Generating Semantic Adversarial Examples with Differentiable Rendering

Anonymous authors
Paper under double-blind review

Abstract

Machine learning (ML) algorithms, especially deep neural networks, have demonstrated success in several domains. However, several types of attacks have raised concerns about deploying ML in safety-critical domains, such as autonomous driving and security. An attacker perturbs a data point slightly in the pixel space and causes the ML algorithm to misclassify (*e.g.* a perturbed stop sign is classified as a yield sign). These perturbed data points are called adversarial examples, and there are numerous algorithms in the literature for constructing adversarial examples and defending against them. In this paper we explore *semantic adversarial examples (SAEs)* where an attacker creates perturbations in the semantic space. For example, an attacker can change the background of the image to be cloudier to cause misclassification. We present an algorithm for constructing SAEs that uses recent advances in differential rendering and inverse graphics.

1 Introduction

Machine learning (ML) techniques, especially Deep Neural Networks (DNNs), have been successful in several domains, such as finance and healthcare. 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 safety-critical applications, such as autonomous driving and security, 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 paper we consider the problem of generating *semantic adversarial examples (SAEs)* (Hosseini & Poovendran, 2018; Joshi et al., 2019; Qiu et al., 2019). In these examples, there is a richer set of transformations $\mathcal{T}$. 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 space (*e.g.*, texture of the image) rather than just the concrete (pixel) space. Consequently, SAEs are 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., 2018b; Huang et al., 2019), 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 is to support a richer set of transformations (*e.g.* making the background of the image cloudy or changing the time of day from morning to evening). There is evidence that SAEs can help with domain adaptation (Volpi et al., 2018) or making the control loop more robust (Dreossi et al., 2018b), further motivating our approach.

To summarize, the main contributions of this paper are the following:

- We present a new class of test-time attacks in the form of SAEs. We demonstrate how to generate SAEs that support a rich set of transformations (refer § 3) using an inverse graphics framework (refer § 2). Specifically, we show how one can systematically take techniques to perform attacks in the pixel space such as FGSM (Goodfellow et al., 2015) and PGD (Madry et al., 2017) and transform them to their semantic counterparts.
- We evaluate the generated SAEs on the popular object detector SqueezeDet (Wu et al., 2016). By correctly choosing the semantic parameters, SAEs degrade performance (characterized by the mean average precision or mAP) by 28 percentage points (refer § 5.1).
- We also show that by augmenting the dataset using SAEs, we can boost the robustness of SqueezeDet (characterized by mAP) by up to 15 percentage points (refer § 5.2). While augmentation with SAEs improves robustness against SAEs, augmentation using traditional pixel-based perturbations does not produce the same effect (refer § 5.3).

2 Related Work

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. (2018a) 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: 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). 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. 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.

3 Semantic 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).

3.1 TRADITIONAL ADVERSARIAL EXAMPLES

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 $F : \Re^n \to Y$. The adversary wishes to solve the following optimization problem:

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

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 ($p \geq 2$). If δ is the solution of the optimization problem given above, then the adversarial example $A(\mathbf{x}) \; = \; \mathbf{x}+\delta$.

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:

$$\delta = \varepsilon \, \text{sign}(\nabla_{\mathbf{x}} L_F(\mathbf{x})) \tag{1}$$

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 $\text{sign}(\nabla L_F(\mathbf{x}))$ 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.

PGD. In *Projected Gradient Descent (PGD)* (Madry et al., 2017), we find a perturbation in an iterative manner. The PGD attack can be thought of an iterative version of FGSM. Assume that we are using the l_p norm. Assume $\mathbf{x}_0$ is the original sample $\mathbf{x}$.

$$\mathbf{x}_{k+1} = \Pi_{B_p(\mathbf{x},\varepsilon)}(\mathbf{x}_k + \varepsilon \, \text{sign}(\nabla_{\mathbf{x}} L_F(\mathbf{x}))) \tag{2}$$

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).

3.2 SEMANTIC ADVERSARIAL EXAMPLES (SAES)

Let $\mathcal{T} : (\Re^n \times \Theta) \to \Re^n$ 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:

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

In other words, we want to find a "small perturbation" in the parameter space Θ that will misclassify the sample. 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:

$$\theta^\star = \varepsilon\, \text{sign}\left(\left[\tfrac{\partial\tau}{\partial\theta}\right]^\top \Big|_{(\mathbf{x},\perp)} \nabla_{\mathbf{z}} L_F(\mathbf{z})|_{\mathbf{z}=\tau(\mathbf{x},\perp)} \right) \tag{3}$$

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 (*sPGD*) can be constructed. Let $\theta_0 =\perp$ and $\mathbf{x}_0 = \mathbf{x}$. The update steps correspond to the following two equations:

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

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). 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 $\tau(\mathbf{x}, \theta)$.

