers are more susceptible to in-distribution adversarial attacks due to their tendency to overfit, particularly when trained under data limited regimes—a well-documented challenge that stems from their higher model capacity and lack of inductive biases [44, 42].

Table 2: *Results with Co3D dataset.* All models suffer from high attack rates, confirming the widespread presence of in-distribution failures for object recognition models.

	ResNet	Anti-Aliased Networks	ViT	DeIT
Test Accuracy	0.92	0.94	0.82	0.85
Attack Rate	0.51	0.39	0.72	0.72

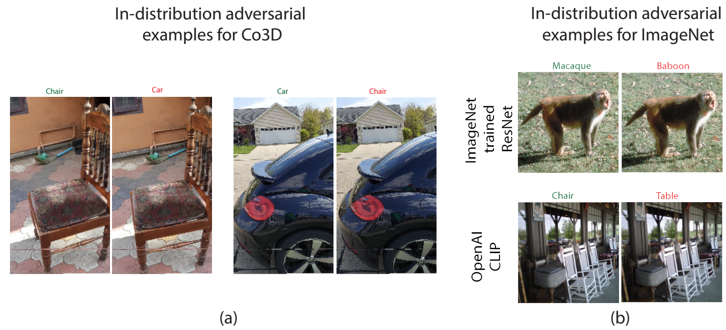

Figure 4: *In-distribution adversarial attacks on natural images.* (a) Misclassifications in ImageNet caused by CMA-Search + novel view synthesis. Examples are presented for a ResNet model trained on ImageNet, and OpenAI's CLIP model. (b) Sample errors for the Co3D dataset searched within $1-5$ frames of a correctly classified image. Attack rates are reported in Table 2.

For lighting changes, CMA-Search can find a misclassification in 42% cases with just a 6.5% change. The supplement presents additional results (See Sec. S2) and clean accuracies for these models (Table S3). Combined, these results confirm that object recognition models are plagued by in-distribution adversarial attacks. Fig. 3(b) shows the distribution of parameters for failures—errors are distributed across the space with no clear, strong patterns characterizing failures.

6 Results on Natural Image Data

Results on Co3D: Table 2 reports the average accuracy and attack rate for models trained on Co3D. Despite a high test accuracy of 92%, a ResNet model suffered from an attack rate of 51%. Thus, there were in-distribution adversarial examples within 1-5 frames of the correctly classified frame for over half the images. Sample failures are provided in Fig. 4(a) Transformers struggled even more, with ViT and DeIT having an attack rate near 72%. The shift invariant architecture was more robust, but attack rate was still high at 39% (see Table 2). These trends are consistent with the results in Table 1.

Results on ImageNet: We also confirmed that these results extend to ImageNet. We present empirical results for a ResNet18 model trained on ImageNet, and OpenAI's transformer-based CLIP model [45] in Fig. 4(b). Additional ImageNet failures found using CMA-Search are provided in Fig. S3.

7 Conclusions

Susceptibilities of recognition models have often been attributed to biased training data. We put this hypothesis to test by training and testing with a large-scale, unbiased dataset and propose a new search method for investigating the brittleness of neural networks. Our findings show that while data augmentation, unbiased datasets, and specialized shift-invariant architectures would certainly be helpful, the real problem runs far deeper. Despite high test accuracies, networks are plagued by adversarial examples that lie within the training distribution. This presents a grave challenge for AI, as these errors are hiding in plain sight, with no malicious agent needed to induce an error.

8 Acknowledgements

This research was partially supported by Fujitsu Limited (Contract No. 40009105), NSF grant IIS-1901030, IIS-2127544, and NIH R01HD104969 grant.

References

[1] Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. A rotation and a translation suffice: Fooling CNNs with simple transformations. `https://openreview.net/forum?id=BJfvknCqFQ`, 2018.

[2] Sanjana Srivastava, Guy Ben-Yosef, and Xavier Boix. Minimal images in deep neural networks: Fragile object recognition in natural images. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[3] Aharon Azulay and Yair Weiss. Why do deep convolutional networks generalize so poorly to small image transformations? *Journal of Machine Learning Research*, 20(184):1–25, 2019.

