Understanding Adversarial Examples Through Deep Neural Network's Classification Boundary and Uncertainty Regions

Juan Shu[1] , **Bowei Xi**[1*†] , **Charles Kamhoua**[2]
[1]Department of Statistics, Purdue University
[2]US Army Research Laboratory
{shu30, xbw}@purdue.edu, charles.a.kamhoua.civ@army.mil

Abstract

Although AI is developing rapidly, AI's vulnerability under adversarial attacks remains an extraordinarily difficult problem. In this paper we study the root cause of adversarial examples through studying the deep neural network's (DNN) classification boundary. The existing attack algorithms can generate from a handful to a few hundred adversarial examples given one clean sample. We show there are a lot more adversarial examples given one clean sample, all within a small neighborhood of the clean sample. We then define DNN uncertainty regions and show the transferability of adversarial examples is not universal. The results lead to two conjectures regarding the size of the DNN uncertainty regions and where DNN function becomes discontinuous. The conjectures offer a potential explanation for why the generalization error bound – the theoretical guarantee established for DNN – cannot adequately capture the phenomenon of adversarial examples.

1 Introduction

DNN is a powerful tool for complex tasks. Soon after DNN gained popularity [Krizhevsky *et al.*, 2012], researchers noticed that targetedly adding minor perturbations to a clean image can cause a DNN to misclassify the perturbed image [Szegedy *et al.*, 2014]. Despite decades of theoretical research on DNN, there are still many unanswered questions regarding DNN's properties. For example, we do not know the shape of DNN classification boundary. There is also a discrepancy between the established generalization error bounds for DNN and the existence of adversarial examples.

We know the shape of the decision boundary of many well known models, such as linear regression, generalized linear regression, non-parametric regression, support vector machine (SVM), to name a few. Despite many work on building a robust DNN and to evaluate DNN robustness, we are yet to know the shape of DNN classification boundary. A lack of understanding of DNN's classification boundary naturally leads to the fact that we do not know where are the regions containing the adversarial examples. There are conflicting conjectures about the regions containing the adversarial examples. [Szegedy *et al.*, 2014; Crecchi *et al.*, 2019] believed adversarial examples lie in "dense pockets" in lower dimensional manifold, caused by DNN's non-linearity. On the other hand [Goodfellow *et al.*, 2014] believed it is DNN's linear nature and the very high dimensional inputs that lead to the adversarial examples. Furthermore [Goodfellow *et al.*, 2014] believed "adversarial examples occur in contiguous regions of the 1-D subspace defined by the fast gradient sign method, not in fine pockets." Currently many papers show DNN's classification boundary as a curve separating two classes (e.g., [Tramer *et al.*, 2017]), similar to Figure 1. And often it is thought that the adversarial examples lie in the other side of the boundary. Without the knowledge of DNN classification boundary, building a robust DNN model will remain an elusive task.

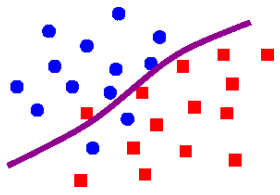


Figure 1: Popular Description of DNN Classification Boundary

There are real world implications when using DNN in critical applications without fully understanding its basic properties, such as its classification boundary and vulnerabilities. For example, DNN is used to process the videos received from cameras, as described on the webpage [Tesla AutopilotAI, 2022]. In a system where the algorithm used to process sensor data has inherent vulnerabilities, they become part of the security vulnerabilities the attackers can explore.

DNN is a popular phrase sometimes meaning different model structures. In this paper we focus on convolutional neural network (CNN) and fully connected neural network (multilayer perceptron (MLP)), and examine their classification boundary. Through experiments, we show the problem of adversarial examples is not as simple as linear vs. nonlinear. It is a far more complex structural problem. The most significant contributions of this paper are the following.

1. We show DNN classification boundary is highly fractured, unlike other classifiers. There are lower dimensional regions containing adversarial examples within a

*Corresponding Author is Bowei Xi

small neighborhood surrounding every clean image.

2. Our first conjecture is that the union of these lower dimensional bounded regions containing adversarial examples has zero probability mass. Our second conjecture is that a DNN function is discontinuous at the boundary of these lower dimensional bounded regions, and may be discontinuous inside some of these bounded regions. The two conjectures could be the reason that the theoretical guarantees established for DNN, such as the generalization error bounds, co-exist with the adversarial examples. Hence new theory is needed to evaluate DNN robustness.
3. We show that transferability of adversarial examples is not universal, contrary to [Szegedy *et al.*, 2014; Goodfellow *et al.*, 2014; Tramer *et al.*, 2017], which suggested that adversarial examples generated against one DNN are misclassified by other DNNs, even if they have different model structures or are trained on different subsets of the training data. We show that adversarial examples against one DNN can be correctly classified by some other DNN models, simply by using different initial random seeds in the training process. This leads to our definition of DNN uncertainty regions.

Besides the three major contributions, additional contributions of this paper are the following.

1. Given one clean image, existing attack algorithms generate up to a few hundred adversarial examples. Sampling from the lower dimensional region lead to a stronger attack, generating a lot more adversarial examples given one clean image.
2. Far fewer pixels are perturbed to form these hyper-rectangles compared to the existing attack algorithms. Therefore we reduce the total amount of perturbations added to a clean image to create adversarial examples.

The paper is organized as follows. Section 1.1 discusses the related work. Section 2 conducts experiments to establish the shape of DNN classification boundary and introduces the concept of DNN uncertainty regions. Section 3 discusses the discrepancy between the theoretically proven DNN large sample property, its generalization error bound, and the existence of adversarial examples. Section 4 concludes this paper.

