$\mathcal{SATYA}$: Defending against Adversarial Attacks using Statistical Hypothesis Testing

Sunny Raj[1], Laura Pullum[2], Arvind Ramanathan[2], and Sumit Kumar Jha[1]

[1] Computer Science Department,
University of Central Florida, Orlando, FL, USA
sraj,jha@cs.ucf.edu
[2] Computational Science and Engineering Division,
Oak Ridge National Laboratory, Oak Ridge, TN, USA
pullumll,ramanathana@ornl.gov

Abstract. The paper presents a new defense against adversarial attacks for deep neural networks. We demonstrate the effectiveness of our approach against the popular adversarial image generation method DeepFool. Our approach uses Wald's Sequential Probability Ratio Test to sufficiently sample a carefully chosen neighborhood around an input image to determine the correct label of the image. On a benchmark of 50,000 randomly chosen adversarial images generated by DeepFool we demonstrate that our method $\mathcal{SATYA}$ is able to recover the correct labels for 95.76% of the images for CaffeNet and 97.43% of the correct label for GoogLeNet.

1 Introduction

Over the last few years, it has been shown that small perturbations to an input can cause machine learning algorithms to produce incorrect answers [5,16,18,12]. In particular, computer implementations of vision algorithms including approaches based on deep learning have been shown to be vulnerable to such adversarial attacks. These attack approaches cover a broad spectrum from random sampling of images to the framing of an optimization problem often solved using variants of stochastic gradient descent. This knowledge of adversarial synthesis can be leveraged by an attacker to generate unwanted or malicious output from machine learning systems. Tampering with machine learning systems using adversarial attacks that are directly interacting with humans such as autonomous driving can lead to immediate catastrophic results [17]. As the adoption of machine learning systems is increasing rapidly the security and robustness of these systems gain even more importance. Given the ease with which adversarial inputs can be generated for deep learning algorithms, two questions are of natural interest:

1. Can we detect the adversarial nature of the input to a neural net?
2. Can we recover the correct results even when deep neural networks are exposed to adversarial inputs?

DNN	DNN Accuracy on original image	$\mathcal{SATYA}$ Accuracy on original image	$\mathcal{SATYA}$ Accuracy on adversarial image
CaffeNet	73.76%	71.99%	95.76%
GoogLeNet	78.19%	77.97%	97.43%

Table 1: $\mathcal{SATYA}$ correctly identifies 95.76% of adversarial images generated by DeepFool against the Caffe deep learning framework for 50,000 random images. The accuracy is 97.43% for adversarial versions of 50,000 randomly selected images for GoogLeNet. The accuracy for original images is within 2% of the DNN classification accuracy.

In this paper, we make progress towards answering both these questions for image classification using deep neural networks. We show that the sampling of a suitably-selected neighborhood of the input image that spans two or more classes can be used to correctly classify the input image with high probability. The Sequential Probability Ratio Test (SPRT) allows our approach to adaptively sample this carefully-crafted neighborhood of the input image and decide the label of a (possibly adversarial) image in a computationally efficient manner [23]. In our experimental studies, $\mathcal{SATYA}$ is able to correctly classify 95.76% of adversarial images generated by the DeepFool system for the CaffeNet [22] deep learning framework. We are also able to correctly classify 97.43% of the adversarial images generated from the GoogLeNet [8] deep learning framework. For comparison the method shown in [7] detects DeepFool adversarial images with only 85-90% accuracy. This method detects 50% of the original non-adversarial image as adversarial. In comparison, our method detects less than 2% of non-adversarial images as adversarial. To the best of our knowledge, $\mathcal{SATYA}$'s accuracy on adversarial images synthesized by DeepFool [12] is the highest reported in the literature so far.

The idea that sampling can act as a defense against adversarial attacks is simple and intuitive, though no concrete result of detecting adversarial examples using sampling around the space of input image has been shown in literature. In this paper, we show that simply sampling around the input image is enough to get good results. We show that $\mathcal{SATYA}$ gives us an improvement of more than 1% over this simple sampling approach. We also show that $\mathcal{SATYA}$ is more resilient to variations in the location of adversarial image.

2 Related Work

A variety of machine learning approaches, including deep learning [5] and human-crafted vision algorithms [18] such as histogram-of-gradients, have been shown to be susceptible to adversarial inputs. Small but carefully crafted perturbations in an input can cause a machine learning algorithm to produce an incorrect output. In fact, it has been observed that adversarial examples designed for one deep learning classifier can transfer to another unrelated deep learning classifier and produce incorrect results even in the second classifier [15].

2.1 Adversarial Networks

The design of algorithms for generating adversarial images has received significant attention in the machine learning [9,14] and in software security [16] communities. A framework for generating adversarial nets using backpropagation for multiplayer perceptrons was proposed in [4], and the approach was illustrated on multiple datasets including MNIST, TFD and CIFAR-10. The approach was extended in [10] to conditional generative adversarial nets and experimental results on both MNIST and MIT Flickr 25,000 dataset have been reported. A pyramidal hierarchy of generative adversarial nets have also been used to create image models that are confused to be natural by human evaluators [1].