[4] Jeet Mohapatra, Tsui-Wei Weng, Pin-Yu Chen, Sijia Liu, and Luca Daniel. Towards verifying robustness of neural networks against a family of semantic perturbations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 244–252, 2020.

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

[6] Ali Shahin Shamsabadi, Ricardo Sanchez-Matilla, and Andrea Cavallaro. Colorfool: Semantic adversarial colorization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1151–1160, 2020.

[7] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. In *Advances in Neural Information Processing Systems*, 2019.

[8] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Advances in neural information processing systems*, 31, 2018.

[9] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280*, 2017.

[10] Naveen Karunanayake, Ravin Gunawardena, Suranga Seneviratne, and Sanjay Chawla. Out-of-distribution data: An acquaintance of adversarial examples–a survey. *arXiv preprint arXiv:2404.05219*, 2024.

[11] David Stutz, Matthias Hein, and Bernt Schiele. Disentangling adversarial robustness and generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6976–6987, 2019.

[12] Jeremy Reizenstein, Roman Shapovalov, Philipp Henzler, Luca Sbordone, Patrick Labatut, and David Novotny. Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10901–10911, 2021.

[13] Richard Tucker and Noah Snavely. Single-view view synthesis with multiplane images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 551–560, 2020.

[14] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, 2009.

[15] Leander Kurscheidt and Matthias Hein. Lost in translation: Modern image classifiers still degrade even under simple translations. *ICML Shift Happens Workshop*, 2022.

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

[17] Richard Zhang. Making convolutional networks shift-invariant again. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 7324–7334, 2019.

[18] Anadi Chaman and Ivan Dokmanić. Truly shift-invariant convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3773–3783, 2021.

[19] Hsueh-Ti Derek Liu, Michael Tao, Chun-Liang Li, Derek Nowrouzezahrai, and Alec Jacobson. Beyond pixel norm-balls: Parametric adversaries using an analytically differentiable renderer. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[20] Xiaohui Zeng, Chenxi Liu, Yu-Siang Wang, Weichao Qiu, Lingxi Xie, Yu-Wing Tai, Chi-Keung Tang, and Alan L Yuille. Adversarial attacks beyond the image space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4302–4311, 2019.

[21] Rakshith Shetty, Mario Fritz, and Bernt Schiele. Towards automated testing and robustification by semantic adversarial data generation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 489–506, 2020.

[22] Lakshya Jain, Steven Chen, Wilson Wu, Uyeong Jang, Varun Chandrasekaran, Sanjit Seshia, and Somesh Jha. Generating semantic adversarial examples with differentiable rendering. `https://openreview.net/forum?id=SJlRF04YwB`, 2019.

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

[24] Ameya Joshi, Amitangshu Mukherjee, Soumik Sarkar, and Chinmay Hegde. Semantic adversarial attacks: Parametric transformations that fool deep classifiers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4773–4783, 2019.

[25] Philip Yao, Andrew So, Tingting Chen, and Hao Ji. On multiview robustness of 3D adversarial attacks. In *Practice and Experience in Advanced Research Computing*, pages 372–378. 2020.

[26] Justin Gilmer, Luke Metz, Fartash Faghri, Samuel S Schoenholz, Maithra Raghu, Martin Wattenberg, and Ian Goodfellow. The relationship between high-dimensional geometry and adversarial examples. arXiv preprint, arXiv:1801.02774, 2018.

[27] Alhussein Fawzi, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Robustness of classifiers: from adversarial to random noise. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

[28] Alhussein Fawzi, Hamza Fawzi, and Omar Fawzi. Adversarial vulnerability for any classifier. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

[29] Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Analysis of classifiers' robustness to adversarial perturbations. *Machine learning*, 107(3):481–508, 2018.

[30] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 740–755, 2014.

[31] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[32] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, 2013.

[33] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[34] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An information-rich 3D model repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

[35] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[36] Ian J Goodfellow. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[37] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[38] Nikolaus Hansen and Andreas Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *Proceedings of IEEE International Conference on Evolutionary Computation*, pages 312–317, 1996.

[39] Nikolaus Hansen, Youhei Akimoto, and Petr Baudis. CMA-ES/pycma on Github. Zenodo, DOI:10.5281/zenodo.2559634, February 2019.

