STARS

University of Central Florida
STARS

Electronic Theses and Dissertations, 2020-

2020

Towards Robust Artificial Intelligence Systems

Sunny Raj
University of Central Florida

 Part of the Artificial Intelligence and Robotics Commons
Find similar works at: https://stars.library.ucf.edu/etd2020
University of Central Florida Libraries http://library.ucf.edu

This Doctoral Dissertation (Open Access) is brought to you for free and open access by STARS. It has been accepted for inclusion in Electronic Theses and Dissertations, 2020- by an authorized administrator of STARS. For more information, please contact STARS@ucf.edu.

STARS Citation

Raj, Sunny, "Towards Robust Artificial Intelligence Systems" (2020). *Electronic Theses and Dissertations, 2020-*. 121.
https://stars.library.ucf.edu/etd2020/121

TOWARDS ROBUST ARTIFICIAL INTELLIGENCE SYSTEMS

by

SUNNY RAJ
B.Tech. Indian Institute of Technology, Dhanbad, 2011

A dissertation submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy
in the Department of Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Spring Term
2020

Major Professor: Sumit Kumar Jha

© 2020 Sunny Raj

ABSTRACT

Adoption of deep neural networks (DNNs) into safety-critical and high-assurance systems has been hindered by the inability of DNNs to handle adversarial and out-of-distribution input. State-of-the-art DNNs misclassify adversarial input and give high confidence output for out-of-distribution input. We attempt to solve this problem by employing two approaches, first, by detecting adversarial input and, second, by developing a confidence metric that can indicate when a DNN system has reached its limits and is not performing to the desired specifications. The effectiveness of our method at detecting adversarial input is demonstrated against the popular DeepFool adversarial image generation method. On a benchmark of 50,000 randomly chosen ImageNet adversarial images generated for CaffeNet and GoogLeNet DNNs, our method can recover the correct label with 95.76% and 97.43% accuracy, respectively. The proposed attribution-based confidence (ABC) metric utilizes attributions used to explain DNN output to characterize whether an output corresponding to an input to the DNN can be trusted. The attribution based approach removes the need to store training or test data or to train an ensemble of models to obtain confidence scores. Hence, the ABC metric can be used when only the trained DNN is available during inference. We test the effectiveness of the ABC metric against both adversarial and out-of-distribution input. We experimental demonstrate that the ABC metric is high for ImageNet input and low for adversarial input generated by FGSM, PGD, DeepFool, CW, and adversarial patch methods. For a DNN trained on MNIST images, ABC metric is high for in-distribution MNIST input and low for out-of-distribution Fashion-MNIST and notMNIST input.

This work is dedicated to my parents and my sisters.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

Artificial Intelligence systems have found applications in a variety of domains ranging from computer vision, speech recognition to playing games such as chess and go [6, 7, 8, 9]. The availability of big data and fast GPUs has allowed deep neural network (DNN) based AI systems to equal or sometimes exceed human-level performance on various benchmarks [10, 11, 12]. Despite these remarkable achievements, the adoption of deep learning systems in safety-critical systems is hindered by the inability of DNNs to handle adversarial or out-of-distribution inputs. While it is important to develop DNN models for higher accuracy, there is also a need to develop methods that characterize the limitations of these systems.

It has been shown that small perturbations to input can cause machine learning and, in particular, deep learning systems to produce incorrect answers [4, 13, 14, 2]. 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 a 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 catastrophic results [15].

Machine learning systems are unable to handle out-of-distribution images. ML systems can generate high confidence prediction for irrelevant input. ML systems trained only on a certain input type can still give high confidence prediction output for a compatible unrelated input and not generate any alarms. This behaviour can create problems for safety critical tasks such a medical diagnosis where a significant change in the input can go undetected. As the adoption of ML systems is

increasing rapidly the security and robustness of these systems gain even more importance.

In this thesis, we attempt to solve the problem of adversarial attacks on deep learning systems by attempting to answer the following two questions:

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?

We introduce a new method $\mathcal{SATYA}$ that makes 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 [16]. In our experimental studies, $\mathcal{SATYA}$ is able to correctly classify 95.76% of adversarial images generated by the DeepFool system for the CaffeNet [17] deep learning framework. We are also able to correctly classify 97.43% of the adversarial images generated from the GoogLeNet [18] deep learning framework. For comparison the method shown in [19] 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 [2] is the highest reported in the literature so far. Another method for detecting adversarial perturbations has been presented in [19], 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 iden-

tifying normal images as adversarial is around 50%. In comparison $\mathcal{SATYA}$ has a false positive rate of less than 2%.

Table 1.1: $\mathcal{SATYA}$ 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.

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%

The idea that sampling can act as a defense against adversarial attacks is intuitive and straightforward, though no concrete result of detecting adversarial examples using sampling around the space of input image has been shown in the literature. In this thesis, we show that merely 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 the adversarial image.

We then extend the sampling strategy to develop a confidence metric that can indicate when the DNN system has reached its limits and is not able to perform to the desired specification. A confidence metric would help increase trust between the human operator and the DNN system and allow further integration of DNNs in safety-critical systems such as medical diagnosis [20] and autonomous driving [21] by letting DNNs operate under standard input while giving a clear indication when the DNN prediction becomes unreliable and enabling a human or a simplex system architecture [22] to intervene. A low confidence score can also be used to detect distribution shifts or adversarial attacks. A straightforward approach to computing such a score is to use the logit values prior to the softmax layer. But this raw confidence score is poorly calibrated [23, 24] and does not correlate well with prediction accuracy. The logits reflect high confidence even on wrong

predictions over adversarial examples [25, 26, 27]. In fact, the improved accuracy in deep learning models over the last decade has been accompanied with worsening calibration of logit output to network accuracy [23] compared to earlier models [28]. It indicates overfitting in the negative log likelihood space even though DNNs avoid overfitting in the output accuracy [29]. This has motivated the development of confidence measures that use a held-out validation set for training a separate calibration model on the output of the DNNs [30, 31, 23]. But additional training or validation data may not be available with the trained machine learning (ML) model due to privacy, security or other practical constraints. Another class of confidence measures use sampling over model ensembles or training data to estimate conformance and data density [32, 33, 34]. In contrast, the focus of our work is to compute confidence for a DNN model on a given input without access to training data or the possibility of retraining a new model or model ensembles.

Figure 1.1: (From left to right) The original image with a label of yawl; masking its top 0.2% attribution; masking its top 4% of attribution; image with a banana adversarial patch [1]; masking its top 0.2% attribution; (rightmost) masking its top 4% of attribution. The classification result for the original image is robust and conforming in the neighborhood obtained by removing top positive attributions. The model predicts the original label (yawl) in the neighborhood. But the prediction changes from banana to yawl for many samples in the attribution neighborhood of the adversarial image. We observe a similar difference in conformance for the out-of-distribution examples and the adversarial examples generated by state-of-the-art digital attack methods such as DeepFool [2], PGD [3], FGSM [4] and Carlini-Wagner (CW) [5]. This observation motivates the use of conformance in the attribution-neighborhood as a confidence metric.

The confidence of a model on a given input can be measured by sampling the neighborhood of the input and observing whether the model's output changes or conforms to the original output. But accurately estimating the conformance by sampling in the neighborhood of the input becomes

exponentially difficult with an increase in the dimension of the input [35]. We propose a novel approach which samples the high-dimensional neighborhood effectively using attributions provided by methods [36, 37, 38] developed recently to improve interpretability of DNNs. In particular, we adopt the use of Shapley values with roots in cooperative game theory for computing attribution. Approaches using Shapley values [38, 39] satisfy intuitively expected axiomatic properties over attributions [40]. Our attribution-based confidence metric is theoretically motivated by these axiomatic properties. The key idea is to use attributions for importance sampling in the neighborhood of an input. This importance sampling is more likely to select neighbors obtained by changing features with high attribution. The conformance of the machine learning model over these samples is computed as the fraction of neighbors for which the output of the model does not change. Figure 1.1 illustrates how an input that triggers incorrect responses in DNNs, such as an out-of-distribution sample or an adversarial example, does not conform in its neighborhood. Thus, the conformance of the model's prediction in the input's neighborhood, sampled using feature attributions, can be used as an effective measure of the confidence of the model on that input.

We make the following new contributions in the thesis.

- We propose a novel attribution-based confidence (ABC) metric computed by importance sampling in the neighborhood of a high-dimensional input using relative feature attributions, and estimating conformance of the model. It does not require access to training data or additional calibration. We mathematically motivate the proposed ABC metric using axioms on Shapley values.
- We empirically evaluate the ABC metric over MNIST and ImageNet datasets using (a) out-of-distribution data, (b) adversarial inputs generated using digital attacks such as FGSM, PGD, CW and DeepFool, and (c) physically-realizable adversarial patches and LaVAN attacks.

CHAPTER 2: RELATED WORK

The set of inputs to a machine learning system is significantly larger than the training set of the machine learning system. The set of inputs that the machine learning system has not been trained on can potentially confuse the machine learning system. This confusing input can be generated synthetically or obtained naturally. In this chapter, we discuss some already existing ways to generate synthetic images that fool image recognition systems. Synthetic images that fool the machine learning system are called adversarial images. We further discuss some datasets of natural images that confuse machine learning systems. These naturally existing images are classified as out-of-distribution images or natural adversarial images. Out-of-distribution images are images that are significantly different from the images that were used to train the machine learning system. Natural adversarial images are naturally occurring images that contain labels that the machine learning system was trained on but still is misclassified by the machine learning system. Finally, we discuss some methods to make the machine learning systems robust against these confusing inputs.

2.1 Synthetic Adversarial Images

Szegedy et al. were the first to show that synthetic adversarial images can easily fool deep neural networks [27]. They used box constrained L-BGFS optimization to solve the following optimization problem:

$$\text{Minimize } c|\boldsymbol{r}| + \text{loss}_f(\boldsymbol{x} + \boldsymbol{r}, y) \text{ subject to } \boldsymbol{x} + \boldsymbol{r} \in [0, 1]^m \tag{2.1}$$

Where the variables $\boldsymbol{x}$, y, $\boldsymbol{r}$, and c denote the original image, the target label, the generated per-

turbations, and the line search step size, respectively. The function $f : \mathbb{R}^m \rightarrow \{1 \cdots n\}$ is the classifier mapping an image to a label in $Y \in \{1 \cdots n\}$. The function $\text{loss}_f : \mathbb{R}^m \times Y \rightarrow \mathbb{R}^+$ is the loss which maps a pair of image and label to a positive loss. In the follow up to this paper, Goodfellow et al. proposed the fast gradient sign method (FGSM) [4]. They hypothesized that neural networks are too linear, and linear perturbations to input are enough to create adversarial images. They used the sign of the loss function to generate adversarial images. The perturbation $\boldsymbol{r}$ for this method is calculated using the following equation:

$$\boldsymbol{r} = \epsilon \text{sgn}(\nabla_x J(\boldsymbol{\theta}, \boldsymbol{x}, f(\boldsymbol{x}))) \tag{2.2}$$

Where the variables ϵ and θ are the stepsize and model parameters, respectively. Function J is the cross-entropy cost function used to train the neural network. This method can be used to generate adversarial images with a pre-specified target label or adversarial images without any pre-specified target label. The perturbation $\boldsymbol{r}$ required to generate an adversarial image with a fixed class y is calculated using the following equation: :

$$\boldsymbol{r} = -\epsilon \text{sgn}(\nabla_x J(\boldsymbol{\theta}, \boldsymbol{x}, y)) \tag{2.3}$$

Kurakin et al. proposed the basic iterative method (BIM) that extends the FGSM method by iteratively applying the FGSM equation and clipping the adversarial image after each iteration to maintain a fixed distance from the original image [41]. They used the following iterative equation to generate adversarial images:

$$\boldsymbol{x}_0^{adv} = \boldsymbol{x} \tag{2.4}$$

$$\boldsymbol{x}_{t+1}^{adv} = \Pi_{\boldsymbol{x}+\boldsymbol{S}} \left(\boldsymbol{x}_t^{adv} + \alpha \text{sgn}(\nabla_x J(\boldsymbol{\theta}, \boldsymbol{x}_t^{adv}, f(\boldsymbol{x}))) \right) \tag{2.5}$$

Where t is the iteration count, α is the step size count, and S is the set of allowed perturbation.

