SafetyNet: Detecting and Rejecting Adversarial Examples Robustly

Jiajun Lu, Theerasit Issaranon, David Forsyth
University of Illinois at Urbana Champaign
{jlu23, issaran1, daf}@illinois.edu

Abstract

We describe a method to produce a network where current methods such as DeepFool have great difficulty producing adversarial samples. Our construction suggests some insights into how deep networks work. We provide a reasonable analyses that our construction is difficult to defeat, and show experimentally that our method is hard to defeat with both Type I and Type II attacks using several standard networks and datasets. This SafetyNet architecture is used to an important and novel application SceneProof, which can reliably detect whether an image is a picture of a real scene or not. SceneProof applies to images captured with depth maps (RGBD images) and checks if a pair of image and depth map is consistent. It relies on the relative difficulty of producing naturalistic depth maps for images in post processing. We demonstrate that our SafetyNet is robust to adversarial examples built from currently known attacking approaches.

1. Introduction

Adversarial examples are images with tiny, imperceptible perturbations that fool a classifier into predicting the wrong labels with high confidence. $\mathbf{x}$ denotes the input to some classifier, which is a *natural* example and has label l. A variety of constructions [9, 14, 20, 25] can generate an *adversarial* example $\mathbf{a}(\mathbf{x})$ to make the classifier label it $m \neq l$. This is interesting, because $\| \mathbf{a}(\mathbf{x}) - \mathbf{x} \|_2$ is so small that we would expect $\mathbf{a}(\mathbf{x})$ to be labelled l.

Adversarial examples are a persistent problem of classification neural networks, and of many other classification schemes. Adversarial examples are easy to construct [30, 22, 3], and there are even universal adversarial perturbations [19]. Adversarial examples are important for practical reasons, because one can construct physical adversarial examples, suggesting that neural networks in current status are unusable in some image classification applications (*e.g.* imagine a small physical modification that could reliably get a stop sign classified as a go faster sign [25, 16]). Adversarial examples are important for conceptual reasons too, because an explanation of why adversarial examples are easy to construct could cast some light on the inner life of neural networks. The absence of theory means it is hard to defend against adversarial examples (for example, distillation was proposed as a defense [26], but was later shown to not work [2]).

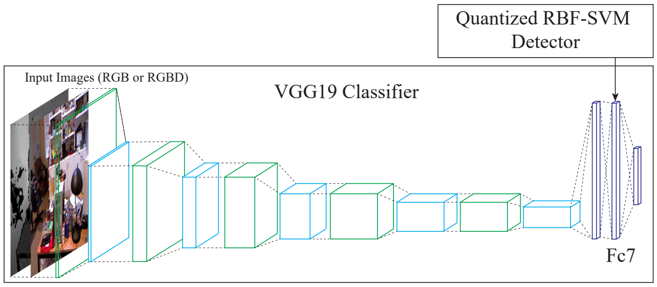

Figure 1: SafetyNet consists of a conventional classifier (in our experiments, either VGG19 or ResNet) with an RBF-SVM that uses discrete codes computed from late stage ReLUs to detect adversarial examples. We show that (a) SafetyNet detects adversarial examples reliably, even if they are produced by methods not represented in the detectors' training set and (b) it is very difficult to produce examples that are both misclassified and slip past SafetyNet's detector.

Adversarial example constructions (*e.g.*, line search along the gradient [9]; LBFGS on an appropriate cost [30]; DeepFool [20]) all rely on the gradient of the network, but it is known that using the gradient of another similar network is sufficient [25], so concealing the gradient does not work as a defense for current networks. An important puzzle is that networks that generalize very well remain susceptible to adversarial examples [30]. Another important puzzle is that examples that are adversarial for one network tend to be adversarial for another as well [30, 15, 27]. Some network architectures appear to be robust to adversarial examples [13], which still need more empirical verification. At least some adversarial attacks appear to apply to many distinct networks [19].

We denote the probability distribution of examples by $P(X)$. At least in the case of vision, $P(X)$ has support on some complicated subset of the input space, which is known as the "manifold" of "real images". Nguyen *et al.* show how to construct examples that appear to be noise, but are confidently classified as objects [23]. This construction yields $\mathbf{a}(\mathbf{x})$ lies outside the support of $P(X)$, so the classifier's labeling is unreliable because it has not seen such examples. However, most adversarial examples "look like" images to humans, such as figure 5 in [30], so they are likely to lie within the support of $P(X)$.

One way to build a network that is robust to adversarial examples is to train networks with enhanced training data (adding adversarial samples [18]); this approach faces difficulties, because the dimension of the images and features in networks means an unreasonable quantity of training data is required. Alternatively, we can build a network that detects and rejects an adversarial sample. Metzen *et al.* show that, by attaching a detection subnetwork that observes the state of the original classification network, one can tell whether it has been presented with an adversarial example or not [17]. However, because the gradients of their detection subnetwork are quite well behaved, the joint system can be attacked (Type II attack) easily in both their and our experiments. Both their and our experiments also show that their detection subnetwork is easily fooled by adversarial samples produced by attacking methods which are not used in detector training process.

Our method focuses on codes produced by quantizing individual ReLUs in particular layers of the classification network ("patterns of activation"), and proceed from the hypothesis:

Hypothesis 1 *Adversarial attacks work by producing different patterns of activation in late stage ReLUs to those produced by natural examples.*

These patterns lie outside the family for which the softmax layer would be reliable. This hypothesis suggests that: (a) the presence of an adversarial example can be detected (as in Metzen *et al.* [17]); (b) such detectors can be made very difficult to defeat (unlike Metzen *et al.* [17]; section 3); (c). such detectors should be good at generalization for different adversarial attacks (unlike Metzen *et al.* [17]); (d) transfer attacks work because an example that generates unfamiliar patterns in one network tends to generate unfamiliar patterns in other networks too; (e) transfer attacks could be defended as well (section 3).

