Analyzing And Improving Neural Networks By Generating Semantic Counterexamples Through Differentiable Rendering

Lakshya Jain [1] **Varun Chandrasekaran** [2] **Uyeong Jang** [2] **Sanjit Seshia** [1] **Somesh Jha** [2]

Abstract

Even as deep neural networks (DNNs) have achieved remarkable success on vision-related tasks, their performance is brittle to transformations to the input. Of particular interest are semantic transformations that model changes that have a basis in the physical world, such as rotations, translations, changes in lighting or camera pose. In this paper, we show how *differentiable rendering* can be utilized to generate images that are *informative*, yet *realistic*. These images can be used to analyze DNN performance and improve its robustness through data augmentation. Given a differentiable renderer and a DNN, we show how to use off-the-shelf attacks from adversarial machine learning to generate *semantic counterexamples* — images where semantic features are changed to induce misclassifications or misdetections. We validate our approach on DNNs for image classification and object detection. For classification, we show that semantic counterexamples, when used to augment the dataset, (i) improve generalization performanc, and (ii) enhance robustness to semantic transformations. Additionally, in comparison to sampling-based semantic augmentation, our technique generates more informative data in a *sample efficient manner*.

1. Introduction

Machine Learning (ML) models, such as deep neural networks (DNNs), have shown remarkable success in several domains, including visual tasks such as image classification and object detection. Thus, ML models have started being deployed in safety-critical applications such as in autonomous driving and other cyber-physical systems (CPS). At the same time, it has been well documented that DNN performance can be brittle to small perturbations to the input (Goodfellow et al., 2015; Carlini & Wagner, 2017; Madry et al., 2017). Such brittleness in safety-critical CPS can have disastrous consequences.

Figure 1. **Semantic counterexamples for object detection.** Benign re-rendered VKITTI image (left), semantic counterexample generated by our approach (right). The semantic changes involve changes in car positions, orientations, and color. This causes the network to fail to detect the grey car in the immediate foreground.

Semantic modifications to input images, capturing changes that have a basis in the physical world, are particularly important in the context of CPS. These modifications (*c.f.* Appendix J) include translations, rotations, changes in lighting, contrast, or color, changes in camera pose, time of day, object deformations. They capture perturbations to the environment of the ML-based system that are *semantically meaningful*, and more likely to occur in a physical environment. Small semantic modifications to the input should not negatively affect the output of the ML model. In other words, we want the ML model to possess *semantic robustness*. For example, for an object detector that must identify cars in an image and draw bounding boxes around them, the output should remain unchanged to a change of car colors. Inputs that violate semantic robustness are termed as *semantic counterexamples*. In spite of the impressive depth and volume of work in adversarial ML (AML) applied to computer vision, most of that literature focuses on pixel-

[1]University of California, Berkeley [2]University of Wisconsin, Madison. Correspondence to: Lakshya Jain <lakshya.jain@berkeley.edu>.

Presented at the ICML 2021 Workshop on Uncertainty and Robustness in Deep Learning., Copyright 2021 by the author(s).

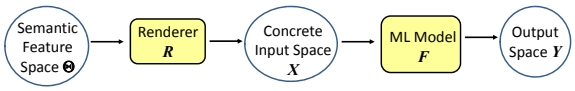

Figure 2. Our approach analyzes the composition of a differentiable renderer R and an ML model F.

level transformations to the input (Goodfellow et al., 2015; Carlini & Wagner, 2017; Madry et al., 2017). There is a need for effective techniques for generating semantic counterexamples and for using them to improve the semantic robustness of the ML model (DNN) deployed in CPS.

In this paper, we address this need through an approach that leverages advances in AML and differentiable rendering. We begin by defining a semantic feature space capturing features of the environment that, together with a rendering process, determine the input image. Through the notion of *semantic robustness*, we are interested in exploring modifications to points in the semantic feature space that produce incorrect outputs. AML techniques currently provide an effective way to produce such modifications in the pixel space for images. We show how advances in differentiable rendering allow us to use *off-the-shelf* attacks from AML literature to generate semantic counterexamples in a sample-efficient fashion. For object detection, we show how our approach can produce semantically-meaningful images that are misdetected (see example in Figure 1). For classification, we show how dataset augmentation with semantic counterexamples results in (i) improved generalization performance, and (ii) enhanced semantic robustness across adversarial attacks.

2. Semantic Counterexamples

The components of the problem considered in this paper are shown in Figure 2. We begin by defining a semantic feature space Θ capturing the features of the environment that determine the input image. An element of Θ is a vector of semantic features describing the environment of the ML model, including the pose of objects, their color, and other characteristics, camera pose, lighting, time of day and weather conditions, background of the scene, etc. Given a point $\theta \in \Theta$, a process R (that models rendering, simulation, or camera capture) produces an input image x. An ML model (such as a DNN) F then maps x to an output y. In this section, we define the notions of semantic robustness and semantic counterexamples in this context. We then show how a combination of differentiable rendering and off-the-shelf adversarial-ML techniques can be used to find semantic counterexamples.

2.1. Notation

Consider a space Z of the form $X \times Y$, where X is the sample space and Y is the set of labels. Each point $x \in X$ is a *concrete input vector*, *e.g.,* a pixel-level encoding of an image; we will assume that $X = \mathbb{R}^n$. Let H be a hypothesis space (*e.g.,* weights of a DNN). We assume a loss function $\mathcal{L} : H \times Z \mapsto \mathbb{R}$ such that that given a hypothesis $w \in H$ and a labeled data point $(x, y) \in Z$, the loss is $\mathcal{L}_w(x, y)$. The machine learning model is a function F from $\mathbb{R}^n$ to Y. Sometimes, to emphasize that the ML model depends on a hypothesis $w \in H$, we will denote it as F_w (if w is clear from the context, we will simply write F, and use $\mathcal{L}_F(x, y)$ in place of $\mathcal{L}_w(x, y)$). When F is a classifier, the *softmax* output corresponding to F is denoted by $s(F)$ and is a function $s(F) : X \to \Delta(Y)$, where $\Delta(Y)$ is the set of distributions over Y. Typically, $F(x)$ is defined as $\arg\max_{y \in Y} s(F)(y)$. The formalism for object detection is similar, and is described in Appendix H.1.

2.2. Semantic Robustness

The semantic robustness property captures the requirement that a small change in the semantic feature space must only result in a small change in the output space. Given $\theta \in \Theta$, a renderer R and an ML model F, this property is:

$$\forall \theta' \in \Theta, \forall x, x' \in X \,.\, \big([\theta \approx_\Theta \theta' \wedge R(\theta) = x \wedge R(\theta') = x'] \Rightarrow [F(x) \approx F(x')]\big) \quad (1)$$

where $\approx_\Theta$ specifies that θ and θ' are *close* in Θ, while $\approx$ specifies that the outputs are close/unchanged as required by the application. A $\theta' = \tilde{\theta}$ that falsifies Property 1 is termed as a *semantic counterexample*.

In practice, the relation $\approx_\Theta$ is typically defined as having θ' within an ε neighborhood of θ using a suitable norm μ such as ℓ_∞, ℓ_0, ℓ_1, or ℓ_p ($p \geq 2$). Similarly $\approx$ is defined as either requiring $F(x) = F(x')$ or having them within a specified neighborhood of each other. In this case, a semantic counterexample $\tilde{\theta}$ is an element of Θ in an ε neighborhood of θ that violates Property 1. For example, if θ includes the position of a car, its color, and the camera pose, $\tilde{\theta}$ can be a small change to the camera pose that causes F to misidentify the car.

Note that x and $\tilde{x} = R(\tilde{\theta})$ may be far from each other in X even when θ and $\tilde{\theta}$ are close in Θ; *e.g.,* changing the color of a car while keeping the scene largely unchanged, as shown in Figure 1. Thus, semantic counterexamples supersede small adversarial perturbations in the pixel space.