The function $\Pi_{\boldsymbol{x}+\boldsymbol{S}}$ is the projection function that projects an image so that the image perturbations belong to the set of S. Kurakin et al. generated adversarial images using both FGSM and BIM methods, printed it using a color printer and then re-acquired the color prints for classification with the machine learning system. They used the term "photo transformation" to describe this procedure. They showed that the adversarial properties of images persist through physical printouts of images. They observed that FGSM adversarial images are more persistent to photo transformation when compared to BIM adversarial images. They hypothesized that the low robustness of BIM adversarial images to photo transformation was due to the more subtle nature of the perturbations generated by BIM. Dong et al. incorporated momentum in BIM to improve the success rate of adversarial attacks [42]. This method called the moment iterative fast gradient sign method (MI-FGSM) uses the history of past gradient directions to predict new adversarial images. The equation to calculate MI-FGSM adversarial images with decay μ and accumulated gradient $\boldsymbol{g}$ is as follows:

$$\boldsymbol{x}_0^{adv} = \boldsymbol{x} \tag{2.6}$$

$$\boldsymbol{g}_0 = 0 \tag{2.7}$$

$$\boldsymbol{g}_{t+1} = \mu \boldsymbol{g}_t + \frac{\nabla_x J(\boldsymbol{\theta}, \boldsymbol{x}_t^{adv}, f(\boldsymbol{x}))}{||\nabla_x J(\boldsymbol{\theta}, \boldsymbol{x}_t^{adv}, f(\boldsymbol{x}))||_1} \tag{2.8}$$

$$\boldsymbol{x}_{t+1}^{adv} = \boldsymbol{x}_t^{adv} + \alpha \text{sgn}(\boldsymbol{g}_{t+1}) \tag{2.9}$$

Madry et al. introduced the projected gradient descent (PGD) method to generate multiple adversarial images for adversarial training; they chose a random initial image within $\boldsymbol{S}$ to start the BIM iteration [3]. The equation to calculate PGD adversarial images is as follows:

$$\boldsymbol{x}_0^{adv} = \boldsymbol{x} + \boldsymbol{s} \text{ where } \boldsymbol{s} \in \boldsymbol{S} \tag{2.10}$$

$$\boldsymbol{x}_{t+1}^{adv} = \Pi_{\boldsymbol{x}+\boldsymbol{S}} \left(\boldsymbol{x}_t^{adv} + \alpha \text{sgn}(\nabla_x J(\boldsymbol{\theta}, \boldsymbol{x}_t^{adv}, f(\boldsymbol{x}))) \right) \tag{2.11}$$

Moosavi-Dezfooli et al. proposed the DeepFool adversarial image generation algorithm [2]. They assume that the deep neural network classifier is linear and used this assumption to calculate the minimum perturbation required to generate adversarial images. They compared their method to FGSM and show that the average perturbation required to generate adversarial images is up to 5 times lower.

Our experimental studies use the state-of-the-art DeepFool 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 Strategies to Make Robust AI Systems

Virtual adversarial training [43] uses a KL-divergence based robustness metric of a model against local perturbation around a data point 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 than random noise perturbations [44, 45], specifically in the context of linear classifiers. The observation of adversarial attacks in the context of high-dimensional data has been explained by 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 [46], thereby making it harder to generate adversarial examples for such robust networks. They report up to 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 statistical method to detect adversarial examples has been proposed in [47], this method has been shown to work on MNIST, DREBIN and MicroRNA data with attack vectors chosen using FGSM, JSMA, SVM, and DT attacks. An impressive performance of about 100% detection in certain tests has 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 the ImageNet data set that we are using for our method. Another method for detecting adversarial perturbations has been presented in [19]. 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%.

2.3 Explainable AI

There has been a continued interest in interpretability of results produced by DNNs. Attribution methods assign a positive or negative value (attribution) to features of input that contribute to the

decisions made by DNNs [48, 49]. Various methods to calculate attributions have been proposed. Input-perturbation based methods occlude or perturb the input and note the change in activation of later layers to calculate attributions [50, 51]. Gradient-based methods use the gradient of the predictor function with respect to the input to calculate attributions [36, 37, 38]. Shrikumar et al. proposed the DeepLIFT method; they derived a reference activation map of individual neurons from a reference input and then used the difference between the current activation and the reference activation to calculate the contributing score of individual neurons [52]. The contributing score of individual neurons is backpropagated to the input to obtain attributions. Lundberg et al. proposed the DeepSHAP method; they used the backpropagation technique of DeepLIFT to recursively backpropagate the shapely value of individual neurons to obtain attributions [53].

CHAPTER 3: SATYA - 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 3.1. 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 3.1, 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.

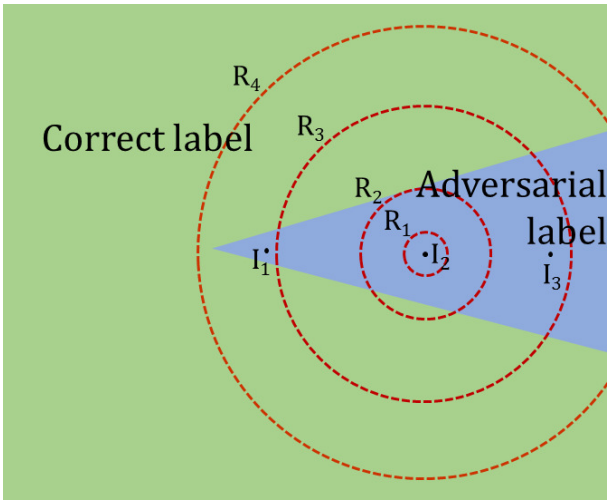


Figure 3.1: The correct non-adversarial label space is shown in green, while the adversarial space is shown in blue. The adversarial images I_1, I_2 and I_3 are located at different locations inside the adversarial space. The dotted lines denote hyphersphere with varying radii drawn with the image I_2 as the center. Sampling on a hypherspheres of radius R_1 will give incorrect results while sampling on a hypherphere of radius R_4 will give correct results.

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 [54] to generate uniformly distributed points on the surface of a d-dimensional hypherspheres. 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 μ = 0, σ = 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 hypherspheres with radius R is given by Equation 3.1. This equation is implemented by the function $\mathcal{N}$ in Algorithm 1.

$$P_i = \frac{R}{\|G_i\|} G_i^T \tag{3.1}$$

If an adversarial image is deep inside the adversarial space, sampling on low radius hyphersphere will only give adversarial samples as shown by R_1 and R_2 in Figure 3.1. An image like I_3 that is further inside the adversarial space will require a larger hyphersphere 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 3.1. 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.

Table 3.1: 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.

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

One trend that we observe in Table 3.1 is the decline in accuracy as the radius of the hypersphere increases beyond 1500 units. Hyphersphere with higher radii have larger volume and can accommodate samples of multiple labels as shown in Figure 3.2. Multiple labels on the hyphersphere can decrease the chance of 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 hypher-sphere 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.

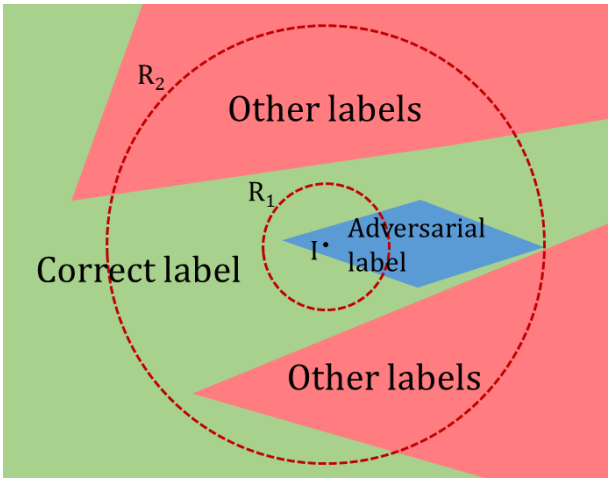


Figure 3.2: Sampling around the image I. A low radius R_1 will give correct results, while a very high radius R_2 is likely to give incorrect results.

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 3.3. The location of I_{mid} reduces the need for finding the optimal hypersphere 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.

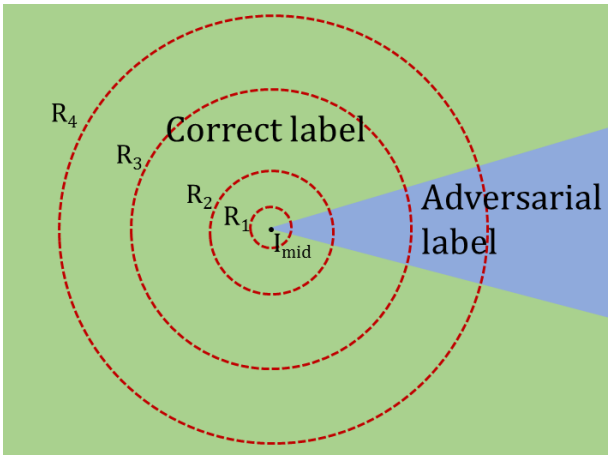


Figure 3.3: Sampling around the image I_{mid} at the border of the space of correct non-adversarial label and adversarial label. Sampling on most hyphersphere radii will give correct results.

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 3.2. 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 3.1 the standard deviation for CaffeNet is 29.98 whereas the standard deviation in Table 3.2 is 10.16. Similarly for GoogLeNet, the standard deviation in Table 3.1 is 37.05 where as it is 4.34 for Table 3.2. This simple metric shows us that the results obtained from $\mathcal{SATYA}$ is more resilient to variations in the location of the adversarial image.

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.

Table 3.2: Sampling the neighborhood of images at varying sampling radii and percentage of those images classified correctly for 1000 images.

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 %

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

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 [16]. 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 5.3 and 5.4, 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 5.5 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 $\mathcal{SATYA}$.

CHAPTER 4: ATTRIBUTION-BASED CONFIDENCE METRIC

Our proposed confidence metric (ABC) is closely related to the literature on confidence metrics, attribution methods and techniques for adversarial attacks and defenses for DNNs.