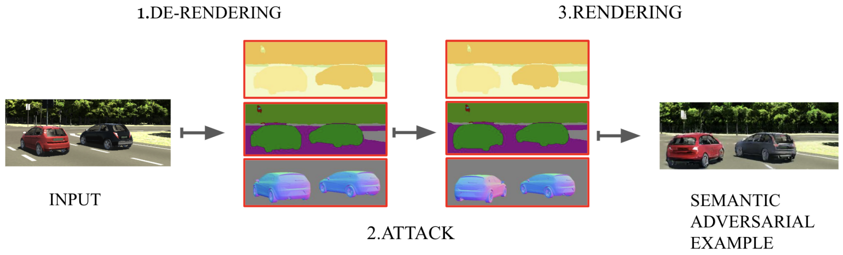

Figure 1: The input is de-rendered (step 1) to its intermediary representation (IR) - semantic, graphic, and textural maps. Then, this is adversarially perturbed (*e.g.* the red car is rotated) as described in § 3.2 (step 2). The resulting IR is then re-rendered to the generate the SAE (step 3).

Differentiable rendering and inverse graphics. We apply the above framework to images by employing a differentiable renderer/de-renderer in an inverse graphics setting. Such an inverse graphics setting can be thought of two transformations: (a) a de-renderer $\beta : \Re^n \to \mathcal{S}$, and (b) a renderer $\gamma : \mathcal{S} \to \Re^n$. Here, $\mathcal{S}$ is the intermediate representation (IR). In the differentiable renderer/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 $\alpha : (\mathcal{S} \times \Theta) \to \mathcal{S}$ 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 $\gamma(\alpha(\beta(\mathbf{x}), \theta))$. We use the fact that for differentiable renderers/de-renderers the functions β, γ, α are differentiable and hence attacks like sFGSM and sPGD can be implemented.

4 Validation

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.

4.1 Implementation Details

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 each of these below.

To obtain the semantics associated with our inputs and to generate the final SAEs, we use the inverse graphics (*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 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. 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.

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.* distortions 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). We stress that our choice of the number of iterations is restricted by our choice of the differentiable graphics framework. Using more iterations resulted in unintelligble outputs. We also stress that the exact choice of three components are irrelevant; our constructions are general (refer § 3.2 for more details).

4.2 Selecting Semantic Parameters

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* hyperparameter $\varepsilon \in [0, 1]$. In the context of pixel perturbations, the step-size corresponds to the maximum permissible change of a pixel. 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.

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*: ε = 0.05, (b) *weather*: ε = 0.25, (c) *foliage*: ε =

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

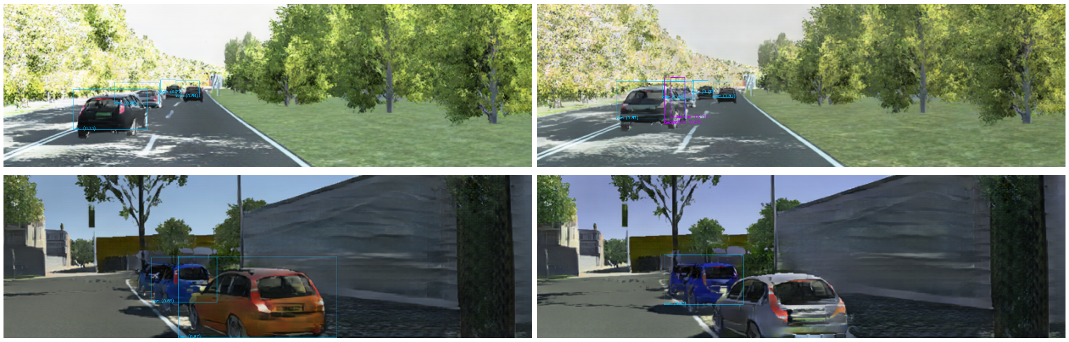

Figure 2: **Semantic space adversarial examples.** Benign re-rendered VKITTI image (left), adversarial examples generated by iterative sFGSM over a combination of semantic features (right). Cyan boxes indicate car detected, purple indicated pedestrian, and yellow indicate cyclist. The adversarial example 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).

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[2].

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 1: 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.

From Table 1, 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 2. 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 § 5.1). 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 2: 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.

[2] This fact is useful when we evaluate model robustness through retraining the models with SAEs as inputs, which we discuss in § 5.2

5 Evaluation

We designed and carried out experiments to answer the following questions: (1) Do SAEs cause performance degradation in SqueezeDet?, (2) Can the generated SAEs be used for improving robustness?, and (3) How does the degradation (and robustness) caused by SAEs compare to that caused by pixel perturbations?

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 § 4.1. 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: (1) 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 § 5.1), (2) 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 (refer § 5.2), and (3) Pixel perturbation-based augmentation is ineffective against SAEs (refer § 5.3).

We do not report other metrics (classification accuracy, background error, etc.) associated with detection as our experiments are not designed to alter them.