An interesting white-box approach to adversarial attack [20] relies on exploring the internal layers of the deep neural network representation of an image and making minimal possible perturbation to the image so that its internal representation matches a completely different natural image – thereby leading to incorrect classification of the image. This approach has the ability to trick a deep neural network to confound any image with any other chosen image through cleverly chosen perturbations, and can generate multiple adversarial examples.

Our experimental studies use the state-of-the-art DeepFool [12] algorithm to generate adversarial examples. The DeepFool algorithm is known to compute perturbations that efficiently create adversarial images for deep neural networks. To the best of our knowledge, $\mathcal{SATYA}$'s ability to defend against adversarial perturbations generated by DeepFool with more than 95% probability is new and has not been reported before in the literature.

2.2 Defense against adversarial attacks

Virtual adversarial training [11] uses a KL-divergence based robustness metric of a model against local perturbation around a datapoint to regularize the model. This approach has been reported to work better than ordinary adversarial training on several benchmarks, including MNIST, SVHN and NORB.

Adversarial attacks have theoretically been shown to be more powerful that random noise perturbations [2,3] specifically in the context of linear classifiers. The observation of adversarial attacks in the context of high-dimensional data has been explained by a formal proof demonstrating that robustness to random noise is $\sqrt{d}$ times more than that to adversarial perturbations. Our work is motivated by this observation. Instead of feeding a single adversarial image as an input to a deep neural network, we sample a carefully-constructed neighborhood of the adversarial image and hence avoid making a decision on a single image.

Robust optimization has been used to increase the local stability of artificial neural networks [21], thereby making it harder to generate adversarial examples for such robust networks. They report upto 79.96% accuracy on adversarial images generated from the MNIST benchmark and about 65.01% accuracy on adversarial images generated from the CIFAR-10 benchmark. Our work is different from their approach as we do not seek to optimize the training of the network itself but instead seek to query enough samples so as to prevent an adversarial

attack on a pre-trained classifier like GoogLeNet or CaffeNet. Of course, our approach also happens to produce better experimental results with accuracies as high as 95.76% on adversarial examples for CaffeNet and 97.43% on adversarial examples for GoogLeNet.

A stastical method to detect adversarial examples has been proposed in [6], this method has been shown to work on MNIST, DREBIN and MicroRNA data with attack vectors chosen using FGSM, JSMA, SVM and DT attacks. Impressive performance of about 100% detection in certain tests have been reported. Though no method to recover the original label of the image has been shown, one thing to note is that the data sets used in this method are significantly less complex than ImageNet data set that we are using for our method. Another method for detecting adversarial perturbations has been presented in [7]. This method has been shown to work on ImageNet datasets against DeepFool perturbations. The detection probability for adversarial images has been shown to be around 85-90% for DeepFool perturbed images, though the false positive rate of identifying normal images as adversarial is around 50%. In comparison $\mathcal{SATYA}$, has a false positive rate of less than 2%.

3 The $\mathcal{SATYA}$ algorithm: Defending against adversarial attacks

Our approach to detecting and recovering from adversarial inputs is based on the SPRT-driven sampling of a carefully-crafted neighborhood around a (possibly adversarial) input image. In Section 3.1, we first present an intuitive method of sampling the neighborhood of input image. In Section 3.2, we improve upon the intuitive method to sample in a carefully crafted subspace and discuss its advantage over Section 3.1. The use of Wald's sequential probability ratio test to drive an efficient exploration of this neighborhood is discussed in Section 3.3. An overview of the full method is presented in Algorithm 1.

3.1 Sampling as a defense against adversarial images

A very intuitive approach for developing a defense against adversarial images would be to sample the neighborhood of adversarial images. The underlying idea is simple: *If the adversarial image happens to be adversarial only because it is carefully crafted, its neighbors may still be correctly classified by a deep neural network and hence may help void the adversarial nature of the input.* We show the arrangement of adversarial space in Figure 1. One important point to keep in mind is the dimensionality of the input image; even though we have shown a two-dimensional space in Figure 1, the search space of images has very high dimensionality. The number of dimensions d for an input image to CaffeNet is 227x227x3, where 227 is the input dimension and 3 is the number of channels corresponding to RGB values of the input image.

