Things You May Not Know About Adversarial Example: A Black-box Adversarial Image Attack

Yuchao Duan
Nanjing University

Zhe Zhao
ShanghaiTech University

Lei Bu
Nanjing University

Fu Song
ShanghaiTech University

ABSTRACT

Numerous methods for crafting adversarial examples were proposed recently with high attack success rate. Most of the existing works normalize images into a continuous vector, domain firstly, and then craft adversarial examples in the continuous vector space. However, "adversarial" examples may become benign after de-normalizing them back into discrete integer domain, known as the discretization problem. The discretization problem was mentioned in some work, but was despised and have received relatively little attention.

In this work, we conduct the first comprehensive study of this discretization problem. We theoretically analyzed 34 representative methods and empirically studied 20 representative open source tools for crafting discretization images. Our findings reveal that almost all of existing works suffer from the discretization problem and the problem is far more serious than we thought. For instance, almost all of black-box attack methods downgrade to white-box ones, methods having higher success rates drop down to lower high success rates, e.g., from 100% to 10%, and there are 10 tools out of 20 tools whose attack successor rate gap between continuous vector space and discrete integer space exceeds 50%. This suggests that the discretization problem should be taken into account when crafting adversarial examples. As a first step towards addressing the discretization problem, we propose a black-box attack method to encode the adversarial example searching problem as a derivative-free optimization problem. Our method is able to craft "real" adversarial images by derivative-free search on the discrete integer domain. Experimental results show that our method achieves significantly higher attack success rates on the discrete integer domain than most of the other tools, no matter white-box or black-box. Moreover, our method is able to handle any model that is not differentiable and we successfully break the winner of NIPS 17 competition on defense with a 95% success rate.

CCS CONCEPTS

•**Security and privacy** → **Usability in security and privacy;**
•**Computing methodologies** → *Machine learning;*

KEYWORDS

Adversarial examples; deep neural networks; discretization; black-box attacks; derivative-free optimization

Anonymous Submission to ACM CCS 2019, London, UK
2019. 978-x-xxxx-xxxx-x/YY/MM…$15.00
DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In the past 10 years, machine learning algorithms, fueled by massive amounts of data, achieve human-level performance or better on a number of tasks. Models produced by machine learning algorithms, especially deep neural networks, are increasingly being deployed in a variety of applications, including safety-critical applications, such as autonomous driving [3, 29, 77], medical diagnostics [16, 55, 64], speech processing [28], computer vision [34, 39], robotics [43, 82], natural language processing [2, 57], and cyber-security [60, 65, 68].

In the early stage of machine learning, people pay more attention on the basic theory and application research, although it is known in 2004 that machine learning models are often vulnerable to adversarial manipulation of their input intended to cause misclassification [17]. In 2014, Szegedy et al. proposed the concept of adversarial example for the first time in deep learning and deep neural networks setting [73]. By adding a subtle perturbation to the input of the deep neural network, it results in a misclassification with high confidence. Moreover, a relatively large fraction of adversarial examples can be used to attack all different models that have different architectures and training data.

Since the findings of Szegedy et al. [73], a plethora of studies have shown that the state-of-the-art deep neural networks suffer from adversarial example attacks which can lead to severe consequences when applied to real-world applications [4, 6, 8–10, 12–14, 22, 24, 31, 32, 37, 40, 44, 49, 51, 53, 54, 56, 63, 75, 78, 80, 83]. In the literature, there are mainly two types of complementary techniques: testing based [4, 6, 8, 14, 22, 31, 32, 40, 46, 49, 51, 53, 56, 73, 75, 78] and verification based [23, 25, 35, 56, 58, 66, 67, 78] for crafting adversarial examples. According to the adversary's knowledge and capabilities, both white-box [4, 22, 23, 25, 35, 46, 49, 51, 56, 56, 58, 66, 73] and black-box [6, 8, 14, 31, 32, 40, 53, 75, 78, 78] attacks have been proposed.

Prior explorations of adversarial examples largely focus on digital adversarial examples [10, 14, 23, 24, 49, 51, 78]. More recently, researchers started to study physical adversarial examples. Kurakin et al. showed that there is a big gap between adversarial examples in the digital world and in the physical world, which means the adversarial perturbations that generalize well in the digital world may not generalize to the physical world [40].

In this work, we refine digital adversarial examples into adversarial *vector* examples: adversarial examples in continuous real domain (e.g., $[0, 1]^m$), and adversarial *image* examples: adversarial examples concrete domain (e.g., $\{0, \cdots, 255\}^m$). Traditional attacks craft adversarial vector examples and then transform them into concrete domain via discretization, namely, transforming floating-point numbers back into the concrete domain. However, examples

transformed from adversarial vector examples may become benign, which is so-called the discretization problem [10]. This discretization problem was initially observed by Goodfellow et al. [24] and Papernot et al. [54]. Two follow-up papers [10, 54] also mentioned this. However, it seems that the discretization problem was underestimated in the literature. For instance, Carlini and Wanger [10] stated that "*This rounding will slightly degrade the quality of the adversarial example.*"

In this work, we conduct a systematic study on the discretization problem on adversarial examples in image classification domain which has a plethora of studies [1]. We firs discuss the difference between adversarial *vector* examples and adversarial *image* examples, as well as distance metrics which are commonly used to approximate human's perception of visual difference. Then, we revise the definition of black-box attacks. We found that almost all the traditional black-box attacks that are assumed to have access to the transformation between image and vector images.