2.3. Finding Semantic Counterexamples

A common goal in AML is for an adversary A to take any input vector $x \in \mathbb{R}^n$ and produce a minimally altered version denoted by $x+\delta$, that has the property of being misclassified by a classifier F. As in AML, we formulate the procedure of finding a semantic counterexample as an optimization problem, but this now involves the semantic space Θ and

the renderer R. Some differentiable renderers, such as Redner (Li et al., 2018b), directly generate an image x from a semantic feature vector θ. However, others such as 3D-SDN (Yao et al., 2018) first de-render an input image x to θ and then re-render. In either case, the key aspects are that the perturbation is to be performed on θ and the function R is differentiable. Note that in practice, *the semantic feature space is determined by the differentiable renderer in consideration*. Given this knowledge, the optimization problem is:

$$\begin{array}{ll} \min_{\pi \in \Theta} & \nu(\pi) \\ \text{such that} & F \circ R(\theta + \pi) \neq F \circ R(\theta) \end{array}$$

where ν is a norm on Θ. In other words, we want to find a *small perturbation* in the semantic space Θ that will misclassify the sample. If the result of the optimization is π^*, then the generated semantic counterexample is $\tilde{\theta} = \theta + \pi^*$.[1]

Let $A(F, x)$ be an attack algorithm to generate traditional adversarial examples. We outline a general technique for transforming A to the semantic equivalent sA to generate semantic counterexamples. The technique works as by transforming algorithm A using the following rules: (1) Replace δ with π; (2) Replace $x + \delta$ with $\theta + \pi$; (3) Use chain rule to compute the gradients of terms, *e.g.,* the loss function, that involve $R(\theta)$.

3. Evaluation

We conducted an empirical evaluation seeking to answer the following questions: (1) Are the semantic counterexamples realistic and effective at degrading the accuracy of the original network? and (2) Does augmenting with semantic counterexamples improve model robustness?

Details regarding our implementation and experimental procedure are presented in Appendices E, F, and G. Our experiments were performed on two servers. The first has an NVIDIA Titan GP102 GPU, 8 CPU cores, and 15GB memory. The second has 264 GB memory, 8 NVIDIA's GeForce RTX 2080 GPUs, and 48 CPU cores. Due to space constraints, we restrict our discussion to the experiments involving image classification (using VGG-16 (Simonyan & Zisserman, 2014) as the target[2]). The experiments involving object detection are detailed in Appendix H.

We observe that: (1) Semantic counterexamples are shown to be both realistic (using both quantitative and qualitative measures) and effective at degrading accuracy (see § 3.1 and § 3.2), and (2) Classification models augmented using semantic counterexamples do not suffer from generalization degradation, but show improved robustness (*c.f.* § 3.3) across *all* adversarial ML methods utilized. We also discuss cross-model transferability in Appendix L.

3.1. Accuracy and Augmentation

Train	Test: benign	si-FGSM	sGD	sCW
benign	0.986	0.529	0.657	0.485
si-FGSM	0.979	0.936	0.955	0.907
sGD	0.978	0.939	0.946	0.928
sCW	0.98	0.942	0.959	0.95

Table 1. **Accuracy metrics of robust networks retrained** with adversarial training using semantic counterexamples generated on the benign model. The rows indicate the method used to generate the retraining semantic counterexamples and the columns indicate the dataset used for evaluation. Observe that retraining on semantic counterexamples improves robustness across datasets, regardless of the initial train dataset (larger values are better). All experiments are carried out on VGG-16.

A discussion on informativeness of samples is presented in Appendix C.

We measure the accuracy degradation induced by the semantic counterexamples on the VGG-16 model they were generated from. This is compared against counterexamples generated by random sampling and Halton sampling (our 2 baselines). For each baseline approach, we provide 2 ranges to sample rotation (the dominant semantic transformation, keeping the vertex translation parameter fixed). These are (i) the large range *i.e.,* [-0.75, +0.75] radians, and (ii) the small range *i.e.,* [-0.3, +0.3] radians. Table 6 (*c.f.* Appendix F) contains the weighted average accuracy degradation of various strategies. Observe that semantic counterexamples generated using our baseline approach, Halton sampling (in the large range) cause the most accuracy degradation (the difference between the overall accuracy in the benign setting and the overall accuracy for this particular approach). This is followed by semantic counterexamples (sCW and si-FGSM). The same experiment is repeated with the ResNet-50 architecture (He et al., 2016) and reported in Table 14 (*c.f.* Appendix I). The results are slightly different, where si-FGSM induces the most accuracy degradation; this suggests that the hyperparameter choices impact the method.

When the datasets are augmented using semantic counterexamples, the accuracy of the newly learned models are presented in Table 1. These results suggest that retraining improves semantic robustness. Additionally, retraining us-

[1] We note that the semantic parameter space is not homogeneous and it is unclear if one function can be used to capture a suitable notion of distance. Not only should the norm measure the changes in the semantic space, it should also approximate human perception. Additionally, in practice the "+" operator can be more complex for semantic parameters such as weather or background.

[2] Experiments with ResNet-50 follow a similar trend and are described in Appendix I

ing samples generated from one strategy provide robustness against samples generated from other strategies as well.

Takeaway: The results suggest that semantic counterexamples are useful augmentation samples as they do not harm generalization performance, but improve performance on unseen semantic counterexamples.

3.2. Realism

Results are from semantic counterexamples generated from VGG-16; results from ResNet-50 are in Appendix I.

Method	Strategy	Score
FID	si-FGSM	10.75
	sGD	**8.94**
	sCW	11.68
	Halton (large)	13.71
	Random (large)	13.96
LPIPS	si-FGSM	0.41
	sGD	**0.38**
	sCW	0.43
	Halton (large)	0.42
	Random (large)	0.46

Table 2. **Realism measures** for augmentation samples generated using VGG-16. Lower the scores, more realistic the images. Refer Table 21 in Appendix M for more details.

1. FID & LPIPS Distance: To obtain a quantitative measure of realism, we measure: (i) the Fréchet Inception Distance (FID) (Heusel et al., 2017), a metric known to correlate with human visual quality, and (ii) the LPIPS distance (Zhang et al., 2018), another metric for perceptual similarity. In both cases, the lower the better. The results are summarized in Table 2. We omit scores for Halton (small) and Random (small) as they do not create informative samples. Each cell contains the average score (per class) when the augmenting samples are compared with the samples they were generated from. The results suggest that across both metrics, the semantic counterexamples are realistic.

2. Survey: To validate if (i) humans are able to correctly classify the objects in semantic counterexamples, and (ii) if humans consider the semantic counterexamples realistic, we conducted out an online survey on Amazon Mechanical Turk with 30 participants[3]. Each participant was asked 2 questions about 15 semantic counterexamples. The first question was to classify the object. The second question was to rate the realism of the modification induced (in comparison to the original, unmodified image placed next to it) on a scale of 1 (lowest) to 10 (highest). For the classification task, we observe that the human participants are able to correctly classify the object 98% of the time (on average). The average median realism for samples generated using Redner was 6.67 (*mean realism* = 6.35). These results further validate the realism of semantic counterexamples.

[3]Each participant was a master worker and was compensated $8 for the study, approved by our IRB.

Takeaway: Our results suggest that our approach generates *realizable/realistic* images.

3.3. Robustness after Retraining

Results in Table 1 suggest that semantic counterexamples are useful for augmentation. To measure if the retrained models are more robust, we test these models by generating new semantic counterexamples using methods formalized in Appendix D. Each row in Table 3 contains normalized accuracy by evaluating the augmented model (as in Table 6) with test samples generated using the augmented model (as opposed to the benign model as in Table 1); this difference is highlighted by the subscript $robust$. In comparison to the benign model (in row 1), we can see that the augmented models are indeed more robust to all adversarial methods tested, regardless of the technique used to generate training semantic counterexamples. This hints at potential cross-method robustness transferability.

	Test			
Train	benign	si-FGSM$_{robust}$	sGD$_{robust}$	sCW$_{robust}$
benign	0.991	0.547	0.685	0.483
si-FGSM	0.978	0.797	0.871	0.819
sGD	0.978	0.836	0.910	0.797
sCW	0.991	0.884	0.918	0.879

Table 3. **Accuracy metrics for benign and adversarially retrained networks** on semantic counterexamples generated by using various methods (in the columns) on individual networks trained to be robust to individual methods (in the rows). Retraining against one method helps provide some robustness against *all* tested methods.