[40] David Warde-Farley and Ian Goodfellow. Adversarial perturbations of deep neural networks. In Tamir Hazan, George Papandreou, and Daniel Tarlow, editors, *Perturbations, Optimization, and Statistics*, pages 311–342. MIT Press, Cambridge, MA, USA, 2016.

[41] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[42] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[43] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pages 10347–10357, 2021.

[44] Andreas Steiner, Alexander Kolesnikov, Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer. How to train your vit? data, augmentation, and regularization in vision transformers. *arXiv preprint arXiv:2106.10270*, 2021.

[45] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pages 8748–8763, 2021.

[46] Radoslav Harman and Vladimír Lacko. On decompositional algorithms for uniform sampling from n-spheres and n-balls. *Journal of Multivariate Analysis*, 101(10):2297–2304, 2010.

[47] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[48] Jae Shin Yoon, Kihwan Kim, Orazio Gallo, Hyun Soo Park, and Jan Kautz. Novel view synthesis of dynamic scenes with globally coherent depths from a monocular camera. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5336–5345, 2020.

[49] Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, and Alexei A Efros. View synthesis by appearance flow. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 286–301, 2016.

[50] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. Synsin: End-to-end view synthesis from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7467–7477, 2020.

S1 Graphics pipeline to generate dataset of camera and lighting variations

S1.1 3D Scene Setup

Each scene contains one camera, one 3D model and 1-4 lights. To ensure no spuri- ous correlations with object texture [17], texture for all ShapeNet objects was replaced with a simple diffuse material and the background was kept constant to ensure no spurious correlations between foreground and background. Thus, every scene is completely parametrized by the camera and the light parameters. As shown in Fig. 1, camera parameters are 10 Dimensional: one dimension for the FOV (field of view of camera lens), and three dimensions each for the Camera Position (coordi- nates of camera center), Look At (point on the canvas where the camera looks), and the UP vector (rotation of camera). Analogously, lights are represented by 11 dimensions - two dimensions for the Light Size, and three each for Light Position, light Look At and RGB color intensity. Multiple lights ensure that scenes contain complex mixed lighting, including self-shadows. Thus, our scenes are (11n + 10) dimensional, where n is the number of lights. There is a one-to-one mapping between the pixel space (rendered images) and this low dimensional scene representation.

S1.2 Unbiased, uniform sampling of scene parameters

To ensure an unbiased distribution over different viewpoints, locations on the frame, perspective projections and colors, we ensured that scene parameters follow a uniform distribution. Concretely, camera and light positions were sampled from a uniform distribution on a spherical shell with a fixed minimum and maximum radius. The Up Vector was uniformly distributed across range of all possible camera rotations, and RGB light intensities were uniformly distributed across all possible colors. Camera and light Look At positions were uniformly distributed while ensuring the object stays in frame and is well-lit (frame size depends on Camera Position and FOV). Finally, Light Size and camera FOV were uniformly sampled 2D and 1D vectors. Hyper-parameters for rendering, along with the exact distribution for each scene parameter and the corresponding sampling technique used to sample from these distributions are reported in the supplement.

Below we specify the hyper-parameters for rendering, along with the exact distribution for each scene parameter and the corresponding sampling technique used to sample from these distributions.

Camera Position: For scene camera, first a random radius r_c is sampled while ensuring $r_c \sim \text{Unif}(0.5, 8)$. Then, the camera is placed on a random point denoted (x_c, y_c, z_c) on the spherical shell of radius r_c. To generate a random point on the sphere while ensuring an equal probability of all points, we rely on the method which sums three randomly sampled normal distributions [46]:

$$X, Y, Z \sim \mathcal{N}(0, 1), \tag{2}$$

$$v = (X, Y, Z), \tag{3}$$

$$(x_c, y_c, z_c) = r_c * \frac{v}{\|v\|}. \tag{4}$$

Camera Look At: To ensure the object is shown at different locations within the camera frame, the camera Look At needs to be varied. However, range of values such that the object is visible can be present across the entire range of the frame depends on the camera position. So, we sample camera Look At as l_c as follows:

$$l_c \sim \text{Unif}(K * x_c, K * y_c, K * z_c), \text{where } K = 0.3. \tag{5}$$

The value $K = 0.3$ was found empirically. We found it helped ensure that objects show up across the whole frame while still being completely visible within the frame.