Next, we conduct a comprehensive theoretical analysis of 34 representative white-based and black-based adversarial attacks. We study whether these attacks craft adversarial image examples or adversarial vector examples, whether there is gap between the crafted adversarial vector examples and the examples after discretization, and whether the authors are aware of the discretization problem and take it into account. We find that 26 out of 34 works are affected by the discretization problem and several works that are claimed as black-box attacks downgrade to white-box attacks. In order to confirm that these works do be affected by the discretization problem in real world, we carry out a comprehensive empirical evaluation of 20 representative open source tools for crafting adversarial vector examples. Our empirical evaluation shows that 16 out of 20 tools are affected, and many of adversarial vector examples become benign after transforming them back into concrete domain. Moreover, there are 10 tools whose gaps of the attack success rate between adversarial vector examples and adversarial image examples exceed 50%. *Our study reveals that the discretization problem is far more serious than we thought and suggests that the discretization problem should be taken into account when crafting digital adversarial examples, measuring distance and success rate of attacks.*

A naive idea to avoid the discretization problem is to guarantee that no round error occurs when transforming the adversarial vector examples back into concrete domain. However, this requires access to the transformation between image and vector become grey-box attacks, which is not always feasible in practice. A "real" black-box attack should not have access to the model, as well as the the transformation between image and vector images. Existing black-box attacks that reply on the transferability property of adversarial examples may can avoid this problem, but require a model that is similar to the target and re-trained on a similar dataset.

As the second main contribution of this work, we propose a black-box algorithm for crafting adversarial image examples for both target and untargeted attacks. Our attack only requires access to the probability distribution of classes for each test input, hence is a real black-box attack. In order to craft adversarial image examples in black-box setting, we compute integer perturbations of images in discrete domain, which always results in valid images. We formalize the computation of adversarial image examples as a black-box discrete optimization problem constrained with a distance threshold of $\mathbb{L}_\infty$ distance metric, where the $\mathbb{L}_\infty$ norm is defined in discrete domain. However, this discrete optimization problem cannot be solved using gradient-based methods, as the model is non-continuous. In this work, we leverage a model-based derivative-free discrete optimization method that does not rely on the gradient of the objective function, but instead, learns from samples of the search space. It is suitable for optimizing functions that are non-differentiable, with many local minima, or even unknown but only testable.

We demonstrate the effectiveness of our attacks on the MNIST dataset [42], a digit-recognition task (0-9), using the LeNet classifier [41]; and the ImageNet dataset [18], a large-image recognition task with 1000 classes, using the ResNet50 [26] and InceptionV3 [72] classifiers. Our attack achieves a comparable success rate against to popular white-box based attacks: Fast Gradient Sign Method (FGSM) [24] and C&W [10], and significantly outperforms popular "black"-box based attacks: ZOO [12], decision-based attack [8], query-limited attack NES-PGD [32], FGSM [24] and C&W [10] with a substitute model. In terms of efficiency, our attack is on the same level of time usage as the other black-box methods. Nevertheless, the query time of our method is significantly smaller than NES-PGD and Bandit [33], which are specially designed for query-limited scenarios. Moreover, our attack is able to break the HGD defense [45], which won the first place of NIPS 2017 competition on defense against adversarial attacks, with 95% success rate. Our attack also achieves the so-far best success rate of white-box attacks in the online MNIST Adversarial Examples Challenge.

Our contributions in this paper include:

- We conduct a comprehensive theoretical analysis of 34 representative white-based and black-based adversarial attacks against the discretization problem. 26 out of 34 works are affected by the discretization problem. Almost all the black-box attacks downgrade to white-box.
- We carry out a comprehensive empirical evaluation of 20 representative white-based and black-based open source tools. 16 out of 20 tools have a gap of the success rate of attacks between adversarial vector examples and adversarial image examples, and the gap of 10 tools exceeds 50%.
- We propose a "real" black-box algorithm for crafting adversarial image examples for both target and untargeted attacks by leveraging a derivative-free discrete optimization method. Our method can attack any machine learning models that not differentiable.
- We show that our attack can achieve better success rate on the real image attack, compared to the existing popular white-box and black-box tools.
- We demonstrate that our attack can be used as a query-efficient black-box attack. It uses 600 to 2000 query times less than [32] and [33], which are specially designed for query-limited scenarios.
- Our attack is able to break the HGD defense [45], which won the first place of NIPS 2017 competition on defense against adversarial attacks, with 95% success rate. Our attack can also achieve the same result as the best white-box attacks in MNIST Challenge.

[1] Learning algorithms on other problems using discretization may have similar drawback. We leave these for future work.