Contributions: Section 2 describes our SafetyNet architecture, which consists of the original classifier network and a detector that rejects adversarial examples. A **type I attack** on SafetyNet consists of a standard adversarial example crafted to be (a) similar to a natural image; (b) misclassified by the original network. A **type II attack** consists of an example that is crafted to be (a) similar to a natural image; (b) misclassified; *and* (c) not rejected by SafetyNet. We show that SafetyNet is robust to both types of attacks and generalize well. Concealing the gradients is highly effective for SafetyNet, and it produces a black box that is strongly resistant to the best attacks we have been able to construct. This is in sharp contrast to all other known methods [25, 17].

In section 3, we demonstrate SceneProof, a robust and reasonably effective proof that an image is an image of a real scene (a "real" image; contrast a "fake" image, which is not an image of a real scene). We identify images of real scenes by checking a match between the image and a depth map, which is hard to manipulate. We show that SceneProof is (a) accurate and (b) strongly resistant to attacks that try to get manipulated scenes identified as authentic scenes.

In section 4, we propose a model that explains why our approach works, and it also demonstrates that SafetyNet is difficult to attack in principle.

2. SafetyNet: Spotting Adversarial Examples

SafetyNet consists of the original classifier, and an adversary detector which looks at the internal state of the later layers in the original classifier, as in Figure 1. If the adversary detector declares that an example is adversarial, then the sample is rejected.

2.1. Detecting Adversarial Examples

The adversary detector needs to be hard to attack. We force an attacker to solve a hard discrete optimization problem. For a layer of ReLUs at a high level in the classification network, we quantize each ReLU at some set of thresholds to generate a discrete code (binarized code in the case of one threshold). Our hypothesis 1 suggests that different code patterns appear for natural examples and adversarial examples. We use an adversary detector that compares a code produced at test time with a collection of examples, meaning that an attacker must make the network produce a code that is acceptable to the detector (which is hard; section 3). The adversary detector in SafetyNet uses an RBF-SVM on binary or quaternary codes (activation patterns) to find adversarial examples.

We denote a code by $\mathbf{c}$. The RBF-SVM classifies by

$$f(\mathbf{c}) = \sum_{i}^{N} \alpha_i y_i \exp(-||\mathbf{c} - \mathbf{c}_i||^2 / 2\sigma^2) + b \qquad (1)$$

In this objective function, when σ is small, the detector produces essentially no gradient unless the attacking code $\mathbf{c}$ is very close to a positive example $\mathbf{c}_i$. Our quantization process makes the detector more robust and the gradients even harder to get. Experiments show that this form of gradient obfuscation is quite robust, and that confusing the detector is very difficult without access to the RBF-SVM, and still difficult even when access is possible. Experiments in section 3 and theory in section 4 confirm that the optimization problem is hard.

2.2. Attacking Methods

We use the following standard and strong attacks [2], with various choice of hyper-parameters, to test the robustness of the systems. Each attack searches for a nearby $\mathbf{a}(\mathbf{x})$ which changes the class of the example and does not create visual artifacts. We use these methods to produce both type I attack (fool the classifier) and type II attack (fool the classifier *and* sneak past the detector).

Fast Sign method: Goodfellow et al [9] described this simple method. The applied perturbation is the direction in image space which yields the highest increase of the linearized cost under l_∞ norm. It uses a hyper-parameter ϵ to govern the distance between adversarial and original image.

Iterative methods: Kurakin et al. [14] introduced an iteration version of the fast sign method, by applying it several times with a smaller step size α and clipping all pixels after each iteration to ensure that results stay in the ϵ neighborhood of the original image. We apply two versions of this method, one where the neighborhood is in L_∞ norm and another where it is in L_2 norm.

DeepFool method: Moosavi-Dezfooli et al. [20] introduced the DeepFool adversary, which is able to choose which class an example is switched to. DeepFool iteratively perturbs an image x_0^{adv}, linearizes the classifier around x_n^{adv} and finds the closest class boundary. The minimal step according to the l_p distance from x_n^{adv} to traverse this class boundary is determined and the resulting point is used as x_{n+1}^{adv}. The algorithm stops once x_{n+1}^{adv} changes the class of the actual classifier. We use a powerful L_2 version of DeepFool.

Transfer method: Papernot et al. [25] described a way to attack a black-box network. They generated adversarial samples using another accessible network, which performs the same task, and used these adversarial samples to attack the black-box network. This strategy has been notably reliable.

2.3. Type I Attacks Are Detected

Accuracy: Our SafetyNet can detect adversarial samples with high accuracy on CIFAR-10 [12] and ImageNet-1000 [4]. For classification networks, we used a 32-layer ResNet [10] for CIFAR-10 and a VGG19 network [29] for ImageNet-1000. Figures 2 shows the detection accuracy of our Binarized RBF-SVM detector on the x5 layer of ResNet for Cifar10 and on the fc7 layer of VGG19 trained for ImageNet-1000. Adversarial samples are generated by **Iterative-L2**, **Iterative-Linf**, **DeepFool-L2** and **FastSign** methods. Figure 2 compares our RBF-SVM detection results with the detector subnetwork results of [17]. The RoC for our detector for Cifar-10 and ImageNet-1000 appears in Figure 3.