Confidence metrics: The need for confidence metric to reflect the uncertainty in the output of machine learning models was recognized very early in the literature [31, 30]. The high accuracy but brittleness of deep learning models has revived interest in defining confidence metrics that reflect the accuracy of the model. DNNs are not well-calibrated [23, 32, 33] and so, the straightforward use of logit layer before softmax as a confidence measure is not reliable. Several post-processing based confidence metrics have been proposed in the literature which can be grouped into three classes:

- Calibration models trained using held-out validation set: [31] proposed a parametric approach where the logits are used as features to learn a calibration model from a held-out validation set. An example calibration model is $q_i = \max_k \sigma(W z_i + b)^k$ where z_i are the logits, σ is the standard sigmoid function, k denotes a class, $(\cdot)^k$ denotes the k-th element of a vector, and W, b are parameters [28]. Temperature scaling is a special case of Platt scaling with a single parameter. Nonparametric learning of calibration models from held-out validation data has been also proposed in the form of histogram binning [55], isotonic regression [56] and Bayesian binning [57].
- Model ensemble approaches: [58] use ensembles of networks to obtain uncertainty estimates. Bayesian neural networks [30, 59] return a probability distribution over outputs as an alternative way to represent model uncertainty. Sampling models by dropping nodes in a DNN has been shown to estimate probability distribution over all models [60].

- Training-set-based uncertainty estimation: [33] compute the trust score on deeper layers of a DNN than input to avoid the high-dimensionality of inputs. They propose a trust score that measures the conformance between the classifer and a modified nearest-neighbor classifier on the testing example. [32] use k-nearest neighbors regression using training set on the intermediate representations of the network which showed enhanced robustness to adversarial attacks and leads to better calibrated uncertainty estimates.

In contrast to these approaches, ABC metric only needs the trained model at inference time, and does not require training data, separate held-out validation data or training a model ensemble.

DNN robustness, adversarial inputs and defense: Szegedy et al. used L-BFGS method to generate adversarial examples [27]. Goodfellow et al. proposed a fast gradient sign method (FGSM) to generate faster adversarial images as compared to the L-BFGS method; this method performed only a one-step gradient update at each pixel along the direction of the gradient's sign [4]. Rozsa et al. replaced the sign of the gradients with raw gradients [61]. Dong et al. applied momentum to FGSM [42], and Kurakin et al. further extended FGSM to a targeted attack [41]. A number of adversarial example generation techniques [2, 3, 5] are now available in tools such as Cleverhans [62] which we use in our experiments. In addition to digital attacks, we also study how the proposed ABC metric measures uncertainty on physically realizable attacks in form of patches or stickers [1, 63]. Further, the adversarial examples have been shown to transfer [64, 65] across models making them agnostic to availability of model parameters and effective against even ensemble approaches. Efforts over the last few years to defend against adversarial attacks have met with limited success. While approaches such as logit pairing [66], defensive distillation [67], manifold-based defense [68, 69], and adversarial training methods that exploit knowledge of the specific attack [70], have shown effectiveness against particular attack methods, more principled techniques such as robust optimization [71, 72, 3] and formal methods [73] are limited to pertur-

bations with bounded L_p norm. Schott et al. present an insightful study of the state of the art on attacks and defenses [74], and proposes an analysis by synthesis defense. This arms race between attacks and defenses are not limited to computer vision, but extend to audio and natural language processing. Schonner et al. were able to fool the state-of-the-art speech recognition system Kaldi using audio noise that were barely distinguishable to a human listener [75]. Qin et al. then improved the construction of adversarial examples to create audio noise that was imperceptible to humans [76].

Our approach to addressing the challenge of DNN's susceptibility to adversarial examples using ABC metric differs from the majority of prior work in that it measures confidence of a model's prediction to characterize the credibility of a DNN on a given input instead of attempting to classify all legitimate and malicious inputs correctly or make particular adversarial strategies fail.

Model interpretability and attribution methods: A number of explanation techniques [39, 38, 48, 49, 77, 78] have been recently proposed in the literature that either find a complete logical explanation or just the relevant features or assign quantitative importance (attributions) to input features for a given model decision. Many of these methods are based on the gradient of the predictor function with respect to the input [36, 37, 38]. Different attribution methods are compared in [79]. The sensitivity of these attributions to perturbations in the input are studied in [80], and indicate the potential of adversarial attacks on the proposed and other existing confidence metrics. But attacking the model and its confidence measure is more difficult than just attacking the model. This motivates the use of ABC metric to measure confidence. The brittleness of learning has been related to anti-causal direction of learning [81]. We observe that the wrong decisions for out-of-distribution and adversarial inputs often hinge on a relatively small concentrated set of high attribution features, and thus, mutating these features and generating samples in the neighborhood of the original input is an effective top-down inference that is robust to adversarial attacks. In recent work [82], we have demonstrated the use of attributions for detecting adversarial examples.

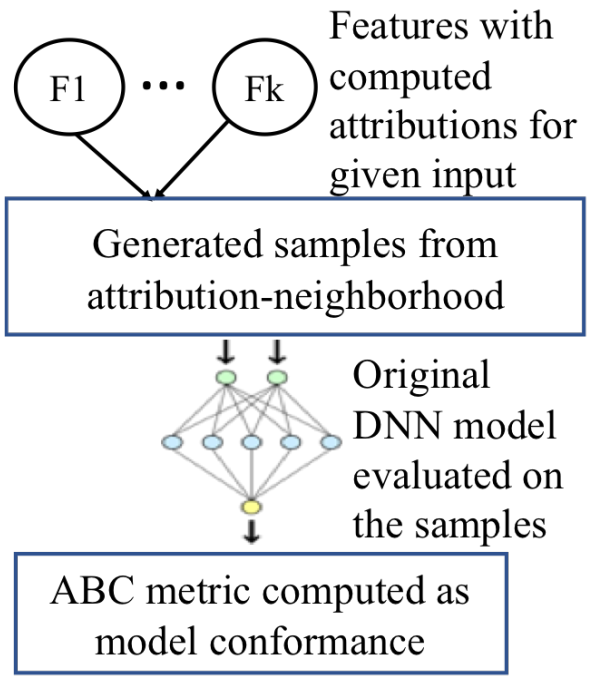


Figure 4.1: ABC complements the bottom-up inference of the original DNN model with the top-down sample generation and conformance estimation.

4.1 ABC Metric Motivation

Motivation: The proposed ABC metric is motivated by Dual Process Theory [83, 84] and Kahneman's decomposition [85] of cognition into System 1, or the intuitive system, and System 2, or the deliberate system. The original DNN model represents the bottom-up System 1, and the ABC metric computation is a deliberative top-down System 2 that uses the attributions in System 1 to generate new samples in the neighborhood of the original input. [81] argue that causal mechanisms are typically continuous but most learning problems such as classification are anti-causal. The lack of resilience to adversarial examples is hypothesized to be the result of learning in an anti-causal direction. Combining both the anti-causal System 1 DNN model and the attribution-driven System 2 that computes the ABC metric creates a relatively more resilient cognition model. ABC metric

uses the attribution over features for the decision of a machine learning model on a given input to construct a generator that can sample the attribution-neighborhood of the input and observe the conformance of the model in this neighborhood. While learning is still in the anti-causal direction, ABC adds causal deliberative System 2 that reasons in the forward generative direction to evaluate the conformance of the model.

Computational challenge: The computation of ABC metric of an ML model on an input requires accurately determining conformance by sampling in the neighborhood of high-dimensional inputs. This can be addressed by sampling over lower-dimensional intermediate or output layers of a DNN [32, 30], or relying on topological and manifold-based data analysis [33]. But these methods require training data that may not always be available at inference time. ABC addresses this challenge by biasing our sampling using the quantitative attribution obtained via Shapley values [38]. Deep learning models demonstrate a concentration of features, that is, few features have relatively very high attributions for any decision. Figure 4.2 illustrates feature concentration for ImageNet where the attribution is computed via Integrated Gradients [38]. Sampling over low attribution features will likely lead to no change in the label. Low attribution indicates that model is *equivariant* along these features. By focusing on high attribution features during sampling, our method can efficiently sample even high-dimensional input spaces to obtain a conservative estimate of the confidence score.

ABC uses feature attributions for dimensionality reduction followed by importance sampling in the reduced-dimensional neighborhood of the input to estimate DNN model's conformance. But unlike typical principal component analysis techniques that search for globally important features, we identify features that are locally relevant for the given input. This enables our approach to conservatively approximate conformance measure of a model in even high-dimensional input's neighborhood and, thus, efficiently compute the ABC confidence metric of the model on the input.

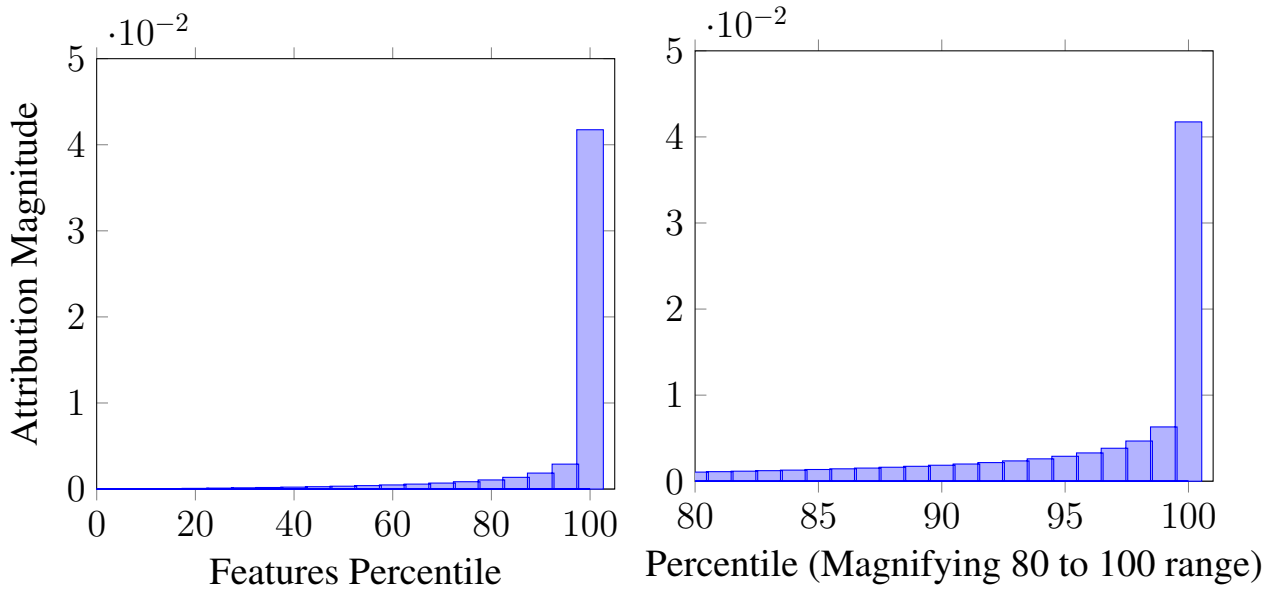


Figure 4.2: Attributions concentrate over few features in ImageNet.