Notations	Description
w, h, ch	the width, height, and number of channels of an image
P	the set of coordinates $w \times h \times ch$.
$\mathbb{V}$	the continuous domain of vector images $\vec{v}$, e.g., $\mathbb{R}_{[0,1]}^{w \times h \times ch}$.
$\vec{v}[p] = (r_1, \cdots, r_{ch})$	the element at the coordinates p of a vector image $\vec{v} \in \mathbb{V}$.
$\mathbb{D}$	the discrete domain of (valid) images $\vec{d}$, e.g., $\mathbb{N}_{[0,255]}^{w \times h \times ch}$.
$\vec{d}[p] = (r_1, \cdots, r_{ch})$	the element at the coordinate p of an image $\vec{d} \in \mathbb{D}$.
$\mathbb{T} : \mathbb{D} \to \mathbb{V}$	the *normalizer* that transforms an image into a vector image.
$\mathbb{T}^{-1} : \mathbb{V} \to \mathbb{D}$	the denormalizer that transforms a vector image to an image such that $\forall \vec{d} \in \mathbb{D},\ \mathbb{T}^{-1}(\mathbb{T}(\vec{d})) = \vec{d}$.
$\mathbb{C}_t$	the set of mutually exclusive classes for the classification task t.
$\vec{v}^{\mathsf{adv}} \in \mathbb{V}$	adversarial vector example.
$\vec{d}^{\mathsf{adv}} \in \mathbb{D}$	adversarial image example.

Table 1: Notations used in this paper

To the best of our knowledge, this is the first comprehensive study of the discretization problem on adversarial examples.

2 BACKGROUND

In this section, we recap the traditional basic notations of deep learning based image classification, adversarial example attacks and distance metrics.

2.1 Deep Learning based Image Classification

For convenient reference, we summarize the notations in Table 1.

The goal of an image classification task t is to construct a classifier $f_t : \mathbb{D} \to \mathbb{C}_t$. With the normalizer $\mathbb{T}$, the goal can be seen as the construction of a classifier $g_t : \mathbb{V} \to \mathbb{C}_t$, namely, $f_t = g_t \circ \mathbb{T}$.

Deep learning methods are used to compute a classifier $\widehat{g_t} : \mathbb{V} \to \mathbb{C}_t$ as an approximation of g_t. Consider the supervised learning task t, let $(d, c_d)_{d \in D}$ be a finite set of labelled data, where $c_d \in \mathbb{C}_t$ denotes the class of the image $d \in D$. The labelled dataset $(d, c_d)_{d \in D}$ is normalized into the dataset $(v, c_v)_{v \in V}$ by the normalizer $\mathbb{T}$, namely $V = \{\mathbb{T}(d) \mid d \in D\}$ and $c_v = c_d$ if $c = \mathbb{T}(d)$ for every $d \in D, v \in V$. A deep neural network is trained and tuned based on labelled dataset $(v, c_v)_{v \in V}$, which results in the approximate classifier $\widehat{g_t}$. Finally, the classifier f_t is approximated by the function $\widehat{f_t} := \widehat{g_t} \circ \mathbb{T}$. Figure 1 depicts the relationship of these notations for convenient.

2.2 Adversarial Vector Example Attacks

Given a classifier $\widehat{g_t}$ and a vector image $\vec{v} \in \mathbb{V}$ that may be obtained from an image $\vec{d} \in \mathbb{D}$ via the normalizer $\mathbb{T}$, an *adversarial vector example* for $\vec{v}$ and $\widehat{g_t}$ is an image $\vec{v}^{\mathsf{adv}} \in \mathbb{V}$ such that

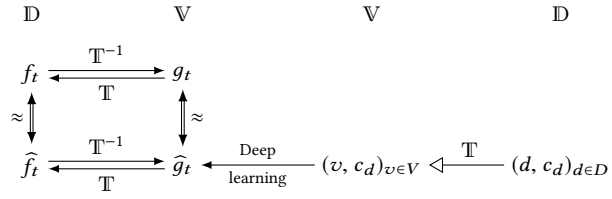

Figure 1: Relationship of classifiers in continuous and discrete domains, where $\xleftarrow{\mathbb{T}}$ denotes applying the normalizer $\mathbb{T}$ on $(d, c_d)_{d \in D}$ and other arrows labelled by $\mathbb{T}$ or $\mathbb{T}^{-1}$ denote function composition.

$$\widehat{g_t}(\vec{v}) \neq \widehat{g_t}(\vec{v}^{\mathsf{adv}}),$$

i.e., the classifier $\widehat{g_t}$ misclassifies the example $\vec{v}^{\mathsf{adv}}$.

An *untargeted attack* of the classifier $\widehat{g_t}$ is to craft an adversarial vector example $\vec{v}^{\mathsf{adv}}$ for a given image $\vec{v} \in \mathbb{V}$. A more powerful attack, *targeted attack*, for a given target class c is to craft an adversarial vector example $\vec{v}^{\mathsf{adv}}$ such that $\widehat{g_t}(\vec{v}^{\mathsf{adv}}) = c$. It is often expected that the adversarial vector example $\vec{v}^{\mathsf{adv}}$ and the original example $\vec{v}$ are visually indistinguishable by humans.

Depending on the knowledge of the classifier $\widehat{g_t}$ by the adversary, adversarial example attacks are broadly categorized into white-box and black-box attacks. White-box attack knows the parameters and architecture of $\widehat{g_t}$, even defense and detection algorithms if they exist, while black-box attack can only access to the probabilities of top-k classes for each test input. In a less powerful black-box attack scenario, the number of queries is also limited.