For a simple sampling approach, we sample on the surface of the hypersphere centered around the input image. We use the sampling method described in [13]

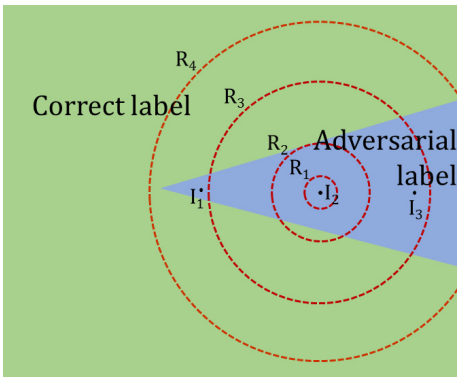


Fig. 1: The correct non-adversarial label space is shown in green, while the adversarial space is shown in blue. The adversarial images I_1, I_2 and I_3 are located at different locations inside the adversarial space. The dotted lines denote hyphersphere with varying radii drawn with the image I_2 as the center. Sampling on a hypersphere of radius R_1 will give incorrect results while sampling on a hypherphere of radius R_4 will give correct results.

to generate uniformly distributed points on the surface of a d-dimensional hypersphere. To generate a random point P_i, we generate d random numbers $r_{i0}, r_{i1} \ldots r_{id}$ from d independent standard normal distribution with $\mu = 0$, $\sigma = 1$ and $d = x \times y \times c$, where x and y are the dimensions of the image and c is the number of channels in the image. Let $G_i = [r_{i0}\ r_{i1}\ \ldots\ r_{id}]$, then the random point P_i on the surface of d-dimensional hypersphere with radius R is given by Equation 1. This equation is implemented by the function $\mathcal{N}$ in Algorithm 1.

$$P_i = \frac{R}{\|G_i\|} G_i^T \tag{1}$$

If an adversarial image is deep inside the adversarial space, sampling on low radius hyperspheres will only give adversarial samples as shown by R_1 and R_2 in Figure 1. An image like I_3 that is further inside the adversarial space will require a larger hypersphere radius to give correct samples when compared to the image I_1. We test the accuracy of detection at various radii for 1000 sample images for both CaffeNet and GoogLeNet; we show the results in Table 2. The performance of CaffeNet is optimal for a radius of 500 units where it reaches the peak accuracy of 92.6%. The performance of GoogLeNet is optimal at 1000 units where the accuracy is 97.0%. We suspect that this difference in peak accuracy at different radii is due to the generally different performance of DeepFool on these two different networks. In the case of CaffeNet, DeepFool might be creating adversarial images closer to the boundary of the adversarial space whereas for GoogLeNet the adversarial image might be further inside the adversarial space.

One trend that we observe in Table 2 is the decline in accuracy as the radius of the hypersphere increases beyond 1500 units. Hyperspheres with higher radii have larger volume and can accommodate samples of multiple labels as shown in Figure 2. Multiple labels on the hypersphere can decrease the chance of

Index	Hyperhsphere radius	Correct Percentage CaffeNet	Correct Percentage GoogLeNet	Average Number of Labels CaffeNet	Average Number of Labels GoogLeNet
1	50	5.7%	6.7%	1.94	1.91
2	100	21.9%	8.0%	1.80	1.71
3	200	58.3%	33.2%	1.60	1.48
4	500	**92.6**%	94.0%	1.43	1.31
5	1000	90.8%	**97.0**%	1.36	1.27
6	1500	89.2%	96.3%	1.38	1.29
7	2000	86.4%	94.5%	1.45	1.34
8	3000	79.2%	91.0%	1.65	1.49
9	5000	64.1%	83.7%	2.31	2.06

Table 2: Sampling the neighborhood of images at varying sampling radii. Third and fourth columns show the percentage of image correctly classified by CaffeNet and GoogleNet. Fifth and sixth column show the average number of different labels on the hyphersphere.

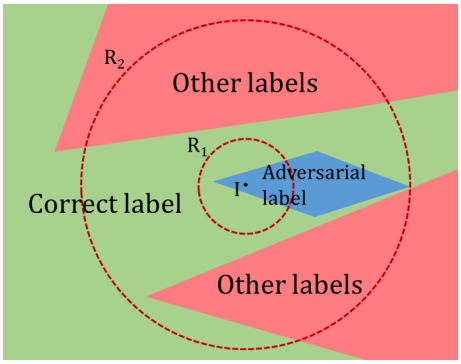


Fig. 2: Sampling around the image I. A low radius R_1 will give correct results, while a very high radius R_2 is likely to give incorrect results.

the correct label having the maximum number of samples. This hypothesis is confirmed by the fact that we observe an increase in the average number of labels on the hyphersphere as the radius increases from 1500 units. One natural conclusion from these experimental observations is that an algorithmic approach to defend against adversarial attacks should construct a consistent search space unaffected by the position of the adversarial image inside the adversarial space.

3.2 Constructing a suitable sample space

One suitable candidate for a consistent sampling space is the neighborhood of an image that is on the boundary between the correct non-adversarial space and the adversarial space. This image is shown as I_{mid} in Figure 3. The location of I_{mid} reduces the need for finding the optimal hyphersphere radius for sampling.

We present the procedure to calculate the image I_{mid} in this section. We iterate that the figures shown here are a simplification of the more complicated high dimensional space.