4.2 ABC Algorithm

In this section, we theoretically motivate the proposed ABC metric and present an algorithm for its computation. Attribution methods using Shapley values often employ the notion of a baseline input $\mathbf{x}^b$; for example, the all dark image can be the baseline for images. The baseline can also be a set of random inputs where attribution is computed as an expected value. Let the attribution for the j-th feature and output label i be $\mathcal{A}_j^i(\mathbf{x})$. The attribution for the j-th input feature depends on the complete input x and not just x_j. The treatment for each logit is similar, and so, we drop the logit/class and denote the network output simply as $\mathcal{F}(\cdot)$ and attribution as $\mathcal{A}_j(\mathbf{x})$. For simplicity, we use the baseline input $\mathbf{x}^b = 0$ for computing attributions. We make the following two assumptions on the DNN model and the attributions, which reflect the fact that the model is well-trained and the attribution method is well-founded:

- The attribution is dominated by the linear term. This is also an assumption made by at-

tribution methods based on Shapley values such as Integrated Gradient [38] which define attribution as the path integral of the gradients of the DNN output with respect to that feature along the path from the baseline $\mathbf{x}^b$ to the input $\mathbf{x}$, that is,

$$\mathcal{A}_j^i(\mathbf{x}) = (\mathbf{x}_j - \mathbf{x}_j^b) \times \int_{\alpha=0}^{1} \partial_j \mathcal{F}^i(\mathbf{x}^b + \alpha(\mathbf{x} - \mathbf{x}^b)) d\alpha \tag{4.1}$$

where the gradient of i-th logit output of the model along the j-th feature is denoted by $\partial_j \mathcal{F}^i(\cdot)$.

- Attributions are complete i.e. the following is true for any input $\mathbf{x}$ and the baseline input $\mathbf{x}^b$:

$$\mathcal{F}(\mathbf{x}) - \mathcal{F}(\mathbf{x}^b) = \sum_{k=1}^{n} \mathcal{A}_k(\mathbf{x}) \text{ where } \mathbf{x} \text{ has } n \text{ features.} \tag{4.2}$$

  Shapley value methods such as Integrated Gradient and DeepShap [38, 39] satisfy this axiom too.

We first establish a relationship between attributions and the sensitivity of the model's output to change in an input feature. This is useful in attribution-based dimensionality reduction of a high-dimensional $\mathbf{x}$ and defining its attribution-neighborhood.

Theorem 1 *The sensitivity of the output $\mathcal{F}(\mathbf{x})$ with respect to an input feature $\mathbf{x}_j$ in the neighborhood of $\mathbf{x}$ is approximately the ratio of the attribution $\mathcal{A}_j(\mathbf{x})$ to the value of that feature $\mathbf{x}_j$, that is, $\frac{\mathcal{A}_j(\mathbf{x})}{\mathbf{x}_j}$.*

Given an input $\mathbf{x}$ and its neighbor $\mathbf{x}' = \mathbf{x} + \delta\mathbf{x}$, we can use Taylor series expansion to express the output $\mathcal{F}(\mathbf{x}')$ as:

$$\mathcal{F}(\mathbf{x}') = \mathcal{F}(\mathbf{x}) + \sum_{k=1}^{n} (\frac{\partial \mathcal{F}(\mathbf{x})}{\partial \mathbf{x}_k} \delta\mathbf{x}_k) + \max_{k=1,\ldots,n} O(\delta\mathbf{x}_k^2) \ . \tag{4.3}$$

Assuming the completeness of attribution over the features of the input $\mathbf{x}'$, we obtain the following:

$$\mathcal{F}(\mathbf{x}') - \mathcal{F}(\mathbf{x}^b) = \sum_{k=1}^{n} \mathcal{A}_k(\mathbf{x}') \ . \tag{4.4}$$

Subtracting Equation 4.2 from 4.4, we can eliminate the baseline input $\mathbf{x}^b$ and use Taylor series to obtain:

$$\begin{aligned} \mathcal{F}(\mathbf{x}') - \mathcal{F}(\mathbf{x}) &= \sum_{k=1}^{n} \left(\mathcal{A}_k(\mathbf{x}') - \mathcal{A}_k(\mathbf{x})\right) \\ &= \sum_{k=1}^{n} \left(\frac{\partial \mathcal{A}_k(\mathbf{x})}{\partial \mathbf{x}_k} \delta \mathbf{x}_k \right) + \max_{k=1,\ldots,n} O(\delta \mathbf{x}_k^2) \end{aligned} \ . \tag{4.5}$$

While computing conformance, we sample only neighbors of the input and so we can drop the higher order terms in both Equations 4.5 and 4.3. These equations hold for all neighbors including those which differ in only one of the features $\mathbf{x}_j$ and so, we can conclude that the sensitivity of the model with respect to a feature $\mathbf{x}_j$ is $\frac{\partial \mathcal{A}_j(\mathbf{x})}{\partial \mathbf{x}_j}$.

Taking the derivative of Equation 4.1 on both sides with respect to feature $\mathbf{x}_j$ and ignoring the non-linear attribution terms, we obtain the following:

$$\begin{aligned} \frac{\partial \mathcal{A}_j(\mathbf{x})}{\partial \mathbf{x}_j} &= \int_{\alpha=0}^{1} \frac{\partial \mathcal{F}(\mathbf{x^b} + \alpha(\mathbf{x} - \mathbf{x}^b))}{\partial \mathbf{x}_j} d\alpha + \mathbf{x}_j \frac{\partial}{\partial \mathbf{x}_j} \left(\int_{\alpha=0}^{1} \frac{\partial \mathcal{F}(\mathbf{x}^b + \alpha(\mathbf{x} - \mathbf{x}^b))}{\partial \mathbf{x}_j} d\alpha \right) \\ &= \int_{\alpha=0}^{1} \frac{\partial \mathcal{F}(\mathbf{x^b} + \alpha(\mathbf{x} - \mathbf{x}^b))}{\partial \mathbf{x}_j} d\alpha + \mathbf{x}_j \left(\int_{\alpha=0}^{1} \frac{\partial^2 \mathcal{F}(\mathbf{x}^b + \alpha(\mathbf{x} - \mathbf{x}^b))}{\partial \mathbf{x}_j^2} d\alpha \right) \\ &\approx \int_{\alpha=0}^{1} \frac{\partial \mathcal{F}(\mathbf{x^b} + \alpha(\mathbf{x} - \mathbf{x}^b))}{\partial \mathbf{x}_j} d\alpha = \frac{\mathcal{A}_j(\mathbf{x})}{\mathbf{x}_j} \text{ from Eqn. 4.1 with baseline feature } \mathbf{x}_j^b = 0. \end{aligned} \tag{4.6}$$

Thus, we have shown that the sensitivity of the model's output with respect to an input feature in the neighborhood of the input is approximately given by the ratio of the attribution for that input feature

to the value of that feature. Note that $\frac{\mathcal{A}_j(\mathbf{x})}{\mathbf{x}_j}$ does not vanish even when the traditional sensitivity given by $\frac{\partial \mathcal{F}(\mathbf{x})}{\partial \mathbf{x}_j}$ vanishes exploiting the non-saturating nature of Shapley value attributions. The model is almost equivariant with respect to features with low $\mathcal{A}_j(\mathbf{x})/\mathbf{x}_j$ and, thus, the attribution-neighborhood is constructed by mutating features with high $\mathcal{A}_j(\mathbf{x})/\mathbf{x}_j$. The overall algorithm for computing ABC metric of a DNN model on an input is as follows.

Algorithm 2 Evaluate ABC confidence metric $c(\mathcal{F}, \mathbf{x})$ of machine learning model $\mathcal{F}$ on input $\mathbf{x}$

Input: Model $\mathcal{F}$, Input $\mathbf{x}$ with features $\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_n$, Sample size S
Output: ABC metric $c(\mathcal{F}, \mathbf{x})$

$\mathcal{A}_1, \ldots \mathcal{A}_n \leftarrow$ Attributions of features $\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_n$ from input $\mathbf{x}$
$i \leftarrow \mathcal{F}(\mathbf{x})$ ▷*Obtain model prediction*
for $j = 1$ to n **do**
    $P(\mathbf{x}_j) \leftarrow \frac{\mathcal{A}_j/\mathbf{x}_j}{\sum_{k=1}^{n} \mathcal{A}_k/\mathbf{x}_k}$
end for
Generate S samples by mutating feature $\mathbf{x}_j$ of input $\mathbf{x}$ to baseline $\mathbf{x}_j^b$ with probability $P(\mathbf{x}_j)$
Obtain the output of the model on the S samples.
$c(\mathcal{F}, \mathbf{x}) \leftarrow S_{conform}/S$ where model's output on $S_{conform}$ samples is i
return $c(\mathcal{F}, \mathbf{x})$ as confidence metric (ABC) of prediction by the model $\mathcal{F}$ on the input $\mathbf{x}$

CHAPTER 5: SATYA - EXPERIMENTAL RESULTS

We evaluated $\mathcal{SATYA}$ on 50,000 images from the ILSVRC2013 training dataset [86] for both the CaffeNet [17] and the GoogLeNet [18] deep learning frameworks. Adversarial versions of these images were created using the state-of-the-art DeepFool [2] 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?

5.1 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 1.1. 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 5.1. We can see that an accuracy gain of 2.12% for CaffeNet and 1.14% for googleNet was achieved using $\mathcal{SATYA}$.

Table 5.1: 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.

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

5.2 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 5.2. 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.

Table 5.2: $\mathcal{SATYA}$ 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.

	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%

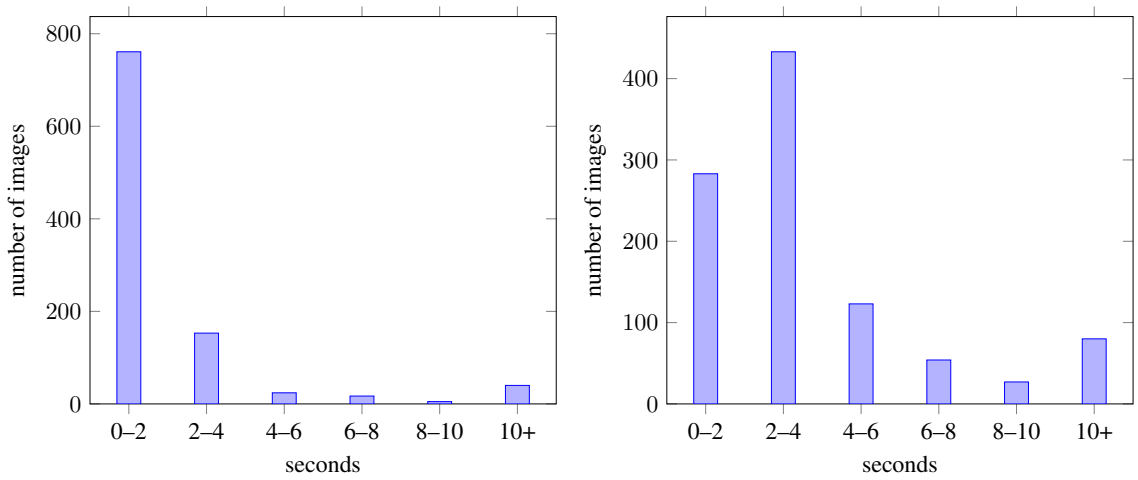
5.3 Runtime Performance

The time required by $\mathcal{SATYA}$ to analyze both original and adversarial images for CaffeNet is shown in Figure 5.1 and for GoogLeNet is shown in Figure 5.2. 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 runtime performance of the underlying classifier by using massively parallel calls from $\mathcal{SATYA}$ to the underlying classifier such as CaffeNet or GoogLeNet.