2.3 The Discretization Problem

Given an image $\vec{d} \in \mathbb{D}$, and a classifier $\widehat{g_t}$ with the normalizer $\mathbb{T}$ and denormalizer $\mathbb{T}^{-1}$, suppose we have crafted an adversarial vector example $\vec{v}^{\mathsf{adv}}$ from the vector image $\mathbb{T}(\vec{d})$, i.e.,

$$\widehat{g_t}(\mathbb{T}(\vec{d})) \neq \widehat{g_t}(\vec{v}^{\mathsf{adv}}).$$

To show or store $\vec{v}^{\mathsf{adv}}$, $\vec{v}^{\mathsf{adv}}$ will be transformed into the image $\mathbb{T}^{-1}(\vec{v}^{\mathsf{adv}})$. The *discretization problem* occurs when

$$\mathbb{T}(\mathbb{T}^{-1}(\vec{v}^{\mathsf{adv}})) \neq \vec{v}^{\mathsf{adv}}.$$

This discretization problem may results in severe consequence, namely,

$$\widehat{g_t}(\mathbb{T}(\mathbb{T}^{-1}(\vec{v}^{\mathsf{adv}}))) \neq \widehat{g_t}(\vec{v}^{\mathsf{adv}}),$$

which further, results in failure of targeted/untargeted attacks, i.e.,

$$\widehat{g_t}(\mathbb{T}(\mathbb{T}^{-1}(\vec{v}^{\mathsf{adv}}))) = \widehat{g_t}(\mathbb{T}(\vec{d})) \text{ or } \widehat{g_t}(\mathbb{T}(\mathbb{T}^{-1}(\vec{v}^{\mathsf{adv}}))) \neq c.$$

where c denotes the target class. Therefore, we introduce the concept of adversarial image examples.

2.4 Adversarial Image Example Attacks

Due to the discretization problem, an image classifier should be defined as a pair $(\widehat{g_t}, \mathbb{T})$ consisting of an classifier $\widehat{g_t}$ and its normalizer $\mathbb{T}$ rather than the individual classifier $\widehat{g_t}$.

Given a classifier $(\widehat{g_t}, \mathbb{T})$ and an image $\vec{d} \in \mathbb{D}$, an *adversarial image example* for $\vec{d}$ is an image $\vec{d}^{\mathsf{adv}} \in \mathbb{D}$ such that

$$\widehat{g_t}(\mathbb{T}(\vec{d})) \neq \widehat{g_t}(\mathbb{T}(\vec{d}^{\mathsf{adv}})).$$

Consequently, an untargeted attack of a classifier $(\widehat{g}_t, \mathbb{T})$ is to craft an adversarial image example $\vec{d}^{\mathsf{adv}}$ for a given image $\vec{d} \in \mathbb{D}$. A targeted attack for a given target class c is to craft an adversarial image example $\vec{d}^{\mathsf{adv}}$ such that

$$\widehat{g}_t(\mathbb{T}(\vec{d}^{\mathsf{adv}})) = c.$$

In terms of white-box, the adversary may knows the normalizer, denormalizer, parameters and architecture of the classifier, then, the adversary can search adversarial image examples of $\vec{d}$ by first searching adversarial vector examples $\vec{v}^{\mathsf{adv}}$ of $\mathbb{T}(\vec{d})$ and then checking whether $\mathbb{T}^{-1}(\vec{v}^{\mathsf{adv}})$ is adversarial or matches the targeted class. However, for black-box attacks, the adversary can only access to the probabilities of top-k classes for each test input, then the adversary cannot search adversarial image examples in vector domain, unless the adversary knows the normalizer and denormalizer, hence, downgrades to white-box. (Indeed, it downgrades to grey-box. In this work, we regard grey-box as white-box for simplifying the presentation.)

2.5 Distance Metrics

The distortion of adversarial examples should be visually indistinguishable from their normal counterparts by humans. However, it is hard to model human perception, hence several distance metrics were proposed to approximate human's perception of visual difference. In the literature, there are three common distance metrics $\mathbf{L}_0$, $\mathbf{L}_1$, $\mathbf{L}_2$ and $\mathbf{L}_\infty$ which are usually defined over samples in continuous domain $\mathbb{V}$. All of them are $\mathbf{L}_n$ norms which is defined as

$$\|\vec{v} - \vec{v}^{\mathsf{adv}}\|_n = \left(\sum_{p \in P} |\vec{v}[p] - \vec{v}^{\mathsf{adv}}[p]|^n\right)^{-n},$$

where $\vec{v}, \vec{v}^{\mathsf{adv}} \in \mathbb{V}$.