Our results show: When our detector is tested on the same adversary as it is trained on, its performance is similar to the detector subnetwork [17], even though our detector works on quantized activation patterns while the detector subnetwork works on original continuous activation patterns. DeepFool is a strong attack. Increasing the number of categories in the problem makes it easier for DeepFool to produce an undetected adversarial example, likely because it becomes easier to exploit local classification errors without producing strange ReLU activations. If DeepFool is required to produce a label outside the top-5 for the original example, the attack is much weaker.

Generalization across attacks: Generally, a detector cannot know at training time what attacks will occur at test time. We test generalization across attacks by training a detector on one class of attack, then testing with other classes of attack. Figure 2 shows that our RBF-SVM generalizes across attacks more reliably than a detector subnetwork. We believe this is because the representation presented to the RBF-SVM has been aggressively summarized (by quantization), so that the classifier is not distracted by subtle but irrelevant features. Note this kind of generalization is not guaranteed just by using a neural network; for example, Table 3 shows networks trained on normal quality JPEG images are confounded by low quality JPEG test images.

Our supplementary materials contain dataset description, rejecting by classification confidence, and Type II attack results.

3. Application: SceneProof

SceneProof is a model application of our SafetyNet, because it would not work with a network that is subject to adversarial examples. We would like Alice to be able to prove to Bob that her photo is real without the intervention of a team of experts, and we'd like Bob to have high confidence in the proof. This proof needs to operate at large scales (i.e. anyone could produce a proof while taking a picture), and automatically.

Current best methods to identify fake images require careful analysis of vanishing points [8], illumination angles [6], and shadows [11] (reviews in [8, 7]). Such analyses are difficult to conduct at large scales or automatically. RGB image editing is easy, with very powerful tools available. We construct a proof by capturing an RGBD image (easily accessible with consumer depth sensors), which changes the security aspect because it's quite hard to edit a depth map convincingly and those edits need to be consistent with the image. The proof of realness is achieved by a classifier that checks both image and depth and determines whether they are consistent. Such a system works if (a) the classifier is acceptably accurate (i.e. it can deter-

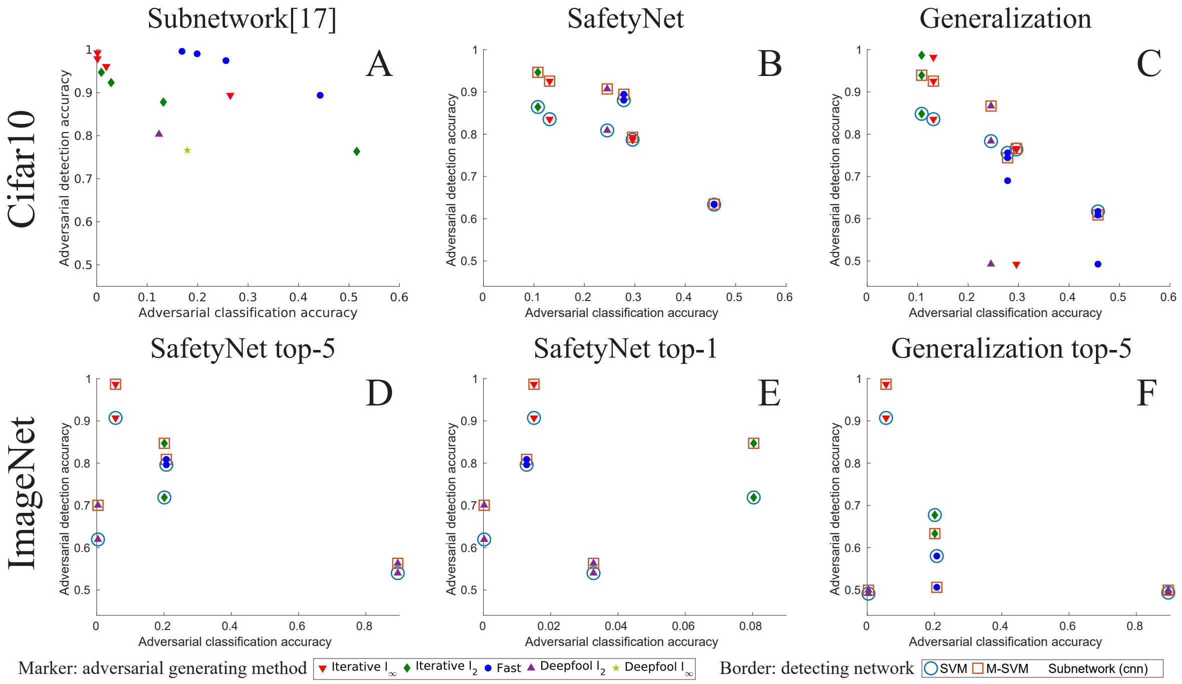

Figure 2: SafetyNet accurately detects adversarial attacks. To facilitate comparison, we follow the conventions of [17], plotting the success of the adversary (i.e. its ability to fool the classifier; leftward is better) on the horizontal axis and the accuracy of the detector on the vertical axis (higher is better). We show results for binary (SVM) and quaternary (M-SVM) codes, and for a variety of attacks. **A:** Results for the detection subnetwork on CIFAR-10 from [17]. **B:** Results for SafetyNet on CIFAR-10, where the detector was trained and tested on adversarial samples generated by the same attacking method (same setting as **A**). **C:** Results for SafetyNet *and* the detection subnetwork (cnn) of [17] on CIFAR-10, where the detector was trained on L_∞ attack and tested on other attacking methods; SafetyNet generalizes better than detection subnetwork to different adversarial attacking methods. **D:** Results for SafetyNet on ImageNet-1000, where the detector was trained and tested on the same adversarial method. The classifier is evaluated with top-5 accuracy (**E** is evaluated with top-1 accuracy, note difference in x axis); using top-5 accuracy significantly advantages the adversary detector, because forcing an adversarial example to move out of top-5 requires larger changes. **F:** Results for SafetyNet on ImageNet-1000 (top-5), where the detector was trained on L_∞ attack and tested on other attacking methods; SafetyNet has relatively small loss of detection accuracy (compared to **E**). We cannot compare to the detection subnetwork of [17], because they do not provide results for ImageNet-1000.

