Generating Semantic Adversarial Examples through Differentiable Rendering

Lakshya Jain

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2020-85
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2020/EECS-2020-85.html

May 28, 2020

Acknowledgement

Thank you to the following people, who helped immensely in the course of this work: Wilson Wu, Steven Chen, Varun Chandrasekaran, Uyeong Jang, Somesh Jha, Andrew Lee, Andy Yan, Tzu-Mao Li, David Wagner, and Sanjit Seshia.

Due to length constraints, specific acknowledgement breakdowns are available in the thesis PDF itself.

Generating Semantic Adversarial Examples through Differentiable Rendering

by Lakshya Jain

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

Professor Sanjit Seshia
Research Advisor

5/28/2020

(Date)

* * * * * * *

Professor David Wagner
Second Reader

May 25, 2020

(Date)

Generating Semantic Adversarial Examples through Differentiable Rendering

Abstract

Generating Semantic Adversarial Examples through Differentiable Rendering

by

Lakshya Jain

Masters of Science in Electrical Engineering and Computer Science

University of California, Berkeley

Sanjit Seshia, Chair

Deep neural networks (DNNs) are being increasingly applied in mission-critical situations, where high accuracy and robustness are essential. While deploying DNNs for computer vision and image processing tasks, the focus of adversarial analysis has been mainly on finding pixel-level changes that may impact the network's robustness. We examine an alternate approach - generating adversarial images by inducing perturbations in the *semantics* of an image, yielding interesting scenes that are more likely to occur in the real world. One can then use the generated examples to improve the robustness of DNNs through using them in data augmentation and adversarial retraining. In this report, we provide and implement a pipeline that can generate such scenes on demand by combining a differentiable rendering framework with gradient-based attacks. We demonstrate that the semantic adversarial examples generated by the pipeline can fool an object classification or detection framework, that retraining on these counterexamples is effective in making the network more robust to such attacks, and that the semantic robustness achieved against one attack appears to help achieve semantic robustness against other gradient-based attacks.

To my parents, Jayshree and Jawahar, my sister Priya, and my godparents Darsan, Toral, and Saumil.

Contents

Acknowledgments

Thank you to the following people, who helped immensely in the course of this work:

- Wilson Wu, for his immense help with the work and the code for the object detection portion of the pipeline.
- Steven Chen, for helping us get the object detection pipeline up and running.
- Varun Chandasekaran and Uyeong Jang, for their help throughout, from the beginnings with 3D-SDN and object detection to the work with Redner and classification, and for providing valuable guidance and insights regarding what experiments to run, what to investigate, and so forth.
- Somesh Jha, for being a valuable source of feedback in the direction of the research and in helping to formalize the theory behind these ideas.
- Sanjit Seshia, for advising me throughout the course of this work and for providing guidance, insights, consideration, and help on everything ranging from theory to experiments to new areas and tasks to target.
- Andrew Lee and Andy Yan, who helped immensely in the creation of the publicly available classification pipeline.
- Tzu-Mao Li and Tzu-Ming Harry Hsu, who were of immense help in understanding Redner and 3D-SDN, respectively.
- David Wagner, who provided excellent insights and feedback on my thesis and also gave valuable suggestions on new experiments to run.

Chapter 1

Introduction

1.1 Overview

Machine learning (ML) techniques, especially Deep Neural Networks (DNNs), have been successful in several domains, such as finance and healthcare, and are increasingly being used in safety-critical and mission-critical domains such as autonomous driving, medicine, and security (Amodei et al., 2016) (Seshia et al., 2016). However, several test-time (Biggio et al., 2013; Szegedy et al., 2014; Goodfellow et al., 2015; Kurakin et al., 2016) and training-time (Jagielski et al., 2018; Shafahi et al., 2018) attacks have made their adoption in high-assurance applications problematic. ML techniques, such as generative models, have also been used for nefarious purposes such as generating "deepfakes" (Liu et al., 2017; Zhu et al., 2017). Our focus in this paper is on test-time attacks in which an adversary generates a slightly perturbed sample to fool a classifier or an object-detector.

Let X be the sample space and Y be the space of labels. A classifier F is a function from X to Y. Given a sample $\mathbf{x} \in X$, most attacks for constructing adversarial examples find a perturbation δ with a small norm (typical norms that are used are l_∞, l_0, and l_2) such that $\mathbf{x}+\delta$ has a different label than $\mathbf{x}$, *i.e.* $F(\mathbf{x}) \neq F(\mathbf{x}+\delta)$.

In this report, we consider the problem of generating *semantic adversarial examples (SAEs)* (Hosseini & Poovendran, 2018; Joshi et al., 2019; Qiu et al., 2019; Dreossi et al., 2018c). In these examples, there is a richer set of transformations $\mathcal{T}$ that capture semantically-meaningful changes to inputs to the ML model. We assume a norm on $\mathcal{T}$ (this norm is induced by various parameters corresponding to the transformations, such as angle of rotation and size of the translation). In our universe, an adversary is given a sample $\mathbf{x}$ and wishes to find a transformation parameterized by $\theta \in \Theta$ with small norm such that $F(\tau(\mathbf{x}, \theta)) \neq F(\mathbf{x})$ (we consider untargeted attacks, but our ideas extend to targeted attacks as well). SAEs can also be viewed as outcomes of perturbations in a "rich" semantic feature space (*e.g.,*, texture of the image) rather than just the concrete feature space (*e.g.,*, pixels). Consequently, SAEs are typically physically realizable, and it is easy to understand how the changes in semantics results in an adversarial example. SAEs have been

considered in the literature (Xiao et al., 2018; Dreossi et al., 2018c; Huang et al., 2019; Dreossi et al., 2017), but prior works typically consider a small set of fixed transformations (*e.g.,* rotation and translation, or modifying a single object's texture).

Our goal in this work is to provide the framework for a pipeline that can generate semantic adversarial examples on demand by leveraging a differentiable renderer and gradient-based attacks. We examine and demonstrate the efficacy of our pipeline through evaluating gradient-based semantic adversarial attacks and and showcasing the utility of the resulting SAEs in serving as data augmentation tools to improve the robustness of a neural network.

We pick two tasks to showcase our approach on: object classification, in which a neural network aims to classify an image of an object with the correct label, and object detection, in which a network aims to classify and draw accurate bounding boxes around objects in the scene. We implement our pipeline for each approach and perform gradient-based attacks on each to generate SAEs. We then evaluate the effectiveness of the generated SAEs, both in degrading the performance of the original model and in data augmentation to improve the model's robustness.

1.2 Related Work

The notion of SAEs has been proposed in the literature in specific contexts. For example, in the NLP domain, Lei et al. (2018) propose a gradient guided greedy algorithm to make semantic changes to text documents. Sharif et al. (2016) propose attacks against facial recognition systems that are both inconspicuous and realizable. Similarly, Evtimov et al. (2017) obtain patches that can be added to road signs, rendering object detectors useless. In both these works, however, the modifications to semantics are not found by searching the space of semantic modifications (and found instead by *realizing* changes in pixel intensities); such approaches will fail to generalize for other domains. 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. Dreossi et al. (2018c) present an approach to generate SAEs based on systematic sampling of the semantic space coupled with verification techniques for a closed-loop cyber-physical system containing the ML model, where they treat the ML model as a black box. However, this sampling-based "black box" approach faces scalability issues when dealing with high-dimensional spaces and also fails to exploit structure of the ML model to perform a targeted search for SAEs.

Xiao et al. (2019) utilize a differentiable rendering framework to introduce changes in shape and texture that are capable of fooling a variety of ML models for various tasks. However, our approach builds on this work by taking into account advances in both differentiable rendering (using (Li et al., 2018) allows, for the first time, full and accurate backpropogation through the renderer, allowing for the retrieval of *exact* gradients) and *differentiable inverse graphics*, allowing us change a richer set of semantic parameters and utilize off-the-shelf gradient-based attacks to generate counterexamples. An interesting contribution by Qiu et al. (2019) suggests how a generative model

can be used to introduce semantic modifications. However, generative models are notoriously difficult to train and operate, making them unusable for a wide variety of tasks. Similar issues occur in the work by Joshi et al. (2019). To the best of our knowledge, however, our method is the first to use off-the-shelf gradient-based methods to generate semantic adversarial counterexamples with which to augment datasets.

Adversarial Examples and Robustness: There is extensive research for generating adversarial examples in the pixel space; we henceforth refer to these as *pixel-perturbations*. Goodfellow et al. (2015) propose the fast gradient sign method (FGSM) where inputs are modified in the direction of the gradients of the loss function with respect to input, causing a variety of models to misclassify their inputs. Madry et al. (2017) generalize this approach and propose the projected gradient descent (PGD) approach working using the same intuition. While these approaches suggest modifications to the raw pixel values, other methods of generating adversarial examples exist. Athalye et al. (2017) introduce an approach to generate 3D adversarial examples (over a chosen distribution of transformations). Engstrom et al. (2019) observe that modifying the spatial orientation of images results in misclassifications. Similarly, Geirhos et al. (2018) discovered that certain models are biased towards textural cues.

To improve robustness, current approaches include adversarial training (Madry et al., 2017), smoothing-based approaches (Cohen et al., 2019; Lécuyer et al., 2018), or through specific regularization (Raghunathan et al., 2018). An alternative approach, utilizing some notion of semantics, is advocated in the work of Guo et al. (2017). The authors augment the training set with transformed versions of training images, utilizing basic image transformations (*e.g.,*, scale and re-cropping) and total variance minimization, and demonstrate an improvement in robustness. Dreossi et al. (2018b) improve the robustness of SqueezeDet (Wu et al., 2016) through counterexample guided data augmentation; these counterexamples are synthetically generated by sampling from a space of transformations and applying them to original training images.

Inverse Graphics and Differentiable Rendering Frameworks The process of finding 3D scene parameters (geometric, textural, lighting, etc.) given images is referred to as inverse graphics (Baumgart, 1974). There is a history of using gradients to solve this problem (Blanz & Vetter, 2002; Shacked & Lischinski, 2001; Barron & Malik, 2015). Kulkarni et al. (2015) 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. Pipelines for general differential rendering were proposed by Loper & Black (2014) and Kato et al. (2018). (Yao et al., 2018) builds on Kato's work by incorporating this renderer into its inverse-graphics framework to render the meshes of cars.