In more detail, $n = 0$ for $\mathbf{L}_0$ counts the number of different pixels, i.e., $\sum_{p \in P}(\vec{v}[p] \neq \vec{v}^{\mathsf{adv}}[p])$. $n = 1$ for $\mathbf{L}_1$ denotes the sum of absolute differences of each pixel value, i.e., $\sum_{p \in P}(|\vec{v}[p] - \vec{v}^{\mathsf{adv}}[p]|)$. $n = 2$ for $\mathbf{L}_2$ denotes Euclidean or squared root distance. $n = \infty$ for $\mathbf{L}_\infty$ measures the largest change introduced. Remark that

$$\lim_{n \to \infty} \|\vec{v} - \vec{v}^{\mathsf{adv}}\|_n = \max\{(\vec{v}[p] - \vec{v}^{\mathsf{adv}}[p]) \mid p \in P\}.$$

However, due to the discretization problem, for a given distance metric $\mathbf{L}_n$, $\|\vec{v}^{\mathsf{adv}} - \mathbb{T}(\vec{d})\|_n$ may differ from $\|\mathbb{T}(\mathbb{T}^{-1}(\vec{v}^{\mathsf{adv}})) - \mathbb{T}(\vec{d})\|_n$. Consequently, it is not reasonable to approximate human's perception of visual difference using distance metrics defined between adversarial vector examples. Instead, we think it is much better to measure the distance between images rather than vector images. For this purpose, we revise distance metrics and introduce $\mathbb{L}_p$ norm which is defined between images directly. Formally, $\mathbb{L}_n$ is defined as follows:

$$\|\vec{d} - \vec{d}^{\mathsf{adv}}\|_n = \left(\sum_{p \in P} |\vec{d}[p] - \vec{d}^{\mathsf{adv}}[p]|^n\right)^{-n},$$

where $\vec{d}, \vec{d}^{\mathsf{adv}} \in \mathbb{D}$. Accordingly, we can define $\mathbb{L}_0 = \|\vec{d} - \vec{d}^{\mathsf{adv}}\|_0$, $\mathbb{L}_1 = \|\vec{d} - \vec{d}^{\mathsf{adv}}\|_1$, $\mathbb{L}_2 = \|\vec{d} - \vec{d}^{\mathsf{adv}}\|_2$ and $\mathbb{L}_\infty = \|\vec{d} - \vec{d}^{\mathsf{adv}}\|_\infty$. Obviously, $\mathbb{L}_n$ differs from $\mathbf{L}_n$ for any n.

3 DISCRETIZATION PROBLEM IN THE WILD

In this section, we first conduct a comprehensive theoretical study on 34 of representative testing based and verification based methods, then report a comprehensive empirical study on 19 of representative open source tools for crafting adversarial examples, with respect to the discretization problem.

3.1 Theoretical Study

Let us fix a classifier $\widehat{g}_t$ for an image classification task t, normal vector image $\vec{v} \in \mathbb{V}$ from an image $\vec{d} \in \mathbb{D}$ using a normalizer $\mathbb{T}$ and the target class c for targeted attack. We assume that $\widehat{g}_t(\vec{v}) = c_{\vec{v}}$ and $\mathtt{loss} : \mathbb{V} \times \mathbb{C}_t \to \mathbb{R}$ is the continuous loss function of $\widehat{g}_t$.

We theoretically analyze existing works in the follows aspects: testing vs. verification, and black-box vs. white-box. The summary of results is given in Table 2 and Table 3.

3.1.1 **Testing based White-box Methods.** We classify testing based methods along two dimensions: gradient-based and non-gradient-based methods.

Gradient-based methods. Goodfellow et al. [24] proposed the first and fastest gradient-based methods (named FGSM). By linearizing the loss function $\mathtt{loss}(\vec{v}^{\mathsf{adv}}, c_{\vec{v}})$, FGSM perturbs an image by maximizing the loss subject to a $\mathbf{L}_\infty$ constraint: $\vec{v}^{\mathsf{adv}} = \vec{v} + \epsilon\,\mathrm{sign}(\nabla_{\vec{v}}\mathtt{loss}(\vec{v}, c_{\vec{v}}))$, where the hyper-parameter ϵ denotes changed value for each pixel and $\mathrm{sign}(\cdot)$ is the sign function. Inspired by FGSM, several gradient-based methods were proposed. Kurakin et al. [40] introduced an extension of FGSM, called *Basic Iterative Method* (named BIM), that iteratively takes multiple small steps while adjusting the direction after each step. Madry et al. [47] proposed to apply the *projected gradient descent* algorithm with random start (named PGD). Kurakin et al. [40] proposed *Iterative Least-Likely Class Method* (named ILLC) for targeted attack which does gradient descent on the benign image. Similar idea can be applied to FGSM and BIM for targeted attacks. Dong et al. [19] integrated the momentum techniques into BIM and ILLC (named MBIM and MILLC) for the purpose of stabilizing the updating direction and escaping from poor local maximum during iterations. Pei et al. [56] introduced neuron coverage and a gradient-based algorithm to solve joint optimization problem (named DeepXplore), in order to craft corner adversarial vector examples. Inspired by DeepXplore, Sun et al. [70] (named DeepCover) and Ma et al. [46] (named DeepGauge) introduced more test coverage criteria and craft adversarial vector examples guided by their coverage criteria using gradient-based algorithms.