Camera Up Vector: Note that the camera Up Vector is implemented as the vector joining the camera center (0,0,0) to a specified position. We sample this position and therefore the Up Vector u_c as follows:

$$x, y, z \sim \text{Unif}(-1, 1), \tag{6}$$

$$u_c = (x, y, z). \tag{7}$$

Camera Field of View (FOV): We sample the field of view f_c while ensuring:

$$f_c \sim \text{Unif}(K_1, K_2). \tag{8}$$

Again, the values $K_1 = 35, K_2 = 100$ were found empirically to ensure objects are completely visible within the frame while not being too small.

Light Position: For every scene we first sample the number of lights n between 1-4 with equal probability. For each light i, a random radius r_i is sampled ensuring $r_i \sim \text{Unif}(R_1, R_2)$, then the light is placed on a random point (x_i, y_i, z_i) on the sphere of radius r_i. $R_1 = 1$ and $R_2 = 8$ were found empirically to ensure that the light is able to illuminate the 3D model appropriately.

Light Look At: To ensure that the light is visible on the canvas, light Look At is sampled as a function of the camera position:

$$l_i \sim \text{Unif}(K * x_c, K * y_c, K * z_c), \text{where } K = 0.3. \tag{9}$$

As in the case of the Camera Look At parameter mentioned above, the value $K = 0.3$ was found empirically.

Light Size: Every light in our setup is implemented as an area light, and therefore requires a height and width to specify the size. We generate the size s_i for light i as:

$$h, w \sim \text{Unif}(L_1, L_2), \tag{10}$$

$$s_i = (h, w). \tag{11}$$

$L_1 = 0.1, L_2 = 5$ were found empirically to ensure the light illuminates the objects appropriately.

Light Intensity: This parameter specifies the RGB intensity of the light. For light i, RGB color intensity c_i was sampled as:

$$r, g, b \sim \text{Unif}(0, 1), \tag{12}$$

$$c_i = (r, g, b). \tag{13}$$

Object Material: To ensure no spurious correlations between object texture and category, all object textures were set to a single diffuse material. Specifically, the material is a linear blend between a Lambertian model and a microfacet model with Phong distribution, with Schilick's Fresnel approximation. Diffuse reflectance was set to 1.0, and the material was set to reflect on both sides.

S1.3 3D models used for generating two different test sets

Our dataset contains 11 categories, with 40 3D models for every category chosen from ShapeNet [34]. Neural networks were evaluated on two test sets - one with the 3D models seen during training, and the second with new, unseen 3D models. The first test set was generated by simply repeating the same procedure as described above. Thus, the *(Geometry* $\times$ *Camera* $\times$ *Lighting)* joint distribution matches exactly for the train set and this test set. The second test set was created by the exact same generation procedure, but with 10 new 3D models for every category chosen from ShapeNet. The motivation for this second test set was to ensure our models are not over-fitting to the 3D models used for training. Thus, the *(Camera* $\times$ *Lighting)* joint distribution matches exactly for this test set and the train set, but the *Geometry* is different in these two sets.

S2 Generating nearby views for Natural Image Datasets

S2.1 Views in the vicinity of ImageNet images

ImageNet contains only one viewpoint per object. While several variations of ImageNet have been proposed by adding noise in the form of corruptions and perturbations [47], these variations are designed to study the impact of out-of-distribution shifts on object recognition models. Like these variations, our camera manipulations correspond to transforming input images to study its impact on object recognition models. However, the key difference is that our work focuses on in-distribution adversarial examples, due to which these datasets designed for out-of-distribution shifts cannot be repurposed for our experiments. Thus, a major challenge in extending our results to ImageNet is generating natural images in the vicinity of a correctly classified image by slightly modifying the camera parameters. To do so for ImageNet is equivalent to novel view synthesis (NVS) from single images, which has been a long-standing challenging task in computer vision.

Table S3: Performance of object recognition models on seen and new 3D models.