Li et al. (2018) design a general-purpose differentiable ray tracer; gradients can be computed with respect to arbitrary semantic parameters such as camera pose, scene geometry, materials, and lighting parameters. This pipeline is fully differentiable and, with the help of edge sampling and path tracing, can render images with much more detail, including texture, shadows, and global illumination. Finally, 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.

1.3 Thesis Outline and Contributions

Section 1 of this thesis (above) describes the overall idea and the prior work in this field. Section 2 will discuss the adversarial learning space and the techniques that we utilize, as well as the rendering and object detection/classification frameworks we use for our work. Section 3 will discuss the implementation of our pipeline and the experimental evaluations.

In this work, the main contribution is the creation of a data generation framework that can identify a network's "weak points" and generate counterexamples to train on, reinforcing the network and making it more robust overall to attacks and adversarial scenes. As an added contribution, we also showcase the transferability of robustness against gradient-based semantic attacks and show that robustness to one attack helps with robustness against other types of attacks.

We implement and provide a fully differentiable pipeline that supports a variety of gradient-based semantic attacks that degrade the performance of an object detector or classifier. The classification variant of this pipeline is publicly available and a link can be found in the acknowledgements. The object-detection variant is available on request, as we do not currently have the rights to make the code publicly available.

Portions of section 1 and 2 were partially sampled from our writeup Jain et al. (2019), which was coauthored with Varun Chandrasekaran, Somesh Jha, Wilson Wu, Steven Chen, and Uyeong Jang. The object detection portions of Section 3 were sampled from the same coauthored writeup as well. The code for the classification attack was co-written with Andrew Lee and Andy Yan and can be found at `github.com/BerkeleyLearnVerify/rednercounterexamplegenerator/`.

Chapter 2

Adversarial Learning

Consider a space Z of the form $X \times Y$, where X is the sample space and Y is the set of labels. From here on we will assume that $X = \Re^n$. Let H be a hypothesis space (*e.g.,*, weights of a DNN). We assume a loss function $\ell : H \times Z \mapsto \mathbb{R}$ so that given a hypothesis $w \in H$ and a labeled data point $(x, y) \in Z$, the loss is $\ell(w, x, y)$. The output of the learning algorithm is a *classifier*, which is a function from $\Re^n$ to Y. To emphasize that a classifier depends on a hypothesis $w \in H$, which is output of the learning algorithm, we will denote it as F_w (if w is clear from the context, we will sometimes simply write F). If a datapoint x is perturbed to be x^* and the aim is to make a classifier be robust to all such x^* within a certain space of admissible inputs such that the distance between x^* and x is no more than α, where robustness would mean being $F(x)$ and $F(x^*)$ being within β of each other, then an adversarial example is one that violates these properties, as stated in Dreossi et al. (2019).

2.1 Adversarial Attacks

We will focus our discussion on untargeted attacks, but our discussion also applies to targeted attacks. An adversary A's goal is to take any input vector $\mathbf{x} \in \Re^n$ and produce a minimally altered version of $\mathbf{x}$, an *adversarial example* denoted by $A(\mathbf{x})$, that has the property of being misclassified by a classifier $F : \Re^n \rightarrow Y$. The adversary wishes to solve the following optimization problem:

$$\begin{array}{ll} \min_{\delta \in \Re^n} & \mu(\delta) \\ \text{such that} & F(\mathbf{x}+\delta) \neq F(\mathbf{x}) \end{array}$$

The various terms in the formulation are: μ is a norm on $\Re^n$; μ can be l_∞, l_0, l_1, or l_p ($p \geq 2$). While not representative, these norms are commonly used as an approximation of a human's visual perception (Sen et al., 2019). If δ is the solution of the optimization problem given above, then the adversarial example $A(\mathbf{x}) = \mathbf{x}+\delta$.

Fast Gradient Sign Method (FGSM): The *fast gradient sign method (FGSM)* (Goodfellow et al., 2015) was one of the first untargeted attacks developed in literature. The adversary crafts an

adversarial example for a given legitimate sample $\mathbf{x}$ by computing (and then adding) the following perturbation:

$$\delta = \varepsilon \operatorname{sign}(\nabla_{\mathbf{x}} L_F(\mathbf{x})) \tag{2.1}$$

The function $L_F(\mathbf{x})$ is a shorthand for $\ell(w, \mathbf{x}, l(\mathbf{x}))$, where w is the hypothesis corresponding to the classifier F, $\mathbf{x}$ is the data point and $l(\mathbf{x})$ is the true label of $\mathbf{x}$ (essentially we evaluate the loss function at the hypothesis corresponding to the classifier). The gradient of the function L_F is computed with respect to $\mathbf{x}$ using sample $\mathbf{x}$ and label $y = l(\mathbf{x})$ as inputs. Note that $\nabla_{\mathbf{x}} L_F(\mathbf{x})$ is an n-dimensional vector and $\operatorname{sign}(\nabla L_F(\mathbf{x}))$ is a n-dimensional vector whose i^{th} element is the sign of the $\nabla_{\mathbf{x}} L_F(\mathbf{x})[i]$. The value of the *input variation parameter* ε factoring the sign matrix controls the perturbation's amplitude. Increasing its value increases the likelihood of $A(\mathbf{x})$ being misclassified by the classifier F but also makes adversarial examples easier to "detect" by humans.

The key idea is that FGSM takes a step *in the direction of the gradient of the loss function with respect to the input*, thus attempting to maximize the loss function using its first-order approximation. Recall that stochastic gradient descent (SGD) takes a step in the direction that is on expectation opposite to the gradient of the loss function because it is trying to minimize the loss function. For our work, we use an "iterative" version of FGSM that repeats this procedure for k steps, like the iterative gradient sign technique described in Kurakin et al. (2018).

Projected Gradient Descent (PGD): In *Projected Gradient Descent (PGD)*, where we use the optimization variant described in Carlini & Wagner (2017), we find a perturbation in an iterative manner. Standard gradient descent is performed and the perturbation is then clipped into a ball of radius ε. Assume that we are using the l_p norm (the choice of p is an implementation choice; we used the l_∞ norm). Assume $\mathbf{x}_0$ is the original sample $\mathbf{x}$.

$$\mathbf{x}_{k+1} = \Pi_{B_p(\mathbf{x},\varepsilon)}(\mathbf{x}_k + \alpha(\nabla_{\mathbf{x}} L_F(\mathbf{x}))) \tag{2.2}$$

The operator $\Pi_{B_p(\mathbf{x},\varepsilon)}(y)$ is the projection operator, *i.e.* it takes as input a point y and outputs the closest point in the ε-ball (using the l_p-norm) around $\mathbf{x}$. The iteration stops after a certain number of steps (the exact number of steps is a hyperparameter).

Carlini-Wagner (CW): In the *Carlini-Wagner* (Carlini & Wagner, 2017) (CW) attack algorithm, the perturbation distance metric $D(x, x+\delta)$ is minimized subject to $f(x+\delta) \leq 0$ and $x+\delta \in [0,1]^n$, where $f(x)$ is a function indicating whether or not the desired misclassification has been achieved. The formulation can thus be simplified to

$$\min \|\delta\|_p + c \cdot f(x+\delta) \text{ such that } x + \delta \in [0,1]^n \tag{2.3}$$

For the purposes of our attack, we use the 2-norm in our distance metric.

2.2 Semantic Adversarial Examples (SAEs)

The presentation of SAEs in this section is a specific formalization of an attacker seeking to violate semantic robustness, specifications for which were surveyed in Seshia et al. (2018). Let $\mathcal{T}$:

$(\Re^n \times \Theta) \rightarrow \Re^n$ be a set of transformations parameterized by a space Θ, and μ is a norm over Θ. The reader can think of Θ as parameters that control the transformations (*e.g.,* the angle of rotation). Given $\theta \in \Theta$, $\tau(x, \theta)$ is the image transformed according to the parameters θ. We assume that there is a special identity element in Θ (which we call $\perp$) such that $\tau(\mathbf{x}, \perp) = \mathbf{x}$. An adversarial attack in this universe is characterized as follows:

$$\begin{array}{ll} \min_{\theta \in \Theta} & \mu(\theta) \\ \text{such that} & F(\tau(\mathbf{x}, \theta)) \neq F(\mathbf{x}) \end{array}$$

In other words, we ideally want to find a "small perturbation" in the parameter space Θ that will misclassify the sample[1]. Consider the function $L_F(\tau(\mathbf{x}, \theta))$. The derivative with respect to θ is $\left[\frac{\partial \tau}{\partial \theta}\right]^\top \Big|_{\mathbf{x}} \nabla_{\mathbf{z}} L_F(\mathbf{z})|_{\mathbf{z}=\tau(\mathbf{x},\theta)}$ (the notation $\left[\frac{\partial \tau}{\partial \theta}\right]^\top \Big|_{\mathbf{x}}$ is the transposed Jacobian matrix of τ as a vector-valued function of θ, evaluated at $\mathbf{x}$, and $\nabla_{\mathbf{z}} L_F(\mathbf{z})|_{\mathbf{z}=\tau(\mathbf{x},\theta)}$ is the derivative evaluated at $\tau(\mathbf{x}, \theta)$). The semantic version of FGSM (*sFGSM*) will produce the following θ:

$$\theta^\star = \varepsilon \operatorname{sign}\left(\left[\frac{\partial \tau}{\partial \theta}\right]^\top \Big|_{(\mathbf{x},\perp)} \nabla_{\mathbf{z}} L_F(\mathbf{z})|_{\mathbf{z}=\tau(\mathbf{x},\perp)} \right) \tag{2.4}$$

The adversarial example $A(\mathbf{x})$ is $\tau(\mathbf{x}, \theta^\star)$. Note that we do not assume any special properties about τ, such as linearity. We only assume that τ is differentiable.