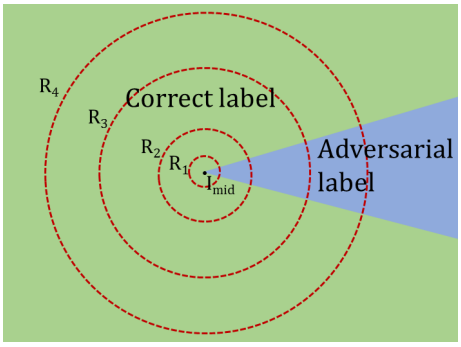


Fig. 3: Sampling around the image I_{mid} at the border of the space of correct non-adversarial label and adversarial label. Sampling on most hyphersphere radii will give correct results.

Given an input image I and a deep neural network (DNN) classifier $\mathcal{C}$, $\mathcal{SATYA}$ first calculates the best classification label l_1 for the image I. It also computes the second-best classification label l_2 for the same image. For an adversarial image generate by DeepFool, the second label l_2 is the correct label. Similarly for the non adversarial image, the label l_2 is the incorrect label and is the label of the adversarial space. We generate the image I_{mid} using the gradient generated by the backpropagation function of the DNN. The error function $\mathcal{E}(I, l)$ of the classifier $\mathcal{C}$ gives the backpropagated error at the input layer with the input image I and the correct label l. At each step j of the iteration, for an input image I^j the error $\mathcal{E}(I^j, l_2)$ of the input layer of DNN is generated by assuming l_2 as the correct label of the image. The new image I^{j+1} is generated using the update $I^{j+1} = I^j + \mathcal{E}(I^j, l_2)$. Each iteration generates an image with higher confidence of l_2. We continue adding $\mathcal{E}(I^j, l_2)$ until at iteration k, l_2 is the highest confidence label of the image. The image I^k has the label l_2 and the image I^{k-1} has the label l_1. The image I_{mid} is calculated by doing a binary search between the images I^k and I^{k-1} to get an image on the separating boundary of the two labels l_1 and l_2. The algorithm then samples images on the surface of an n-dimensional hypersphere of a fixed radius around I_{mid} using the method shown in Section 3.1.

The results of sampling with various radii for 1000 images is shown in Table 3. For both CaffeNet and GoogLeNet, good accuracy is obtained at the smallest sampled radius of 50 units. We note that the accuracy of $\mathcal{SATYA}$ is better than simple sampling around the adversarial image. We revisit this comparison with higher number of samples in the experimental section. We can also note that there is less variation between the accuracy at different hyphersphere radii for

$\mathcal{SATYA}$. In Table 2 the standard deviation for CaffeNet is 29.98 whereas the standard deviation in Table 3 is 10.16. Similarly for GoogLeNet, the standard deviation in Table 2 is 37.05 where as it is 4.34 for Table 3. This simple metric shows us that the results obtained from $\mathcal{SATYA}$ is more resilient to variations in the location of the adversarial image.

Algorithm 1 $\mathcal{SATYA}$ Adversarial Image Classification

Input: Image I, Set of labels L, Deep Neural Network Classifier $\mathcal{C}$, Input layer error function of classifier $\mathcal{E}$, Type I/II error e, Maximum number of samples N, Indifference region $[p_0, p_1]$, Maximum number of iterations for searching middle image M, Sampling radius R
Output: Classification label for image I

$l_1 = \arg\max_{l \in L} \mathcal{C}(I, l)$ ▷*Find best label for image I*
$l_2 = \arg\max_{l \in \{L \setminus l_1\}} \mathcal{C}(I, l)$ ▷*Find second-best label for I*
$l_c = l_1$, $I_1 = I$, $I_2 = I$, $m = 0$, $n = 0$, $s = 0$
while $l_c \neq l_2$ **do**
  $I_1 = I_2$
  $I_2 = I_2 + \mathcal{E}(l_2)$
  $l_c = \mathcal{C}(I_2)$
end while
▷*Perform binary search to compute the boundary between l_1 and l_2*
$I_{mid} = \mathcal{B}(I_1, I_2)$
▷*Compute SPRT stopping criteria for Type I/II error*
$S_{min} = \log(\frac{e}{1-e})$, $S_{max} = \log(\frac{1-e}{e})$
repeat
  $n = n + 1$ ▷*Increment total number of samples*
  $J =$ sample *i.i.d.* from $\mathcal{N}(I_{mid}, R)$
  if $\mathcal{C}(J) = l_2$ **then**
    $s = s + 1$ ▷*Increment no. of successful samples*
  end if
  ▷*Update Sequential Probability Ratio*
  $S = log\left(\frac{p_1^s(1-p_1)^{n-s}}{p_0^s(1-p_0)^{n-s}}\right)$
until $S < S_{min}$ **or** $S > S_{max}$ **or** $n \geq N$
if $s > n - s$ **then**
  print Class label: l_2
else
  print Class label: l_1
end if

In our current implementation, we have only considered the top-2 labels l_1 and l_2 for deciding the correct label of the image. The limitation of top-2 labels works in the case of DeepFool as the adversarial attack algorithm works gradually towards an adversarial label and the algorithm terminates at the first instance of a wrong label thus leaving the correct label as l_2. For implementing

a top-n variant of this algorithm, a competition between the top-n labels can be organized and the winner declared as the current label.