$\mathcal{SATYA}$ 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.

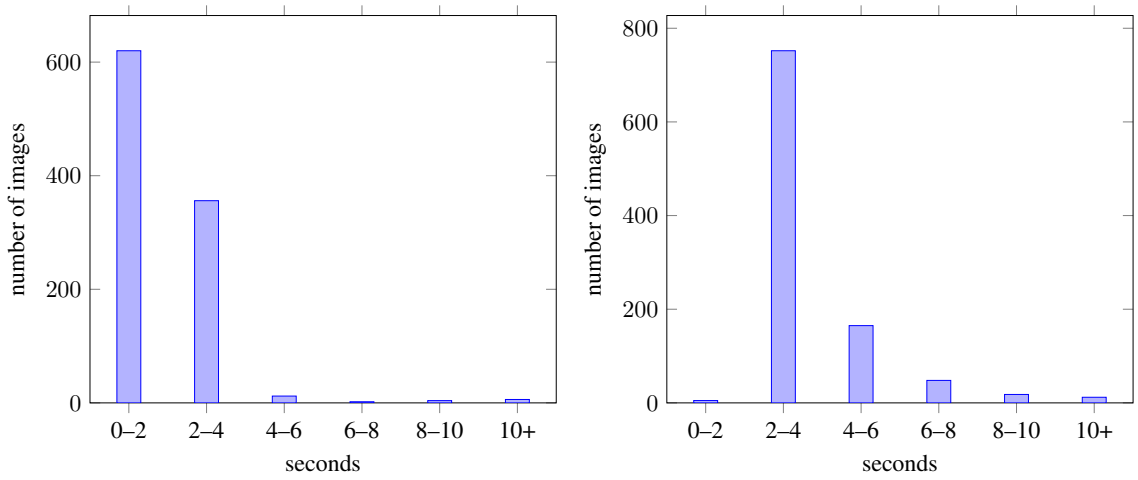
The performance of $\mathcal{SATYA}$ on images from the GoogLeNet is qualitatively similar to the results on CaffeNet. About 98% of the adversarial images and 75% of the original images can be analyzed

within 4 seconds. The worst case runtime of $\mathcal{SATYA}$ on adversarial images is 27 seconds for GoogLeNet.



(a) Adversarial Images (b) Original Images

Figure 5.1: Time taken by $\mathcal{SATYA}$ for predicting the label of adversarial and ordinary images used in CaffeNet.

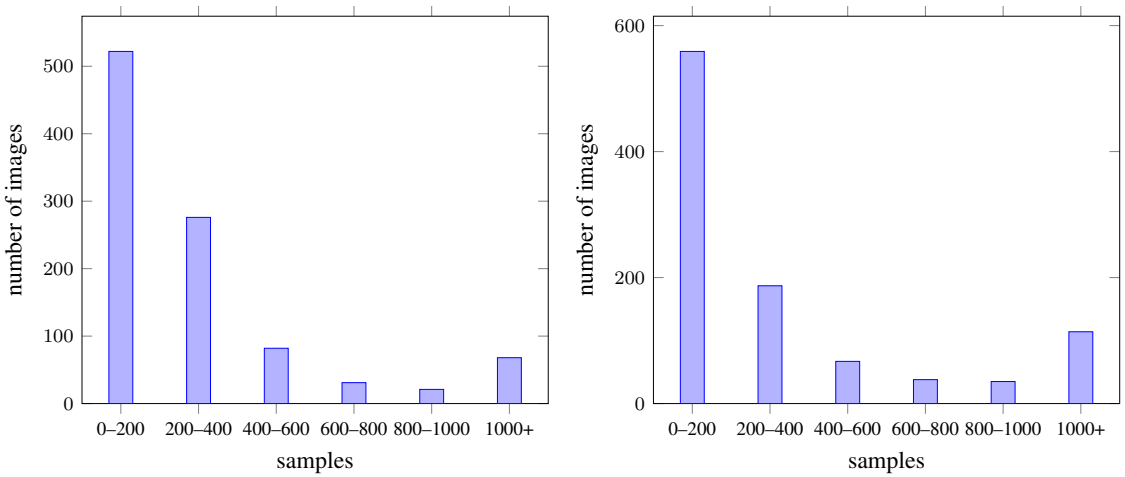


(a) Adversarial Images (b) Original Images

Figure 5.2: 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.

5.4 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 5.3, 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.

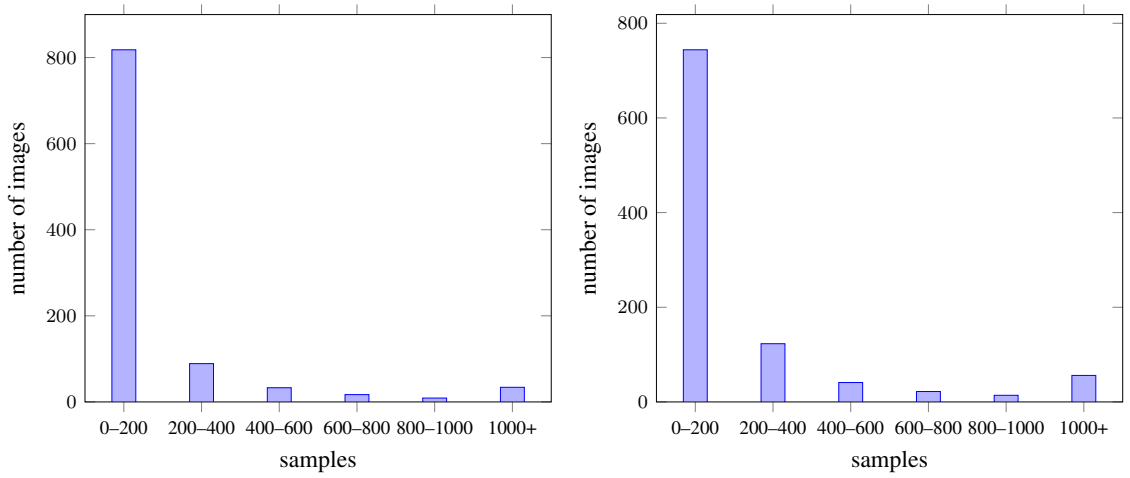


(a) Perturbed Images (b) Original Images

Figure 5.3: Number of samples tested to determine the label of an original as well as an adversarial image for CaffeNet.

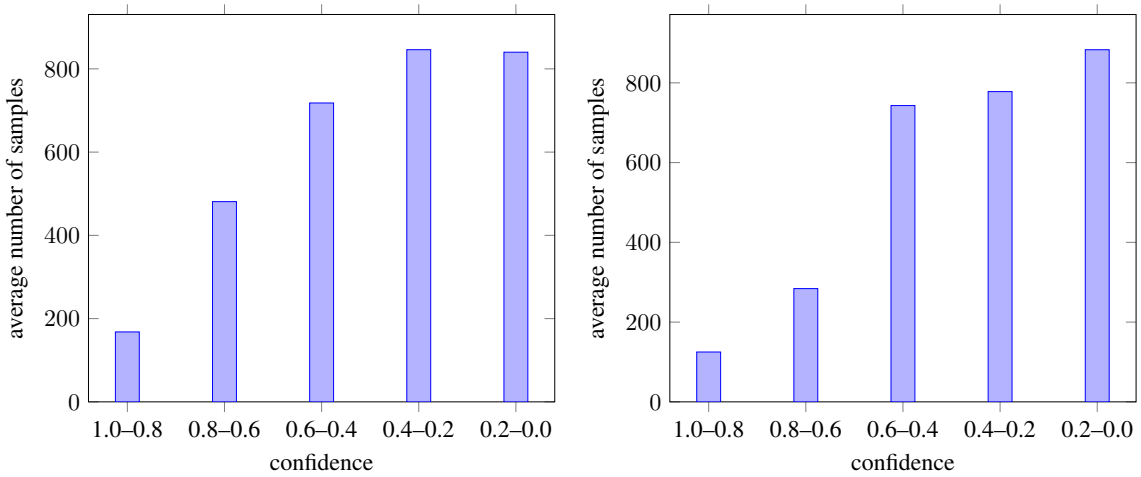
Figure 5.4 shows the number of samples required by original and adversarial images for the GoogLeNet deep learning framework. About 80% of the perturbed images and more than 75% of the original images were analyzed by sampling fewer than 200 samples. In the light of this variation in number of samples for a small fraction of the images, a natural question that arises is the source of this variability. Using Figures 5.5 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.



(a) Adversarial Images (b) Original Images

Figure 5.4: Number of samples tested to determine the label of an original as well as an adversarial image for GoogLeNet.



(a) CaffeNet (b) GoogLeNet

Figure 5.5: 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.

CHAPTER 6: ABC - EXPERIMENTAL RESULTS

We evaluate the attribition-based confidence metric on out-of-distribution data and adversarial attacks. All experiments were conducted on a 8 core Intel®Core™i9-9900K 3.60GHz CPU with NVIDIA Titan RTX Graphics and 32 GB RAM.

6.1 Out-of-distribution Images

We computed ABC metric for MNIST, MNIST [87] with rotation and background, notMNIST [88], and FashionMNIST [89] datasets. The DNN model was trained on MNIST images. Input compatible images such as rotated MNIST, notMNIST and FashionMNIST act as out-of-distribution datasets for the DNN model. In our experiments we observed that the average ABC metric of MNIST dataset is higher than the average ABC metric of out-of-distribution datasets. Only19% of the FashionMNIST dataset and 26% of the notMNIST dataset have a confidence higher than 0.85 while 70% of MNIST dataset had confidence higher than 0.85. The results of the experiments are shown in Figure 6.1

We compare the predicted accuracy vs confidence below to evaluate how well the attribution-based confidence metric reflects the reduced accuracy on the rotated MNIST dataset with a background image [90]. The dataset has MNIST images randomly rotated by 0 to 2π, and with a randomly selected black/white background image. The accuracy and confidence of the model drops with increase in rotation angle (from 0 to 50 degrees) and decrease in accuracy as illustrated in Figure 6.2. The three examples show how the ABC metric reflects the confusability of inputs. We compare the ABC metric with a trained calibrated scaling model in Figure 6.3.