1.1 Related Work

There are two broad categories of attacks, poisoning attacks and evasion attacks [Xi, 2020]. Poisoning attacks inject malicious samples into the training data, to cause the resulting learning model to make a mistake with certain test samples. Assuming there is no easy access to the training process, evasion attacks generate test samples that the learning model cannot handle correctly. The adversarial examples generated to attack DNN belong to evasion attacks. Depending on adversaries' knowledge of a DNN model, there are white-box attacks and black-box attacks. For white-box attacks, adversaries know the true DNN model, including model structure and parameter values. For black-box attacks, adversaries don't know the true model. Instead, adversaries query the true model, build a local substitute model based on the queries, and attack the local model. A targeted attack generates adversarial examples that are misclassified into a pre-determined class, while an untargeted attack simply generates misclassified samples. Several survey papers are published, introducing the current state and the timeline of attacks and defenses, e.g., [Biggio and Roli, 2018; Xi, 2020]. In general, the attack algorithms follow an optimization approach, i.e., generating adversarial examples through minimizing a loss function.

Adversarial evasion attacks against DNN are the earliest attacks. Recently there are attacks designed to break graph neural network (e.g., [Tang *et al.*, 2020]), recurrent neural network (e.g., [Gao *et al.*, 2018]) etc. In this paper, we examine the classification boundary and uncertainty regions of CNN and MLP. There are many existing attacks against such DNN models. In our experiments we use Foolbox [Rauber *et al.*, 2020], which implements a large collection of adversarial attack algorithms.

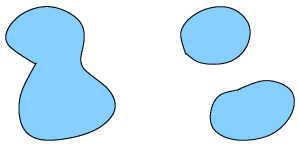
Let W be a clean image and W^a be an adversarial example. Let M be a trained DNN model that assigns a class label to W. $M(W^a) \neq M(W)$. W is a matrix for a gray-scale image, and a tensor for a color image. The size of the matrix/tensor is determined by the image resolution. The individual elements (pixels) in W represent the light level, having integer values ranging from 0 (no light) to 255 (maximum light). The pixels are rescaled to $[0,1]$ by dividing the pixel value by 255. W can be vectorized. Assume a vectorized W is $d-$dimensional, i.e., $W \in [0,1]^d$. Some attack algorithms generate a single W^a or only a handful of W^as are not used in our experiments, because there are not enough adversarial examples to locate the region containing these W^as. We also exclude attack algorithms that need large perturbations to generate W^a. Here are the attack algorithms that are used in our experiments: (1) Pointwise (PW) Attack; (2) Carlini & Wagner L_2 (CW2) Attack; (3) NewtonFool (NF) Attack; (4) Fast Gradient Sign Method (FGSM); (5) Basic Iterative Method (BIM) L_1, L_2, L_∞ attacks; (6) Moment Iterative (MI) Attack.

2 DNN's Uncertainty Regions

DNN function is described by $M(W) = c$, where c is the object class assigned to image W by a trained DNN model M. In this paper we assume M assigns hard labels. The function is more complex when W is a high resolution color image with M assigning soft labels and classification accuracy is assessed using top 5 classes. We leave it to the future work.

The concept of uncertainty regions was first proposed for support vector machine (SVM) facing multi-class classification task. For one-against-all SVM, multiple separating hyper-planes are used to classify the samples. The areas within the margins of the binary hyper-planes are the SVM uncertainty regions [Voichita *et al.*, 2008; Tuia *et al.*, 2009]. If a data point is very close to several binary decision boundaries, SVM is uncertain which class it belongs to.

We propose a different definition of DNN uncertainty regions. Given a training dataset D_n with n samples, let $\mathscr{M} = \{M_1, M_2, ...\}$ be the set of DNN models trained on D_n with identical model structure, i.e., same number of layers, same activation, etc, while varying the initial random seed.



(a) (b)

Figure 2: Conceptual Plots of Uncertainty Regions

Since DNN training is a non-convex optimization problem, the training process converges to different local optima using different initial seed. Thus the training process produces multiple DNN models (M_is) based on one training data D_n. For $i \neq j$, M_i and M_j have different parameter values, i.e., different weights and biases. We define a DNN uncertainty region as a set of points (Ws) in a bounded region in $[0,1]^d$ that cannot be separated into disjoint regions, where at least two DNN models M_i and M_j disagree on the hard label of W.

Definition 1. *An uncertainty region is defined as*

$$U := \left\{W : \exists\, M_i,\, M_j \in \mathscr{M},\, s.t.\, M_i(W) \neq M_j(W)\right\},$$

where U cannot be separated into disjoint regions in $[0,1]^d$.

We use the L_2 distance between a clean image W and an adversarial image W^a, $d(W,W^a) = ||W - W^a||_2$. Let δ be the radius of a $\delta-$ball with the clean image W at the center, $\delta > 0$. Denote the $\delta-$ball by $B(\delta,W)$. $B(\delta,W) := \{W : d(W^a,W) \leq \delta\}$. When δ is sufficiently small, the points in $B(\delta,W)$ are noisy versions of W and should be labeled to the same object class as W. Given a clean image W, we can determine the value of δ based on the amount of adversarial perturbations. We choose δ to be slightly larger than the minimum amount of adversarial perturbations calculated from a number of attack algorithms. Figure 2 (b) conceptually shows two separate uncertainty regions while Figure 2 (a) is one region. We discover there exists multiple uncertainty regions inside a $B(\delta,W)$.