Index	Hypersphere radius	Correct Prediction CaffeNet	Correct Percentage CaffeNet	Correct Prediction GoogLeNet	Correct Percentage GoogLeNet
1	50	974	**97.4** %	970	97.0 %
2	100	963	96.3 %	973	97.3 %
3	200	952	95.2 %	974	97.4 %
4	500	926	92.6 %	977	**97.7** %
5	1000	896	89.6 %	963	96.3 %
6	1500	886	88.6 %	956	95.6 %
7	2000	858	85.8 %	941	94.1 %
8	3000	783	78.3%	915	91.5 %
9	5000	635	63.5 %	834	83.4 %

Table 3: Sampling the neighborhood of images at varying sampling radii and percentage of those images classified correctly for 1000 images.

3.3 Statistical Hypothesis Testing

The sampling neighborhood around the transition image I_{mid} constructed in the previous subsection is quantitatively different for different input images. For adversarial images generated from images that were correctly recognized by the DNN classifier $\mathcal{C}$ with high-confidence, we find that the sampling neighborhood contains an overwhelming majority of images that are correctly labeled by the DNN classifier $\mathcal{C}$. On the other hand, the sampling neighborhood only contains a thin majority of images correctly labeled by the DNN classifier $\mathcal{C}$ if the original image was correctly classified by the classifier with a very low confidence.

$\mathcal{SATYA}$ uses the Sequential Probability Ratio Test (SPRT) to adaptively sample the neighborhood constructed in the previous subsection [23]. The test rejects one of the following two hypotheses:

Null Hypothesis: $\mathcal{C}$ assigns the label l_2 to images in the neighborhood of I_{mid} with probability more than p_1

Alternate Hypothesis: $\mathcal{C}$ assigns the label l_2 to images in the neighborhood of I_{mid} with probability less than p_0

The user specifies an indifference region $[p_0, p_1]$, Type I/II error e and the maximum number of samples to be obtained N. SPRT then samples the neighborhood recording the total number of images sampled (n) and the number of images (s) labeled by the classifier as l_2. Using these inputs, SPRT computes the likelihood ratio:

$$\frac{p_1^s(1-p_1)^{n-s}}{p_0^s(1-p_0)^{n-s}}$$

If the likelihood ratio falls below a threshold derived from the Type I/II error, SPRT rejects the null hypothesis. If the likelihood ratio exceeds a threshold, SPRT rejects the alternate hypothesis.

If the probability of sampling an image with the label l_2 is more than p_1, the algorithm will produce this label with probability $1 - e$. For example, if $p_1 = 0.51$ and $e = 0.01$, our algorithm produces this label with 99% accuracy if the sampling neighborhood has at least 51% correctly labeled images. Of course, greater accuracy can be achieved by reducing the Type I/II error and by setting the value of p_1 to $0.5 + \epsilon$ for a small $\epsilon > 0$. However, this comes at the expense of a larger number of samples needed to reach a conclusion.

In Figures 6 and 7, we show how the number of samples required by our statistical hypothesis testing algorithm can vary widely among different input images. In particular, adversarial images that are generated from images for which the DNN classifier $\mathcal{C}$ was making a correct but low-confidence prediction tend to require larger number of samples. Figure 8 shows that the number of samples required to disambiguate an adversarial image generated from an original image classified with confidence between 0.4 and 0.6 is more than 5 times the number of samples required for a high-confidence (0.8-1.0) prediction. Thus, the SPRT-driven adaptive sampling is critical to ensure an efficient performance of $\mathcal{SATYA}$.

4 Experimental Results

We evaluated $\mathcal{SATYA}$ on 50,000 images from the ILSVRC2013 training dataset [19] for both the CaffeNet [22] and the GoogLeNet [8] deep learning frameworks. Adversarial versions of these images were created using the state-of-the-art DeepFool [12] adversarial attack system. In our experiments, we have used the following parameters for the $\mathcal{SATYA}$ algorithm: Type I/II error $e = 0.000001$, maximum number of samples $N = 2000$, indifference regions $p_0 = 0.47$ and $p_1 = 0.53$, maximum number of iterations for searching the transition image I_{mid} $M = 500$ and hyphersphere radius for sampling $R = 200$ units. Our experiments were carried on a 16GB Intel(R) Core(TM) i7-4770K CPU @ 3.50GHz workstation with an NVIDIA GeForce GTX 780 GPU. Our experimental evaluation has four goals:

1. How well does $\mathcal{SATYA}$ perform on the adversarial images generated by the DeepFool algorithm working with CaffeNet and on adversarial images generated for GoogLeNet?
2. What is the impact of $\mathcal{SATYA}$ on original non-adversarial benchmarks?
3. How is the runtime performance of $\mathcal{SATYA}$?
4. Does the runtime of our approach vary significantly depending on the image being investigated?