4. Conclusions

In this paper, we introduce the notion of semantic counterexamples, which are examples that adversely affect an ML model and which are generated via semantically-meaningful modifications. We demonstrate an approach for generating semantic counterexamples that adapts and combines two key techniques: algorithms from AML and differentiable rendering. We present an extensive evaluation to assess the realism and efficacy of semantic counterexamples. Future directions include testing the operation of these semantically robust ML networks within the control loop of a CPS, and making our algorithms aware of the rest of the control loop.

References

Barron, J. T. and Malik, J. Shape, illumination, and reflectance from shading. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(8):1670–1687, Aug 2015. doi: 10.1109/TPAMI.2014.2377712.

Baumgart, B. G. *Geometric Modeling for Computer Vision.* PhD thesis, Stanford, CA, USA, 1974. AAI7506806.

Bhattad, A., Chong, M. J., Liang, K., Li, B., and Forsyth, D. Unrestricted adversarial examples via semantic manipulation. In *International Conference on Learning Representations*, 2020.

Blanz, V. and Vetter, T. A morphable model for the synthesis of 3d faces. *SIGGRAPH'99 Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, 09 2002. doi: 10.1145/311535.311556.

Carlini, N. and Wagner, D. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 39–57. IEEE, 2017.

Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

Dreossi, T., Ghosh, S., Yue, X., Keutzer, K., Sangiovanni-Vincentelli, A., and Seshia, S. A. Counterexample-guided data augmentation. In *27th International Joint Conference on Artificial Intelligence (IJCAI)*, 2018.

Engstrom, L., Tran, B., Tsipras, D., Schmidt, L., and Madry, A. Exploring the landscape of spatial robustness. In *International Conference on Machine Learning*, pp. 1802–1811, 2019.

Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A., Kohno, T., and Song, D. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1625–1634, 2018.

Gaidon, A., Wang, Q., Cabon, Y., and Vig, E. VirtualWorlds as proxy for multi-object tracking analysis. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4340–4349, 2016.

Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F. A., and Brendel, W. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *CoRR*, abs/1811.12231, 2018. URL `http://arxiv.org/abs/1811.12231`.

Goodfellow, I., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015. URL `http://arxiv.org/abs/1412.6572`.

Halton, J. H. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numer. Math.*, 2(1):84–90, December 1960. ISSN 0029-599X. doi: 10.1007/BF01386213. URL `https://doi.org/10.1007/BF01386213`.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Klambauer, G., and Hochreiter, S. Gans trained by a two time-scale update rule converge to a nash equilibrium. *CoRR*, abs/1706.08500, 2017. URL `http://arxiv.org/abs/1706.08500`.

Kato, H., Ushiku, Y., and Harada, T. Neural 3d mesh renderer. pp. 3907–3916, 06 2018. doi: 10.1109/CVPR.2018.00411.

Kulkarni, T. D., Whitney, W. F., Kohli, P., and Tenenbaum, J. Deep convolutional inverse graphics network. In *Advances in neural information processing systems*, pp. 2539–2547, 2015.

Kurakin, A., Goodfellow, I., and Bengio, S. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.

LaValle, S. M. *Planning algorithms.* Cambridge university press, 2006.

Li, T.-M., Aittala, M., Durand, F., and Lehtinen, J. Differentiable monte carlo ray tracing through edge sampling. In *SIGGRAPH Asia 2018 Technical Papers*, pp. 222. ACM, 2018a.

Li, T.-M., Aittala, M., Durand, F., and Lehtinen, J. Differentiable monte carlo ray tracing through edge sampling. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 37 (6):222:1–222:11, 2018b.

Loukadakis, M., Cano, J., and O'Boyle, M. Accelerating deep neural networks on low power heterogeneous architectures. 01 2018.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *ArXiv*, abs/1706.06083, 2017.

Qiu, H., Xiao, C., Yang, L., Yan, X., Lee, H., and Li, B. Semanticadv: Generating adversarial examples via attribute-conditional image editing. *CoRR*, abs/1906.07927, 2019. URL `http://arxiv.org/abs/1906.07927`.

Redmon, J. and Farhadi, A. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

Shacked, R. and Lischinski, D. Automatic lighting design using a perceptual quality metric. *Comput. Graph. Forum*, 20, 09 2001. doi: 10.1111/1467-8659.00514.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Song, Y., Shu, R., Kushman, N., and Ermon, S. Generative adversarial examples. *CoRR*, abs/1805.07894, 2018. URL http://arxiv.org/abs/1805.07894.

Wu, B., Iandola, F. N., Jin, P. H., and Keutzer, K. SqueezeDet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving. *CoRR*, abs/1612.01051, 2016. URL http://arxiv.org/abs/1612.01051.

Xiao, C., Yang, D., Li, B., Deng, J., and Liu, M. Meshadv: Adversarial meshes for visual recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Yao, S., Hsu, T.-M. H., Zhu, J.-Y., Wu, J., Torralba, A., Freeman, W. T., and Tenenbaum, J. B. 3D-aware scene manipulation via inverse graphics. In *Advances in neural information processing systems*, 2018.

Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, pp. 586–595, 2018.

Appendix

A. Related Work

Our work builds upon the literature on differentiable rendering and adversarial ML, which we review below.

Differentiable Rendering: The process of finding 3D scene parameters (geometric, textural, lighting, etc.) given images is referred to as de-rendering or inverse graphics (Baumgart, 1974; Blanz & Vetter, 2002; Shacked & Lischinski, 2001; Barron & Malik, 2015). A pipeline for differentiable rendering (the opposite of inverse graphics) were proposed by Kato *et al.* (Kato et al., 2018). Kulkarni *et al.* (Kulkarni et al., 2015) combine both de-rendering and rendering, and propose a model that learns interpretable representations of images (similar to image semantics), and show how these interpretations can be modified to produce changes in the input space. Similarly, Yao *et al.* (Yao et al., 2018) propose a pipeline that, through de-rendering obtains various forms of semantics, geometry, texture, and appearance, which can be rendered using a generative model. Our experimental setup uses both of these differentiable renderers.

Adversarial ML: Adversarial examples are test-time inputs that result in incorrect outputs produced by the ML model. Extensive prior work (Madry et al., 2017; Goodfellow et al., 2015; Carlini & Wagner, 2017) focuses on generating norm-bounded pixel-level changes to input features to induce incorrect actions. However, these manipulations are not usually realizable in the real world. Our focus is on those attacks that result in *semantically meaningful* adversarial samples. Generating real world adversarial examples has resulted in several efforts (Eykholt et al., 2018; Bhattad et al., 2020), some of which are more realistic than others. Eykholt *et al.* (Eykholt et al., 2018) show how physical changes can fool neural networks. Bhattad *et al.* (Bhattad et al., 2020) give techniques for colorizing or changing the texture of images to generate adversarial examples. Geirhos *et al.* (Geirhos et al., 2018) discovered that certain models are biased towards textural cues, while Engstrom *et al.* (Engstrom et al., 2019) observe that modifying the spatial orientation of images results in misclassifications. Song *et al.* (Song et al., 2018) develop an approach for creating semantic adversarial examples, but utilize GANs to do this instead, using class-conditional searches on the latent space. This work, however, constructs examples from scratch instead of taking existing datapoints and perturbing them, and does not utilize gradient-based approaches, limiting the flexibility of their pipeline. Meanwhile, approaches that involve explicitly sampling the semantic parameter space are slow and expensive (Dreossi et al., 2018). The work that is closest to ours is that of Xiao *et al.* (Xiao et al., 2019) and Qiu *et al.* (Qiu et al., 2019); the former uses a differentiable renderer to induce changes in shapes and textures, while the latter induces a different set of semantic changes using a generative model. Our approach stands out by being general, in that it can apply to any vector of semantic features as long as we can render from those features using a differentiable renderer. Further, we show how our generated semantic counterexamples can be effectively used for data augmentation and that they can transfer between DNNs.