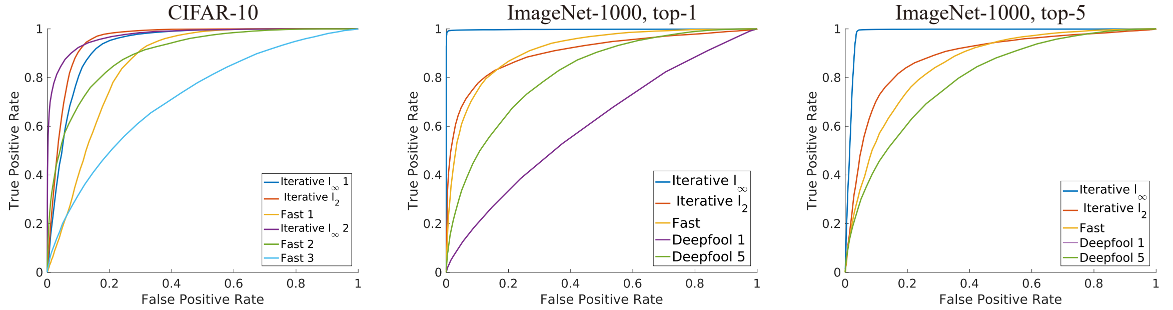

Figure 3: ROC curve for our adversary detector on various adversaries. **Left:** CIFAR-10; **center:** ImageNet-1000, top-1; **right:** ImageNet-1000, top-5. Deepfool-5 is a variant of deepfool that is required to force the adversarial example out of the original example's top 5 classes. Deepfool is a strong adversarial attack, and seems to benefit from being able to choose the target class from multiple classes.

	Non Attack			Type I Attack			Type II Attack		
Method	F→T	T→F	T→T reject	F→T	T→F	T→T reject	F→T	T→F	T→T reject
Non Attack Data	9.7%	0%	9.4%	N/A	N/A	N/A	N/A	N/A	N/A
Unfamiliar Data Average	17.3%	0%	0%	N/A	N/A	N/A	N/A	N/A	N/A
Gradient Descent Attack	N/A	N/A	N/A	9.9%	5.0%	6.1%	16.3%	3.7%	6.2%
Transfer Attack Average	N/A	N/A	N/A	4.6%	9.4%	33.6%	7.9%	9.8%	26.6%

Table 1: Summary of our fc7 RBF-SVM detector's reaction on various non attack data and Type I, Type II attacks (smaller is better). **F→T** means the rate at which false label images are classified as true and the detector does not spot, same for **T→F**. **T→T reject** means the rate at which true label images are classified as true, however, they are rejected by the detector. This number only matters for non attack data because attacks are likely to distort activation patterns even when the label keeps same. As expected, Type I attacks are less successful than Type II attacks. This is because a Type I attack does not explicitly try to fool the detector.

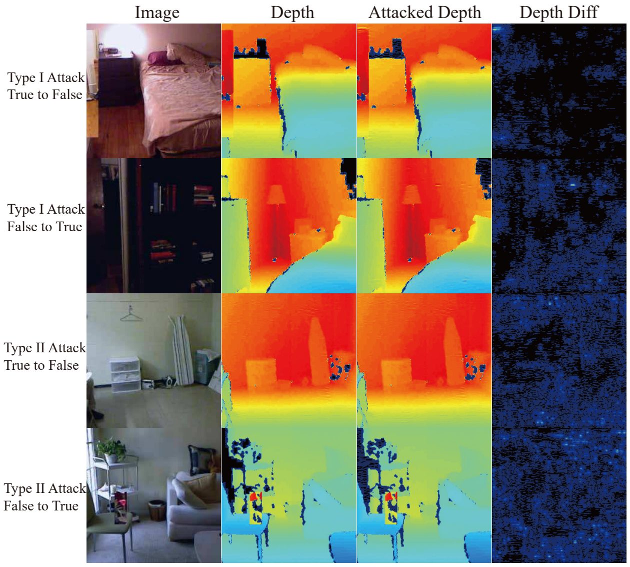

Figure 4: We show figures for successful Type I attacks (fool the classifier) on the original classifier network, and successful Type II attacks (fool both the classifier and detector) on our SafetyNet. Attackers are only allowed to manipulate the depth. Our SafetyNet is very difficult to attack and attacks changing label from False to True is harder. Successful attacks on our SafetyNet requires the original inputs hard to classify and the attacks also need to manipulate the images more.

mine whether the pair is real or not accurately); (b) it can detect a variety of adversarial manipulations of depth or image or both (i.e. type I attacks fail) ; and (c) type II attacks generally fail. We achieve this by using the SafetyNet architecture.