4.1 Accuracy on adversarial images

$\mathcal{SATYA}$ correctly identifies more than 95% of all the adversarial images generated for both the CaffeNet deep neural network and the GoogLeNet deep

learning framework. The results for the execution on 50,000 ILSVRC images is shown in Table 1. The hyphersphere radius for sampling was taken to be 200 units for both CaffeNet and GoogLeNet. The accuracy of detection of the correct label was 95.76% for CaffeNet and 97.43% for googleNet.

To compare $\mathcal{SATYA}$ with simple sampling approch around input adversarial image we ran the benchmark on the same set of 10,000 random ILSVRC images. The hyphersphere radius for best accuracy was 500 units for CaffeNet and 1000 units for GoogLeNet. We show the experimental results in Table 4. We can see that an accuracy gain of 2.12% for CaffeNet and 1.14% for googleNet was achieved using $\mathcal{SATYA}$.

Benchmark	Simple sampling	$\mathcal{SATYA}$	Accuracy gain
CaffeNet	93.40%	95.52%	2.12%
GoogLeNet	96.55%	97.69%	1.14%

Table 4: Accuracy of $\mathcal{SATYA}$ compared to simple sampling approach for a sample set of 10,000 ILSVRC images. The hypersphere radius for CaffeNet was taken to be 500 units and for googleNet it was taken to be 1000 units.

4.2 Accuracy on original unperturbed images

While $\mathcal{SATYA}$'s performance on adversarial images is very good, it would not be a useful algorithm if its performance on non-adversarial images turned out to be poor. $\mathcal{SATYA}$ performs very well even on original non-adversarial images. The accuracy of CaffeNet on the original non-adversarial image is 73.76% while the accuracy of $\mathcal{SATYA}$ on CaffeNet is 71.99%. The accuracy of GoogLeNet on the original non-adversarial image is 78.19% while the accuracy of $\mathcal{SATYA}$ on original image is 77.54%. We can see that the underlying detection algorithms perform only slightly better than $\mathcal{SATYA}$. The breakup for images correctly and incorrectly classified by CaffeNet is given in Table 5. We can see that $\mathcal{SATYA}$ correctly classifies 7.53% of image originally incorrectly classified by CaffeNet and 2.45% of image originally incorrectly classified by GoogLeNet.

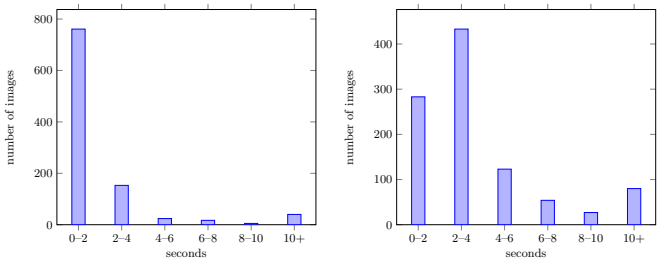
4.3 Runtime Performance

The time required by $\mathcal{SATYA}$ to analyze both original and adversarial images for CaffeNet is shown in Figure 4 and for GoogLeNet is shown in Figure 5. The runtime performance of $\mathcal{SATYA}$ is acceptable for high-fidelity applications like cyber-physical systems. An overwhelming majority of the images were analyzed by $\mathcal{SATYA}$ within 4 seconds. We should note that the availability of enough parallel computational resource can be used to speed up $\mathcal{SATYA}$ to match the

	Prediction		Correct
Benchmark	Wrong	Correct	Percentage
CaffeNet+	1808	35074	95.10%
CaffeNet-	12199	919	7.53%
GoogLeNet+	540	38553	98.62%
GoogLeNet-	10646	261	2.45%

Table 5: $\mathcal{SATYA}$ correctly identifies 95.10% of the 36882 original images correctly recognized by the Caffe deep learning framework (called CaffeNet+ here) and recognizes 7.53% of the 13118 original images not correctly recognized by Caffe (called CaffeNet- here). The accuracy is 98.62% for original versions of 39093 correctly recognized images and 2.45% for original version of 10907 images not correctly recognized by GoogLeNet. Here, the two classes are referred as GoogLeNet+ and GoogLeNet- respectively.

runtime performance of the underlying classifier by using massively parallel calls from $\mathcal{SATYA}$ to the underlying classifier such as CaffeNet or GoogLeNet.

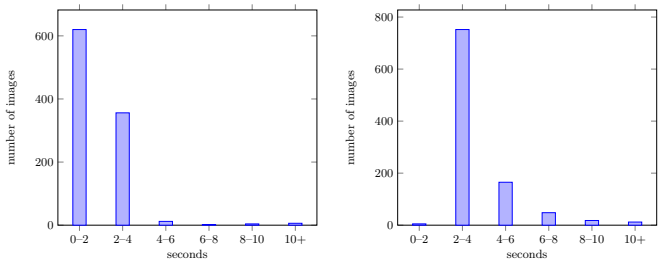


(a) Adversarial Images (b) Original Images

Fig. 4: Time taken by $\mathcal{SATYA}$ for predicting the label of adversarial and ordinary images used in CaffeNet.