B. Augmentation Quality

Suppose there is a dataset D and we have two examples for augmentation (x, y) and (x', y'). Which one is better for augmentation? In this section, we formalize the notion of *informativeness* to address this question.

B.1. Entropy & Information Worth

A common measurement used to measure the classifier's predictive capabilities makes use of the concept of Shannon entropy, which measures the unpredictability of outcome from a given set. Let the label set Y have L labels $\{l_1, \cdots, l_L\}$. Given set of datapoints $D = \{(x_1, y_1), \cdots, (x_m, y_m)\}$ where $y_t \in Y$ is the true label of a point x_t; a classifier F on the set D divides the set into subsets $D_1, \ldots, D_L$ where D_i contains points whose predicted label is l_i. Let E_i be the entropy measured on each subset D_i of size m_i *i.e.,* $E_i = -\sum_{j\in[L]} p_j^i \log p_j^i$, where p_j^i is the estimated probability of observing the class j in the subset D_i. If the entropy E_i for each subset D_i is low, then the classifier divides D in a well-organized manner. Thus, the following weighted average of E_i's (henceforth referred to as *information worth*) is used to estimated the quality of a classifier *i.e.,* $\hat{E} = \sum_{i\in[L]} \gamma_i \cdot E_i$, where $\gamma_i = \frac{|D_i|}{|D|} = \frac{m_i}{m}$.

We also want to make use $\hat{E}$ as a measure of the quality of augmented data. As long as the datapoints used for augmentation are realistic, it is better to have datapoints that do not agree with the current classifier, as we expect the re-training to improve the classifier by fixing the incorrect classification. However, the previous definition of $\hat{E}$ depends only on the predictions of the classifier. Thus, it is hard to distinguish between augmenting samples generated using the methodology we propose from those generated using random sampling, for example.

B.2. Classifier Confidence & Realism

In the previous definition, each point x could belong to only 1 subset (*i.e.,* binary membership). However, with most DNN-based models, a point could belong to many classes (and consequently subsets) depending on its softmax values (*i.e.,* fractional membership). Thus, let $\mu_i(x)$ denote the membership of a point to subset D_i. The most natural choice of $\mu_i(x) = s(F)(x)_i$ which denotes the softmax value of x for class i. We also assume that there exists a quantifiable measure of realism ρ such that $\rho(x) \in [0, 1]$ (larger value means the image is more realistic). In this work, since we generate a new datapoint x' from a reference datapoint x^*, we first define the similarity σ of x' to x^* as $\sigma(x'; x^*) = 1 - \frac{d(x',x)}{d_{\text{max}}}$, where d is the LPIPS distance (Zhang et al., 2018) and d_{max} is the maximum LPIPS score achievable. This metric assigns $\rho(x^*; x^*) = 1$, because when there is no perturbation, *i.e.,* $x' = x^*$, then $d(x^*, x^*) = 0$. Therefore, we define $\rho(x)$ for $x \in S$ as $\sigma(x; x^*)$. Since ρ captures *realism*, if the value $\rho(x)$ is small for some data point x, we do not trust the classification result $F(x)$. Based on this intuition, we control the participation of each datapoint x in the information worth by weighting it by its realism $\rho(x)$. These ideas are captured by using the following p_j^i and γ_i.

$$p_j^i = \frac{\sum_{t=1}^m \rho(x_t)\mu_i(x_t)\mathbb{1}[y_t = j]}{\sum_{l\in[L]} \sum_{t=1}^m \rho(x_t)\mu_i(x_t)\mathbb{1}[y_t = l]}$$

$$\gamma_i = \frac{\sum_{t=1}^m \rho(x_t)\mu_i(x_t)}{\sum_{l\in[L]} \sum_{t=1}^m \rho(x_t)\mu_i(x_t)}$$

C. Notes On Augmentation Quality

C.1. Defining Information Worth

Based on the notation formalized in § B.1, let E_i be the entropy measured on each subset D_i of size m_i *i.e.,* $E_i = -\sum_{j \in L} p_j^i \log p_j^i$, where p_j^i is the estimated probability of observing the class j in the subset D_i.

$$\begin{aligned} p_j^i &= \frac{\sum_{t=1}^{m_i} \mathbb{1}[x_t \in D_i]\mathbb{1}[y_t = j]}{m_i} \\ &= \frac{\sum_{t=1}^{m_i} \mathbb{1}[f(x_t) = i]\mathbb{1}[y_t = j]}{m_i} \end{aligned}$$

If the entropy E_i for each subset D_i is low, then the classifier divides D in a "well-organized" way, *i.e.,* the subset is extremely likely to consist of datapoints of single label. Thus, the following weighted average of E_i's (henceforth referred to as *information worth*) is used to estimated the quality of a classifier *i.e.,* $\hat{E} = \sum_{i \in [N]} \alpha_i \cdot E_i$, where each weight α_i is determined by the proportion of the size of the subset D_i to the size of the entire set D *i.e.,* $\alpha_i = \frac{|D_i|}{|D|} = \frac{m_i}{m}$.

C.2. Salient Features of Information Worth

We make the following observations:

- A classifier achieving high accuracy is likely to achieve low value of $\hat{E}$ (however, the converse is not true in general). Therefore, a high value $\hat{E}$ can be an indicator of a bad classifier.
- If we update a classifier (through training) from F to F', we can measure the quality of the change in terms of the *information gain* which is the amount of decrease $\hat{E}_F - \hat{E}_{F'}$ (larger, the better).

C.3. Results

Membership	None	Halton (large)	Halton (small)	Random (large)	Random (small)	sCW	sGD	si-FGSM
Binary	0.1082	1.1125	0.6165	0.8085	0.431	0.9565	0.7218	0.887
Fractional	0.1224	1.1158	0.5964	0.8604	0.4742	0.9464	0.7213	0.8783

Table 4. **Information worth** of augmentation samples generated using various strategies. Binary membership uses the model prediction, *i.e.,* $\mu_i(x) = \mathbb{1}[F(x) = i]$ and fractional membership uses the model confidence, *i.e.,* $\mu_i(x) = s(F)(x)_i$. Larger values are better. All experiments are carried out on VGG-16.

D. Semantic Counterexample Constructions

1. Semantic FGSM Recall that $z = (x, y)$ where $x = R(\theta)$ and $y = F(x) = F \circ R(\theta)$. Consider the loss function $\mathcal{L}_F(x, y)$. Let θ_0 be a starting semantic feature vector that we wish to perturb into a semantic counterexample. The derivative with respect to θ, evaluated at θ_0 is

$$\left[\frac{\partial R}{\partial \theta}\right]^\top \Bigg|_{\theta_0} \nabla_z \mathcal{L}_F(z)|_{z=(R(\theta_0),F\circ R(\theta_0)))}$$

where the notation $\left[\frac{\partial R}{\partial \theta}\right]^\top \Big|_{\theta_0}$ is the transposed Jacobian matrix of R as a vector-valued function of θ, evaluated at θ_0, and $\nabla_z \mathcal{L}_F(z)|_{z=(R(\theta_0),F\circ R(\theta_0)))}$ is the derivative evaluated at $(R(\theta_0), F \circ R(\theta_0)$.

The semantic version of FGSM (*i.e.,* sFGSM) will, given the step-size hyperparameter α, produce the following semantic counterexample $\tilde{\theta}$:

$$\tilde{\theta} = \theta_0 + \alpha \operatorname{sign}\left(\left[\tfrac{\partial R}{\partial \theta}\right]^\top \Big|_{\theta_0} \nabla_z \mathcal{L}_F(z)|_{z=(R(\theta_0),F\circ R(\theta_0)))}\right) \tag{2}$$

Note that we only assume that the rendering function R is differentiable (which is true in all our experiments).