We are mainly concerned with attacks label "fake" images "real". Natural attacks on our system are: produce a depth map for an RGB image using some regression method to obtain an RGBD image (regression); manipulate RGBD image by inserting new objects; take an RGBD image labeled "fake" and manipulate it to be labeled "real" (type I adversarial); take an RGBD image labeled "fake" and manipulate it to be labeled "real" in a way that fools SafetyNet's adversary detector (type II adversarial). There is a wide range of available regression/adversarial attacks, and our system needs to be robust to various methods which might be used to prepare the regression/adversarial attack.

Real test data is easily obtained. We use the raw Kinect captures of LivingRoom and Bedroom from NYU v2 dataset [21]. However, fake data requires care. To evaluate generalization over different attacks, we omit some "regression" methods from the training data and use them only in test. "Regression" methods used in both train and test are: random swaps of depth and image planes; single image predicted depth [5]; rectangle cropped region insertion and random shifted or scaled misaligned depth and image. "Regression" methods used *only* in test are: all zero depth values; nearest neighbor down-sample and up-sampled images and depths; low quality JPEG compressed images and depths; Middlebury stereo RGBD dataset [28] and Sintel RGBD dataset [1](which should be classified "fake" because they are renderings). Refer to Figure 4 for dataset and attacks.

Type I attacks on SafetyNet fail: Type I attacks on SceneProof using a familiar adversary (i.e. one used to train the detector) fail. We report results for two detectors A (applied to fc7 of VGG19) and B (applied to fc6 of VGG19) in Table 2. Type I attacks on SceneProof using an unfamiliar adversary (i.e. one not used to train the detector) generally fail. We report results for two detectors A (applied to fc7 of VGG19) and B (applied to fc6 of VGG19) in Table 3.

A type II attack must both fool the classifier *and* sneak past the detector. We distinguish between two conditions. In non-blackbox case, the internals of the SafetyNet system is *accessible* to the attacker. Alternatively, the network may be a black box, with internal states and gradients concealed. In this case, attackers must probe with inputs and gather outputs, or build another approximate network as in [25].

Type II attacks on accessible SafetyNet fail: a type

Test example type	Classifier Acc	B	A	B $\wedge$ A	A$\vee$B	A$\vee$B, T
Natural RGBD, False	91.8%	15.2%	17.1%	14.3%	18.8%	19.6%
Natural RGBD, True	97.7%	10.1%	11.6%	9.2%	12.7%	10.8%
Adversarial RGBD, False	33.1%	89.1%	88.6%	87.3%	90.4%	88.9%
Adversarial RGBD, True	15.3%	81.3%	81.0%	79.1%	83.3%	83.7%

Table 2: This table shows detailed information about behavior of classifiers and detectors, summarized in Table 1. In this table, results are calculated on the whole SceneProof dataset testing split. Type I attacks on SceneProof using a familiar adversary (i.e. one used to train the detector) fail. We report results for two detectors: A is applied to fc7 of VGG19; B is applied to fc6 of VGG19. Each detector column gives the rate at which the detector labels examples as adversarial. A$\vee$B means the example is reported adversarial if either detector says it is adversarial. A$\wedge$B means the example is reported adversarial if both detectors say it is adversarial. A$\vee$B, T gives the fraction of examples that were detected as adversarial among these that were labelled True by the classifier. We break out results by type and ground truth label. For example, for natural RGBD images that do not represent real scenes (first row), the classifier labels 91.8% correctly; of the 8.2% incorrectly labelled true, the A$\vee$B adversary detector rejects 19.6% as adversarial examples (last column).

Test example type	Classifier Acc	B	A	B $\wedge$ A	A$\vee$B	A$\vee$B,T
zero D channel	76.5%	6.5%	25.6%	6.1%	26.0%	82.0%
down-up sampled	75.2%	54.9%	60.6%	51.3%	63.4%	87.6%
low quality JPEG	36.4%	80.1%	79.2%	77.2%	82.2%	81.8%
Sintel RGBD [1]	27.6%	45.3%	51.7%	39.7%	57.2%	61.4%
Middlebury RGBD [28]	24.0%	39.7%	40.3%	33.4%	46.6%	47.8%

Table 3: This table shows detailed information about behavior of classifiers and detectors, summarized in Table 1. The table arrangement is same to Table 2. Type I attacks on SceneProof using an unfamiliar adversary (i.e. one not used to train the detector) generally fail. All these examples should be labelled false, *OR* rejected as adversarial. The column for each detector reports the rate at which the detector identifies examples as adversarial. For example, in the first row, 76.5% of zero D channel RGBD images are correctly labelled as false by the classifier; of those labelled "true", 82.0% are rejected as adversarial (last column). This means that a total of 4.2% of zero D channel RGBD images pass through SafetyNet with "true" labels.

II attack involves a search for an adversarial example that will be (a) mislabelled and (b) not detected. This search is made difficult by the quantization procedure and by the narrow basis functions in the RBF-SVM, so we smooth the quantization operation and the RBF-SVM kernel operation. Smoothing is essential to make the search tractable, but can significantly misapproximate SafetyNet (which is what makes attacks hard). Our smoothing attack uses a sigmoid function with parameter λ to simulate the quantization process. We also help the search process by increasing the size of the RBF parameter σ to form smoother gradients. Even after smoothing the objective function, attacks tend to fail, likely because it is hard to make an effective tradeoff between easy search and approximation. Table 4 includes Type I and Type II, blackbox and non-blackbox attacking results on SceneProof dataset. Our SafetyNet is the most robust architecture to various attacks.

Type II attacks on black box SafetyNet fail: Assume the state of SafetyNet is concealed. We follow [24, 19] by building attacks on various alternative networks, then transferring these network's adversarial samples. These attacks fail for our SafetyNet, refer to Table 4. In contrast to SafetyNet, the detector subnetwork of [17] is generally susceptible to type II attacks in both blackbox and non-blackbox settings. This is because of quantization process and detection subnetwork's classification boundary problem [19].