In a similar manner a semantic version of the PGD attack variant that we implemented (*sPGD*) can be constructed. Let $\theta_0 = \perp$ and $\mathbf{x}_0 = \mathbf{x}$. The update steps correspond to the following two equations:

$$\begin{aligned} \theta_{k+1} &= \Pi_{B_\mu(\theta_0,\varepsilon)} \left(\theta_k \oplus \alpha \left(\left[\frac{\partial \tau}{\partial \theta}\right]^\top \Big|_{(\mathbf{x}_k,\theta_k)} \nabla_{\mathbf{z}} L_F(\mathbf{z})|_{\mathbf{z}=\tau(\mathbf{x}_k,\theta_k)} \right) \right) \\ \mathbf{x}_{k+1} &= \tau(\mathbf{x}_0, \theta_{k+1}) \end{aligned}$$

Note that $\Pi_{B_\mu(\cdot,\cdot)}$ 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). It is important to note that in our semantic implementation, the projection spaces are more precisely defined by considering each semantic space individually, instead of the more loosely defined global vertex coordinates. For example, in the classification setting, the projection space S_r for the rotation vector r for the object would be different and independent from the projection space S_v for individual vertex perturbations, which

[1]However, the existence of such a function for the semantic space is an open research question; the semantic parameter space is not homogeneous and it is unclear if one function can be used to capture the distance amongst all these transformations. Not only should the norm measure the changes in the semantic space, it should also approximate human perception. In its presence, one would be able to effectively measure the distortion introduced by the adversary in the semantic space.

would deal with the vertex's perturbation distance from its initial position relative to the rest of the shape.

The operator $\oplus$ is the aggregation operator (similar to addition in $\Re^n$), but in the parameter space Θ. The precise axioms satisfied by $\oplus$ depends on Θ, but one axiom we require is:

$$\tau(\mathbf{x}, \theta_1 \oplus \theta_2) \quad = \quad \tau(\tau(\mathbf{x}, \theta_1), \theta_2)$$

In fact, our recipe can be used to transform any attack algorithm such as Carlini & Wagner (2017) that adds a perturbation δ to its "semantic version" as follows:

- Replace δ with θ.
- Replace $\mathbf{x} + \delta$ with $\tau(\mathbf{x}, \theta)$.
- Use chain rule to compute the gradients of terms that involve $\tau(\mathbf{x}, \theta)$.

For example, semantic Carlini-Wagner (**sCW**) can be constructed as

$$\min \|\theta\|_p + c \cdot f(\tau(\mathbf{x}, \theta)) \text{ such that } \tau(\mathbf{x}, \theta) \in [0, 1]^n \tag{2.5}$$

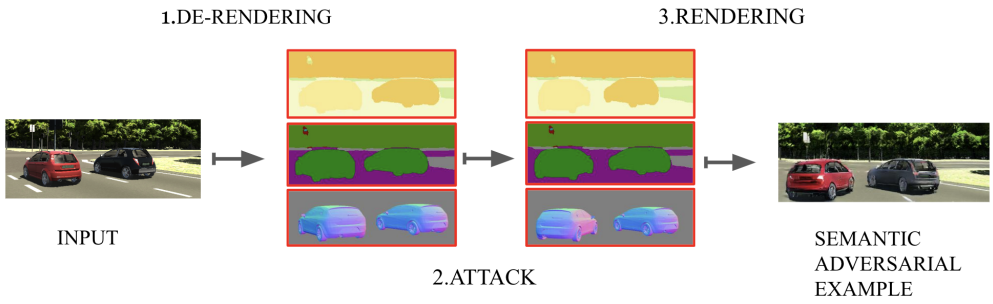

Figure 2.1: *Our procedure for attacking an object detection framework: The input is de-rendered (step 1) to its intermediary representation (IR) - semantic, graphic, and textural maps. Then, this is adversarially perturbed* (e.g., *the red car is rotated) as described in* § 2.2 *(step 2). The resulting IR is then re-rendered to the generate the SAE (step 3).*

2.3 Graphics Frameworks

We apply the above framework to images by employing either a differentiable renderer or a de-renderer in an inverse graphics setting.

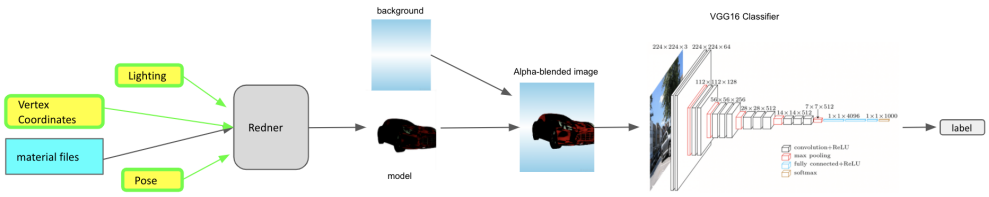

Figure 2.2: *Our procedure for attacking a classification framework: The renderer (Redner) is fed vertex, lighting, material, and pose information. The output model is then alpha-blended with a background and the resulting image is classified by VGG16 (Simonyan & Zisserman, 2014; Loukadakis et al., 2018). Classifier gradients are retrieved 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 rendered and reclassified.*

2.3.1 Differentiable Rendering

For a differentiable renderer, the vertex coordinates v and materials map m of the scene are accepted as inputs, along with the light source ℓ and the camera angle ζ. The renderer can be viewed as a transform $\gamma(v, m, \zeta, \ell) \rightarrow \Re^n$, where the input involves all the semantic parameters available and the output is an image in the pixel space $\Re^n$. By perturbing the semantic inputs of the scene using gradient-based attacks and feeding them into the renderer, one can thus create or generate new images that would serve as adversarial counterexamples that the model would be weak against. For our classification attack, as shown in figure 2.2, we utilize `redner`, the renderer proposed and designed in (Li et al., 2018), which serves as a fully differentiable graphics framework[2] that utilizes Monte-Carlo ray tracing and edge-based sampling. With redner, exact semantic gradients with respect to v, ζ, and ℓ can be retrieved, allowing us to perform the attacks in 2.1 with respect to the semantic feature spaces in question.

2.3.2 Inverse Graphics

An alternative rendering means for our flexible framework is achieved by using inverse graphics. The setting can be thought of as consisting of two transformations: (a) a de-renderer $\beta : \Re^n \rightarrow \mathcal{S}$, and (b) a renderer $\gamma : \mathcal{S} \rightarrow \Re^n$. Here, $\mathcal{S}$ is the intermediate representation (IR). In the de-renderer we utilize (Yao et al., 2018), the IR contains a semantic map, texture codes, and 3D attributes. Let Θ be the set of changes to the IR (*e.g.,* change to the texture code to make it more cloudy) and $\perp \in \Theta$ corresponds to the identity. Suppose there is an operator $\alpha : (\mathcal{S} \times \Theta) \rightarrow \mathcal{S}$ that, given a $\theta \in \Theta$, transforms the IR, *i.e.* $\alpha(s, \theta) = s'$ for $s, s' \in \mathcal{S}$. In this case, the function $\tau(\mathbf{x}, \theta)$ is equal to $\gamma(\alpha(\beta(\mathbf{x}), \theta))$. As is the case in differentiable renderers, the functions β, γ, α are differentiable and hence attacks like sFGSM and sPGD can be implemented. The full pipeline for object detection and the use of inverse graphics can be found in figure 2.2.

[2]This is distinct from inverse graphics, which is used for our object detection case.

Chapter 3

Implementation and Evaluation

3.1 Implementation

In this section, we describe the various components used in our implementation to generate SAEs, and describe experiments carried out to determine the impact of choice of semantic parameters towards generating effective SAEs for object detectors. We stress that the exact choice of components are not important; our contribution is our automated approach to combine these components to generate SAES in a manner that is general (see § 2.2 for more details) and would work with any inverse graphics framework. We also believe that our methodology will work for other tasks such as classification and semantic segmentation; these tasks require optimizing over a different cost function. In other words, if there is an existing pixel attack, we should be able to easily transform it into a semantic attack (with the right inverse graphics framework).

The three main components required to successfully generate SAEs include: (a) a differentiable inverse graphics framework, (b) a victim model (which is also differentiable), and (c) an attack strategy. We describe our choices for the proof-of-concepts below.

3.1.1 Object Classification

For object classification, we used the VGG16 network (Simonyan & Zisserman, 2014), further trained on our custom dataset, to classify images. We used ShapeNet2.0 (Chang et al., 2015), a dataset that contains 3D modeling information for hundreds of Imagenet classes, to acquire the data needed to assemble scenes, and chose 12 classes to render images of, train the classifier on, and generate attack images for. These classes were the airplane, bus, car, boat, trashcan, bench, helmet, motorcycle, mailbox, skateboard, tower, and train classes – all of which are objects that one may see while out driving on roads.

Our dataset is generated by `redner` via the object files in the ShapeNet2.0 Dataset (Chang et al., 2015), in which a scene/object is represented by a sequence files that encode information about an object's vertices and its material composition. These files are read and the data passed into

`redner` along with a camera angle, and this generates a 3D model of the scene as a result. The model image is then combined with a sky-blue and white gradient background using alpha blending to generate an output image of the scene.

We first create our dataset by generating 18,348 images; this is done by taking 4,587 unique meshes and applying 4 different camera angle poses on them – a forward, top, right, and left view. We split this up into training, validation, and test sets; the disjoint nature of these datasets means that there is a one-to-one correspondence between SAEs and benign images for training, validation, and test datasets, meaning we will never train on an SAE whose benign counterpart is found in the test set (or vice versa). We then retrain the original VGG16 ImageNet network as a 12-class classifier on our new dataset of benign images to ensure that the classifier is tuned to the dataset in question (this classifier, trained on this dataset, will be referred to as `VGG_benign` from hereon out). Finally, we attack this classifier adversarially and generate adversarial datasets through three standard gradient attacks: sFGSM, sPGD, and the semantic Carlini-Wagner (sCW) attack. The adversarial test sets we create for **evaluation** purposes, as in 3.2.2, thus consist entirely of SAEs generated through attacking the benign test set images with the specified attack algorithm. Similar to Simonyan et al. (2013), we target the raw (unnormalized) class score of the correct label instead of the normalized softmax probability.