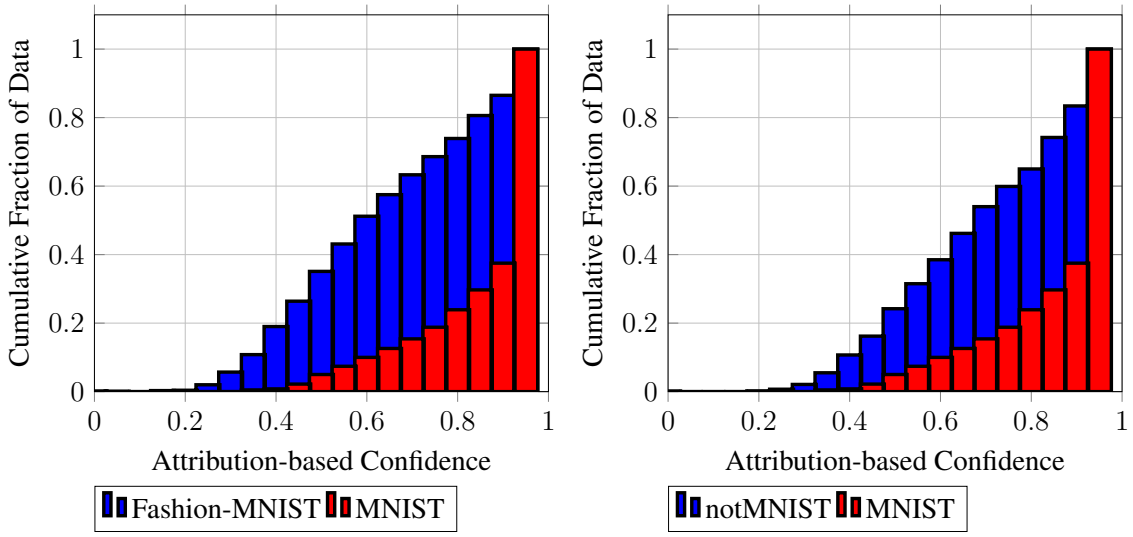


Figure 6.1: Cumulative data fraction vs. ABC for FashionMNIST and nonMNIST compared with MNIST. Only 19% of the FashionMNIST dataset and 26% of the notMNIST dataset have a confidence higher than 0.85 while 70% of MNIST dataset had confidence higher than 0.85.

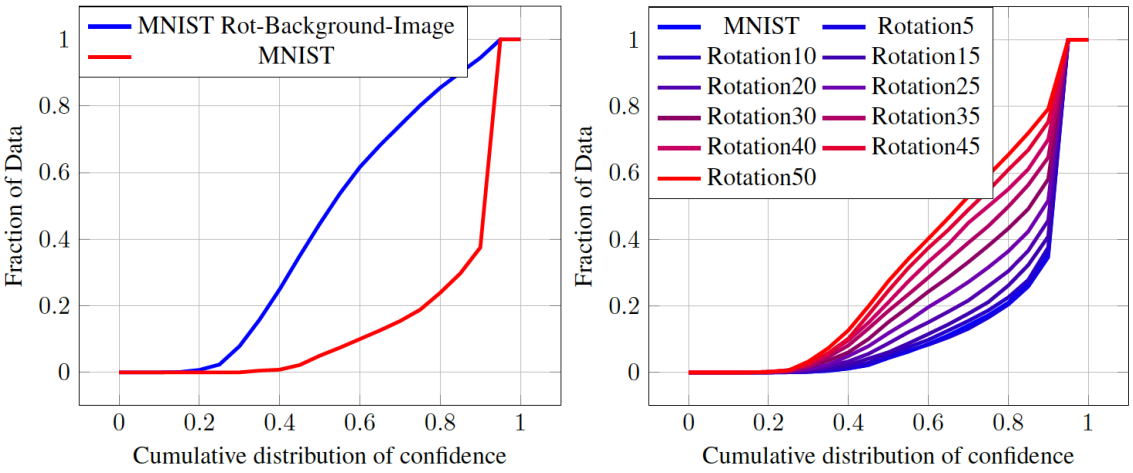


Figure 6.2: (left) Comparison of ABC metric for rotated-background-MNIST and MNIST. Rotated-background-MNIST has lower ABC confidence than MNIST. (right) ABC metric of rotated-MNIST at different angles ranging from 0 to 50 degrees. The accuracy and confidence of the model drops with increase in rotation angle.

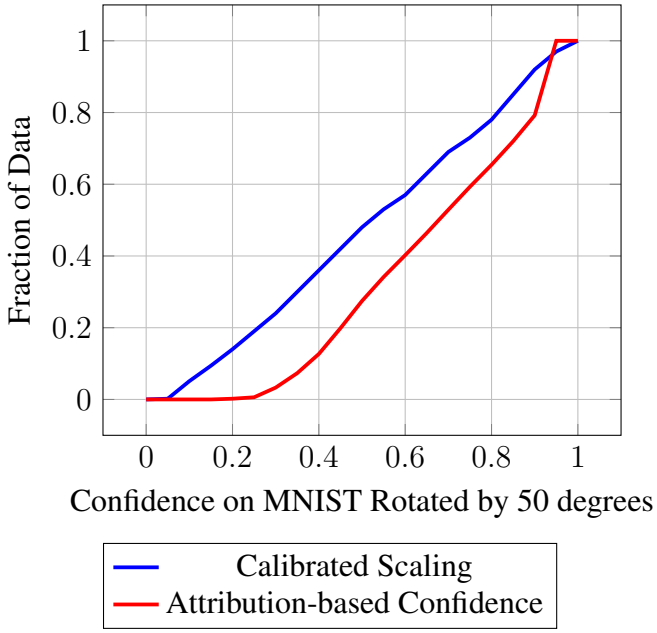


Figure 6.3: Comparison of attribution-based confidence metric with calibrated Platt scaling model for MNIST dataset rotated by 50 degrees.

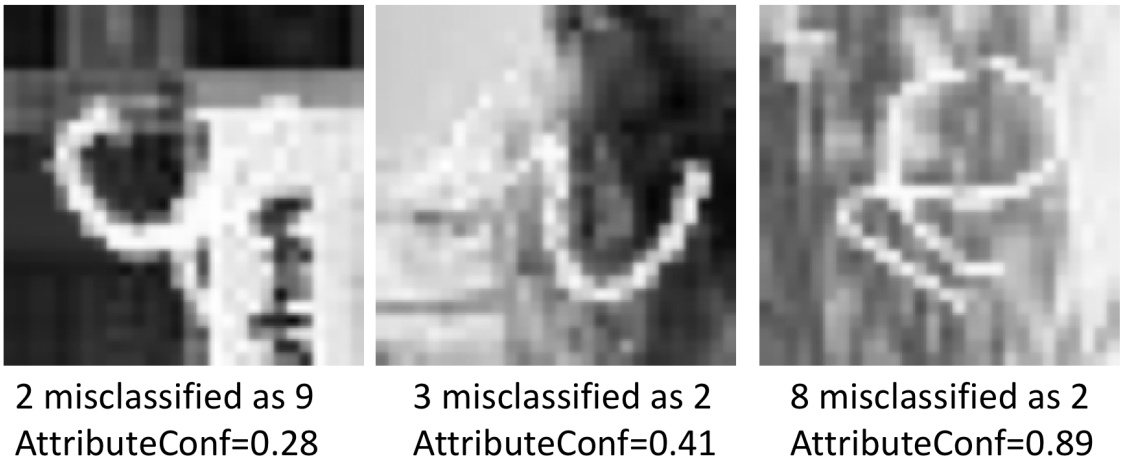


Figure 6.4: Selected examples from rotated-background-MNIST with confidence showing quantitative analysis of ABC metric showing connection between confidence and interpretability. Confusing images have lower ABC values.

6.2 Adversarial Images

We calculated the ABC metric for both digital and physical adversarial images. Figure 6.5 illustrates how the decreased ABC metric reflects the decrease in accuracy on MNIST adversarial examples. The accuracy with PGD attack drops close to zero, and so, we only plot the fraction of data with different confidence levels. We calculated ImageNet adversarial images using FGSM, PGD, CW, and DeepFool attacks. Figure 6.6 illustrates how the ABC metric reflects the decrease in accuracy under adversarial attack.

We applied physically realizable adversarial patch [1] and LaVAN [63] attacks on 1000 images from ImageNet. For the adversarial patch attack, we used a patch size of 25%for two patch types: banana and toaster. For LaVAN, we used a baseball patch of size 50×50 pixels. Figure 6.7 illustrates how the ABC metric is low for most of the adversarial examples reflecting the decrease in the accuracy of the model.

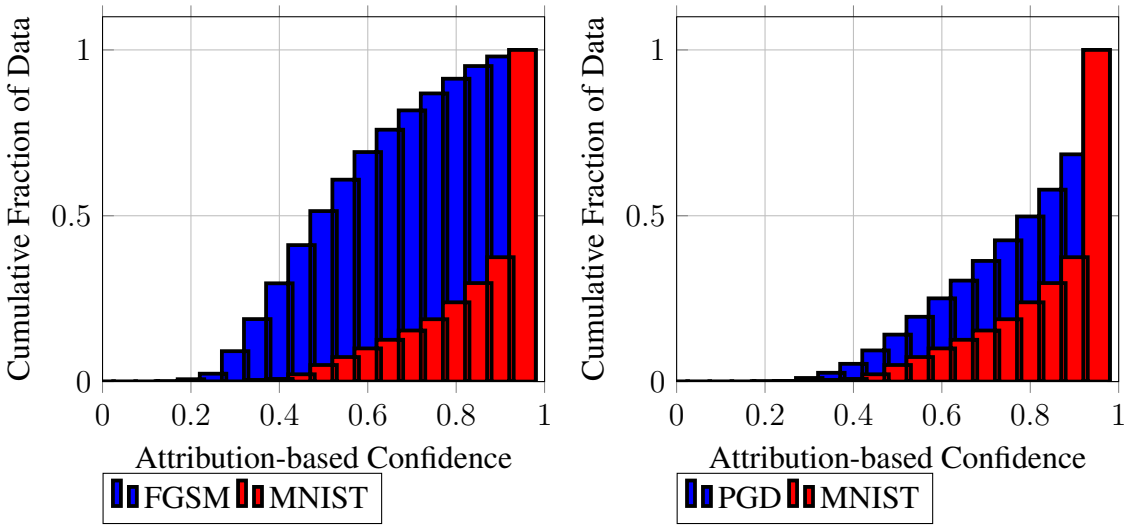


Figure 6.5: ABC metric for FGSM and PGD attacks on MNIST.