$\mathcal{SATYA}$ correctly recognizes about 91% of the adversarial images and 70% of the original images within 4 seconds on our single GPU machine with only sequential calls to the DNN CaffeNet classifier. The worst case runtime of our approach on adversarial images is 22 seconds for the CaffeNet deep neural network.

The performance of $\mathcal{SATYA}$ on images from the GoogLeNet is qualitatively similar to the results on CaffeNet. About 98% of the adversarial images and 75% of the original images can be analyzed within 4 seconds. The worst case runtime of $\mathcal{SATYA}$ on adversarial images is 27 seconds for GoogLeNet.

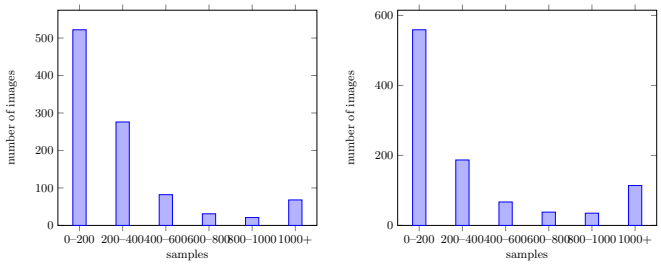


(a) Adversarial Images (b) Original Images

Fig. 5: Time taken by $\mathcal{SATYA}$ in calculating the label for images used in GoogLeNet. The performance of both adversarial and original images have been analyzed.

4.4 Dependence of performance on the confidence of classification

The performance of $\mathcal{SATYA}$ depends upon the number of images sampled by the sequential probability ratio test. In Figure 6, we illustrate the number of samples required by $\mathcal{SATYA}$ while analyzing original and adversarial images for CaffeNet. About 50% of the adversarial images can be analyzed by studying only 200 samples. Similarly, about 55% of the original images can be classified by analyzing only 200 samples. Only a small fraction of images require more than $1,000$ samples.

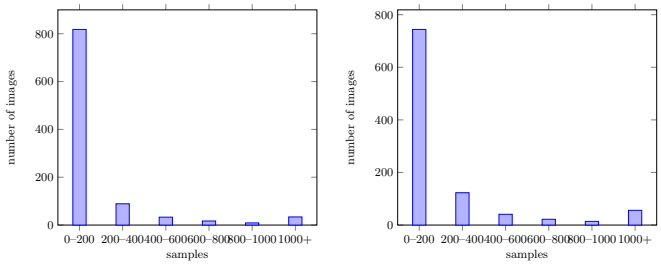


(a) Perturbed Images (b) Original Images

Fig. 6: Number of samples tested to determine the label of an original as well as an adversarial image for CaffeNet.

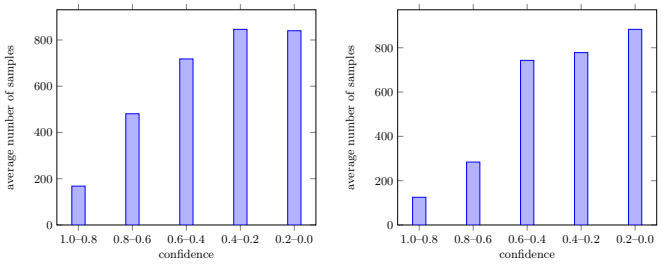
Figure 7 shows the number of samples required by original and adversarial images for the GoogLeNet deep learning framework. About 80% of the perturbed images and more than 75% of the original images were analyzed by sampling fewer than 200 samples. In the light of this variation in number of samples for a small fraction of the images, a natural question that arises is the source of this

variability. Using Figures 8 we establish an empirical relationship between the number of samples required to disambiguate an image and the confidence with which the classifier assigns a label to the image. If CaffeNet is able to correctly label an image with a confidence of 0.6 or more, $\mathcal{SATYA}$ only needs 500 or fewer samples to assign a label to an adversarial input created using this image.



(a) Adversarial Images (b) Original Images

Fig. 7: Number of samples tested to determine the label of an original as well as an adversarial image for GoogLeNet.



(a) CaffeNet (b) GoogLeNet

Fig. 8: The average number of samples needed to classify an image increases as the classifier's confidence in the label of the image decreases.

We observed qualitatively similar results for adversarial images generated for the GoogLeNet deep learning classifier. Adversarial inputs generated using images that were assigned high-confidence labels by GoogLeNet (0.6 or more) were easily labeled by $\mathcal{SATYA}$ using fewer than 300 samples. The fact that $\mathcal{SATYA}$ is able to recover the labels of adversarial inputs generated from high-confidence images is extremely desirable. Such a performance behavior implies that high-

confidence predictions from a classifier may be difficult to distort in a manner where they cannot be recovered by $\mathcal{SATYA}$ and other defensive approaches.