DNN model structure: Because we focus on studying the classification boundary of DNN, here the DNN model structure must strictly remain the same. We discover that even a minor change to the model structure, such as adding or removing a batch normalization layer, will lead to a different classification boundary.

Clean natural sample: We consider a clean natural image as the result of taking a photo using a camera. Regarding how many clean images we can have, let's consider the volume of a $d-$dimensional $B(\delta,W)$, $|B(\delta,W)| = \frac{\pi^{d/2}}{\Gamma(1+d/2)}\delta^d$. The volume of the feature space $[0,1]^d$ is 1. Let δ be sufficiently small, hence the images in $B(\delta,W)$ are noisy versions of W. For a fixed δ and d, there are only a finite number of non-overlapping δ balls in the feature space $[0,1]^d$. However, as $d \longrightarrow \infty$, we have $|B(\delta,W)| \longrightarrow 0$. Hence the feature space for higher resolution color images can contain increasingly more clean images.

Uncertainty Region Construction: We compare an adversarial example W^a with the corresponding clean W. If a pixel value in W^a is different than that in W, it is perturbed by the attack. Given a clean W, we use one attack algorithm and generate a sufficient amount of adversarial examples that are all mis-classified into the same wrong object class. We examine how many pixels are perturbed by the attack. Then we compute the interval for each perturbed pixel (the original W has a single value for this pixel). The perturbed pixels are ordered by the interval sizes from the largest to the smallest. We then construct a hyper-rectangle starting from the largest interval, and stop at where the subsequent intervals can be considered as nearly a constant (which may not equal to the original pixel value for clean W). The detailed procedure is described as follows.

Table 1: Re-Trained LeNet Mis-classification Rates on Clean MNIST Test Data

M_1	M_2	M_3	M_4	M_5
0.033	0.035	0.025	0.019	0.017
M_6	M_7	M_8	M_9	M_{10}
0.015	0.013	0.012	0.012	0.012

Assume M_1 is the model under attack. For a given attack algorithm and a object class t, $t \neq c$ where c is the true object class of W, we combine the adversarial examples W^a from both the targeted attack and the untargeted attack, s.t. $M_1(W^a) = t$. We then construct the subspace spanned by W^as. This step requires an attack algorithm to generate sufficient amount of perturbed images W^a, at least 80-100 images, given one clean image. We notice that different attack algorithms discover different regions containing the adversarial examples for one clean image. Only a handful of adversarial examples is not enough to locate a region containing these adversarial examples. The more adversarial examples an attack algorithm can generate for a clean image, the easier to locate the region containing these adversarial examples.

Although there are a large collection of attacks algorithms in Foolbox, most of them cannot satisfy this requirement, including several famous attacks – DeepFool attack, L-BFGS attack, PGD attack (on MNIST). Furthermore we only consider adversarial examples W^a in $B(\delta,W)$ with a small δ. Some attacks generate large perturbations that are barely recognizable, such as Spatial Transform Attack, Additive Gaussian Noise Attack, and Additive Uniform Noise Attack. They are also excluded from the experiments. We only use the attack algorithms listed in Sec. 1.1 in the experiments.

Let $I_k(t) := \left\{W_k^a = (W_{k,i}^a) :\, M_1(W_k^a) = t,\, t \neq c\right\}$. $I_k(t)$ is the set of adversarial images misclassified to class t by attack algorithm k. If $\exists\, W_{k,i}^a \neq W_i$, i.e., attack k adds perturbation to the ith pixel, we compute the interval size of the ith pixel as the difference between the maximum and the minimum values on the ith pixel from the group of adversarial images. Let $s_i^{t,k} = max_{I_k(t)}(W_{k,i}^a) - min_{I_k(t)}(W_{k,i}^a)$. Assume m pixels are perturbed by attack k, $m \leq d$. Then the intervals are ranked by interval size as $s_{(1)}^{t,k} \geq s_{(2)}^{t,k} \geq \cdots s_{(m)}^{t,k}$. We construct a hyper-rectangle $R_k(t)$ using b largest intervals with $b \leq m$ as

$$R_k(t) \;=\; \otimes_{i=1}^{b}\, [min_{I_k(t)}(W_{k,(i)}^a),\, max_{I_k(t)}(W_{k,(i)}^a)].$$

Table 2: Re-Trained LeNet Misclassification Rates in Hyper-Rectangles

		$s_{(i)}$	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
NF $1 \rightarrow 2$	60d	0.030	0	0	0	0.001	0.004	0.925	1	0.58	0.991	1
FGSM $1 \rightarrow 2$	375d	0.012	0.915	0	0	0	0	0	0	0	0	0
CW2 $1 \rightarrow 6$	150d	0.013	0	0.099	0	0.993	0.007	0.945	0.807	0.042	1	0.652
BIM L_∞ $1 \rightarrow 7$	380d	0.002	0.782	0	0	0	0	0	0	0	0	0
PW $1 \rightarrow 8$	35d	1	0.809	0.818	0.48	0.856	0.896	0.974	0.932	0.754	0.92	0.937
MI $1 \rightarrow 0$	230d	0.032	0.996	1	0	0.235	0	0	0	0	0	1

Table 3: 10 Re-Trained LeNet Misclassification Rates Under Different Attacks

		M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
NF $1 \rightarrow 2$	403d	1	0.82	0.62	0.92	0.96	1	1	0.95	1	1
FGSM $1 \rightarrow 2$	403d	1	0	0	0	0	0	0	0	0	0
CW2 $1 \rightarrow 6$	318d	1	0	0	0	0	0	0	0	0	0
BIM L_∞ $1 \rightarrow 7$	491d	1	0	0	0	0	0	0	0	0	0
PW $1 \rightarrow 8$	40d	1	0.04	0.01	0.15	0.16	0.55	0.37	0.41	0.33	0.48
MI $1 \rightarrow 0$	462d	1	1	0.95	1	0.87	0.85	0.82	0.62	0.63	0.67