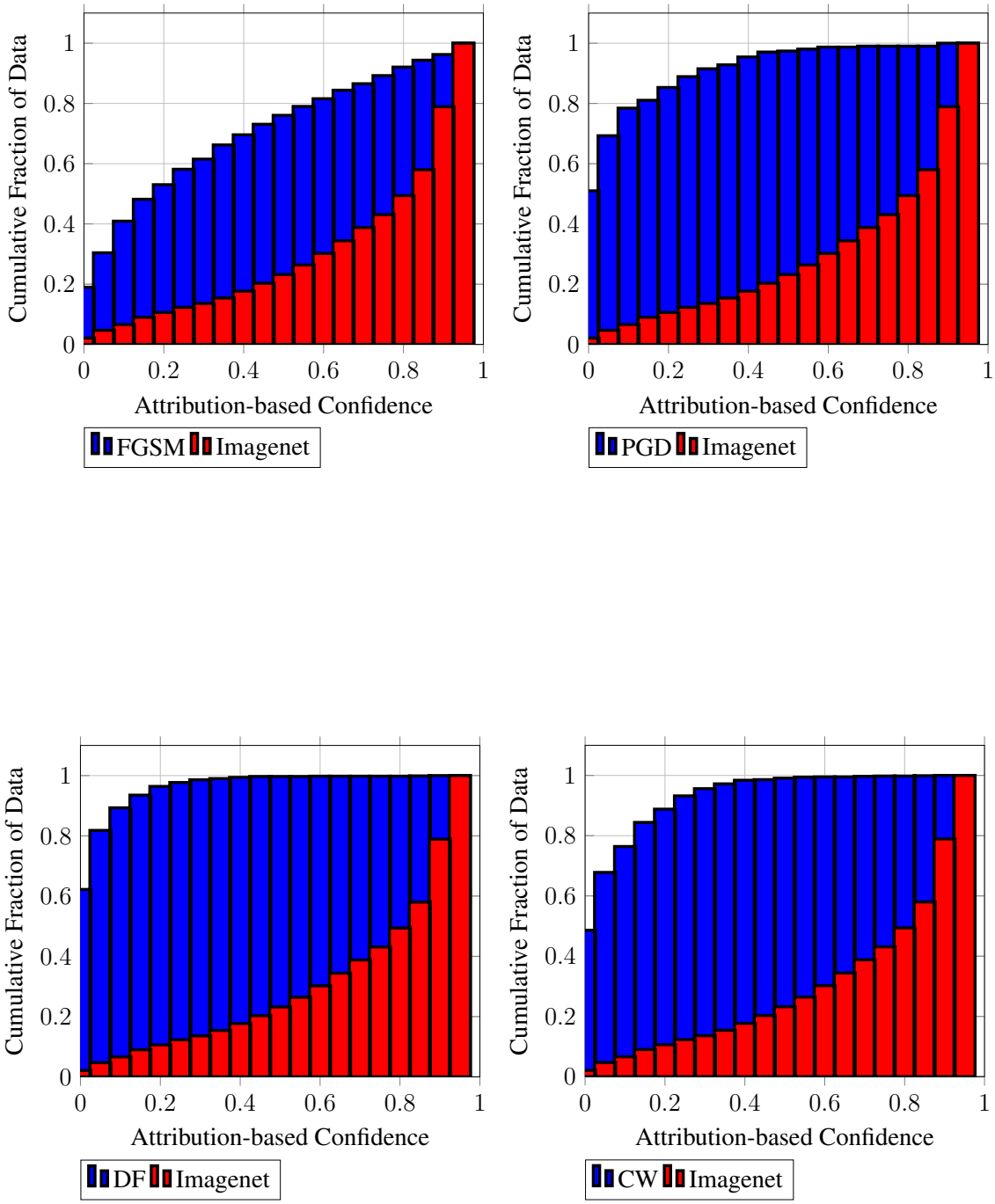


Figure 6.6: ABC metric for FGSM, PGD, DeepFool and CW attacks on ImageNet.

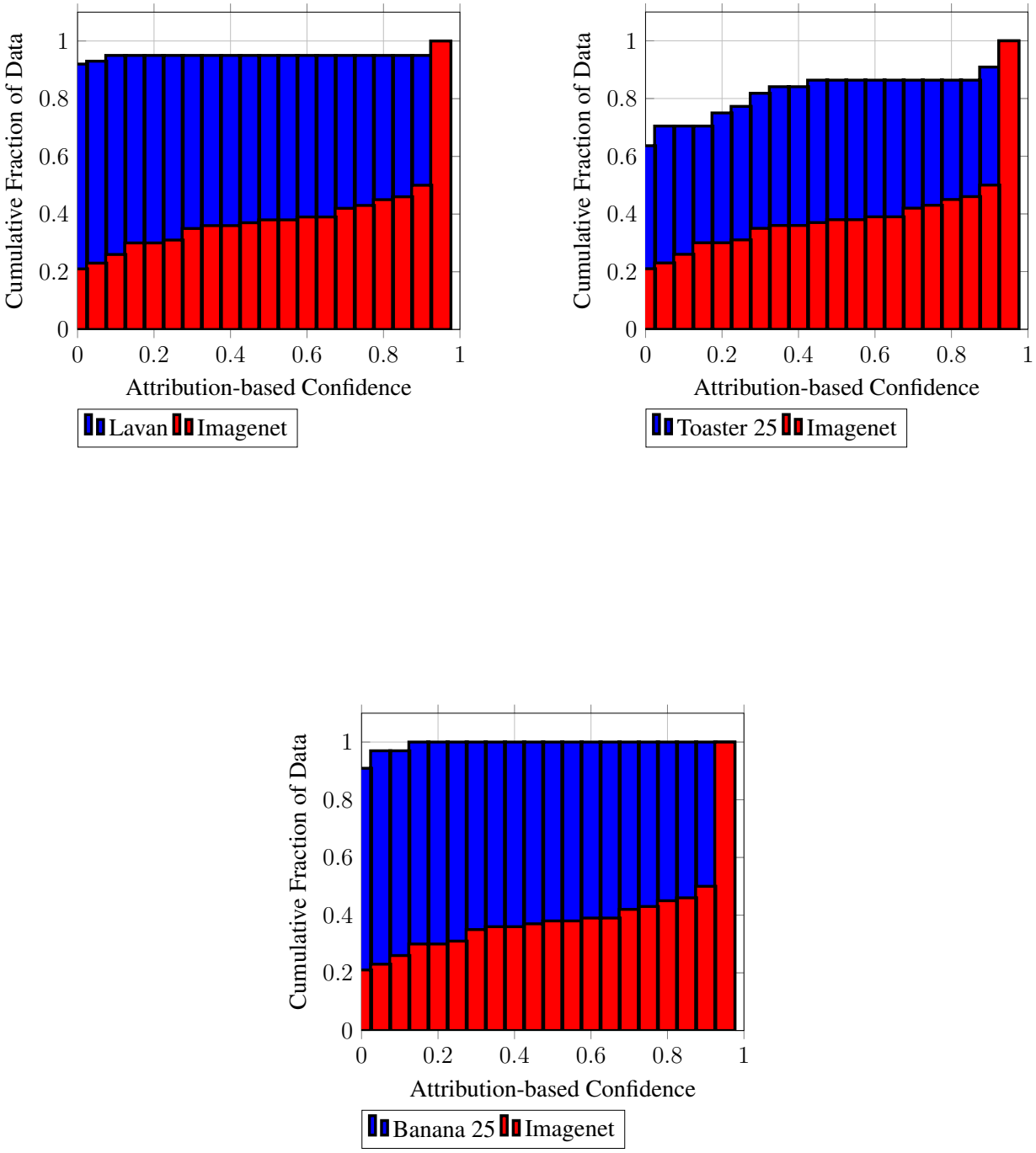


Figure 6.7: Cumulative data fraction vs. ABC metric for ImageNet and adversarial patch attacks.

6.3 Comparison of Attribution Methods

ABC metric can be calculated using various underlying attribution methods. The attribution methods that we used for comparison are IntegratedGradients, DeepShap, and Gradient. We compare the performance of the ABC metric for different attribution methods in Figure 6.8.

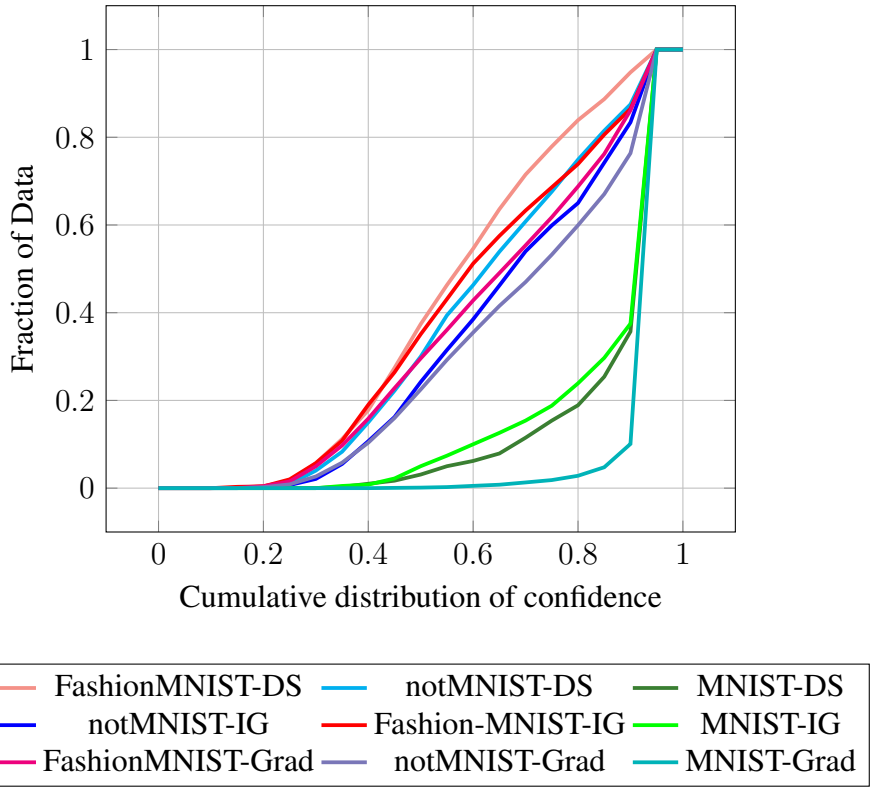


Figure 6.8: Comparing ABC metric using different attribution methods: Gradients (Grad), Integrated Gradient (IG), and DeepSHAP (DS). For out of distribution examples (FashionMNIST and notMNIST), results with DeepShap are slightly better than IG (which is better than Gradient).

CHAPTER 7: CONCLUSION AND 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 the GoogLeNet deep learning 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 the simple sampling approach, $\mathcal{SATYA}$ gives better accuracy and is more resilient to variations in the adversarial image.

We employ an attribution-driven sampling of the neighborhood of a given input and measure the conformance of the model's predictions to compute the attribution-based confidence (ABC) metric for DNN prediction on this input. While directly sampling the neighborhood of a high-dimensional input is challenging, our approach uses attribution-based dimensionality reduction for finding locally relevant features in the vicinity of the input, which enables effective sampling. We theoretically motivate the proposed ABC metric from the axioms of Shapley values, and experimentally evaluate its utility over out-of-distribution data and adversarial examples.

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.

LIST OF REFERENCES

[1] T. B. Brown, D. Mané, A. Roy, M. Abadi, and J. Gilmer, "Adversarial patch," *arXiv preprint arXiv:1712.09665*, 2017.

[2] 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*, pp. 2574–2582, 2016.

[3] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

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

[5] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *ISSP*, pp. 39–57, IEEE, 2017.

[6] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-real: Learning agile locomotion for quadruped robots," 2018.

[7] D. Mahajan, R. Girshick, V. Ramanathan, K. He, M. Paluri, Y. Li, A. Bharambe, and L. van der Maaten, "Exploring the limits of weakly supervised pretraining," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 181–196, 2018.

[8] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, *et al.*, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," *arXiv preprint arXiv:1712.01815*, 2017.

[9] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalch-

brenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–503, 2016.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *ICCV*, pp. 1026–1034, 2015.

[11] W. Xiong, J. Droppo, X. Huang, F. Seide, M. L. Seltzer, A. Stolcke, D. Yu, and G. Zweig, "Toward human parity in speech recognition," *TASLP*, pp. 2410–2423, 2017.