2. Semantic iterative FGSM. Let θ_0 be the initial semantic feature vector. The update steps correspond to the following equation:

$$\theta_{k+1} = \theta_k + \alpha \operatorname{sign}\left(\left[\tfrac{\partial R}{\partial \theta}\right]^\top \Big|_{\theta_k} \nabla_z \mathcal{L}_F(z)|_{z=(R(\theta_k),F(R(\theta_k)))}\right) \tag{3}$$

Similar to Simonyan *et al.* (sim), we define $\mathcal{L}_F$ to be the the raw (unnormalized) class score of the correct label instead of the normalized softmax probability for the attack listed above.

3. Semantic GD. We present a semantic version of the GD attack (sGD). Again, let θ_0 be the initial semantic feature vector. The update steps in semantic GD correspond to the following equation:

$$\theta_{k+1} = \Pi_{B_\nu(\theta_0,\varepsilon)}\left(\theta_k + \alpha \cdot \zeta_k\right) \tag{4}$$

where

$$\zeta_k = \left(\left[\tfrac{\partial R}{\partial \theta}\right]^\top \Big|_{\theta_k} \nabla_z \mathcal{L}_F(z)|_{z=(R(\theta_k),F(R(\theta_k)))}\right) \tag{5}$$

Similar to Simonyan *et al.* (sim), we define $\mathcal{L}_F$ to be the the raw (unnormalized) class score of the correct label instead of the normalized softmax probability for the attack listed above.

Note that $\Pi_{B_\nu(\cdot,\cdot)}$ is the projection operator in the parameter space Θ. We also assume that the projection operator will keep the parameters in the feasible set, which depends on the image (*e.g.,* translation does not take the car off the road).

4. Semantic CW. The semantic Carlini-Wagner (sCW) attack solves the following optimization problem.

$$\min \|\pi\|_p + c \cdot f(\theta + \pi) \tag{6}$$

where $p = 1$, $c = 0.1$ and the objective function f is defined as,

$$f(\theta + \pi) = \max((\max_{i \neq t} \Phi_i(R(\theta + \pi)) - \Phi_t(R(\theta + \pi))), 0) \tag{7}$$

and Φ_i is the output corresponding to class i of the model F before the softmax layer. The gradients are computed using the Adam optimizer, with the learning rate η.

E. Implementation Details

We utilize two differentiable graphics frameworks: (i) 3D-SDN proposed by Yao *et al.* (Yao et al., 2018), and (ii) Redner proposed by Li *et al.* (Li et al., 2018a). Using these frameworks, we augment two popular synthetic datasets: (i) VKITTI (Gaidon et al., 2016): scenes from photo-realistic proxy virtual worlds used for multi-object tracking, and (ii) ShapeNet (Chang et al., 2015): a large-scale repository of shapes represented by 3D CAD models of objects. The former dataset is used for object detection with SqueezeDet (Wu et al., 2016) as the target, and the latter for image classification using VGG-16 (Simonyan & Zisserman, 2014) and ResNet-50 (He et al., 2016) as targets.

We generate semantic counterexamples by adapting three popular methods – i-FGSM (Kurakin et al., 2016) (iterative FGSM), a variant of PGD (Madry et al., 2017) (henceforth called GD), and CW (Carlini & Wagner, 2017) – to generate their semantic counterparts (with the prefix *s*). Our implementation with 3D-SDN involves 1000 lines of code; this involves making changes to the original code to enable end-to-end differentiation, and replacing several image manipulations with their differentiable counterparts. Our implementation with Redner (Li et al., 2018a) involves 618 lines of code. The code is available at `https://anonymous.4open.science/r/1766a57c-d61f-42e4-aea1-f19fa5f0bc37/`[4]. We compare these semantic counterexamples with augmenting samples generated using (i) random sampling and (ii) Halton sampling (Halton, 1960) as baselines. All semantic counterexamples generated involve modifying multiple semantic parameters. More details (needed for reproducibility), such as the exact hyperparameters used for our generation process and the semantic parameters modified, the dataset sizes, model parameters are presented in Appendix G.

[4]We cannot open source the code using 3D-SDN as the license for the 3D-SDN code does not permit it.

F. Experimental Setup

Here, we detail our evaluation setup for the image classification task (detailed through the rest of the paper). Information and results related to the object detection task can be found in Appendices G and H.

Semantic Features. For the classification experiments using Redner, the semantic features include: (i) *pose*: the camera angle at which the object is viewed in the frame, (ii) *vertex*: the vertex coordinates of the object, and (iii) *lighting*: the environment lighting of the scene. We stress that the *choice of semantic parameters are closely tied to the implementation of the renderer we use*.

	Parameters		
Strategy	*vertex*	*pose*	*pose* + *vertex*
sCW	0.892	0.543	0.485
si-FGSM	0.862	0.615	0.529
sGD	0.897	0.741	0.657

Table 5. **Effectiveness of multi-parameter modifications** for Redner. Observe that single parameter modifications are not as effective.

Semantic Parameter Combinations. Before we discuss specifics about the strategies we use to generate semantic counterexamples, we need to understand how the semantic parameter space can be manipulated. In particular, we wish to answer the question: *Which semantic parameters do we change?* In the pixel perturbation setting, all pixels are equal *i.e.,* any pixel can be perturbed. Whether such uniformity naturally exists in the semantic space is unclear.

Experimentally, we observe that modifying individual semantic parameters (as described earlier) are not as effective as multi-parameter modifications (refer Table 5 for normalized accuracy measurements). Additionally, modifications to *pose* induce greater accuracy degradation than modifications to *vertex*. For all results reported in this section, we use semantic counterexamples obtained through multi-parameter modifications. Note that changing the *lighting* parameter did not impact the accuracy of classification, and is consequently omitted from all tabulations.

Sampling Approaches. We provide additional information about the sampling strategies we use (as baselines).

1. Random Sampling: For each benign point, we obtain 5 samples using Halton sampling in 2 distinct parameter ranges (only for *pose* transformations): (i) **large** *i.e.,* [-0.75, +0.75] radians, and (ii) **small** [-0.3, +0.3] radians. For each range, among the 5 samples generated, we first verify if the sample induces a misclassification. If so, we pick the one with the highest softmax value for the incorrect prediction.

2. Halton Sampling: We utilize Halton sampling as implemented by Dreossi *et al.* (Dreossi et al., 2018); for each benign point, we obtain 5 samples using Halton sampling in 2 distinct parameter ranges (only for *pose* transformations): (i) **large** *i.e.,* [-0.75, +0.75] radians, and (ii) **small** [-0.3, +0.3] radians. For each range, among the 5 samples generated, we first verify if the sample induces a misclassification. If so, we pick the one with the highest softmax value for the incorrect prediction.

Inefficiency of Sampling: From our results, it is clear that the semantic counterexamples generated using our approach are effective for data augmentation purposes. However, the performance of Halton sampling degrades for high dimensions (LaValle, 2006): as the number of semantic features increase, the number of trials required to obtain a realistic Halton sample useful for augmentation increases dramatically. This further validates the scalability of our approach.

Strategy	benign	sCW	si-FGSM	sGD	Random (large)	Random (small)	Halton (large)	Halton (small)
Accuracy	0.986	**0.485**	0.529	0.657	0.582	0.830	0.274	0.757

Table 6. **Accuracy degradation** induced by different semantic counterexample-generation strategies. Observe that semantic counterexamples are effective at inducing accuracy degradation. All experiments are carried out using VGG-16 as the target model. Refer Table 22 in Appendix M for a more detailed version of this table.

Training Procedure. A pretrained ImageNet model is taken, and the final fully connected layer is retrained in both VGG-16 and ResNet-50, for 20 epochs, using the images (information found in Table 12 in Appendix G), to obtain the benign models. To obtain the robust models, we replace *half of the training set (at random)* with the corresponding semantic counterexamples, and retrain the benign model for 20 epochs.

G. Implementation Specifics

Below, we provide some salient features related to the implementation of our approach. Details are mostly focussed on the implementation of the approach for object detection networks.

G.1. Semantic Features

Recall that we generate semantic counterexamples by strategically perturbing semantic features.