BIM [40], ILLC [40] and their extensions MBIM and MILLC [19] craft adversarial samples in discrete domain $\mathbb{V}$, yielding adversarial image examples, therefore, not affected by the discretization problem. However, to our knowledge, all the BIM implementations we found actually craft adversarial *vector* examples. FGSM searches adversarial examples in continuous domain $\mathbb{V}$, it seems that they are aware of the discretization problem, as the perturbation ϵ of $\mathbf{L}_\infty$ is sometimes selected corresponding to the magnitude of the smallest bit of an 8-bit image, e.g., $\epsilon = 8/255$ on CIFAR-10 dataset. But, it is still a bit confusing why 0.1 and other similar decimal ϵ is used on image related dataset, under $\mathbf{L}_\infty$ norm, which do not correspond to the magnitude of the smallest bit of an 8-bit image. Moreover, ϵ in default is 0.3 in several tutorials of FGSM on the MNIST and CIFAR10 in cleverhans[2] maintained by Google Inc., OpenAI and Pennsylvania State University. The similar problem

[2] https://github.com/tensorflow/cleverhans

		Reference	(Un)targeted	Domain	Distance	Affected	B→W	Gap
Testing based methods	White-box	FGSM [24]	Untargeted	Continuous	$\mathbf{L}_\infty$	✗	-	✗
		BIM [40]	Untargeted	Discrete	$\mathbb{L}_\infty$	✗	-	✗
		PGD [47]	Untargeted	Continuous	$\mathbf{L}_\infty$	✓	-	✓
		ILLC [40]	Targeted	Discrete	$\mathbb{L}_\infty$	✗	-	✗
		MBIM/MILLC [19]	Untargeted/Targeted	Discrete	$\mathbb{L}_2$, $\mathbb{L}_\infty$	✗	-	✗
		C&W [10]	Both	Continuous	$\mathbf{L}_0$, $\mathbf{L}_2$, $\mathbf{L}_\infty$	✓	-	✓
		OptMargin [27]	Untargeted	Continuous	$\mathbf{L}_\infty$	✓	-	✓
		EAD [11]	Both	Continuous	$\mathbf{L}_1$, $\mathbf{L}_2$	✓	-	✓
		DeepXplore [56]	Untargeted	Continuous	$\mathbf{L}_1$	✓	-	✓
		DeepCover [70]	Untargeted	Continuous	$\mathbf{L}_\infty$	✓	-	✓
		DeepGauge [46]	Untargeted	Continuous	$\mathbf{L}_\infty$	✓	-	✓
		BLB [73]	Targeted	Continuous	$\mathbf{L}_2$	✓	-	✓
		JSMA [54]	Targeted	Continuous	$\mathbf{L}_0$	✗	-	✗
		DeepFool [48]	Untargeted	Continuous	$\mathbf{L}_{p\geq 1}$	✓	-	✓
		UAP [49]	Untargeted	Continuous	$\mathbf{L}_{p\geq 1}$	✓	-	✓
		DeepConcolic [71]	Untargeted	Continuous	$\mathbf{L}_0$, $\mathbf{L}_\infty$	✗	-	✗
	Black-box	PMGJCS [53]	Both	Continuous	$\mathbf{L}_1$	✓	✓	✓
		PMG [52]	Untargeted	Continuous	$\mathbf{L}_1$	✓	✓	✓
		One pixel [69]	Both	Discrete	-	✗	✗	✗
		LSADV [50]	Both	Continuous	-	✓	✓	✓
		ZOO [12]	Targeted	Continuous	$\mathbf{L}_2$	✓	✓	✓
		FD [7]	Both	Continuous	$\mathbf{L}_2$, $\mathbf{L}_\infty$	✓	✓	✓
		NES-PGD [32]	Targeted	Continuous	$\mathbf{L}_\infty$	✓	✓	✓
		DBA [8]	Targeted	Continuous	$\mathbf{L}_2$	✓	✓	✓

Table 2: Summary of theoretical study results (I). *(Un)targeted* column shows the type of attack. *Affect* column shows whether the method is affected by the discretization problem. *Domain* column shows the domain of images. *Distance* column shows the considered distance metrics. *B→W* column shows whether black-box downgrades to white-box or not. *Gap* column shows whether there is a gap between the crafted vector image and (valid) image.

	Reference	Domain	Distance	W/B-box	Completeness	Affected	C→InC	B→W	Gap
Verification based methods	BILVNC [5]	Continuous	$\mathbf{L}_\infty$	White	✓	✓	✓	-	✓
	DLV [30]	Continuous	$\mathbf{L}_1$, $\mathbf{L}_2$	White	✗	✗	-	-	✗
	Planet [21]	Continuous	-	White	✓	✓	✓	-	✓
	MIPVerify [74]	Continuous	$\mathbf{L}_\infty$	White	✓	✓	✓	-	✓
	DeepZ [66]	Continuous	$\mathbf{L}_\infty$	White	✗	✓	-	-	✓
	DeepPoly [67]	Continuous	$\mathbf{L}_\infty$	White	✗	✓	-	-	✓
	DeepGo [61]	Continuous	-	White	✗	✓	-	-	✓
	SaveCV [78]	Continuous	$\mathbf{L}_n$	Black	✗	✓	-	✓	✗
	ReluVal [76]	Continuous	$\mathbb{L}_n$	White	✓	✓	✓	-	✓
	DSGMK [20]	Continuous	$\mathbf{L}_\infty$	White	✗	✓	-	-	✓

Table 3: Summary of theoretical analysis results (II). *Completeness* column shows whether the method is complete or not. *C→InC* column shows whether complete methods downgrades to incomplete.

also exists in PGD [47], DeepXplore [56], DeepCover [70], EAD [11] and DeepGauge [46]. Indeed, the discretization problem of these works can be fundamentally eliminated by selecting proper perturbation step size so that the adversarial vector examples are still adversarial after denormalization.