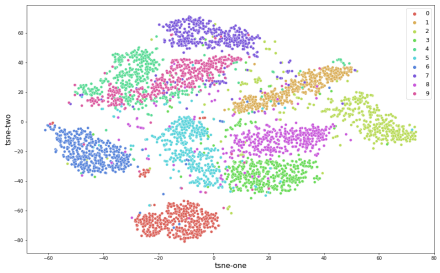


Figure 3: Lower Dimensional Projection of MNIST data

$R_k(t)$ is the subspace based on the adversarial examples generated by attack k and misclassified to class t. We choose the number of intervals b that the remaining interval sizes are very small and the perturbations added can be considered as approximately constant.

We use Pytorch 1.5.0 and Cuda 10.2 to run all the experiments. CPU is Intel Xeon Silver 4114 and the GPU is Nvidia Tesla P100. The code is posted on GitHub. [1]

2.1 MNIST CNN Experiment

Here we conduct an experiment with the task to classify MNIST dataset of 10 handwritten digit. MNIST has 60,000 training images and 10,000 test images. Each image has 28x28 gray-scale pixels. Our model structure is the PyTorch implementation [PyTorch, 2022] of LeNet [LeCun *et al.*, 1998], which has two convolutional layers. The model structure has been published previously. We re-train LeNet on MNIST to optimize the parameter values. The optimizer is SGD with learning rate 0.01. W is a vectorized MNIST image with pixels rescaled to $[0,1]$ in PyTorch implementation. We have $W \in [0,1]^{784}$. Table 1 shows the accuracy of 10 re-trained LeNet models on the MNIST test data using different initial seeds. M_1 to M_{10} have similar performance on clean test data.

Intuitively, the ten handwritten digits have distinct features that facilitate the classification task. Hence LeNet can achieve nearly 99% accuracy. We visualize the digits using t-Distributed Stochastic Neighbor Embedding (t-SNE) technique [Van der Maaten and Hinton, 2008], a nonlinear dimension reduction technique. Figure 3 provides a 2D projection of the ten digits, based on 2000 sampled images. We observe 10 clusters of digits though some clusters overlap slightly. We would expect a classifier to divide up the feature space, and allow a digit class to occupy a portion of the feature space. Then the points away from the classification boundary and their surround neighborhoods would all belong to the same object class. Unfortunately this is not what we see from DNN. We need to draw DNN's classification boundary around every clean image, not along the border between two object classes.

We choose a clean image W, and generate adversarial examples using the attack algorithms listed in Section 1.1. We then construct the hyper-rectangles $R_k(t)$. We have studied many test images and training images, and have obtained similar results. Due to the limited space, here we show the results for a digit 1 from the test data. We run the attacks against M_1. Table 2 shows the following information.

1. The number of intervals used to construct the hyper-rectangles. For example, CW2 $1 \rightarrow 6$ means the digit 1 is mis-classified as 6. 150d means $R_{CW2}(6)$ is spanned by the largest 150 intervals.
2. The smallest interval size in $R_k(t)$, shown in column $s_{(i)}$. For PW attack, we use [0,1] for the selected pixels, since the measured interval sizes are all close to 1. For all other attacks, the interval size is measured from

[1] https://github.com/juanshu30/Understanding-Adversarial-Examples-Through-DNNs-Classification-Boundary-and-Uncertainty-Regions

Table 4: L_2 Distance

	L_2^{min}	L_2^{max}	$\bar{L}_2$	L_2^{min} Attack	L_2^{max} Attack	$\bar{L}_2$ Attack
NF $1 \rightarrow 2$	4.881	5.729	5.305	4.986	8.803	6.806
FGSM $1 \rightarrow 2$	14.259	14.41	14.335	14.413	15.213	14.784
CW2 $1 \rightarrow 6$	7.285	8.435	7.86	7.485	12.687	9.719
BIM L_∞ $1 \rightarrow 7$	8.197	9.124	8.661	13.118	15.11	14.855
PW $1 \rightarrow 8$	5.205	14.84	10.023	12.526	26.526	17.329
MI $1 \rightarrow 0$	21.261	22.607	21.852	43.76	43.297	38.818

(a) PW $1 \rightarrow 8$ (b) PW Sampled $1 \rightarrow 8$

Figure 4: MNIST CNN Experiment $1 \rightarrow 8$

the added perturbations.

3. We sample 1000 images from each $R_k(t)$, and report the misclassification rates by M_1 to M_{10}.

Table 3 shows the 10 re-trained LeNet models' mis-classification rates against the original adversarial images generated by the attacks, and the number of perturbed pixels. The left three columns in Table 4 show the minimum amount of perturbations ($min(||W^a - W||_2)$), the maximum amount of perturbations ($max(||W^a - W||_2)$), and the average amount of perturbations ($mean(||W^a - W||_2)$) of the 1000 sampled images in each hyper-rectangle $R_k(t)$. The right three columns in Table 4 show the same information for the adversarial examples generated by the corresponding attacks.