In the case of VKITTI (using 3D-SDN), the semantic features include (i) *color*: the car's texture codes which change its color, (ii) *weather*: the weather and time of day, (iii) *foliage*: the surrounding foliage and scenery, (iv) *rotate*: the car's orientation, (v) *translate*: the car's position in 2D, and (vi) *mesh*: the 3D mesh which provides structure to the car.

G.2. Semantic Parameter Combinations

Before we discuss specifics about the strategies we use to generate semantic counterexamples, we need to understand how the semantic parameter space can be manipulated. In particular, we wish to answer the question: *Which semantic parameters do we change?* In the pixel perturbation setting, all pixels are equal *i.e.,* any pixel can be perturbed. Whether such uniformity naturally exists in the semantic space is unclear. However, we have additional flexibility; for each rendering framework, we can choose to modify any of the above listed semantic parameters independently without altering the others, *i.e.,* perform *single parameter modifications*. Alternatively, we can modify any subset of the parameters in unison, *i.e.,* perform *multi-parameter modifications*.

Case: 3D-SDN + SqueezeDet + VKITTI

Semantic counterexamples are generated using an iterative variant of FGSM (*i.e.,* si-FGSM). The ineffectiveness of modifying parameters individually is detailed in Table 7.

	Parameters					
Metric	*color*	*weather*	*foliage*	*translate*	*rotate*	*mesh*
recall	100	100	100	100	100	98.7
mAP	99.5	98.8	99.7	99.2	98.2	98.7

Table 7. Ineffectiveness of single parameter modifications. Performance of SqueezeDet on SAEs generated using single parameter modifications. The model had (a) recall = 100, and (b) mAP = 99.4 on benign/non-adversarial inputs.

For multi-parameter modifications, we consider the following combination of semantic parameters: (a) `c1` = *translate* + *rotate*, (b) `c2` = *translate* + *rotate* + *mesh*, (c) `c3` = *translate* + *mesh*, and (d) `c4` = *rotate* + *mesh*. The results are reported in Table 8. We observe that for `c2`, there is $\sim$ 35 PPs decrease in mAP. `c2` was used to generate semantic counterexamples for our experiments.

	Parameters				
Metric	Baseline	`c1`	`c2`	`c3`	`c4`
recall	100	100	100	100	100
mAP	99.4	82	65.9	80.8	98.7

Table 8. Effectiveness of multi-parameter modifications for 3D-SDN.

G.3. Hyperparameters For Generating Semantic Counterexamples

In the context of generating semantic counterexamples, the value of the step-size parameter α is proportional to the magnitude of the geometric and textural changes induced; the effect depends on the semantic parameter under consideration. Large values of α result in unrealistic images rendered. To avoid such issues and to simulate *realistic* transformations, we use a different step-size for each semantic parameter. Similarly, the projection bound ε enables us to clip perturbations that exceed this bound (refer to the construction of PGD (Madry et al., 2017) and sGD for its use).

1. Redner: Below, we report the hypeparameters we use for the various strategies formalized in Appendix D. These are specific to generating semantic counterexamples using VGG-16 (refer Table 9). For sCW, the attack was formulated according based on the work of Carlini *et al.* (Carlini & Wagner, 2017) with the 2-norm used for our distance metric. The attack was untargeted, as we sought just to induce a misclassification of some kind while minimizing the 2-norm of the distance between the old semantic feature vector and the new one.

Strategy	α_{vertex}	α_{pose}	# iterations	ε_{vertex}	ε_{pose}	η_{vertex}	η_{pose}
sCW	-	-	5	-	-	0.01	0.30
si-FGSM	0.002	0.15	5	-	-	-	-
sGD	0.01	0.20	5	0.05	1.0	-	-

Table 9. Hyperaparameters used for generating semantic counterexamples using VGG-16. α is step-size, ε is projection bound, η is the learning rate

These hyperparameters specified below are specific to generating semantic counterexamples using ResNet-50 (refer Table 10).

Strategy	α_{vertex}	α_{pose}	# iterations	ε_{vertex}	ε_{pose}	η_{vertex}	η_{pose}
sCW	-	-	5	-	-	0.01	0.90
si-FGSM	0.01	0.45	5	-	-	-	-
sGD	0.01	0.60	5	0.05	3.0	-	-

Table 10. Hyperparameters used for generating semantic counterexamples using ResNet-50. α is step-size, ε is projection bound, η is the learning rate

2. 3D-SDN: We utilize an iterative variant of FGSM to modify the semantic parameters (*i.e.,* si-FGSM). Recall that in FGSM, the degree of modification of the input is determined by the step-size hyperparameter $\alpha \in [0, 1]$. The hyperparameters are specified in Table 11.

	Parameters					
	color	*weather*	*foliage*	*translate*	*rotate*	*mesh*
α	0.05	0.25	0.10	0.01	0.01	0.025

Table 11. Hyperparameters for generating semantic counterexamples using SqueezeDet. α is the step-size.

We stress that for both differentiable graphics frameworks, these hyperparameters were obtained after extensive visual inspection (by 3 viewers independently). An added benefit of our particular choice of hyperparameters (for 3D-SDN) is that it enables us to use the same ground truth labels throughout our experiments; the produced semantic counterexamples have bounding box coordinates that enable us to use the same ground truth labels as their benign counterparts[5].

G.4. Datasets

Table 12 contains salient features of the ShapeNet dataset we use for experiments with Redner. There are a total of 3670 test images, 12843 train images, and 1835 validation images. Unless specified otherwise, this proportion is used for all experiments. The class imbalance stems from the lack of availability of samples for specific classes in ShapeNet. The experiments in § 3.3 use a smaller test set of 232 images. that roughly maintains the proportions present in the full test dataset.

For experiments with VKITTI, our train dataset consisted of 6339 images, and the test dataset consisted of 2082 images. Each image comprised of the following objects: (a) one or more cars, (b) one or more buses, (c) *no pedestrians*. The object detection task involved detecting the car(s) and/or bus(es).

[5]This fact is useful when we evaluate model robustness through retraining the models with semantic counterexamples as inputs.

Class	Name	# test	# train	# validation
0	airplane	775	2713	388
1	bench	464	1624	232
2	trashcan	119	417	60
3	bus	127	445	64
4	car	1549	5421	774
5	helmet	60	210	30
6	mailbox	15	53	8
7	motorcycle	73	258	37
8	skateboard	49	171	24
9	tower	25	87	12
10	train	99	344	49
11	boat	315	1100	157

Table 12. Salient features of the ShapeNet dataset.

G.5. Training Procedure

We detail the procedure we used for training to obtain both the benign and robust models.

Object Detection: The differentiable graphics framework induces several artifacts on rendering. To this end, we first used identity transforms (*i.e.,* passed the benign image through the differentiable graphics framework without semantically modifying it) to obtain benign re-rendered images. The entire pretrained SqueezeDet model is retrained for 24000 steps using all the 6339 re-rendered images to obtain the benign model. To obtain the robust models, we replace a quarter of the training set at random with semantic counterexamples, and retrain the benign model for 6000 steps.

H. 3D-SDN Experiments

We provide some additional details for the experiments on object detection using 3D-SDN as the differentiable graphics framework.

H.1. Notation

Let x be an image which contains N recognition targets $T = \{t_1, t_2, ..., t_N\}$. Each target t_n, $n = 1, 2, \cdots, N$, is assigned a ground-truth class label $y_n \in \{1, 2, \cdots, L\}$, where L is the number of classes. Denote by $\mathcal{L} = \{l_1, l_2, ..., l_L\}$. The detailed form of T varies among different tasks. In image classification, T contains the whole image. For object detection, T is composed of all pixels. Given a deep network for a specific task, we use $F(x, t_n) \in \mathbb{R}^L$ to denote the classification score vector (before softmax normalization) on the n-th recognition target of x. The exact formulation of the loss functions is dependent on the object detection network's architecture. For example, the loss functions used in SqueezeDet can be found in § 3.3 of Wu *et al.* (Wu et al., 2016).