4. Theory: Bars and P-domains

We construct one possible explanation for adversarial examples that successfully explains (a) the phenomenology and (b) why SafetyNet works. In this explanation, we assume the network uses ReLU and weight decay, because they are representative, make it easier to explain, and likely to extend to other conditions with some modifications. We have a network with N layers of ReLU's, and study $y_i^{(k)}(\mathbf{x})$, the values at the output of the k'th layer of ReLUs. This is a piecewise linear function of $\mathbf{x}$. Such functions break up the input space into *cells*, at whose boundaries the piecewise linear function changes (i.e. is only C^0). Now assume that for some $y_i^{(k)}(\mathbf{x})$ there exist *p-domains* (union of cells) $\mathcal{D}$ in the input space such that: (a) there are no or few examples in the p-domain; (b) the measure of $\mathcal{D}$ under $P(X)$ is small; (c) $|y_i^{(k)}(\mathbf{x})|$ is large inside $\mathcal{D}$ and small outside $\mathcal{D}$. We will always use the term "p-domain" to refer to domains with these properties. We think that the total measure of all

Method	Ori		Subnet Det			Det A			Det ABC		
	F→T	T→F	F→T	T→F	T→T reject	F→T	T→F	T→T reject	F→T	T→F	T→T reject
Non Attack Data	16.3%	0.6%	**8.4%**	**0%**	10.2%	9.7%	**0%**	**9.4%**	**8.4%**	**0%**	9.9%
Gradient Descent (I)	32.8%	55.3%	13.4%	9.5%	6.0%	9.9%	5.0%	6.1%	**8.4%**	**0.3%**	6.3%
VGG FastSign TF (I)	30.6%	2.8%	14.9%	2.2%	54.1%	7.5%	2.5%	44.1%	**6.6%**	**1.9%**	47.2%
ResNet GradDesc TF (I)	28.9%	36.7%	15.3%	22.4%	33.2%	3.6%	13.4%	29.1%	**2.7%**	**11.9%**	30.3%
ResNet FastSign TF (I)	22.2%	29.1%	7.6%	15.1%	29.8%	2.8%	12.2%	27.5%	**2.2%**	**11.6%**	27.8%
Type I Average	28.6%	30.9%	12.8%	12.3%	30.8%	6.0%	8.3%	26.7%	**5.0%**	**6.4%**	27.9%
Gradient Descent (II)	32.8%	55.3%	26.3%	21.9%	11.9%	16.3%	3.7%	6.2%	**13.2%**	**2.6%**	9.6%
VGG Finetune TF (II)	20%	3.1%	**17.1%**	**0%**	43.5%	17.2%	**0%**	45.6%	17.2%	**0%**	48.4%
VGG Subnet Det TF (II)	16.3%	0.6%	13.7%	**0%**	15.6%	10.3%	**0%**	12.5%	**9.1%**	**0%**	13.1%
ResNet Finetune TF (II)	15.6%	40.3%	8.5%	31.3%	29.3%	1.3%	27.2%	20.6%	**0.3%**	**25%**	21.0%
ResNet Subnet Det TF (II)	23.8%	29.7%	17.6%	19.3%	29.8%	2.8%	12.2%	27.5%	**2.2%**	**11.6%**	27.5%
Type II Average	21.7%	25.8%	16.6%	14.5%	26.0%	9.6%	8.6%	22.5%	**8.4%**	**7.84%**	23.9%

Table 4: Type I and Type II attacks, non-blackbox and blackbox attacks on SceneProof all fail. This table is gather by attacking a randomly selected subset of 3200 images from the whole SceneProof dataset test split (contains 80K images). The table compares a VGG19 network (Ori) with the detection subnetwork of [17] (Subnet), and two variants of SafetyNet (Det A, where we have an RBF-SVM on fc7; and Det ABC, where we have an RBF-SVM on each of fc7, fc6 and pool5, and declare an adversary when any detector responds). T→F shows the rate at which true label classified as false and not detected and F→T shows false label classified as true and not detected (i.e. lower is better). T→T reject shows the rate at which true samples are classified as true, but rejected by detector. *This rate only matters for non attack data, and does not matter for all attacks because attacks are likely to distort the activation patterns even if the classification label has not been changed.* There is no manipulation for the non attack data, which represents unforced errors by the classifier; note that each of the adversary detectors catches a high percentage of the false positives committed by the classifier and rejects them as adversarial. We group attacks by type I attack (I) and type II attack (II). The gradient descent shows the performance of an attack by gradient descent method (type I or type II) on an accessible network. Even when the network is accessible, attacks tend to be unsuccessful. TF represents blackbox transfer attacks where adversarial samples are obtrained from another network (VGG - a VGG19 model; ResNet - a ResNet model). The VGG19 (ResNet) FastSign TF gives results for a type I attack by transferring FastSign adversarials from a VGG19 (ResNet) model. VGG (ResNet) Finetune TF finetunes a VGG19 (ResNet) network with adversarial examples labelled false, and generate adversarials; VGG (ResNet) Subnet Det TF uses a VGG19 (ResNet) network with the detection subnetwork of [17]. The results show that original classifier network is easy to attack successfully with all attacking methods. Subnet methods can detect type I attacks, but are not robust to transfer attacks and are vulnerable to type II attacks. Our SafetyNet is robust to Type I and Type II attacks, as well as gradient descent and transfer attacks, likely because: quantization hides irrelevant patterns; SafetyNet works like a matcher, so is hard to differentiate; and the subnetwork suffers from the classification boundary problem noted in [19].

p-domains under $P(X)$ is small.