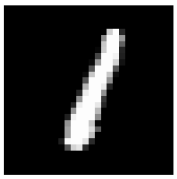


Figure 5: Clean Image 1

Figure 4 shows the adversarial examples generated by the attacks, and the adversarial images generated through sampling in the hyper-rectangles $R_k(t)$. Figure 5 is the corresponding clean image 1. Except for PointWise attack, which changes a pixel value to 0 or 1, the rest of $s_{(i)}$ has the maximum value 0.032, as shown in Table 2. This translates to 8 consecutive integer values on the original 0 – 255 scale. They are very similar light levels, and can be considered as approximately constant. If we add more dimensions to $R_k(t)$, the additional dimensions can be considered as moving the additional pixels to different values. Adding more dimensions do not change the shape and size of $R_k(t)$. Instead that moves a hyper-rectangle to a different location, increasing the amount of perturbation and away from the clean image W. The hyper-rectangles $R_k(t)$ in Table 2 perturbed far fewer pixels than the original attacks. From Table 4, we see that leads to smaller perturbations to create adversarial examples. There are more such hyper-rectangles with the same shape and size, as we add more pixels identified by the attacks. Adding more pixels does not necessarily increase the mis-classification rates by all DNNs. For Carlini & Wagner L_2 attack and FGSM, eventually the hyper-rectangle is moved to a place where M_1 mis-classification rate is close to 100% and M_2 to M_{10} see near 0% mis-classification rate. This is the effect of the optimization approach used in the attack algorithms against M_1. We observe three types of $R_k(t)$ in Table 2.

1. The target DNN mis-classifies most of the adversarial examples while there exists another DNN which correctly classifies the adversarial examples;
2. The target model correctly classifies the adversarial examples while another DNN mis-classifies most of the adversarial examples;
3. The transferable adversarial regions where all DNNs mis-classify a significant proportion of the adversarial examples.

This phenomenon occurs to attacks adding both small and large perturbations. The first two types of $R_k(t)$ belong to DNN uncertainty regions. The existence of DNN uncertainty regions shows transferability of adversarial examples is not universal, contrary to [Szegedy *et al.*, 2014; Goodfellow *et al.*, 2014; Tramer *et al.*, 2017].

Table 5: MLP Mis-classification Rates on Clean MNIST Test Data

M_1	M_2	M_3	M_4	M_5
0.0177	0.0178	0.0183	0.0177	0.0176

Table 6: MLP CW2 Mis-classification Rates in Hyper-Rectangles

		$s_{(i)}$	M_1	M_2	M_3	M_4	M_5
$5 \rightarrow 6$	230d	0.005	0.94	0	0	0	0
$7 \rightarrow 2$	440d	0.01	0.82	0	0.78	0	0

2.2 MNIST MLP Experiment

Here we conduct experiment with a MLP trained on MNIST. It is a fully connected network with 3 layers, 3x512 hidden neurons and ReLU activation. We vary the initial seeds and

(a) CW2 $5 \rightarrow 6$ (b) CW2 Sampled $5 \rightarrow 6$

Figure 6: Images for MLP experiment

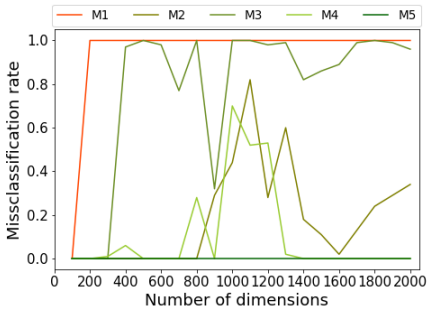


Figure 7: MobileNet misclassification rates in hyper-rectangles

train 5 MLPs. The optimizer is SGD with learning rate 0.01. Table 5 shows the MLP mis-classification rates on the clean MNIST test data. In the interest of space, here we show two examples, a digit 5 and a digit 7, under Carlini & Wagner L_2 attack. Table 6 shows the 5 MLPs' mis-classification rates in the hyper-rectangles. Table 7 shows mis-classification rates against the original adversarial images generated by the attack. Figure 6 shows an adversarial example generated by Carlini & Wagner L_2 attack, and an adversarial example generated through sampling from the hyper-rectangle, based on the same clean image 5. Figure 9 (a) is the corresponding clean image 5. The hyper-rectangle for $7 \rightarrow 2$ lie in one DNN uncertainty region. Again Carlini & Wagner L_2 attack has great success with the target model M_1 but can be correctly classified by some other MLPs.

2.3 CIFAR10 MobileNet Experiment

CIFAR10 [Krizhevsky, 2009] has 60,000 32x32 color images in 10 classes, with 50,000 as training images and 10,000 as test images. A vectorized CIFAR10 image is in $[0,1]^{3072}$, combining three color channels. The dimensionality of a CIFAR10 image is almost 4 times of a MNIST image. We use the MobileNet in this experiment. Similar to Section 2.1, the MobileNet model structure has been published previously

Table 7: MLP Mis-classification Rates Under the Original Attack

		M_1	M_2	M_3	M_4	M_5
CW2 $5 \rightarrow 6$	380d	1	0	0	0	0
CW2 $7 \rightarrow 2$	491d	1	0	0.28	0	0

Table 8: Re-Trained MobileNet Mis-classification Rates on Clean CIFAR10 Test Data

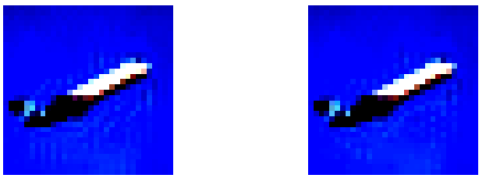
M_1	M_2	M_3	M_4	M_5
0.0767	0.0728	0.0734	0.0727	0.0744