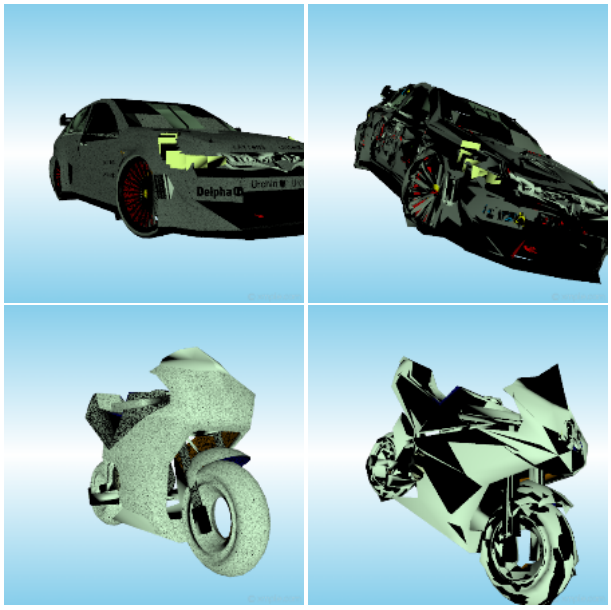

Figure 3.1: ***Benign images vs their adversarial counterparts generated by attacking the object's pose and vertices and feeding it back into `redner`.*** *For the benign "car" class image (top left), the corresponding adversarial example (top right) is misclassified by the network as "boat". For the benign "motorcycle" class image (bottom left), the corresponding adversarial example (bottom right) is misclassified as "car".*

Adversarial attacks are conducted in our pipeline by classifying the image, retrieving the gradient of the classifier's class score with respect to its semantic parameters (such as vertex coordinates and camera angle), perturbing the semantic parameters adversarially, and then re-rendering a new image through passing the information back into `redner` as described in the prior paragraph. For each attack algorithm, the final step described above (semantic adversarial perturbations) was done for 5 iterations; the choice of iterations was decided by a hyperparameter sweep and a qualitative comparison of output images, where the goal of misclassification was balanced by time and the intelligibility of the images.

3.1.2 Object Detection

For object detection, to obtain the semantics associated with our inputs and to generate the final SAEs, we use the inverse graphics framework(*i.e.* a combination of a semantic, textural and geometric de-rendering pipeline and a generative model for rendering) created by Yao et al. (2018). The models in this framework were trained entirely using the VKITTI dataset (Gaidon et al., 2016). These images comprise of simulations of cars in different road environments in virtual worlds. The VKITTI dataset is constructed using a novel real-to-virtual cloning methodology, mirroring many elements that exist in a real world.

The de-rendering pipeline is used to obtain the initial semantic features associated with input images. These semantic features include (a) *color*: the car's texture codes, which change its color, (b) *weather*: the weather and time of day, (c) *foliage*: the surrounding foliage and scenery, (d) *rotate*: the car's orientation, (e) *translate*: the car's position in 2D, and (f) *mesh*: the 3D mesh which provides structure to the car.

The final SAEs were produced using the generative model, which is part of the inverse graphics framework. Specific modifications were made to the differentiable graphics framework we used to ensure that gradients were easy to calculate. The codebase did not originally support end-to-end differentiation as each branch (semantic, geometric, textural) was trained separately. In particular, several image manipulation operations (normalization, rescaling through nearest-neighbor and bilinear interpolation) were implemented in a non-differentiable manner. We implemented the differentiable equivalents of these operations to allow backpropagation. Furthermore, we implemented a weak perspective projection for vehicle objects, as well as an improved heuristic for inpainting of gaps in the segmentation map due to object translations/rotations, in order to improve the quality of the rendering. Additionally, the inverse graphics framework and the object detector (which we describe next) had contrasting dependencies (for libraries, `tensorflow`, cuDNN, and CUDA); resolving these dependencies involved significant code rewriting.

We use the popular and representative SqueezeDet object detector (Wu et al., 2016) as the victim model. This model was originally trained on the KITTI dataset (Geiger et al., 2013). We perform transfer learning on this model using 6339 randomly chosen images from the VKITTI dataset; we wanted the object detector to better adapt to images outside the domain it was initially trained for. However, images produced by the differentiable graphics framework contain artifacts (*i.e.* distor-

tions in the images); these artifacts could be mistaken for pixel perturbations and would impact our evaluation results. To deal with this issue, we retrain SqueezeDet using identity transform re-rendered images[1] produced by the generative model.

Finally, we utilize these gradients and the semantics associated with each input in crafting adversarial attacks using the iterative sFGSM (for 6 iterations). On average, for a CPU-only implementation, generating such a SAE requires 42.81 seconds with about 85% of the time expended in the inverse graphics framework. We also stress that our choice of the number of iterations is restricted by our choice of the inverse graphics framework – in the case of 3DSDN, each re-rendering results in a slight amount of noise or extra artifacts being injected into the image, resulting in a poorer rendering. Using more iterations resulted in unintelligible outputs. We believe that the generation of SAEs is not inhibited by the complexity of the semantic parameter space; better engineering can substantially improve the outputs and the time it takes to generate them.

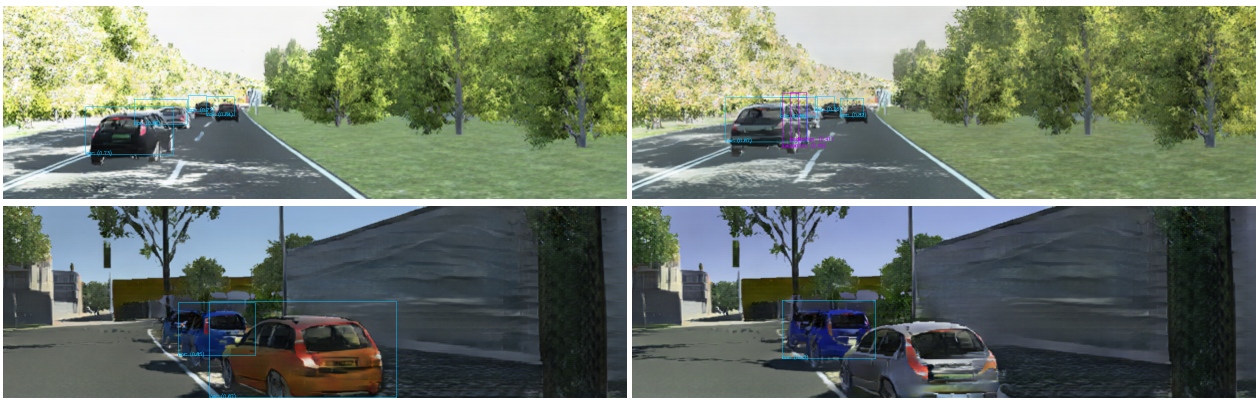

Figure 3.2: ***Semantic space adversarial examples.*** *Benign re-rendered VKITTI image (left), adversarial examples generated by iterative sFGSM over a combination of semantic features (right). Cyan boxes indicate car detected, purple indicated pedestrian, and yellow indicate cyclist. The adversarial 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 (top) and to fail to detect a car in the immediate foreground (bottom).*

3.2 Evaluation

We designed and carried out experiments to answer the following questions: (1) Do SAEs cause performance degradation in their respective tasks? (2) Which parameters is the model most susceptible to, and to what degree? and (3) Can the generated SAEs be used for improving robustness?

[1]Images passed through the de-rendering and rendering framework, without modifying the IR or any associated semantic parameters.

We discuss the answers to (1) in 3.2.2 and 3.2.3.2. The findings for (2) are discussed in 3.2.2 and 3.2.3.1, and the answers to (3) can be found in 3.2.4.

3.2.1 Selecting Hyperparameters

In the pixel perturbation setting, all pixels are equal *i.e. any* pixel can be perturbed. Whether such homogeneity naturally exists in the semantic space is unclear. However, we have additional flexibility; 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*. The degree of modification is determined by the *input variation/step-size/learning-rate* hyperparameter $\varepsilon \in [0, 1]$. For SAEs, the value of ε is proportional to the magnitude of the geometric and textural changes induced; the effect depends on the semantic parameter under consideration.

Our choices of hyperparameter ε was made through extensive visual inspection from several independent sources; we pruned the range of ε to ranges that would give visually intelligble outputs (as qualitatively assessed by independent observers) and then performed hyperparameter sweeps to check for maximal performance degradation.

3.2.2 Adversarial Attack Efficacy: Object Classification

We evaluated the impact of pose and vertex-based semantic attacks on `VGG_benign` through three standard adversarial algorithms: semantic Carlini-Wagner (sCW), semantic FGSM (sFGSM), and semantic PGD (sPGD). We also implemented and evaluated the impact of lighting intensity attacks, but found that they did not result in any noticeable performance degradation; therefore, for brevity, although lighting is supported in our pipeline and included in our released code, we only report pose and vertex attack results here.

Gradients were normalized to provide a sense of scale, and our pose (measured in radians instead of degrees) and vertex hyperparameters for the algorithms are as follows: For sPGD, we used a learning rate of 0.20 for the pose parameter and 0.01 for the vertex parameter. The feasible region, meanwhile, was bounded by ± 1.0 for the pose parameter and ± 0.05 for the vertex perturbation magnitude[2]. For sFGSM, the pose epsilon was 0.15 radians and the vertex epsilon was 0.002 units. For sCW, the learning rate was 0.30 for pose and 0.01 for vertex, and 5 iterations were conducted for all 3 attack algorithms.

As a baseline, `VGG_benign`'s performance on **benign** images was **98.6%**. We report the impact of adversarial attacks on `VGG_benign`'s performance below in 3.1.

[2]The reader may note that the bounds will never actually be hit under this implementation of sPGD with the current feasible region, meaning that no clipping would occur; this is because the feasible region, as referenced in Carlini & Wagner (2017), can be thought of as the space of acceptable perturbations. In practice, the space of perturbations that lead to realistic images is actually larger than the limits that would be hit by sPGD under the current attack; hence, the feasible region is just listed here as the maximum possible perturbations that could occur.