By construction, ReLU networks can represent such p-domains. We construct a p-domain using a basis function with small support. $\mathsf{R}(\mathbf{u})$ denote a ReLU applied to $\mathbf{u}$. We have *basic bar function* ϕ.

$$\phi(\mathbf{x}; i, s, \epsilon) = \frac{1}{\epsilon}\begin{pmatrix} \mathsf{R}((x_i - s) + \epsilon) - \\ 2\mathsf{R}((x_i - s)) + \\ \mathsf{R}((x_i - s) - \epsilon) \end{pmatrix}$$

where ϕ has support when $|\, x_i - s \,| < \epsilon$ and has peak value 1. For an index set $\mathcal{I}$ with cardinality $\#\mathcal{I}$ and vectors $\mathbf{s}$, ϵ, we write *bar function* b as

$$b(\mathbf{x}; \mathcal{I}, \mathbf{s}, \epsilon) = \mathsf{R}(\left(\sum_{i \in \mathcal{I}} \phi(\mathbf{x}; i, s_i, \epsilon_i) - \#\mathcal{I} + 1\right))$$

where b has support when $\| \mathbf{x}_{\mathcal{I}} - \mathbf{s}_{\mathcal{I}} \|_1 < \epsilon_{\mathcal{I}}$. Figure 5 illustrates these functions. It is clear that a CNN can encode bars and weighted sums of bars, and that for at least $k \geq 2$ every $y_i^{(k)}$ could in principle be a bar function. Appropriate choices of $\mathbf{s}$, ϵ and $\mathcal{I}$ choose the location and support of the bar and so can produce bars which have low measure under $P(X)$. Now the functions presented to the softmax layer are a linear combination of the $y_i^{(N)}(\mathbf{x})$. This means that with choice of weight and parameters, a bar can appear at this level, and create a p-domain.

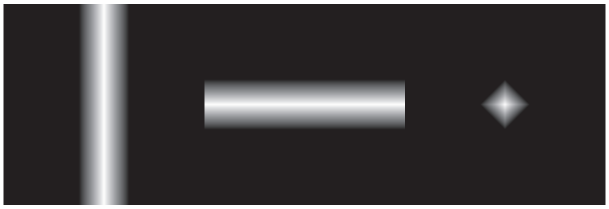

Figure 5: Simple example bar functions on the x, y plane, where black is 0 and white is 1. **Left:** $\phi(\mathbf{x}; 1, 0, 1)$ (i.e. a bar in x, independent of y); **center:** $\phi(\mathbf{x}; 2, 0, 1)$; and **right:** $b(\mathbf{x}; \{1, 2\}, \mathbf{0}, 1)$.

We expect such p-domains to have several important properties. **Adversarial fertility:** P-domains can be used to make adversarial examples by choosing a point in a p-domain close to $\mathbf{x}$. Because there are no or few examples in the p-domain, the loss may not cause the classifier to control the maximum value attained by $y_i^{(k)}(\mathbf{x})$ in this p-domain; and the large range of values inside the p-domain can be used to change the values in layers upstream of k, by moving the example around the p-domain. **Generalization-neutral:** The requirement that p-domains have small measure in $P(X)$ means that both train and test examples are highly unlikely to lie in p-domains. A system with p-domains could generalize well without being immune to adversarial examples. Some subset of p-domains are likely **findable by LBFGS.** Consider the gradient of $y_i^{(N)}(\mathbf{x})$ with respect to $\mathbf{x}$ in two cells separated by a boundary, where some ReLU changes state, weight decay encourages a relatively small change in gradient over these boundaries. If cells neighboring a p-domain have no or few examples in them, we can expect that the gradient change within cell is small too and a second order approximation of $y_i^{(N)}(\mathbf{x})$ could be reliable. We also expect cells to be small, so search and entering a p-domain are possible and requires crossing multiple cell boundaries, which means many changes in ReLU activation. This argument suggests p-domains present **odd patterns of ReLU activation**, particularly in p-domains where some of the $y_i^{(k)}(\mathbf{x})$ are large in the absence of examples.

Why p-domains could exist: As Zhang *et al.* point out, the number of training examples available to a typical modern network is small compared to the relative capacity of deep networks [31]. For example, excellent training error is obtainable for randomly chosen image labels [31]. We expect that $y_i^{(N)}(\mathbf{x})$ will have a number of cells that is exponential in the dimension of $\mathbf{x}$, ensuring that the vast majority of cells lack any example. However, the weight decay term is not sufficient to ensure that $y_i^{(N)}$ is zero in these cells. Overshoot by stochastic gradient descent, caused by poor scaling in the loss, is the likely reason that $y_i^{(N)}(\mathbf{x})$ has support in these cells. Szegedy *et al.* demonstrate that, in practice, ReLU layers can have large norm as linear operators, despite weight decay (see [30], sec. 4.3), so large values in p-domains are plausible. This large norm is likely to be the result of overshoot. Recall that the value of $y_i^{(N)}(\mathbf{x})$ is determined by the *product* of numerous weights, so in some locations in $\mathbf{x}$, the value of $y_i^{(N)}$ could be large, which is a result of multiple layer norms interacting poorly.