[12] S. Dodge and L. Karam, "A study and comparison of human and deep learning recognition performance under visual distortions," in *2017 26th international conference on computer communication and networks (ICCCN)*, pp. 1–7, IEEE, 2017.

[13] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pp. 372–387, IEEE, 2016.

[14] A. Ramanathan, L. L. Pullum, F. Hussain, D. Chakrabarty, and S. K. Jha, "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.

[15] S. Raj, A. Ramanathan, L. L. Pullum, and S. K. Jha, "Testing autonomous cyber-physical systems using fuzzing features derived from convolutional neural networks," in *ACM SIGBED International Conference on Embedded Software (EMSOFT)*, (Seoul, South Korea), ACM, ACM, 2017.

[16] A. Wald, "Sequential analysis.," 1947.

[17] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *CoRR*, vol. abs/1409.4842, 2014.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[19] J. Hendrik Metzen, T. Genewein, V. Fischer, and B. Bischoff, "On Detecting Adversarial Perturbations," *ArXiv e-prints*, Feb. 2017.

[20] X. Jiang, M. Osl, J. Kim, and L. Ohno-Machado, "Calibrating predictive model estimates to support personalized medicine," *Journal of the American Medical Informatics Association*, vol. 19, no. 2, pp. 263–274, 2011.

[21] K. Park, S. Kim, and K. Sohn, "Unified multi-spectral pedestrian detection based on probabilistic fusion networks," *Pattern Recognition*, vol. 80, pp. 143–155, 2018.

[22] D. Seto, B. Krogh, L. Sha, and A. Chutinan, "The simplex architecture for safe online control system upgrades," in *Proceedings American Control Conference.*, vol. 6, pp. 3504–3508, IEEE, 1998.

[23] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1321–1330, JMLR. org, 2017.

[24] V. Kuleshov and P. S. Liang, "Calibrated structured prediction," in *Advances in Neural Information Processing Systems*, pp. 3474–3482, 2015.

[25] Y. Gal, *Uncertainty in deep learning*. PhD thesis, University of Cambridge, 2016.

[26] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. 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.

[27] 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.

[28] A. Niculescu-Mizil and R. Caruana, "Predicting good probabilities with supervised learning," in *Proceedings of the 22nd International Conference on Machine learning*, pp. 625–632, ACM, 2005.

[29] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," *arXiv preprint arXiv:1611.03530*, 2016.

[30] J. S. Denker and Y. Lecun, "Transforming neural-net output levels to probability distributions," in *Advances in Neural Information Processing Systems*, pp. 853–859, 1991.

[31] J. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," *Advances in Large Margin Classifiers*, vol. 10, no. 3, pp. 61–74, 1999.

[32] N. Papernot and P. McDaniel, "Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning," *arXiv preprint arXiv:1803.04765*, 2018.

[33] H. Jiang, B. Kim, M. Guan, and M. Gupta, "To trust or not to trust a classifier," in *Advances in Neural Information Processing Systems*, pp. 5541–5552, 2018.

[34] R. O. Duda, P. E. Hart, *et al.*, *Pattern classification and scene analysis*, vol. 3. Wiley New York, 1973.

[35] T. Bengtsson, P. Bickel, B. Li, *et al.*, "Curse-of-dimensionality revisited: Collapse of the particle filter in very large scale systems," in *Probability and statistics: Essays in honor of David A. Freedman*, pp. 316–334, Institute of Mathematical Statistics, 2008.

[36] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*, 2013.

[37] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *CVPR*, pp. 618–626, 2017.

[38] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *ICML*, pp. 3319–3328, JMLR. org, 2017.

[39] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems*, pp. 4765–4774, 2017.

[40] P. Dubey, "On the uniqueness of the shapley value," *International Journal of Game Theory*, vol. 4, no. 3, pp. 131–139, 1975.

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

[42] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, "Boosting adversarial attacks with momentum," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9185–9193, 2018.

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

[44] A. Fawzi, O. Fawzi, and P. Frossard, "Analysis of classifiers' robustness to adversarial perturbations," *arXiv preprint arXiv:1502.02590*, 2015.

[45] A. Fawzi, O. Fawzi, and P. Frossard, "Fundamental limits on adversarial robustness," in *Proc. ICML, Workshop on Deep Learning*, 2015.

[46] U. Shaham, Y. Yamada, and S. Negahban, "Understanding adversarial training: Increasing local stability of neural nets through robust optimization," *arXiv preprint arXiv:1511.05432*, 2015.

[47] K. Grosse, P. Manoharan, N. Papernot, M. Backes, and P. D. McDaniel, "On the (statistical) detection of adversarial examples," *CoRR*, vol. abs/1702.06280, 2017.

[48] G. Li and Y. Yu, "Visual saliency based on multiscale deep features," in *CVPR*, pp. 5455–5463, 2015.

[49] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua, "Lift: Learned invariant feature transform," in *ECCV*, pp. 467–483, Springer, 2016.

[50] L. M. Zintgraf, T. S. Cohen, T. Adel, and M. Welling, "Visualizing deep neural network decisions: Prediction difference analysis," *arXiv preprint arXiv:1702.04595*, 2017.

[51] J. Zhou and O. G. Troyanskaya, "Predicting effects of noncoding variants with deep learning–based sequence model," *Nature methods*, vol. 12, no. 10, p. 931, 2015.

[52] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3145–3153, JMLR. org, 2017.

[53] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems 30* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), pp. 4765–4774, Curran Associates, Inc., 2017.

[54] M. E. Muller, "A note on a method for generating points uniformly on n-dimensional spheres," *Commun. ACM*, vol. 2, pp. 19–20, Apr. 1959.

[55] B. Zadrozny and C. Elkan, "Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers," in *ICML*, vol. 1, pp. 609–616, Citeseer, 2001.

[56] B. Zadrozny and C. Elkan, "Transforming classifier scores into accurate multiclass probability estimates," in *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 694–699, ACM, 2002.

[57] M. P. Naeini, G. Cooper, and M. Hauskrecht, "Obtaining well calibrated probabilities using bayesian binning," in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[58] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Advances in Neural Information Processing Systems*, pp. 6402–6413, 2017.

[59] D. J. MacKay, "Probable networks and plausible predictions—a review of practical bayesian methods for supervised neural networks," *Network: Computation In Neural Systems*, vol. 6, no. 3, pp. 469–505, 1995.

[60] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *International Conference on Machine Learning*, pp. 1050–1059, 2016.

[61] A. Rozsa, E. M. Rudd, and T. E. Boult, "Adversarial diversity and hard positive generation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 25–32, 2016.

[62] N. Papernot, I. Goodfellow, R. Sheatsley, R. Feinman, and P. McDaniel, "Cleverhans v2.0.0: an adversarial machine learning library," *arXiv preprint arXiv:1610.00768*, 2016.

[63] D. Karmon, D. Zoran, and Y. Goldberg, "LaVAN: Localized and visible adversarial noise," *arXiv preprint arXiv:1801.02608*, 2018.

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

[65] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *ACCS'17*, 2017.

[66] L. Engstrom, A. Ilyas, and A. Athalye, "Evaluating and understanding the robustness of adversarial logit pairing," *arXiv preprint arXiv:1807.10272*, 2018.

[67] 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*, pp. 582–597, IEEE, 2016.

[68] A. Ilyas, A. Jalal, E. Asteri, C. Daskalakis, and A. G. Dimakis, "The robust manifold defense: Adversarial training using generative models," *arXiv preprint arXiv:1712.09196*, 2017.

[69] S. Jha, U. Jang, S. Jha, and B. Jalaian, "Detecting adversarial examples using data manifolds," in *MILCOM*, pp. 547–552, IEEE, 2018.

[70] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," *arXiv preprint arXiv:1705.07204*, 2017.

[71] A. Raghunathan, J. Steinhardt, and P. Liang, "Certified defenses against adversarial examples," *arXiv preprint arXiv:1801.09344*, 2018.

[72] H. Zhang, Y. Yu, J. Jiao, E. P. Xing, L. E. Ghaoui, and M. I. Jordan, "Theoretically principled trade-off between robustness and accuracy," *arXiv:1901.08573*, 2019.

[73] S. Dutta, S. Jha, S. Sankaranarayanan, and A. Tiwari, "Output range analysis for deep feedforward neural networks," in *NASA Formal Methods Symposium*, pp. 121–138, Springer, 2018.

[74] L. Schott, J. Rauber, M. Bethge, and W. Brendel, "Towards the first adversarially robust neural network model on MNIST," *International Conference on Learning Representations (ICLR)*, May 2019.

[75] L. Schönherr, K. Kohls, S. Zeiler, T. Holz, and D. Kolossa, "Adversarial attacks against automatic speech recognition systems via psychoacoustic hiding," *arXiv preprint arXiv:1808.05665*, 2018.

[76] Y. Qin, N. Carlini, I. Goodfellow, G. Cottrell, and C. Raffel, "Imperceptible, robust, and targeted adversarial examples for automatic speech recognition," *arXiv preprint arXiv:1903.10346*, 2019.

[77] S. Jha, V. Raman, A. Pinto, T. Sahai, and M. Francis, "On learning sparse Boolean formulae for explaining AI decisions," in *NASA Formal Methods Symposium*, pp. 99–114, Springer, 2017.

[78] S. Jha, T. Sahai, V. Raman, A. Pinto, and M. Francis, "Explaining AI decisions using efficient methods for learning sparse Boolean formulae," *Journal of Automated Reasoning*, vol. 63, no. 4, pp. 1055–1075, 2019.

[79] J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, and B. Kim, "Sanity checks for saliency maps," in *NIPS*, pp. 9525–9536, 2018.

[80] A. Ghorbani, A. Abid, and J. Zou, "Interpretation of neural networks is fragile," *arXiv preprint arXiv:1710.10547*, 2017.

[81] N. Kilbertus, G. Parascandolo, and B. Schölkopf, "Generalization in anti-causal learning," *arXiv preprint arXiv:1812.00524*, 2018.

[82] S. Jha, S. Raj, S. Fernandes, S. K. Jha, S. Jha, J. Brian, G. Verma, and A. Swami, "Attribution-driven causal analysis for detection of adversarial examples," *Safe Machine Learning workshop at ICLR (No Proceedings)*, 2019.

[83] J. S. B. Evans and K. Frankish, *In two minds: Dual processes and beyond*, vol. 10. Oxford University Press Oxford, 2009.

[84] P. M. Groves and R. F. Thompson, "Habituation: a dual-process theory.," *Psychological review*, vol. 77, no. 5, p. 419, 1970.

[85] D. Kahneman, *Thinking, fast and slow*. Macmillan, 2011.

[86] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[87] Y. LeCun, C. Cortes, and C. J. Burges, "The MNIST database," *URL http://yann. lecun. com/exdb/mnist*, 1998.

[88] Y. Bulatov, "NotMNIST dataset," *Google (Books/OCR), Tech. Rep.[Online]. Available: http://yaroslavvb.blogspot.it/2011/09/notmnist-dataset.html*, 2011.

[89] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[90] U. of Montreal, "The rotated MNIST with background image," *URL https://sites.google.com/a/lisa.iro.umontreal.ca/public_static_twiki/variations-on-the-mnist-digits*, 1998.