Attack \ Parameters	*Vertex*	*Pose*	*Pose + Vertex*
sCW	89.2	54.3	**48.5**
sFGSM	86.2	61.5	52.9
sPGD	89.7	74.1	65.9

Table 3.1: *Performance of* `VGG_benign` *on SAE datasets. Accuracies are reported as percentages of correctly classified images. The model had an accuracy of 98.6% on benign/non-adversarial inputs. Each (row, column) combination represents an evaluation dataset generated by attacking the specified parameters via the listed attack; for example, the model's accuracy on the evaluation dataset of SAEs generated by attacking pose and vertex simultaneously via sCW is 48.5%. We observe that multi-parameter modifications are much more effective than single-parameter modifications. Full class accuracies are included in appendix A.*

We note that the attacks listed above that modify the geometry of the object are extremely effective in reducing the network's overall performance, and that multi-parameter attacks are significantly more effective than single parameter attacks. In fact, we see a 46% drop in the model's overall performance when using the sFGSM attack and a 50% drop with the semantic Carlini-Wagner attack, with the detector's performance dropping from a robust 98.6% to a much more concerning 48.5% in the worst case. This is most likely due to the introduction of unique angles, meshes, and visual perspectives that are not frequently encountered in assembled datasets. It appears that the classifier is most sensitive to pose modifications; this is shown by the best vertex attack only dropping accuracy by 12%, while the best pose attack drops accuracy by 44% (numbers rounded to the nearest percentage point). It is possible that adding rotations to minibatches during training, as discussed in Perez & Wang (2017) may help protect against this to some degrees.

The above results carry dire implications for safety-critical applications, as it speaks to weaknesses in the dataset and the model that our attack discovers; for example, cars that are rotated upwards or downwards to be viewed at angles similar to those they would encounter on steep inclines should still be classified as cars, but this is a common failure case for the network. Vertex perturbations that deform the mesh of the car (simulating, in a sense, the impact of an accident on a car's chassis) are less effective than pose perturbations, but they are still good at decreasing the overall performance of the model. Moreover, they help compound the difficulty of classification, as shown by the efficacy of the dual pose and vertex attack in comparison to the single-parameter attack modification.

Through the efficacy of these attacks, we can thus see that our pipeline successfully leverages these gradient-based techniques to generate semantic counterexamples that would degrade the performance of a network. In particular, changes to the geometry of the scene in ways that generate cases that the network is not used to are key failure points discovered in the process.

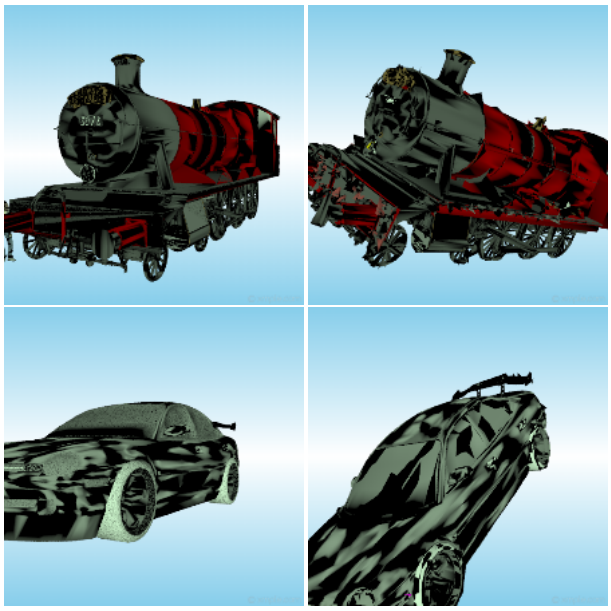

Figure 3.3: ***Benign images vs their adversarial counterparts generated by attacking the object's pose and vertices and feeding it back into `redner`.*** *For the benign "train" class image (top left), the corresponding adversarial example (top right) is misclassified by the network as "car", while the "car" image (bottom left) has its corresponding adversarial example (bottom right) misclassified as "helmet".*

3.2.3 Adversarial Attack Efficacy: Object Detection

3.2.3.1 Selecting Semantic Parameters

Due to the larger variety of semantic parameters available to us in the object detection setting, we had to first conduct a survey of single parameter adversarial attacks before picking the best combination to attack the network.

Large values of ε result in unrealistic images created by the generative model (examples of this include perturbing the mesh to the point where cars are twisted into shapes no longer resembling vehicles). To avoid such issues and to simulate *realistic* transformations, we use a different step-size for each semantic parameter. We test various values of ε for each semantic parameter, and report the best choice for brevity. Specifically, (a) *color*: $\varepsilon = 0.05$, (b) *weather*: $\varepsilon = 0.25$, (c) *foliage*: $\varepsilon = 0.10$, (d) *rotate*: $\varepsilon = 0.01$, (e) *translate*: $\varepsilon = 0.01$, and (f) *mesh*: $\varepsilon = 0.025$. We stress these hyperparameters were obtained after extensive visual inspection (by 3 viewers independently); norm-based approaches typically serve as a proxy for visual verification (Sen et al., 2019). Additionally, our choice in hyperparameters enables us to use the same ground truth labels throughout our experiments; *e.g.,* produced SAEs have bounding box coordinates that enable us

to use the same ground truth labels as their benign counterparts[3]. We also measure the Fréchet Inception Distance (FID) scores (Heusel et al., 2017) for the generated SAEs; we observe that the score is 0.102362 (compared to 0.01651 for generating pixel perturbations), providing quantitative evidence that the SAEs are similar to the benign images they were generated from.

We produce 50 SAEs for each semantic parameter combination choice. We then evaluate the efficacy of generated SAEs on SqueezeDet by measuring its (a) recall percentage, and (b) mean average precision, or mAP, in percentage. These metrics have been used in earlier works (Xie et al., 2017).

Parameter	*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 3.2: *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. We observed that single parameter modifications are ineffective.*

From Table 3.2, it is clear that single parameter modification is ineffective at generating SAEs. Thus, we generate SAEs using the multi-parameter modification method. To this end, we generated SAEs using the 57 remaining combinations of semantic parameters. One could consider a weighted combination of different semantic parameters based on a pre-defined notion of precedence. However, we choose a non-weighted combination. The results of our experiments are in Table 3.3. For brevity, we omit most of the combinations that do not result in significant performance degradation (and discuss the insight we gained from them in § 3.2.3.2). In the remainder of the paper, we report our evaluation using the *translate + rotate + mesh* parameter combination to generate SAEs.

Parameters	*translate + rotate*	*translate + rotate + mesh*	*translate + mesh*	*rotate + mesh*
recall	100	100	100	100
mAP	82	65.9	80.8	98.7

Table 3.3: *Performance of SqueezeDet on SAEs generated using multi-parameter modifications. The model had (a) recall = 100, and (b) mAP = 99.4 on benign/non-adversarial inputs. We observed that certain combinations of multiple parameters are effective towards launching an attack.*

[3]This fact is useful when we evaluate model robustness through retraining the models with SAEs as inputs, which we discuss in § 3.2.4.2

3.2.3.2 SAE Generation and Efficacy

We use 6339 images for training our SqueezeDet model, and evaluate the model using 882 SAEs. To evaluate the robustness, we augment the training dataset with 1547 SAEs and retrain the model. The various components of our framework and the datasets used are highlighted in § 3.1.2. Note that SqueezeDet's loss function comprises three terms corresponding to (a) bounding box regresson, (b) confidence score regression, and (c) classification loss. In our experiments, we target the confidence score regression loss term to impact the mAP and recall of the model. All code was written in `python`. Our experiments were performed on two servers. The SAE generation was carried out on a server with an NVIDIA Titan GP102 GPU, 8 CPU cores, and 15GB memory. All training and evaluation was carried out on a server with 264 GB memory, using NVIDIA's GeForce RTX 2080 GPUs and 48 CPU cores. Our experiments suggest that SAEs are indeed effective in degrading the performance of SqueezeDet. We also observe that the model is susceptible to changes that target the geometry of the input (cars in this case) rather than the changes in the background (refer § 3.2.3.2. The results in Table 3.3 in § 3.2.3.1 demonstrate the effectiveness of SAEs, and offer two insights.

First, the victim model was more susceptible to transformations that modify the geometry of the input (such as *translate* and *mesh*) than other types of transformations. This has dire implications for safety-critical applications; for the cars in our inputs, modifications in the *mesh* parameter results in *deformed* cars as outputs. These are common occurrences in sites of accidents, and need to be detected correctly. A combination of translations and rotations also seem to compound the degradation to the performance of the network (refer Table 3.3). This is most likely due to the introduction of unique angles and visual perspectives that are not frequently encountered in assembled datasets. Unlike pixel perturbations, SAEs are easy to interpret, *i.e.* we are able to understand how the model fails to generalize to specific changes in input semantics. Additionally, they are easier to realize *i.e.* the situations described above (related to translation and deformation of vehicles) occur on a daily basis. Intuitively, changing the geometry of the car can be viewed as targeting the perception of what a car really is – if the human can recognize that the object in question is a car but a model cannot, then the model is not exposed to the sufficient variety of car shapes, positions, and orientations that it may encounter in real-world scenarios; *i.e.* it is unable to domain adapt (Tzeng et al., 2017).

The second insight we gain is that the model was more susceptible to SAEs caused by changing multiple parameters simultaneously. We evaluate the model with 882 SAEs generated using a combination of the parameters listed in § 3.2.3.1. We observe that compared to the baseline performance on non-adversarial/benign inputs (recall = 93.63, mAP = 85.95), SAEs cause a significant performance degradation (recall = 93.17, mAP = 57.78). As stated before, these combinations are easily realizable, and the model's poor performance is indicative of poor domain adaptation.

We note, as an aside, that we do not report other metrics (classification accuracy, background error, etc.) associated with detection as our experiments for object detection are not designed to alter them. The impact of an attack designed for classification can be seen in the prior section.