Table 9: Five Re-Trained MobileNet Misclassification Rates Under BIM L_2 Attack – Airplane Misclassified as Deer

		M_1	M_2	M_3	M_4	M_5
BIM L_2	3017d	1	0	0.88	0	0

[Sandler *et al.*, 2018], which has an initial convolution layers followed by 19 residual bottleneck layers. We re-train the MobileNet on CIFAR10 to optimize the parameters values. The optimizer is SGD with learning rate 0.01; momentum is 0.9; weight decaying is 5e-4. The mis-classification rates of five re-trained MobileNet models on the clean CIFAR10 test data by varying initial seeds are in Table 8. For the interest of space, here we show an example with an airplane image under BIM L_2 attack. The attack success on the five re-trained MobileNet models are in Table 9. Note BIM L_2 attack perturbed 3071 dimensions and left 1 dimension untouched. The images are shown in Figure 8 and Figure 9 (b).

Figure 7 shows the mis-classification rates as we increase the dimensions of the hyper-rectangle. The largest interval size is 0.2 and the 2000th largest interval size is 0.017. $M1$ misclassifies all the sampled images starting from around 200 perturbed dimensions. $M5$ correctly classifies all the sampled images. We see $M2$ and $M4$ misclassification rates increase as more effective dimensions are included, then decrease as we include additional irrelevant dimensions. The 2000-dimensional hyper-rectangle lies in one MobileNet uncertainty region. As noted in [Goodfellow *et al.*, 2014], the direction of adversarial perturbation is important. Adversarial examples cannot be generated by randomly sampling in 3072 dimensional ball $B(\delta, W)$. The lower dimensional hyper-rectangles $R_k(t)$ containing infinitely many adversarial examples are discovered through the attack algorithms. Table 10 shows that the sampled adversarial images from the hyper-rectangle have much smaller perturbations than the original attack on CIFAR10.

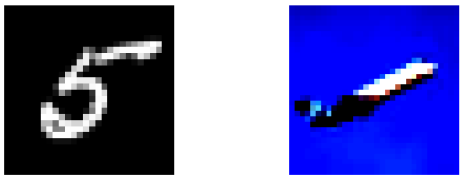


(a) BIM L_2 airplane→deer (b) BIM Sampled airplane→deer

Figure 8: Images for MobileNet experiment

Table 10: L_2 Distance for MLP and Re-Trained MobileNet Experiments

		L_2^{min}	L_2^{max}	$\bar{L}_2$	L_2^{min} Attack	L_2^{max} Attack	$\bar{L}_2$ Attack
CW2 MLP $5 \rightarrow 6$	230d	10.47	10.72	10.61	11.06	11.51	11.26
CW2 MLP $7 \rightarrow 2$	440d	11.18	11.41	11.30	11.51	11.89	11.68
BIM L_2 airplane$\rightarrow$deer	2000d	64.49	67.05	65.88	225	288.32	256.01



(a) Clean Image 5 (b) Clean Airplane

Figure 9: Clean Images

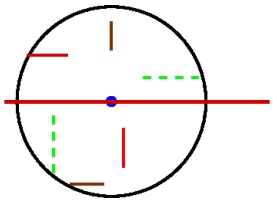


Figure 10: Conceptual Plot of M_1 Classification Boundary Around A Clean Sample

2.4 Uncertainty Regions vs. Transferable Adversarial Regions

Due to the nature of the uncertainty regions, we have to train multiple models. However the classification boundary is established for one model – the model under attack – not the ensemble of all the trained models. The output of a DNN ensemble is either based on the majority vote, or we take the average of the softmax layer outputs from the DNN models in the ensemble. Hence a DNN ensemble has different classification boundary compared with that of a single model used in the ensemble.

For the MNIST CNN experiment, we use Figure 10 as a conceptual plot to show the classification boundary for M_1. M_1 is the model under attack. Let W be the digit 1 used in Section 2.1. Let $\delta = 6$. Hence the hyper-rectangles with larger perturbations are excluded from $B(\delta, W)$.

The blue dot in the center is the clean image. Inside the black circle, the solid line segments are part of the classification boundary for M_1. There are two types, illustrated using two different colors. Type 1 regions are where W^as are misclassified by M_1 but can be correctly classified by some other model M_j; type 2 regions are where W^as are misclassified by all the models, M_1 to M_{10}.

The dashed lines inside the black circle are not part of M_1's classification boundary, but they are model M_1's uncertainty regions, because inside these regions W^as are correctly classified by M_1 but misclassified by some other model M_j. We call them the type 3 regions.

Type 1 and 3 are the uncertainty regions. Type 2 are the transferable adversarial regions which are more difficult to handle. Both the uncertainty regions and the transferable adversarial regions are lower dimensional small "cracks" inside the small neighborhood of a clean image. Only type 1 and type 2 regions where M_1 misclassifies the samples in the $\delta-$ball $B(\delta, W)$ are part of model M_1's classification boundary around the clean image W.

The Shape and Size of Uncertainty Regions: In Table 10 we see a significant reduction of perturbation for BIM L_2 attack and the airplane image, because our hyper-rectangle perturbed far less pixels than the original attack (2000d vs. 3017d). On the other hand, in Table 3, we see only minor reduction for FGSM and the digit 1 because the dimension of the hyper-rectangle is close to the original attack (375d vs 403d). For NF and the digit 1, although our hyper-rectangle used 60d compared with 403d for the original attack, there is only a minor reduction in the total amount of perturbation. Since our approach relies on the existing attacking algorithms to locate the regions, the dimensionality of the regions is related to the original attacks and the clean image itself. Furthermore, although we construct hyper-rectangles, the exact shape of an uncertainty/transferable region may not be a hyper-rectangle. It is important to further investigate how many uncertainty regions and transferable adversarial regions exist in the feature space $[0,1]^d$, and the exact shape and dimensionality of such regions.