Accuracy	ResNet	Anti-Aliased	Truly Shift Invariant	ViT	DeIT	DeIT Distilled
Seen models	0.75	0.82	0.80	0.58	0.63	0.64
New models	0.70	0.74	0.72	0.59	0.64	0.65

However, recent advances in NVS enable us to extend our method to natural image datasets like ImageNet [48, 49, 50, 13].

To generate new views in the vicinity of ImageNet images, we rely on a single-view synthesis model based on multi-plane images (MPI) [13]. The MPI model takes as input an image and the (x, y, z) offsets which describe camera movement along the X, Y and Z axes. Note that unlike our renderer, it cannot introduce changes to the camera Look At, Up Vector, Field of View or lighting changes. An important limitation of this approach is that any noise added by the MPI model in image generation is a confounding variable which we cannot account for. This further highlights the importance of our rendered and Co3D experiments as these experiments do not suffer from such noise.

S2.2 Views in the vicinity of Co3D images

As an additional control for any potential noise introduced by the novel view synthesis pipeline in generating nearby views for ImageNet images, we present additional results on the large-scale, multi-viewpoint Co3D [12] dataset. Co3D was created by capturing short videos of fixed objects placed on a surface by a user moving a mobile phone around the object. Thus, nearby frames in the video represent views in the vicinity of an image. We utilize this to test in-distribution robustness in the vicinity of correctly classified images. The classification dataset is created by picking 5 categories—car, chair, handbag, laptop, and teddy bear. We created the training data by uniformly sampling frames across the whole video for all videos for these categories amounting to $187,200$ training images. Note that this amounts to roughly $38,000$ images per category, which is 32 times the ImageNet training set on a per category basis. An in-distribution test set of $68,854$ images is generated by sampling the remaining frames to measure overall accuracy of the trained models. We then search for in-distribution failures in the vicinity (i.e., nearby frames) from the remaining frames from these videos in the Co3D dataset. Thus, no novel view synthesis pipeline was used. Instead, pre-captured frames from the videos were used to search for in-distribution adversarial examples in the vicinity of viewpoints.

S3 Experimental Details

Below we provide the training details including model architectures, optimization strategies and other hyper-parameters used for the binary classification models trained on simplistic parametrically controlled data, and the object recognition models trained on our rendered images of camera and light variations. All code to run these experiments can be found at `https://github.com/in-dist-adversarials/in_distribution_adversarial_examples`.

S3.1 Training details for MLPs for classifying parametrically controlled uniform data

Let D denote the dimensionality of the input data, and N denote the total number of data points. We used a 5 layer multi-layer perceptron (MLP) with ReLU activations, with the output dimensionality of layers set to $5D$, D, $D/5$, $D/5$, and 2 respectively. However, we found that the number of MLP hidden layers and the number of neurons in these layers had no impact on trends of in-distribution robustness. For experiments with $N < 64,000$ all data was passed in a single batch. For experiments with more data points, each batch contained $64,000$ points. All models were trained for 100 epochs with stochastic gradient descent (SGD) with a learning rate of 0.0001. All experiments were conducted on a compute cluster consisting of 8 NVIDIA TeslaK80 GPUs, and all models were trained on a single GPU at a time. Only models achieving a near perfect accuracy (> 0.99)[2] on a held-out test set were attacked using *CMA-Search*.

[2] Except when dataset size=1000 and dimensions=100 or 500. In these two case the training data was too small for a high test accuracy. These cases are still included for completion.

S3.2 Training details for Object recognition models for classifying images of real-world objects

All CNN models were trained with a batch size of 75 images, while transformers were trained with a batch size of 25. Models were trained for 50 epochs with an Adam optimizer with a fixed learning rate of 0.0003. Other learning rates including 0.0001, 0.001, 0.01 and 0.1 were tried but they performed either similarly well or worse. To get good generalization to unseen 3D models and stable learning, each image was normalized to zero mean and unit standard deviation. As before, all experiments were conducted on our cluster with TeslaK80 GPUs, and each model was trained using a single GPU at a time.