3.2.4 Data Augmentation To Increase Robustness

By this point, we have established the efficacy of our pipeline in generating semantic adversarial examples. The existence of such examples, however, can identify and test concerns and weaknesses regarding the model in question. Unlike pixel perturbations, SAEs are easy to interpret, *i.e.* we are able to understand how the model fails to generalize to specific changes in input semantics. Additionally, they are easier to realize *i.e.* the situations described above (related to translation and deformation of vehicles) occur on a daily basis. Intuitively, changing the geometry of the car can be viewed as targeting the perception of what a car really is – if the human can recognize that the object in question is a car but a model cannot, then the model is not exposed to the sufficient variety of car shapes, positions, and orientations that it may encounter in real-world scenarios; *i.e.* it is unable to domain adapt (Tzeng et al., 2017). This issue can be solved by taking the semantic adversarial examples and augmenting our training dataset with it to improve performance.

The field of semantic data augmentation has been studied in the past; for example, Dreossi et. al developed a scheme for counterexample-guided data augmentation using error tables (Dreossi et al., 2018a;c). However, data augmentation using gradient-based attacks is, to our knowledge, relatively unstudied, and we seek to evaluate the use of our pipeline on large-scale data generation using such techniques.

In theory, our pipeline should provide an easy technique with which to locate generate interesting and unique scenes that may not be encountered in traditional datasets. We seek to evaluate if this is so. In particular, can our adversarial framework be used to generate a large volume of semantic adversarial counterexamples that, when added to the original dataset, can help improve the performance of the classifier through exposing it to corner cases and previously unexplored scenes, making it more robust in the process? Our results for both object classification and detection suggest that this is actually so.

3.2.4.1 Object Classification

After establishing that semantic adversarial counterexamples are effective in generating interesting scenes that `VGG_benign` was ineffective at classifying, our next goal was to use this generated data to improve the network's robustness and make it better at classifying these difficult scenes. To accomplish this, for each attack algorithm, we first set up another training/validation image dataset that consisted of 50% "benign" training images and 50% "adversarial" images that replace their benign counterparts; these were generated from the specified attack algorithm and attacked pose and vertex attributes simultaneously. We then retrained the *original* VGG16 ImageNet network on each "mixed" dataset, giving us 3 "robust" networks that were trained on datasets augmented by one adversarial attack's samples each. Finally, we evaluated the performance of these retrained networks on benign and adversarial test-time datasets created in 3.2.2.

For reference purposes, the names of the retrained network will reference the attack method used to generate augmenting samples for the "mixed" training dataset the model was trained on; e.g. if adversarial images were generated by semantic Carlini-Wagner, we will call the retrained model

`VGG_sCW`.

Our results suggest that data augmentation through semantic adversarial examples does, in fact, strongly improve the model's robustness, and that in each case, semantic robustness against adversarial images generated by one attack also appears to translate to robustness against adversarial images generated via other attacks. For example, we see that, in comparison to `VGG_benign`, `VGG_sCW` achieves near-identical performance on benign images and **substantially** better performance on adversarial images, with performance on all adversarial datasets spiking up to the 95% neighborhood (a performance jump of up to 46 percentage points!). As shown in 3.4, similar results were achieved when sPGD or sFGSM adversarial examples were used as augmenting samples on the training dataset, strongly supporting the augmenting power of semantic adversarial examples and the transferability of robustness across attacks.

Model \ Dataset	benign	sFGSM	sPGD	sCW
`VGG_benign`	98.6	52.9	65.7	48.5
`VGG_sFGSM`	97.9	93.6	95.5	90.7
`VGG_sPGD`	97.8	93.9	94.6	92.8
`VGG_sCW`	98.0	94.2	95.9	95.0

Table 3.4: *Overall accuracies of `VGG_attack` on benign and SAE datasets from 3.2.2. SAE datasets are generated by simultaneously attacking pose and vertex parameters with the specified attack that the dataset is named after. The rows represent models, where each model is named after the attack method used to augment the training dataset the model was trained on, and the columns represent datasets to evaluate on. The benign model's performance is provided for ease of comparison. Full class accuracies are listed in appendix B.*

As the evaluation in 3.4 was on test-time datasets created by attacking the benign networks, we also wished to verify that the robustified networks were less prone to semantic attacks than benign networks by evaluating their performance on test-time datasets created by attacking the robust networks themselves. This would help us verify tightening of the control loop by verifying that the robust networks were now less prone to semantic attacks themselves. To accomplish this, we took a representative subset sample of 232 test images across all classes, maintaining dataset proportions, and now generated SAE datasets by attacking the *adversarially robust* models on these images. The parameters attacked, as before, were the combination of pose and vertex simultaneously. The results for these augmentation experiments are shown in 3.5 and show that semantic attacks are *far* less effective on the robust models than on the original models. For example, the sFGSM attack dropped accuracy by 46 percentage points when attacking the benign model, but only by 11 percentage points when attacking the sCW-robust model. These findings further support the

hypothesis that SAEs are good at improving the robustness of neural networks against such attacks and counterexamples.

Model \ Attack Method	benign	sFGSM	sPGD	sCW
VGG_benign	99.1	54.7	68.5	48.3
VGG_sFGSM	97.9	79.4	87.1	81.9
VGG_sPGD	97.8	83.6	91.0	79.7
VGG_sCW	99.1	88.4	91.8	87.9

Table 3.5: *Overall accuracies of* `VGG_attack` *on benign and SAE datasets of 232 test images generated by attacking the specific model that accuracy is measured for. The rows represent models, where each model is named after the attack method used to augment the training dataset the model was trained on, and the columns represent the attack method used to generate the SAE dataset to evaluate on. Each (row, column) entry represents the accuracy on a specific dataset – for example, the entry (*`VGG_sFGSM`*, sCW) represents accuracy on the SAE dataset generated by using the semantic Carlini-Wagner attack on* `VGG_sFGSM`*. The benign model's performance on these sampled datasets of 232 images is provided for ease of comparison. Full class accuracies are listed in appendix C.*

The strong improvement in overall robustness comes at virtually no cost to performance on benign images, and these results thus indicate the high potential efficacy of our pipeline in data augmentation – through providing an easy-to-use framework that can leverage standard gradient-based techniques to generate scores of adversarial examples and corner cases the network performs poorly on, one can improve the robustness and generalizability of neural networks to safety-critical scenarios and scenes that are often seldom-encountered in training.

Moreover, our work suggests an interesting new aspect of adversarial robustness in the semantic space; namely, that semantic adversarial robustness is general enough to prevent against a variety of attacks, regardless of the semantic attack algorithm the model was trained to defend against. Specifically, making images robust to semantic adversarial examples of a certain adversarial attack type appears to make it robust to a broader set of semantic adversarial examples generated by a wider variety of attack algorithms. For example, robustness on the dataset of semantic Carlini-Wagner attack images, generated through retraining on the aforementioned "mixed" dataset, also enables us to achieve robustness on the sFGSM and sPGD attack datasets with no additional effort. Our findings, in total, appear to thus imply the transferability of semantic adversarial robustness across attacks. This robustness appears to extend to the overall attacks themselves, with the efficacy of each semantic attack dropping significantly on all robust models – the sCW attack, for example, dropped accuracy by 50 percentage points when attacking the benign model, but only by 17 percentage points when attacking the sFGSM-robust model.

Something similar was discussed in Tramèr & Boneh (2019), in which it was shown that robustness against one type of attack could *help* with robustness against another attack with a different perturbation metric (ℓ_2 vs ℓ_∞, for example); however, the multi-perturbation robustness achieved still was not "competitive" when comparing model performance on adversarial images to performance on benign images. This is in contrast to the results shown here, where the multi-perturbation robustness achieved is far more complete, indicating a sort of "transferability" – that is, robustness to one semantic attack greatly helps achieve robustness against others, even if the algorithms are attacking different metrics themselves (e.g. our sFGSM and sPGD target ℓ_∞, whereas our sCW targets ℓ_2).

We hypothesize that the transferability of semantic attack robustness is tied to the physical meaning and nature of semantic modifications; unlike classic pixel perturbation attack algorithms, all pixels are not created equal in the semantic modification space with respect to the objects in the scene. Each modification to the image is done in terms of an object instead of in terms of a pixel, and this means that semantic modifications are not only more physically meaningful, but that different attack algorithms will still generate images with a similar distribution of pixel values across the object and background, respectively. This means that each semantic counterexample would represent a specific weak point in terms of *objects* the network fails to classify, and it is not unreasonable to expect that given enough counterexamples, similar attack algorithms will identify similar weak points in terms of the network's performance on objects alone. This is a highly promising direction for us to explore in future work.

3.2.4.2 Object Detection

As we have established that SAEs are effective in attacking SqueezeDet, we wished to enhance the model's robustness through data augmentation, as in Dreossi et al. (2018c) and Dreossi et al. (2018a). To this end, we carried out two sets of experiments. In the first, we incrementally (re)trained the benign SqueezeDet model on a combination of benign inputs and SAEs (4792+1547) for 24000 iterations. In the second, we tuned our benign model using just SAEs (1547) for 6000 iterations. Our results suggest that the generated SAEs do, in fact, help in improving model robustness. Our experiments show that SAE-based data augmentation can improve mAP by up to 15 percentage points; we present these results in 3.6.

Model	Baseline	Retrained (SAE + Benign)	Tuned (SAE)
recall	93.17	92.97	92.15
mAP	57.78	72.76	72.63

Table 3.6: *Performance of SqueezeDet on SAEs when (b) the model is retrained (on a combination of SAEs + benign inputs), and (c) the model is tuned (on just SAEs), compared to (a) the baseline model (trained on benign images) on SAEs. Both retraining and tuning improve mAP.*

It is clear that both approaches provide comparable increase in mAP while not impacting recall. Additionally, we found that making a model robust to semantic perturbations through either procedure described earlier allowed us to achieve good performance on benign inputs. On benign inputs, we found that for the Retrained (SAE + Benign) model, recall = 93.7 and mAP = 84.73, while for Tuned (SAE), recall = 91.9 and mAP = 79.1. This is comparable to the performance of the baseline model (which was trained and validated on benign inputs), where recall = 93.6 and mAP = 86.17.