5 Conclusion & Future Work

$\mathcal{SATYA}$ provides a highly effective defense against adversarial attacks. In our experimental evaluation, more than 95% of adversarial images generated by DeepFool against CaffeNet deep neural network and against GoogLeNet deep leaning framework are correctly recognized by our approach. $\mathcal{SATYA}$ also performs comparably to the underlying image detection system for non-adversarial images. When compared to simple sampling approach, $\mathcal{SATYA}$ gives better accuracy and is more resilient to variations in the adversarial image.

Several natural avenues for future research are open. A theoretical explanation of the success of our approach perhaps using manifolds will help clarify the interaction of deep neural networks and high-dimensional big data. Practical efforts towards parallelizing $\mathcal{SATYA}$ would help make the tool deployable in real-time settings.

Acknowledgments

The authors would like to thank the US Air Force for support provided through the AFOSR Young Investigator Award to Sumit Jha. The authors acknowledge support from the National Science Foundation Software & Hardware Foundations #1438989 and Exploiting Parallelism & Scalability #1422257 projects. This material is based upon work supported by the Air Force Office of Scientific Research under award number FA9550-16-1-0255. This research was partially supported by ORNL's Laboratory Directed Research and Development (LDRD) proposal 7899. This manuscript has been authored by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy.

References

1. Denton, E.L., Chintala, S., Fergus, R., et al.: Deep generative image models using a laplacian pyramid of adversarial networks. In: Advances in neural information processing systems. pp. 1486–1494 (2015)
2. Fawzi, A., Fawzi, O., Frossard, P.: Analysis of classifiers' robustness to adversarial perturbations. arXiv preprint arXiv:1502.02590 (2015)
3. Fawzi, A., Fawzi, O., Frossard, P.: Fundamental limits on adversarial robustness. In: Proc. ICML, Workshop on Deep Learning (2015)
4. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in neural information processing systems. pp. 2672–2680 (2014)
5. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014)
6. Grosse, K., Manoharan, P., Papernot, N., Backes, M., McDaniel, P.D.: On the (statistical) detection of adversarial examples. CoRR abs/1702.06280 (2017), `http://arxiv.org/abs/1702.06280`

7. Hendrik Metzen, J., Genewein, T., Fischer, V., Bischoff, B.: On Detecting Adversarial Perturbations. ArXiv e-prints (Feb 2017)
8. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 25, pp. 1097–1105. Curran Associates, Inc. (2012), http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
9. Kurakin, A., Goodfellow, I., Bengio, S.: Adversarial machine learning at scale. arXiv preprint arXiv:1611.01236 (2016)
10. Mirza, M., Osindero, S.: Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784 (2014)
11. Miyato, T., Maeda, S.i., Koyama, M., Nakae, K., Ishii, S.: Distributional smoothing with virtual adversarial training. arXiv preprint arXiv:1507.00677 (2015)
12. Moosavi-Dezfooli, S.M., Fawzi, A., Frossard, P.: Deepfool: a simple and accurate method to fool deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2574–2582 (2016)
13. Muller, M.E.: A note on a method for generating points uniformly on n-dimensional spheres. Commun. ACM 2(4), 19–20 (Apr 1959), http://doi.acm.org/10.1145/377939.377946
14. Nguyen, A., Yosinski, J., Clune, J.: Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 427–436 (2015)
15. Papernot, N., McDaniel, P., Goodfellow, I.: Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. arXiv preprint arXiv:1605.07277 (2016)
16. Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z.B., Swami, A.: The limitations of deep learning in adversarial settings. In: Security and Privacy (EuroS&P), 2016 IEEE European Symposium on. pp. 372–387. IEEE (2016)
17. Raj, S., Ramanathan, A., Pullum, L.L., Jha, S.K.: Testing autonomous cyber-physical systems using fuzzing features derived from convolutional neural networks. In: ACM SIGBED International Conference on Embedded Software (EMSOFT). ACM, ACM, Seoul, South Korea (2017)
18. Ramanathan, A., Pullum, L.L., Hussain, F., Chakrabarty, D., Jha, S.K.: Integrating symbolic and statistical methods for testing intelligent systems: Applications to machine learning and computer vision. In: Design, Automation & Test in Europe Conference & Exhibition (DATE), 2016. pp. 786–791. IEEE (2016)
19. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV) 115(3), 211–252 (2015)
20. Sabour, S., Cao, Y., Faghri, F., Fleet, D.J.: Adversarial manipulation of deep representations. arXiv preprint arXiv:1511.05122 (2015)
21. Shaham, U., Yamada, Y., Negahban, S.: Understanding adversarial training: Increasing local stability of neural nets through robust optimization. arXiv preprint arXiv:1511.05432 (2015)
22. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S.E., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. CoRR abs/1409.4842 (2014), http://arxiv.org/abs/1409.4842
23. Wald, A.: Sequential analysis. John Wiley (1947)