S3.3 Visualizing in-distribution adversarial examples using Church-window plots

CMA-Search starts from a correctly classified point and provides an in-distribution adversarial example. We used these two points to define a unit vector in the adversarial direction, and fixed this as one of basis vectors for the space the data occupies. As data dimensionality was D, we calculated the remaining $D - 1$ orthonormal bases. Following the same protocol as past work [40], we randomly picked one of these orthonormal vectors as the orthogonal direction and defined a grid of perturbations with fixed increments along the adversarial and the orthogonal directions. These perturbations were then added to the original sample and the model was evaluated at these perturbed samples. We plotted correct classifications in white, in-distribution adversarial examples in red, and out-of-distribution samples in black.

S4 Computational efficiency of CMA-Search

CMA-Search operates iteratively, generating multiple offsprings in every iteration, and retaining the best in every iteration to calculate parameters for the next iteration. For simplistic parametrically controlled, CMA-Search was set to generate 20 offsprings in every iteration, and the search algorithm was set to stop when an in-distribution adversarial example is found, or if a maximum threshold of 1500 iterations were hit. On average, 51 iterations were needed to find an in-distribution adversarial example for 10 dimensional data. The average number of iterations needed dropped to 20 for 100 dimensional data. Note that as dimensionality increases, all steps become more computationally intensive, this includes training models, generating new offsprings using CMA-Search, and model inference to test offspring fitness. Thus, overall time required to attack increases with dimensionality. However, computational efficiency of CMA-Search improves with dimensionality, as lesser iterations are needed.

For rendered data, which is significantly higher dimensional, we found that CMA-Search is very efficient as extremely low number of iterations are needed to find an in-distribution failure. For both camera and light variation based attacks, CMA-Search was set to generate 10 offsprings in every iteration, and maximum iteration threshold was set to 15. On average, only 2 iterations were needed to find an in-distribution failure with camera variations. For light variations, 3.5 iterations were required on average. This suggests that CMA-Search is more efficient at higher dimensions, despite working well at low dimensions.

Figure S1: *Sample Images from our rendered dataset.*

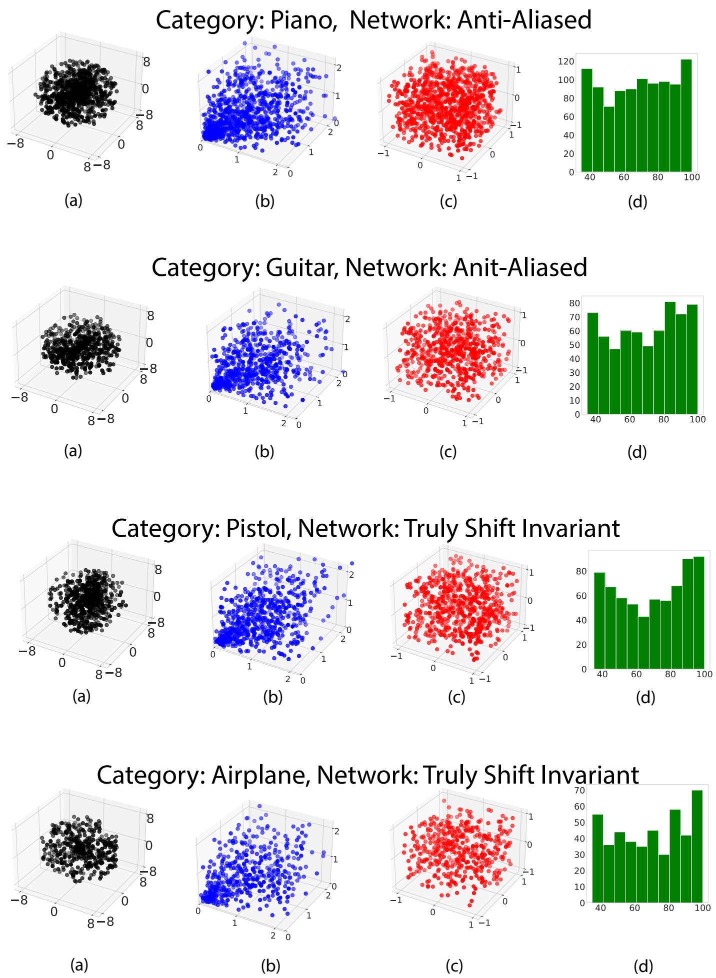

Figure S2: *Camera Parameters that lead to misclassifications for multiple categories and architectures.* (a) Camera Position, (b) Camera Look At, (c) Up Vector, (d) Histogram of Lens Field of View.