3.3 Discussion and Findings

Quite clearly, semantic adversarial examples appear to be easily realizable with the right attacks and parameters and do cause significant performance degradation – for example, we see performance drops of up to 50% for the VGG-16 framework when utilizing adversarial attacks. Interestingly, the parameters that the model appears to be most susceptible to involve anything to do with the geometry of the object instead of any background changes, like background scenery or object lighting. This is a finding that was consistent across both object detection and classification tasks; it appears that the networks are robust enough to be relatively impervious to background changes, but that this robustness is not initially present on images resulting from geometric modifications.

However, this robustness can also be achieved through adversarial retraining. Our results across classification and detection suggest that SAEs are extremely useful for augmentation and help greatly with improving the robustness of the network, with sharp performance gains in both classification and detection tasks. Interestingly, we see in the case of classification that the robustness of the network to one adversarial attack appears to translate to robustness against other gradient-based adversarial attacks as well, as shown by the tables in 3.4 and 3.5.

We see, therefore, that our pipeline serves as a useful data-generation and data-augmentation tool that can find the ”weak points” in the network or dataset and generate images to increase exposure and robustness to such cases. It appears from our results as if SAE-based augmentation is a very promising direction for further exploration; based on insight from § 2.2 and with the use of our newly developed pipeline, we could formulate a more precise framework for semantic adversarial training, similar to (Madry et al., 2017). We leave the exact formulation of these questions to future work.

Chapter 4

Conclusions

In this work, we have successfully created a fully-differentiable pipeline that can use standard gradient-based adversarial attacks on a variety of semantic features to generate interesting scenes that fool a model. This allows our pipeline to act as a fast and efficient data augmentation framework that could be used to enhance the performance of the model by identifying weak points in its training and performance and generating images to address the detected issues.

Concretely, from our results, we first confirmed that semantic space modifications do succeed in fooling a neural network and making it less confident in its classifications/detections, especially when multiple features are perturbed at once (and especially when the geometry of the objects in question is attacked). These semantic counterexamples, however, merely introduce new angles and alter meshes of objects, and the network's poor performance on it would lead to dire safety-critical implications; for example, the rotation of a car to mirror the angle it drives at on an incline should not, under any circumstances, cause a network to misdetect the object or fail to classify it. From this viewpoint, these counterexamples can be viewed as extra data that can be used to improve a network's performance, and our experiments further confirm that retraining a neural network on semantic counterexamples helps improve its robustness by providing vastly improved performance on counterexamples while providing comparable performance on the original images.

The improvement in robustness and performance, however, is not limited to semantic counterexamples generated by a single attack algorithm, as our results indicate that robustness against one attack in the semantic space can actually help provide robustness against other gradient-based semantic attacks. This is an extremely promising direction for future work and one that we hope to explore in much greater detail.

Bibliography

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul F. Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety. *CoRR*, abs/1606.06565, 2016. URL http://arxiv.org/abs/1606.06565.

Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. *CoRR*, abs/1707.07397, 2017. URL http://arxiv.org/abs/1707.07397.

J. T. Barron and J. Malik. 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.

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

Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, pp. 387–402. Springer, 2013.

Volker Blanz and Thomas Vetter. 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.

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

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. (arXiv:1512.03012 [cs.GR]), 2015.

Jeremy M. Cohen, Elan Rosenfeld, and J. Zico Kolter. Certified adversarial robustness via randomized smoothing. *CoRR*, abs/1902.02918, 2019. URL http://arxiv.org/abs/1902.02918.

Tommaso Dreossi, Alexandre Donze, and Sanjit A. Seshia. Compositional falsification of cyber-physical systems with machine learning components. In *Proceedings of the NASA Formal Methods Conference (NFM)*, May 2017.

Tommaso Dreossi, Shromona Ghosh, Xiangyu Yue, Kurt Keutzer, Alberto Sangiovanni-Vincentelli, and Sanjit A. Seshia. Counterexample-guided data augmentation. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pp. 2071–2078. International Joint Conferences on Artificial Intelligence Organization, 7 2018a. doi: 10.24963/ijcai.2018/286. URL https://doi.org/10.24963/ijcai.2018/286.

Tommaso Dreossi, Shromona Ghosh, Xiangyu Yue, Kurt Keutzer, Alberto L. Sangiovanni-Vincentelli, and Sanjit A. Seshia. Counterexample-guided data augmentation. *CoRR*, abs/1805.06962, 2018b. URL http://arxiv.org/abs/1805.06962.

Tommaso Dreossi, Somesh Jha, and Sanjit A. Seshia. Semantic adversarial deep learning. *CoRR*, abs/1804.07045, 2018c. URL http://arxiv.org/abs/1804.07045.

Tommaso Dreossi, Shromona Ghosh, Alberto L. Sangiovanni-Vincentelli, and Sanjit A. Seshia. A formalization of robustness for deep neural networks. In *Proceedings of the AAAI Spring Symposium Workshop on Verification of Neural Networks (VNN)*, March 2019.

Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. Exploring the landscape of spatial robustness. In *International Conference on Machine Learning*, pp. 1802–1811, 2019.

Ivan Evtimov, Kevin Eykholt, Earlence Fernandes, Tadayoshi Kohno, Bo Li, Atul Prakash, Amir Rahmati, and Dawn Song. Robust physical-world attacks on machine learning models. *CoRR*, abs/1707.08945, 2017. URL http://arxiv.org/abs/1707.08945.

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

Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The KITTI dataset. *International Journal of Robotics Research (IJRR)*, 2013.

Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. 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.

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015. URL http://arxiv.org/abs/1412.6572.

Chuan Guo, Mayank Rana, Moustapha Cissé, and Laurens van der Maaten. Countering adversarial images using input transformations. *CoRR*, abs/1711.00117, 2017. URL `http://arxiv.org/abs/1711.00117`.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Günter Klambauer, and Sepp Hochreiter. 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`.

Hossein Hosseini and Radha Poovendran. Semantic adversarial examples. *CoRR*, abs/1804.00499, 2018. URL `http://arxiv.org/abs/1804.00499`.

Lifeng Huang, Chengying Gao, Yuyin Zhou, Changqing Zou, Cihang Xie, Alan Yuille, and Ning Liu. Upc: Learning universal physical camouflage attacks on object detectors, 2019.

Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *2018 IEEE Symposium on Security and Privacy (SP)*, pp. 19–35, 05 2018. doi: 10.1109/SP.2018.00057.

Lakshya Jain, Wilson Wu, Steven Chen, Uyeong Jang, Varun Chandrasekaran, Sanjit Seshia, and Somesh Jha. Generating semantic adversarial examples with differentiable rendering. *arXiv preprint arXiv:1910.00727*, 2019.

Ameya Joshi, Amitangshu Mukherjee, Soumik Sarkar, and Chinmay Hegde. Semantic adversarial attacks: Parametric transformations that fool deep classifiers. *CoRR*, abs/1904.08489, 2019. URL `http://arxiv.org/abs/1904.08489`.

Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. pp. 3907–3916, 06 2018. doi: 10.1109/CVPR.2018.00411.

Tejas D Kulkarni, William F Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. In *Advances in neural information processing systems*, pp. 2539–2547, 2015.

Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *CoRR*, abs/1607.02533, 2016. URL `http://arxiv.org/abs/1607.02533`.

Alexey Kurakin, Ian Goodfellow, Samy Bengio, Yinpeng Dong, Fangzhou Liao, Ming Liang, Tianyu Pang, Jun Zhu, Xiaolin Hu, Cihang Xie, et al. Adversarial attacks and defences competition. In *The NIPS'17 Competition: Building Intelligent Systems*, pp. 195–231. Springer, 2018.

Mathias Lécuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 656–672, 2018.

Qi Lei, Lingfei Wu, Pin-Yu Chen, Alexandros G. Dimakis, Inderjit S. Dhillon, and Michael Witbrock. Discrete attacks and submodular optimization with applications to text classification. *CoRR*, abs/1812.00151, 2018. URL `http://arxiv.org/abs/1812.00151`.

Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. Differentiable monte carlo ray tracing through edge sampling. In *SIGGRAPH Asia 2018 Technical Papers*, pp. 222. ACM, 2018.

Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. *CoRR*, abs/1703.00848, 2017. URL `http://arxiv.org/abs/1703.00848`.

Matthew Loper and Michael Black. Opendr: An approximate differentiable renderer. 09 2014. doi: 10.1007/978-3-319-10584-0_11.

Manolis Loukadakis, José Cano, and Michael O'Boyle. Accelerating deep neural networks on low power heterogeneous architectures. 01 2018.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *ArXiv*, abs/1706.06083, 2017.

Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. *CoRR*, abs/1712.04621, 2017. URL `http://arxiv.org/abs/1712.04621`.

Haonan Qiu, Chaowei Xiao, Lei Yang, Xinchen Yan, Honglak Lee, and Bo Li. Semanticadv: Generating adversarial examples via attribute-conditional image editing. *CoRR*, abs/1906.07927, 2019. URL `http://arxiv.org/abs/1906.07927`.

Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. *CoRR*, abs/1801.09344, 2018. URL `http://arxiv.org/abs/1801.09344`.

Ayon Sen, Xiaojin Zhu, Liam Marshall, and Robert Nowak. Should adversarial attacks use pixel p-norm? *arXiv preprint arXiv:1906.02439*, 2019.

Sanjit A. Seshia, Dorsa Sadigh, and S. Shankar Sastry. Towards Verified Artificial Intelligence. *ArXiv e-prints*, July 2016.

Sanjit A. Seshia, Ankush Desai, Tommaso Dreossi, Daniel Fremont, Shromona Ghosh, Edward Kim, Sumukh Shivakumar, Marcell Vazquez-Chanlatte, and Xiangyu Yue. Formal specification for deep neural networks. In *Proceedings of the International Symposium on Automated Technology for Verification and Analysis (ATVA)*, pp. 20–34, October 2018.