Strategy for Robust Classification: If at least one DNN assigns a label that is different from another DNN, the image triggers an alert and requires additional screening, either involving a human operator or alternative classifiers. This strategy will improve the accuracy over the adversarial examples in DNN uncertainty regions, but won't solve the problem for transferable adversarial examples. Notice although an ensemble can achieve high predictive performance [Dong *et al.*, 2020], a DNN ensemble can be attacked too. Meanwhile there is no guarantee about the number of DNNs that can make correct decision over each uncertainty region. We also need to understand how to measure the size of DNN uncertainty regions vs. DNN transferable adversarial regions. We leave it to the future work.

3 Generalization Error Bound and Adversarial Examples

The accuracy on clean test data is often used to measure a classifier's performance. However, in [Nadeau and Bengio, 2003], the authors argued the test data accuracy is not

the most appropriate performance measure, because the variability due to the randomness of the training data needs to be taken into consideration besides those due to the test data. Let $Z = (W, Y)$ denote a sample, where $W \in [0,1]^d$ is a $d-$dimensional vectorized image and $Y \in \{1, \cdots, c\}$ is the true object class. Z is generated independently and identically from a distribution F over $[0,1]^d$. We denote a training dataset with n sample points by $D_n = (Z_1, \cdots, Z_n)$. [Nadeau and Bengio, 2003] defined generalization error as $E(loss_M(D_n, Z_{n+1}))$, where Z_{n+1} is a test sample, and $loss_M()$ is the loss of applying a classifier M trained on D_n to Z_{n+1}. If $loss_M()$ is a 0-1 loss, the generalization error is defined as the error probability $P(M(W) \neq Y)$ as in [Kaarainen, 2005].

There is an extensive literature on the theoretical generalization error bound for different type of classifiers including DNN. Generalization error bound for DNN is proven to be $O(\frac{c(depth, width)}{\sqrt{n}})$, where $c(depth, width)$ refers to a constant based on the width and depth of a DNN model, e.g., [Bartlett *et al.*, 2017; Golowich *et al.*, 2018].

We observe there is a discrepancy between the theoretically proven generalization error bound for DNN and the existence of adversarial examples. Following the theory, the generalization error on test data should decrease to 0 at a rate proportional to $n^{-1/2}$ where n is the training sample size. But given a clean image, we show there exists infinitely many adversarial images in $B(\delta, W)$ for different network structures and datasets. Adversarial examples also exist for large DNN models trained on ImageNet with millions of training data, where the theoretical asymptotic behavior of DNN should already kick in. Here we have two conjectures.

Conjecture 1: The union of these lower dimensional bounded uncertainty regions and transferable adversarial regions has zero probability mass.

Conjecture 2: A DNN function is discontinuous at the boundary of these lower dimensional bounded regions, and may be discontinuous inside some of these bounded regions. Note lipschitz continuity is an important assumption for proving the generalization error bound.

The two conjectures with Theorem 1 offer a potential explanation for why such a discrepancy exists. Let L_r be a $r-$dimensional region in $[0,1]^d$ with $r < d$. Let $\mathscr{L} = \cup_{i=1}^{\infty} L_{r_i}$ be the union of countably infinite non-overlapping lower dimensional regions L_{r_i} in $[0,1]^d$ with all $r_i < d$.

Theorem 1. *Let M_1 and M_2 be two DNN models trained on D_n. Assume $\forall\ W \in [0,1]^d - \mathscr{L}$, $M_1(W) = M_2(W)$. And assume $\exists\ W \in \mathscr{L}$, s.t. $M_1(W) \neq M_2(W)$. We have*

$$E(loss_{M_1}(D_n, Z_{n+1})) = E(loss_{M_2}(D_n, Z_{n+1})).$$

Proof. For any continuous distribution F on $[0,1]^d$, $F(\mathscr{L}) = 0$, i.e., the lower dimensional $\mathscr{L}$ has 0 probability mass. For two functions that differ only on 0 probability region, we have

$$E(loss_{M_1}(D_n, Z_{n+1})) = E(loss_{M_2}(D_n, Z_{n+1})).$$

□

Remark 1: Theorem 1 means the definition of generalization error cannot tell the difference between a trained classifier that assign correct labels to all the points in $[0,1]^d$ and a different classifier that assign wrong labels only to countably infinite lower dimensional bounded regions. For example, let w_i, $i = 1, 2, ...$, s.t. $w_i \neq w_j$ if $i \neq j$. Assume $L = \cup_{i=1}^{\infty} [0,1]^{d-1} \otimes w_i$, be the union of countably infinite non-overlapping $[0,1]^{d-1}$ regions. A classifier can assign wrong labels to L without any impact on its generalization error. So far we see adversarial examples exist in much lower dimensional regions, leading to Conjecture 1.

Remark 2: Another definition of generalization error involves the empirical error on the training data. Let $\hat{loss}_M(D_n) = \frac{1}{n}\sum_1^n loss(Z_i)$ be the empirical risk estimated from the training data D_n. [Zhang *et al.*, 2021] defined generalization error as $GE^*(M) = E(loss_M(D_n, Z_{n+1})) - \hat{loss}_M(D_n)$, which is also used in some recent papers to establish DNN theoretical guarantees. Corollary 1 in [Zhang *et al.*, 2021] states there exists neural networks with ReLU activation, depth g, width $O(n/g)$ and weights $O(n+h)$, that can fit exactly any function on D_n in $d-$dimensional space. Assume M_1 and M_2 are such models trained on D_n. Hence $\hat{loss}_{M_1}(D_n) = \hat{loss}_{M_2}(D_n) = 0$. Consequently we have $GE^*(M_1) = GE^*(M_2)$.