H.2. Accuracy Degradation

We measure the degradation of mean average precision (mAP) and recall on the SqueezeDet object detector (**?**)§3.3]squeezedet. Note that SqueezeDet's loss function comprises three terms corresponding to (a) bounding box regression, (b) confidence score regression, and (c) classification loss. In our experiment, we target the confidence score regression loss term to impact the mAP and recall of the model. Results in Table 8 show the efficacy of our multi-parameter augmentation strategy.

H.3. Realism

1. FID and LPIPS scores: We observe that for the augmenting samples generated using 3D-SDN, the average FID score is 0.102362. The corresponding average LPIPS distance is 0.523521. Observe that the value of the FID score is much lower than those discussed in § 3.2.

2. Survey: We follow the same protocol (and use the same survey) as described in § 3.2. For those samples generated using 3D-SDN, the average realism rating was 4.74 (average median = 4.26). In the case of 3D-SDN, the lower scores can be attributed to the poor rendering quality of the differentiable graphics framework; the perturbations made do not alter the geospatial positioning/orientation of the objects in the scene.

H.4. Robustness

When the benign model is retrained to be more robust, we notice that (i) its mAP improves by 7 percentage points (in comparison to Table 8) on semantic counterexamples, and (ii) on benign inputs, the map is only 15 percentage points lesser than the baseline (of mAP=99.4).

	Test	
Metric	Semantic	Benign
recall	92.97	93.7
mAP	72.76	84.73

Table 13. Robustness improvements due to semantic adversarial training.

H.5. Transferability

To investigate transferability in the object detection task, we train a YOLOv3 network (Redmon & Farhadi, 2018) on the KITTI dataset and observe if the semantic counterexamples (generated using SqueezeDet) induce performance degradation. On benign test inputs, the mAP of this network is 91.28%. When the test set is populated with the SqueezeDet-generated semantic counterexamples, we observed a drop in mAP to 87.50% (which is not as significant as in the case with SqueezeDet). This suggests that semantic counterexamples (generated for detection) are most effective against the network they are generated from. However, we are not categorically ruling out transferability for the 3D-SDN case (our experiments did not

involve testing a variety of hyperparameters as the semantic counterexample generation process using 3D-SDN is very time consuming).

I. ResNet-50 Experiments

We detail the results related of using ResNet-50 for our image classification experiments.

I.1. Informativeness

We repeat the same experiments as in § 3.1. While the results broadly follow a similar trend, observe that semantic counterexamples are more informative in comparison to samples generated using Halton sampling or random sampling (refer Table 14 and Table 15).

	Class												
Strategy	0	1	2	3	4	5	6	7	8	9	10	11	**overall**
benign	1	1	1	1	0.998	1	1	1	1	1	0.989	0.984	**0.997**
si-FGSM	0.993	0.560	0.512	0.677	0.267	0.85	0.666	0.821	0	0	0	0.025	**0.468**
sCW	0.997	0.849	0.874	0.740	0.683	0.916	0.666	0.863	0	0	0.010	0.025	**0.697**
sGD	0.998	0.760	0.680	0.811	0.51	0.966	0.733	0.904	0.020	0	0	0.031	**0.612**
Random (large)	0.983	0.687	0.571	0.685	0.345	0.916	0.6	0.739	0	0.04	0.010	0.057	**0.520**
Random (small)	1	0.963	0.916	0.818	0.715	1	1	0.945	0.122	0.04	0.020	0.174	**0.749**
Halton (large)	0.984	0.553	0.226	0.629	0.052	0.933	0.466	0.863	0.081	0	0	0.212	**0.383**
Halton (small)	1	0.982	0.932	0.897	0.757	1	1	1	0.571	0.12	0.030	0.317	**0.793**

Table 14. Accuracy degradation induced by different augmentation strategies. Observe that semantic counterexamples are effective at inducing accuracy degradation (lower the **overall** value, the better). All experiments are carried out using ResNet-50 as the target model.

Membership	None	Halton (large)	Halton (small)	Random (large)	Random (small)	sCW	sGD	si-FGSM
Binary	0.108	0.979	0.518	0.968	0.604	0.656	0.236	0.978
Fractional	0.122	1.029	0.548	1.017	0.657	0.697	0.258	1.033

Table 15. Information worth of augmentation samples generated using various strategies. Larger values are better. Binary membership uses the model prediction, *i.e.*, $\mu_i(x) = \mathbb{1}[F(x) = i]$ and fractional membership uses the model confidence, *i.e.*, $\mu_i(x) = s(F)(x)_i$. Larger values are better. All experiments are carried out on ResNet-50.

I.2. Attack Efficacy

We evaluate the efficacy of adversarial attacks on Resnet-50 in Table 16.

	Test			
Train	benign	si-FGSM	sGD	sCW
benign	0.986	0.469	0.612	0.698
si-FGSM	0.978	0.946	0.938	0.941
sGD	0.977	0.914	0.947	0.940
sCW	0.984	0.891	0.925	0.955

Table 16. **Accuracy metrics of robust networks retrained** with adversarial training using semantic counterexamples generated on the benign model. The rows indicate the attack method used to generate the retraining semantic counterexamples and the columns indicate the attack dataset used for evaluation. Observe that retraining on semantic counterexamples improves robustness across datasets, regardless of the initial train dataset (larger values are better). All experiments are carried out on ResNet-50.

I.3. Robustness

Robustness results are found in Table 17.

Train	Test			
	benign	si-FGSM$_{robust}$	sGD$_{robust}$	sCW$_{robust}$
benign	0.991	0.547	0.685	0.483
si-FGSM	0.978	0.5	0.331	0.698
sGD	0.978	0.771	0.569	0.862
sCW	0.991	0.733	0.586	0.711

Table 17. **Accuracy metrics for benign and adversarially retrained networks** on semantic counterexamples generated by using various methods (in the columns) on individual networks trained to be robust to individual methods (in the rows). Retraining against one method helps provide some robustness against *all* tested methods.

I.4. Realism

The results are reported in Table 18.

Method	Strategy	Class											
		0	1	2	3	4	5	6	7	8	9	10	11
FID	si-FGSM	7.99	31.97	33.87	17.62	11.02	19.15	38.60	13.35	27.11	30.10	19.98	12.46
	sGD	5.76	29.27	32.07	14.29	7.83	16.03	40.62	11.58	26.54	27.43	20.10	11.93
	sCW	2.57	6.31	6.08	12.88	3.91	15.03	19.30	12.46	27.62	29.80	19.43	12.84
LPIPS	si-FGSM	0.31	0.60	0.50	0.70	0.52	0.49	0.50	0.48	0.56	0.51	0.56	0.46
	sGD	0.26	0.58	0.45	0.67	0.49	0.45	0.50	0.45	0.55	0.49	0.53	0.45
	sCW	0.19	0.36	0.29	0.68	0.41	0.44	0.37	0.47	0.59	0.51	0.55	0.47

Table 18. Realism measures for augmentation samples generated using ResNet-50.

I.5. Cross-Model Transferability

The results are reported in Table 19.

Strategy	Class												
	0	1	2	3	4	5	6	7	8	9	10	11	**overall**
sCW	0.991	0.803	0.848	0.212	0.785	0.9	0.533	0.616	0.020	0.04	0.272	0.542	**0.761**
si-FGSM	0.878	0.521	0.521	0.078	0.362	0.916	0.2	0.685	0.040	0	0.252	0.562	**0.509**
sGD	0.943	0.618	0.495	0.252	0.635	0.983	0.066	0.822	0.082	0.04	0.353	0.619	**0.667**

Table 19. Transferability of semantic counterexamples generated using ResNet-50 to VGG-16. Lower the **overall** value, the better.