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

Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *Advances in Neural Information Processing Systems*, pp. 6103–6113, 2018.

Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K Reiter. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1528–1540. ACM, 2016.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. URL `http://arxiv.org/abs/1409.1556`.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. 2013.

Yang Song, Rui Shu, Nate Kushman, and Stefano Ermon. Generative adversarial examples. *CoRR*, abs/1805.07894, 2018. URL `http://arxiv.org/abs/1805.07894`.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014. URL `http://arxiv.org/abs/1312.6199`.

Florian Tramèr and Dan Boneh. Adversarial training and robustness for multiple perturbations. *CoRR*, abs/1904.13000, 2019. URL `http://arxiv.org/abs/1904.13000`.

Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7167–7176, 2017.

Bichen Wu, Forrest N. Iandola, Peter H. Jin, and Kurt Keutzer. 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`.

Chaowei Xiao, Jun-Yan Zhu, Bo Li, Warren He, Mingyan Liu, and Dawn Song. Spatially transformed adversarial examples. *CoRR*, abs/1801.02612, 2018. URL `http://arxiv.org/abs/1801.02612`.

Chaowei Xiao, Dawei Yang, Bo Li, Jia Deng, and Mingyan Liu. Meshadv: Adversarial meshes for visual recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Cihang Xie, Jianyu Wang, Zhishuai Zhang, Yuyin Zhou, Lingxi Xie, and Alan Yuille. Adversarial examples for semantic segmentation and object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1369–1378, 2017.

Shunyu Yao, Tzu-Ming Harry Hsu, Jun-Yan Zhu, Jiajun Wu, Antonio Torralba, William T. Freeman, and Joshua B. Tenenbaum. 3d-aware scene manipulation via inverse graphics. In *Advances in neural information processing systems*, 2018.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2242–2251, 10 2017. doi: 10.1109/ICCV.2017.244.

Appendix A

Classification Accuracies For Adversarial Attacks

Below, we provide a class-by-class accuracy breakdown of 3.1

Evaluation Dataset \ Class	airplane	bench	trashcan	bus	car	helmet	mailbox	motorcycle	skateboard	tower	train	boat	**overall**
benign	99.9	99.6	99.2	96.9	99.7	98.3	100	98.6	93.9	84.0	84.9	95.6	**98.6**
sCW - pose + vertex	88.3	40.7	31.9	0	35.3	78.3	13.3	67.1	4.1	12.0	20.0	63.5	**48.5**
sFGSM - pose + vertex	83.6	64.2	46.0	0	40.2	78.3	40.0	50.7	0	24.0	27.3	61.9	**52.9**
sPGD - pose + vertex	91.7	70.7	58.9	0	58.9	88.3	46.7	76.7	4.1	16.0	23.2	78.1	**65.9**
sCW - pose	92.3	50.0	34.5	0	40.7	90.0	20.0	84.9	16.3	12.0	32.3	67.3	**54.3**
sFGSM - pose	93.8	68.8	57.1	30.7	46.7	88.3	40.0	82.2	14.3	8.0	22.2	72.7	**61.5**
sPGD - pose	97.0	74.1	65.6	44.1	69.2	88.3	53.3	93.2	22.5	16.0	34.3	75.9	**74.1**
sCW - vertex	97.9	97.4	88.2	22.8	92.8	93.3	100	84.9	34.6	64.0	58.6	87.0	**89.2**
sFGSM - vertex	95.4	96.3	78.2	10.2	92.8	73.3	80.0	63.0	6.1	68.0	67.7	78.4	**86.2**
sPGD - vertex	97.8	97.8	87.4	22.8	93.8	96.7	80.0	80.8	28.6	64.0	59.6	87.6	**89.7**

Table A.1: *Performance of* `VGG_benign` *on SAEs generated through various attack methods. Accuracies are reported as percentages of correctly classified images, and the datasets are labeled according to the attack method used to generate the images and the parameters they attack. We see that multi-parameter modifications are much more effective than single-parameter modifications*

Appendix B

Classification Accuracies For Augmentation

Below, we report more detailed breakdown (i.e. full class accuracies) of the results from the table 3.4. Accuracies are reported as percentages of correctly classified images. The datasets are labeled according to the attack method used to generate the images. For these experiments, the pose and vertex semantic attributes were attacked simultaneously for each dataset of SAEs.

Evaluation Dataset \ Class	airplane	bench	trashcan	bus	car	helmet	mailbox	motorcycle	skateboard	tower	train	boat	**overall**
benign	99.6	99.5	100	88.2	99.6	96.7	100	98.6	89.8	80.0	87.9	93.3	**98.0**
sCW	99.2	97.4	84.9	76.4	99.0	76.7	933.	93.2	65.3	72.0	73.7	90.2	**95.0**
sFGSM	98.8	97.2	89.2	76.4	99.4	81.7	100	93.2	55.1	64.0	76.7	76.8	**94.2**
sPGD	99.6	99.1	90.8	81.1	99.4	90.0	86.7	97.3	63.3	72.0	66.7	89.8	**95.9**

Table B.1: *Performance of* `VGG_sCW` *on benign and adversarial datasets.*

Evaluation Dataset \ Class	airplane	bench	trashcan	bus	car	helmet	mailbox	motorcycle	skateboard	tower	train	boat	**overall**
benign	99.7	99.6	99.2	89.8	99.6	95.0	93.3	100	87.8	96.0	78.8	93.0	**97.9**
sCW	98.7	89.4	48.7	67.7	95.5	86.7	86.7	93.1	77.6	52.0	59.6	90.7	**90.7**
sFGSM	98.1	94.2	72.3	72.4	97.7	85.0	100	89.0	69.4	68.0	55.6	87.9	**93.6**
sPGD	98.7	98.5	97.5	79.5	98.8	91.7	100	95.9	75.5	68.0	45.4	94.6	**95.5**

Table B.2: *Performance of* `VGG_sFGSM` *on benign and adversarial datasets.*

Evaluation Dataset \ Class	airplane	bench	trashcan	bus	car	helmet	mailbox	motorcycle	skateboard	tower	train	boat	**overall**
benign	99.6	99.6	99.2	90.6	99.5	96.7	100	100	87.8	80.0	68.7	96.8	**97.8**
sCW	98.1	97.4	89.9	72.4	95.8	83.3	93.3	86.3	57.1	60.0	49.5	92.7	**92.8**
sFGSM	98.7	97.8	95.0	83.5	98.3	86.7	100	95.9	87.8	60.0	57.6	84.8	**93.9**
sPGD	99.3	96.6	74.0	77.1	98.0	95.0	80.0	91.8	87.8	64.0	69.7	87.9	**94.6**

Table B.3: *Performance of* `VGG_sPGD` *on benign and adversarial datasets.*

Appendix C

Classification Accuracies For Robust Model Attacks

Below, we report more detailed breakdown (i.e. full class accuracies) of the results from the table 3.5. Accuracies are reported as percentages of correctly classified images. The datasets are labeled according to the attack method used to generate the images. For these experiments, the pose and vertex semantic attributes were attacked simultaneously for each dataset of SAEs. We provide the performance of `VGG_benign` on these smaller evaluation datasets of 232 images for comparison.

Evaluation Dataset \ Class	airplane	bench	trashcan	bus	car	helmet	mailbox	motorcycle	skateboard	tower	train	boat	**overall**
benign	100	100	100	87.5	100	100	100	100	100	100	87.5	100	**99.1**
sCW	79.2	42.9	25.0	0	39.1	75.0	25.0	50.0	25.0	0	25.0	75.0	**48.3**
sFGSM	87.5	67.9	50.0	0	41.3	100	50.0	75.0	0	0	12.5	70.0	**54.7**
sPGD	93.8	82.14	37.5	0	63.0	100	25.0	100	25.0	25.0	25.0	85.0	**68.5**

Table C.1: *Performance of `VGG_benign` on the benign and adversarial datasets.*

Evaluation Dataset \ Class	airplane	bench	trashcan	bus	car	helmet	mailbox	motorcycle	skateboard	tower	train	boat	**overall**
benign	100	100	100	87.5	100	100	100	100	100	100	100	95.0	**99.1**
sCW	97.9	96.4	75.0	37.5	96.7	100	50.0	100	0	50.0	62.5	75.0	**87.9**
sFGSM	97.9	92.9	100	37.5	94.6	100	100	100	25.0	50.0	37.5	80.0	**88.4**
sPGD	97.9	100	75.0	62.5	97.8	100	100	100	75.0	25.0	37.5	90.0	**91.8**

Table C.2: *Performance of `VGG_sCW` on the benign and adversarial datasets where the robust model is attacked.*

Evaluation Dataset \ Class	airplane	bench	trashcan	bus	car	helmet	mailbox	motorcycle	skateboard	tower	train	boat	**overall**
benign	100	100	100	87.5	100	100	100	100	100	100	62.5	95.0	**97.8**
sCW	97.9	85.7	25.0	25.0	91.3	1	50	75.0	75.0	0	12.5	90.0	**81.9**
sFGSM	97.9	89.3	12.5	12.5	87.0	50.0	75.0	100	100	0	0	90.0	**79.7**
sPGD	95.8	96.4	62.5	12.5	94.6	100	100	100	50.0	50.0	25.0	90.0	**87.1**

Table C.3: *Performance of `VGG_sFGSM` on benign and adversarial datasets where the robust model is attacked.*

Evaluation Dataset \ Class	airplane	bench	trashcan	bus	car	helmet	mailbox	motorcycle	skateboard	tower	train	boat	**overall**
benign	100	100	100	87.5	100	100	100	100	100	75.0	62.5	100	**97.8**
sCW	97.9	78.6	87.5	0	87.0	75.0	25.0	100	25.0	0	25.0	90.0	**79.7**
sFGSM	97.9	92.9	100	0	88.0	100	75.0	100	50.0	25.0	25.0	80.0	**83.6**
sPGD	97.9	100	100	37.5	96.7	100	100	100	50.0	50.0	25.0	90.0	**91.0**

Table C.4: *Performance of `VGG_sPGD` on benign and adversarial datasets where the robust model is attacked.*