4 Conclusion

A limitation of our work is that we rely on the existing attack algorithms to locate these hyper-rectangles. Also our approach works with low resolution images. Again we leave it to the future work to capture the shape of the DNN classification boundary in very high dimensional feature space.

We gain important insights from this study. A DNN model draws the classification boundary around every image instead of along the border between the object classes. This helps a DNN model to achieve high accuracy and low generalization error for complex tasks but leaves space for it to be attacked. How to seal these small cracks surrounding every image is a very difficult problem, as we witness the success of the adaptive attacks [Tramer *et al.*, 2020]. The insights gained from this study points to the problem where a robust DNN model should work on.

Understanding the shape of DNN's classification boundary also provides insights to defend against the backdoor attacks [Chen *et al.*, 2017]. As with many other classifiers, we need to understand how the change in the training data moves the classification boundary, in order to firmly close the backdoor.

We conclude that the adversarial examples stem from a structural problem of DNN. DNN's classification boundary is unlike that of any other classifier. Current defense strategies do not address this structural problem. We also need new theory to describe the phenomenon of adversarial examples and measure the robustness of DNN.

Acknowledgment

This work is supported in part by US Army Research Office award W911NF-17-1-0356 and US Army Research Lab.

References

[Bartlett *et al.*, 2017] Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky. Spectrally-normalized margin bounds

for neural networks. *Advances in neural information processing systems*, 30, 2017.

[Biggio and Roli, 2018] Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331, 2018.

[Chen *et al.*, 2017] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

[Crecchi *et al.*, 2019] F Crecchi, D Bacciu, and B Biggio. Detecting adversarial examples through nonlinear dimensionality reduction. In *27th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, pages 483–488, 2019.

[Dong *et al.*, 2020] Xibin Dong, Zhiwen Yu, Wenming Cao, Yifan Shi, and Qianli Ma. A survey on ensemble learning. *Frontiers of Computer Science*, 14(2):241–258, 2020.

[Gao *et al.*, 2018] Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 50–56, 2018.

[Golowich *et al.*, 2018] Noah Golowich, Alexander Rakhlin, and Ohad Shamir. Size-independent sample complexity of neural networks. In *Conference On Learning Theory*, pages 297–299. PMLR, 2018.

[Goodfellow *et al.*, 2014] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[Kaariainen, 2005] Matti Kaariainen. Generalization error bounds using unlabeled data. In *International Conference on Computational Learning Theory*, pages 127–142. Springer, 2005.

[Krizhevsky *et al.*, 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems (NIPS)*, 25, 2012.

[Krizhevsky, 2009] A Krizhevsky. Learning multiple layers of features from tiny images. *Master's thesis, University of Tront*, 2009.

[LeCun *et al.*, 1998] Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[Nadeau and Bengio, 2003] Claude Nadeau and Yoshua Bengio. Inference for the generalization error. *Machine Learning*, 52(3):239–281, 2003.

[PyTorch, 2022] PyTorch: Adversarial Example Generation. https://pytorch.org/tutorials/beginner/fgsm_tutorial.html, 2022.

[Rauber *et al.*, 2020] Jonas Rauber, Roland Zimmermann, Matthias Bethge, and Wieland Brendel. Foolbox native: Fast adversarial attacks to benchmark the robustness of machine learning models in pytorch, tensorflow, and jax. *Journal of Open Source Software*, 5(53):2607, 2020.

[Sandler *et al.*, 2018] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobilenetV2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[Szegedy *et al.*, 2014] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *The 2nd International Conference on Learning Representations (ICLR)*, 2014.

[Tang *et al.*, 2020] Xianfeng Tang, Yandong Li, Yiwei Sun, Huaxiu Yao, Prasenjit Mitra, and Suhang Wang. Transferring robustness for graph neural network against poisoning attacks. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 600–608, 2020.

[Tesla AutopilotAI, 2022] Tesla Artificial Intelligence & Autopilot. https://www.tesla.com/autopilotAI, 2022. Last accessed Feb. 01, 2022.

[Tramer *et al.*, 2017] Florian Tramer, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. The space of transferable adversarial examples. *arXiv preprint arXiv:1704.03453*, 2017.

[Tramer *et al.*, 2020] Florian Tramer, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On adaptive attacks to adversarial example defenses. *Advances in Neural Information Processing Systems*, 33:1633–1645, 2020.

[Tuia *et al.*, 2009] Devis Tuia, Frederic Ratle, Fabio Pacifici, Mikhail F Kanevski, and William J Emery. Active learning methods for remote sensing image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 47(7):2218–2232, 2009.

[Van der Maaten and Hinton, 2008] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[Voichita *et al.*, 2008] Calin Voichita, Purvesh Khatri, and Sorin Draghici. Identifying uncertainty regions in support vector machines using geometric margin and convex hulls. In *IEEE International Joint Conference on Neural Networks*, pages 3319–3324, 2008.

[Xi, 2020] Bowei Xi. Adversarial machine learning for cybersecurity and computer vision: Current developments and challenges. *Wiley Interdisciplinary Reviews: Computational Statistics*, 12(5):e1511, 2020.

[Zhang *et al.*, 2021] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.