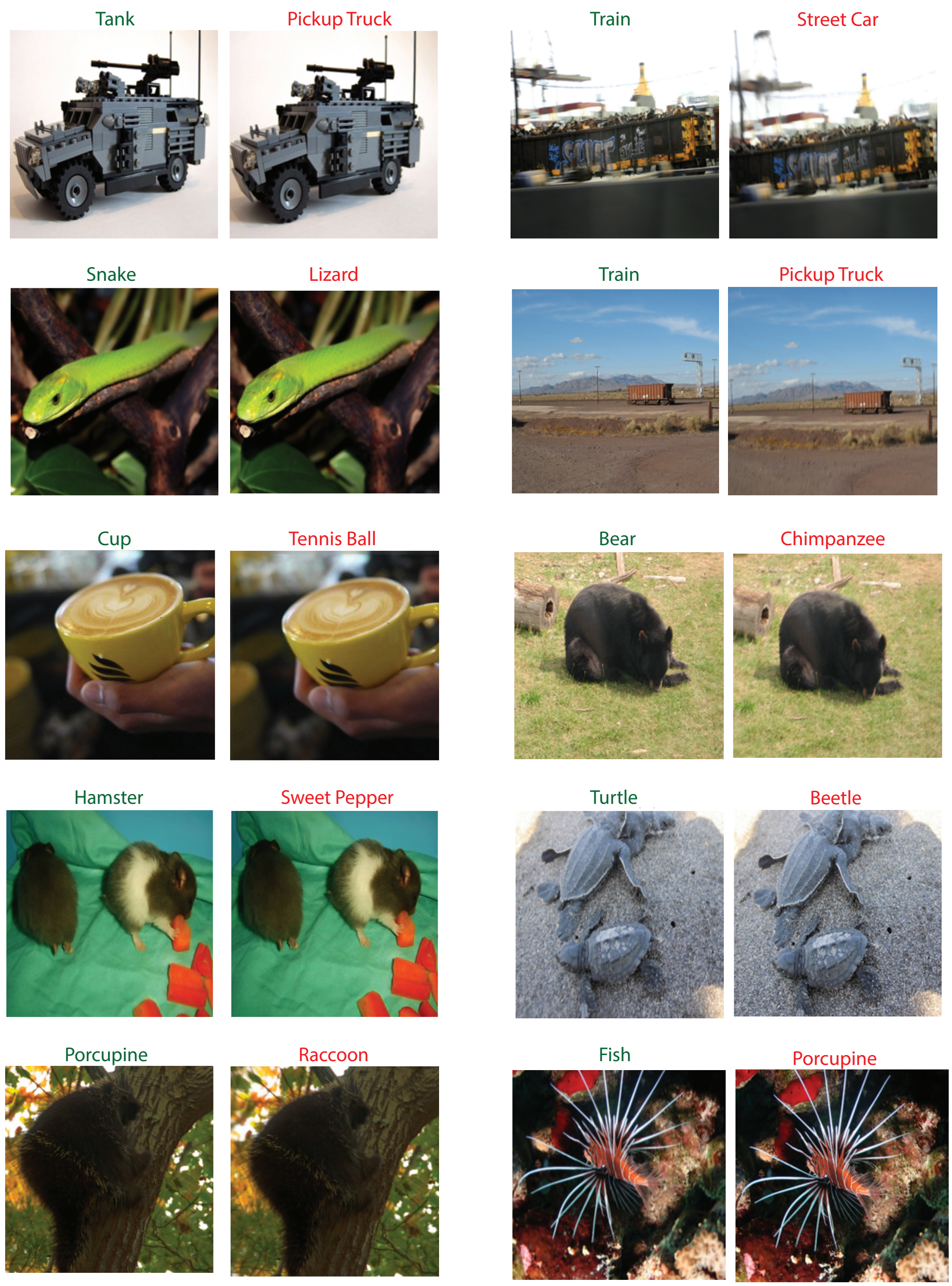

Figure S3: *More examples of misclassified ImageNet-like images discovered by CMA-Search combined with the single view MPI model.*