5.1 Effectiveness of SAEs

The results in Table 2 in § 4.2 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 2). 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 § 4.2. 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.

5.2 Data Augmentation To Increase Robustness

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. (2018b). 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. The results of our experiments are presented in Table 3.

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

Table 3: 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.

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.

Our results suggests that SAE-based augmentation is a promising direction for exploration; based on insight from § 3, we could formulate a framework for semantic adversarial training, similar to (Madry et al., 2017). We leave the exact formulation to future work. In the next subsection, we will compare this approach to augmentation using pixel perturbations, and a combination of both approaches.

5.3 Pixel Perturbations vs. SAEs

Our experiments suggested that pixel perturbations are more effective in degrading SqueezeDet's performance (recall=2.9, mAP=0.05); we conjecture this is due to the larger feature space within which a solution for the optimization can be found *i.e.* the space of pixels is larger than the space of specific semantic parameters/transformations we consider.

To measure the robustness provided by pixel perturbations against semantic perturbations, we performed the same experiment as in § 5.2. In one experiment, we retrained the benign SqueezeDet model on a combination of benign inputs and pixel perturbations (4792+1547) for 24000 iterations. In another experiment, we tuned our model using just pixel perturbations (1547) for 6000 iterations. The results of our experiments are presented in Table 4.

Model	Retrained (Pixel + Benign)	Tuned (Pixel)
recall	92.4	90.43
mAP	56.7	55.35

Table 4: Performance of SqueezeDet on SAEs when the model is (a) retrained (on pixel perturbations + benign inputs), and (b) tuned (on just pixel perturbations).

We observed that data augmentation using pixel perturbations *does not* increase the robustness to SAEs. Pixel perturbations are more general, and do not capture the effects induced by SAEs. Consequently, we wished to understand if we could get the best of both worlds *i.e.* robustness against both pixel perturbations and SAEs. We report the results of this experiment in Appendix A.

6 Conclusions

In this paper, we describe semantic adversarial examples (SAEs), where adversaries perturb the semantics of inputs to produce outputs that are misclassified. Such instances are easier to realize in the physical world, and are more interpretable than their traditional pixel-based counterparts. We propose an algorithm to construct SAEs using advances in differentiable rendering, and evaluate the effectiveness of our approach. We observe that SAEs cause performance degradation in object detector networks (SqueezeDet), that data augmentation using SAEs increases robustness of the model, and that data augmentation using traditional adversarial examples (*i.e.* pixel perturbations) are ineffective against SAEs.

REFERENCES

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.

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

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

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.

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`.

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.

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`.

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.

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.

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

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.

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.

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`.

Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. *arXiv preprint arXiv:1805.12152*, 2018.

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.

Riccardo Volpi, Hongseok Namkoong, Ozan Sener, John C. Duchi, Vittorio Murino, and Silvio Savarese. Generalizing to unseen domains via adversarial data augmentation. *CoRR*, abs/1805.12018, 2018. URL `http://arxiv.org/abs/1805.12018`.

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`.

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.

A Appendix

A.1 Hybrid Tuning

To understand if we are able to obtain robustness against both pixel perturbations and SAEs, we repeated our experiments from earlier sections (§ 5.2); this time, we tuned the *model robust to SAEs* (obtained after 6000 iterations) using pixel perturbations for 6000 iterations - denoted Tuned (Pixel); we also tuned the *model robust to pixel perturbations* (obtained after 6000 iterations) using SAEs for 6000 iterations - denoted Tuned (SAE).

Model	Tuned (SAE)	Tuned (Pixel)
recall	93.2	91.7
mAP	73.8	65.7

(a) Validation set used: SAEs

Model	Tuned (SAE)	Tuned (Pixel)
recall	29.5	90.6
mAP	7.65	75.15

(b) Validation set used: pixel perturbations

Table 5: Performance of dual tuning (*i.e.* tuning a model robust to SAEs with pixel perturbations, and tuning a model robust to pixel perturbations with SAEs)

We validated both the generated models against (a) SAEs, and (b) pixel perturbations. Our results are presented in Table 5. While the order of tuning did not greatly impact robustness in the face of SAEs, it had a direct impact for pixel perturbations (as evident from Table 5b). We explain the observation in Table 5b using manifold learning. The detector is assumed to generalize over unobserved data on the data manifold. Due to the limited amount of sampling, the detector fails to generalize to SAEs lying on these uncovered regions; learning them aids generalization. Pixel perturbations, however, do not lie on the data manifold (due to its larger degree of freedom). Tuning the model using pixel perturbation is analogous to learning a thickened manifold (with the same uncovered regions as before). Now, if we tune this model, the detector learns uncovered regions in a thickened (and incorrect) manifold; this impacts generalization, further making it susceptible to pixel perturbations. Another intuition could be based on the generality of pixel perturbations; tuning first on such general perturbations impacts generalization (Tsipras et al., 2018). When such a model is tuned on SAEs, we gain generalization at the expense of robustness.