Carlini & Wanger [10] proposed a set of white-box attacks (named C&W) for $\mathbf{L}_0$, $\mathbf{L}_2$ and $\mathbf{L}_\infty$. They formalize the problem as an appropriate optimization problem to search for high confidence adversarial vector examples with small magnitude of perturbation. The optimization problem is solved by the Adam [36], a first-order gradient-based optimizer for stochastic objective functions. Although they are aware of the discretization problem and hence the attack success rate is measured on adversarial image examples. However, their algorithms ignored the discretization problem and claimed that "*it slightly degrades the quality of the adversarial example*". Moreover, we find that their tool crafts adversarial vector examples without performing denormalization. Our experiments show that there is a big gap (i.e., 90% for $\mathbf{L}_\infty = 10$) between adversarial vector examples and adversarial image examples crafted by their tool, countering-claim to statements in [10]. Similar problem exists in OptMargin [27] which leverages the algorithms of [10]. Although as mentioned in [10] this problem could be eliminated by performing a greedy search on the lattice defined by the discrete solutions by changing one pixel value at a time. To our knowledge, this idea has not been implemented, hence it's performance in terms of time is unclear.

Non-gradient-based methods. Szegedy et al. [73] proposed the first white-box non-gradient-based attacks using a box-constrained L-BFGS algorithm (named BLB). Moosavi-Dezfooli et al. [48] proposed to search for the closest distance from the source image to the decision boundary of the target model (named DeepFool). It iteratively first calculates decision boundaries of other classes and find the closest one, then calculate the distance r and move to cross the decision boundary. Based on DeepFool, Moosavi-Dezfooli [48] also proposed an universal adversarial perturbation (named UAP) attack which can be used to misclassify almost all images from the dataset.

All these methods craft examples in continuous domain without considering the discretization problem. In fact, DeepFool [48] assumed that the classifier $\widehat{f}_t$ in continuous domain is the same as the classifier in the concrete domain $\widehat{g}_t$, which contradicts to our empirical results. It is possible to avoid the discretization problem by checking examples after denormalization, as done by JSMA [54] and DeepConcolic [71]. By doing this, it may spend a lot of time at crafting *spurious* adversarial examples.

3.1.2 **Testing based Black-box Methods.** Papernot et al. [53] proposed the first black-box method by leveraging transferability property of adversarial examples. It first trains a local substitute model with a synthetic dataset and then crafts adversarial examples from the local substitute model. [52] generalized this idea to attack other machine learning classifier. However, transferability is not always reliable, other methods such as gradient estimation are explored as alternatives to substitute networks. Narodytska & Kasiviswanathan [50] proposed a gradient estimation attack using a local-search based technique (named LSADV). Chen et al. [12] proposed a black-box attack method (named ZOO) with zeroth order optimization. In order to reduce the number of queries, Bhagoji et al. [7] proposed a class of black-box attacks (called FD) that approximates FGSM and BIM via gradient estimation. Independently, Ilyas et al. [32] proposed an alternative gradient estimation method by leveraging NES [62, 79] and employ white-box PGD attack with estimated gradient(named NES-PGD). Brendel et al. [8] proposed a decision-based attack (named DBA), that starts from the target image, moves a small step to raw image every time and check the perturbation cross the decision boundary or not. Su et al. [69] proposed a black-box attack for generating one-pixel adversarial images based on differential evolution.

All these black-box methods (except [69]) have access to the normalizer of the targeted classifier and the adversarial examples are crafted in continuous domain without considering the discretization problem, therefore, black-box attacks (except for [69]) downgrade to white-box ones and crafted adversarial examples may be benign after the discretization post-processing. In contrast, [69] crafts adversarial images directly in discrete domain. We remark that the discretization problem in substitute model based [52, 53] and gradient estimation based [7, 32] methods could be fundamentally eliminated by choosing proper step sizes for the underlying white-box methods, but it is non-trivial for ZOO [12] and DBA [8] due to the underlying algorithms.

3.1.3 **Verification based Methods.** Verification based methods were proposed to provide reliable guarantees on the robust of deep learning classifiers by formal verification. When a neural network is proved non-robust, an adversarial example could be crafted as a *witness*. A verification method/tool is *complete*, if all the robust classifiers can pass the verification, i.e., has no false positives; otherwise it is *incomplete*. In this section, we analyze the discretization problem on verification based methods for image classification networks, hence other works such as NeVer [58] and Reluplex [35] are excluded.

Complete methods includes BILVNC [5], ReluVal [76], Planet [21] and MIPVerify [74], which typically leverage linear programming, SMT solving and iterative refinement. Incomplete methods include DLV [30], DeepZ [66], DeepPoly [67], DeepGo [61], SaveCV [78] and DSGMK [20], which typically verify an over-approximation of the classifier by leveraging abstract interpretation, linear approximations and duality.