J. Sample Outputs

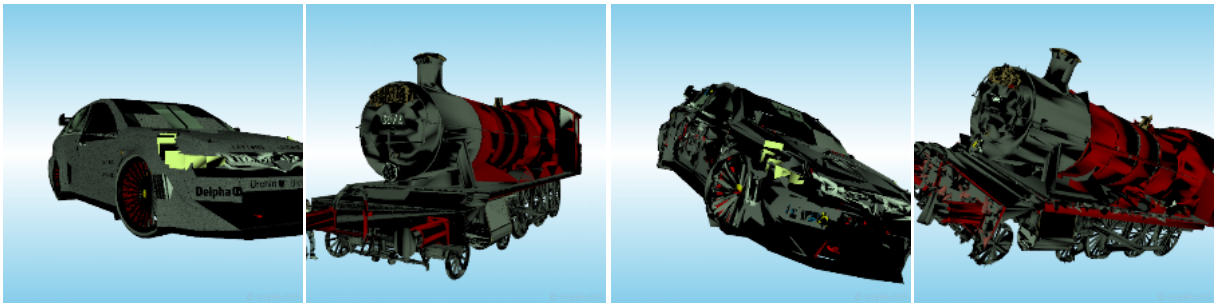

Figure 3. Benign images vs their adversarial counterparts generated by attacking the object's pose and vertices and feeding it back into `redner`. The benign images are on the top row, with the labels "car" and "train", respectively. The adversarial images are on the bottom and are misclassified as "boat" and "car", respectively.

Figure 4. **Semantic space adversarial examples.** Benign re-rendered VKITTI image (top), adversarial examples generated by iterative sFGSM over a combination of semantic features (bottom). Cyan boxes indicate car detected, yellow indicates cyclist, and purple indicates pedestrian. The adversarial example introduces small changes in car positions and orientations, and noticeable changes in their color. This causes the network to detect pedestrians where there are none.

K. Pipelines

The pipeline for classification is presented in Figure 5. The pipeline for object detection can be found in Figure 6.

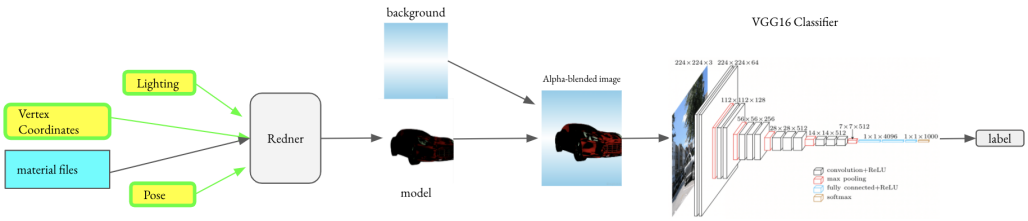

Figure 5. Overall pipeline for generating semantic counterexamples for classification. The renderer (Redner) is fed vertex, lighting, material, and pose information. The output model is then alpha-blended with a background and the resulting image is classified a DNN (say VGG-16). The gradients of the output are computed with respect to the semantic parameters (in yellow with green arrows). They can then be adversarially perturbed at the points indicated by the arrows highlighted in green and the resulting image is re-rendered and re-classified. We borrow the graphic for VGG-16 from (Loukadakis et al., 2018).

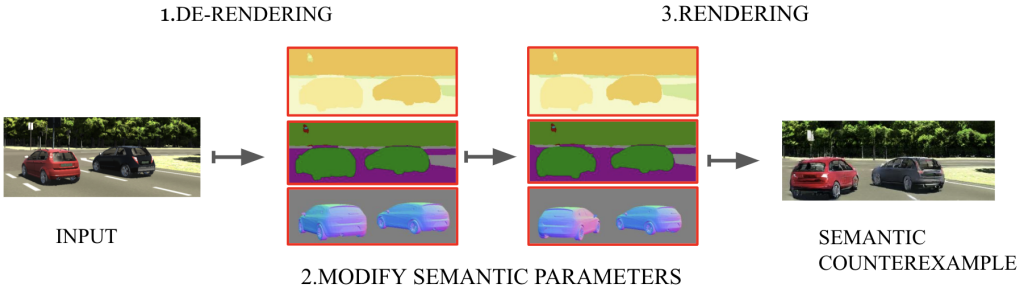

Figure 6. Our procedure for attacking an object detection framework: The input is de-rendered (step 1) to its intermediary representation (IR) - semantic, graphic, and textural maps. Then, this is adversarially perturbed (*e.g.,* the red car is rotated) as described in § 2 (step 2). The resulting IR is then re-rendered to the generate the counterexample (step 3).

L. Cross-Model Transferability

From Table 20, we observe that the semantic counterexamples generated using VGG-16 as the base model transfer to a ResNet-50 model trained on the benign dataset (see Table 6 for a comparison). This suggests that the ResNet-50 model could be retrained using semantic counterexamples generated using a different model. This can potentially be used to reduce the run-time efficiency of generating semantic counterexamples (as the cost of generating a semantic counterexample is proportional to the depth of the victim network). We validate that semantic counterexamples generated using ResNet-50 transfer over to VGG-16 in Appendix I.5.

Strategy	sCW	si-FGSM	sGD
Accuracy	0.588	0.621	0.716

Table 20. **Transferability** of semantic counterexamples generated using VGG-16 to ResNet-50. This suggests that images generated using VGG-16 transfer to a ResNet-50 architecture.

Takeaway: Retraining with semantic counterexamples improves robustness across adversarial attacks (§ 3.3) and the counterexamples transfer across models (§ L).

M. Extended Results

Method	Strategy	Class 0	1	2	3	4	5	6	7	8	9	10	11
FID	si-FGSM	5.62	20.56	23.88	13.40	8.56	16.25	21.14	11.05	18.30	16.91	18.68	8.04
	sGD	4.33	18.73	22.97	10.99	6.20	14.19	25.53	9.33	17.17	16.79	15.39	7.45
	sCW	6.23	26.48	29.74	11.18	8.39	15.85	31.97	11.75	21.65	17.08	15.93	7.79
	Halton (large)	2.48	32.68	21.10	15.42	13.98	9.15	38.27	5.09	21.94	27.71	20.38	5.82
	Random (large)	6.79	27.41	26.33	17.19	12.28	16.42	24.30	12.53	23.92	22.66	19.63	9.49
LPIPS	si-FGSM	0.25	0.53	0.40	0.60	0.45	0.43	0.36	0.41	0.44	0.33	0.47	0.37
	sGD	0.22	0.52	0.37	0.58	0.41	0.42	0.37	0.38	0.46	0.34	0.45	0.35
	sCW	0.27	0.58	0.45	0.60	0.46	0.45	0.42	0.40	0.50	0.35	0.45	0.39
	Halton (large)	0.12	0.59	0.37	0.66	0.55	0.26	0.51	0.22	0.51	0.46	0.47	0.28
	Random (large)	0.29	0.58	0.44	0.67	0.51	0.46	0.45	0.45	0.53	0.41	0.53	0.41

Table 21. **Realism measures** for augmentation samples generated using VGG-16. Lower the scores, more realistic the images.

Strategy	Class 0	1	2	3	4	5	6	7	8	9	10	11	**overall**
benign	0.999	0.996	0.992	0.969	0.997	0.983	1	0.986	0.939	0.84	0.849	0.956	**0.986**
sCW	0.883	0.407	0.319	0	0.353	0.783	0.133	0.671	0.041	0.12	0.2	0.635	**0.485**
si-FGSM	0.836	0.642	0.460	0	0.402	0.783	0.40	0.507	0	0.240	0.273	0.619	**0.529**
sGD	0.917	0.707	0.589	0	0.589	0.883	0.467	0.767	0.041	0.160	0.232	0.781	**0.657**
Random (large)	0.876	0.612	0.462	0	0.436	0.933	0.2	0.712	0.143	0.36	0.576	0.819	**0.582**
Random (small)	0.969	0.935	0.789	0.040	0.810	0.966	0.466	0.904	0.346	0.52	0.535	0.936	**0.830**
Halton (large)	0.744	0.293	0.142	0.008	0.017	0.833	0	0.657	0	0	0.010	0.476	**0.274**
Halton (small)	0.979	0.935	0.832	0.165	0.690	0.967	0.667	0.945	0.245	0.32	0.131	0.721	**0.757**

Table 22. **Accuracy degradation** induced by different semantic counterexample-generation strategies. Observe that semantic counterexamples are effective at inducing accuracy degradation (lower the **overall** value, the better). All experiments are carried out using VGG-16 as the target model.