An alternative to attacking by search using smoothed RBF gradients is as follows. One might pass an example through the main classifier, determine what code it had, then seek an adversarial example that produces that code (and so must fool the RBF-SVM). We sketch a proof that the optimization problem is extremely difficult. Choose some threshold $t > 0$. We use $b_t(u)$ for the function that binarizes its argument with t. Assume we have at least one unit $y_i^{(k)}$ that encodes a weighted sum of bar functions. We wish to create an adversarial example $\mathbf{a}(\mathbf{x}^*)$ that (a) meets criteria for being adversarial and (b) ensures that $b_t(y_i^{(k)}(\mathbf{a}))$ takes a prescribed value (either one or zero). The feasible set for this constraint can be disconnected (*e.g.* a sum of the bump functions of Figure 5 (right)), and so need not be convex, implying that the optimization problem is intractable. As a simple example, the following constraint set is disconnected for $\epsilon < 1/2$

$$\{\mathbf{x} \mid b_t(b(\mathbf{x}; 1, \mathbf{0}, \epsilon) + b(\mathbf{x}; 1, \mathbf{1}, \epsilon)) = 1\}.$$

5. Discussion

We have described a method to produce a classifier that identifies and rejects adversarial examples. Our SafetyNet is able to reject adversarial examples that come from attacking methods not seen in training data. We have shown that it is hard to produce an example that (a) is mislabeled and (b) is not detected as adversarial by SafetyNet. We have sketched one possible reason that SafetyNet works, and is hard to attack. Many interesting problems are opened by our work, and we provides lots of insights into the mechanism that neural network works.

SaferNet: There might be some better architecture than our SafetyNet, whose objective function is harder to optimize. The ideal case would be an architecture that forces the attacker to solve a hard discrete optimization problem which does not naturally admit smoothing.

Neural network pruning: Our work suggests that networks behave poorly for input space regions where no data has been seen. We speculate that this behavior could be discouraged by a post-training pruning process, which removes neurons, paths or activation patterns not touched by training data.

Explicit management of overshoot during training: we have explained adversarial examples using p-domains, which is the result of poor damping of weights during training. We speculate that constructing adversarial examples during training, by identifying locations where this damping problem occurs and exploiting structural insights into network behavior, could control the adversarial sample problem (rather than just using adversarial examples as training data).

6. Acknowledgements

This work is supported in part by ONR MURI Award N00014-16- 1-2007, in part by NSF under Grant No. NSF IIS- 1421521, and in part by a Google MURA award.

References

[1] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In A. Fitzgibbon et al. (Eds.), editor, *European Conf. on Computer Vision (ECCV)*, Part IV, LNCS 7577, pages 611–625. Springer-Verlag, Oct. 2012. 5, 6

[2] N. Carlini and D. Wagner. Defensive distillation is not robust to adversarial examples. 1, 3

[3] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. *arXiv preprint arXiv:1608.04644*, 2016. 1

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009. 3

[5] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374, 2014. 5

[6] H. Farid. Exposing photo manipulation with inconsistent reflections. *ACM Trans. Graph.*, 31(1):4, 2012. 3

[7] H. Farid. Photo forensics. *MIT Press*, 2016. 3

[8] H. Farid. How to detect faked photos. *American Scientist*, 2017. 3

[9] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 1, 3

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 3

[11] E. Kee, J. F. O'Brien, and H. Farid. Exposing photo manipulation with inconsistent shadows. *ACM Transactions on Graphics (ToG)*, 32(3):28, 2013. 3

[12] A. Krizhevsky. Learning multiple layers of features from tiny images. 2009. 3

[13] D. Krotov and J. J. Hopfield. Dense associative memory is robust to adversarial inputs. *arXiv preprint arXiv:1701.00939*, 2017. 1

[14] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016. 1, 3

[15] Y. Liu, X. Chen, C. Liu, and D. Song. Delving into transferable adversarial examples and black-box attacks. *arXiv preprint arXiv:1611.02770*, 2016. 1

[16] J. Lu, H. Sibai, E. Fabry, and D. Forsyth. No need to worry about adversarial examples in object detection in autonomous vehicles. *arXiv preprint arXiv:1707.03501*, 2017. 1

[17] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff. On detecting adversarial perturbations. *arXiv preprint arXiv:1702.04267*, 2017. 2, 3, 4, 6, 7

[18] T. Miyato, S.-i. Maeda, M. Koyama, K. Nakae, and S. Ishii. Distributional smoothing with virtual adversarial training. *arXiv preprint arXiv:1507.00677*, 2015. 2

[19] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard. Universal adversarial perturbations. *arXiv preprint arXiv:1610.08401*, 2016. 1, 6, 7

[20] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2574–2582, 2016. 1, 3

[21] P. K. Nathan Silberman, Derek Hoiem and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 5

[22] A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 427–436, 2015. 1

[23] A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *CVPR*, 2015. 2

[24] N. Papernot, P. McDaniel, and I. Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016. 6

[25] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami. Practical black-box attacks against deep learning systems using adversarial examples. *arXiv preprint arXiv:1602.02697*, 2016. 1, 2, 3, 5

[26] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *Security and Privacy (SP), 2016 IEEE Symposium on*, pages 582–597. IEEE, 2016. 1

[27] N. Papernot, P. D. McDaniel, and I. J. Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016. 1

[28] D. Scharstein, H. Hirschmüller, Y. Kitajima, G. Krathwohl, N. Nešić, X. Wang, and P. Westling. High-resolution stereo datasets with subpixel-accurate ground truth. In *German Conference on Pattern Recognition*, pages 31–42. Springer, 2014. 5, 6

[29] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 3

[30] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013. 1, 2, 8

[31] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. In *ICLR 2016*, 2016. 8