The complete methods BILVNC [5], MIPVerify [74] and Planet [21] consider the robust problem of classifiers in continuous domain, hence indeed is an over-approximation of classifiers in discrete domain. In other words, adversarial vector examples crafted by these complete verification tools as the witness may be benign after denormalization. This means that such complete methods are incomplete for the classifiers in discrete domain. To fundamentally eliminate this problem, the search space should be discretized as done in DLV [30].

Although MIPVerify [74] studied the robust problem of classifiers in continuous domain. Indeed, they use symbolic interval to over approximate the value ranges of pixels without normalization and refine the output interval by repeated splitting of the input intervals. They did not consider non-integer values when verifying the MNIST network, resulting in spurious adversarial examples, hence becomes incomplete. This problem could be fundamentally eliminated by excluding non-integers during interval refinement.

Furthermore, incomplete methods (except DLV [30]) also suffer from the discretization problem. Indeed, the black-box incomplete method, SaveCV [78], becomes white-box, although they used denormalization before checking crafted samples. While, other white-box incomplete methods do not use any denormalization, hence may craft *spurious* adversarial examples. DLV [30] attempts to prove local robustness in a neighborhood of x by means of discretization: they reduce the infinite neighborhood into a finite set of points, and check that the labeling of these points is consistent. This process is then propagated through the network, layer by layer. The discretization used during verification guarantees that the crafted adversarial examples are indeed adversarial.

3.2 Empirical Study

In this section, we conduct an empirical study on 19 of representative open source tools (listed in the first column in Table 4) that can craft adversarial example, in an attempt to understand the discretization problem in the real-world. We consider two research questions:

RQ1: Does perturbation step size affect attack success rate due to the discretization problem?

RQ2: To what extent do the discretization problem affect the attack success rate?

RQ3: Is the discretization problem ubiquitous?

Methods	SR	TSR	GAP	Dataset	Model	Tool & Parameters
FGSM [24]	98.61%	98.58%	0.03%	MNIST	MNIST	Cleverhans, 10000 images, $\epsilon = 0.3$
BIM [40]	100%	14%	86%	ImageNet	ResNet	FoolBox, $\epsilon = 0.3$, stepsize=0.05
MBIM [19]	100%	14%	86%	ImageNet	ResNet	FoolBox, $\epsilon = 0.3$, stepsize=0.06
C&W-$\mathbf{L}_2$ [10]	100%	10%	90%	ImageNet	ResNet	FoolBox
C&W-$\mathbf{L}_\infty$ [10]	100%	68%	32%	ImageNet	ResNet	Original tool, $\mathbf{L}_\infty = 10$
OptMargin[27]	100%	90%	10%	Cifar	Original	Original, optens_attack.py
DeepXplore[56]	33%	7%	78.79%	ImageNet	3 Networks	Original, 100 seeds
BLB[73]	100%	49%	51%	ImageNet	ResNet	FoolBox, $\epsilon = 1e-5$, maxiter=150
DeepFool[48]	100%	1%	99%	ImageNet	ResNet	Foolbox
DeepConclic[71]	2%	2%	0%	MNIST	Original	Original, criterion='nc'
One-pixel[69]	3%	3%	0%	MNIST	LeNet1	AdvBox, max_pixels=9
ZOO[12]	73%	3.3%	95.47%	ImageNet	InceptionV3	Original
NES-PGD[32]	100%	77%	23%	ImageNet	InceptionV3	Original, $\mathbf{L}_\infty = 10$
DBA[8]	100%	44%	56%	ImageNet	ResNet	Original
DLV[30]	90%	90%	0%	MNIST	Original	Original, SR not 100% as time out
Planet[21]	100%	46%	54%	MNIST	testNetworkB.rlv	Use 'GIVE' model obtain 20 images
MIPVerify[74]	42%	0%	100%	MNIST	Original	Quickstart demo with 100 images
DeepPoly[67]	45%	44%	2.22%	MNIST	convBigRELU_DiffAI	Use $\epsilon = 0.3$ and $\epsilon = 76$ to compare
DeepGo[61]	25.4%	25.2%	0.78%	MNIST	Original	Generated 1000 images from 1 image
SafeCV[78]	100%	100%	0%	MNIST	Original	100 images

Table 4: Experiment results on the discretization problem, where SR means the attack success rate of adversarial vector examples, TSR means the attack success rate of adversarial examples, GAP means adversarial percentage lost after transfer to image, all parameters used are default parameters provided by the raw paper or third-party libraries.

3.2.1 Dataset and Setting. Due to diversity (e.g., capabilities, efficiencies, types of networks, platform) of these tools, the dataset and targeted classifiers may be different. For ImageNet images [18], we use 100 images that all can be correctly classified by all the keras tools[3]. For MNIST [42] and CIFAR [38] images, the number of used images are shown in the last column in Table 4. All the targeted models are the models provided by the original tools or third-party libraries or trained according to the instructions of the original tools. In all the experiments of attacks, we conduct untargeted attacks, except that we conduct targeted attacks for Planet. For verification tools that cannot directly attack the model, we use their specific setting (see below).

In order to conduct experiments as fair as possible, we manage to be consistent with these papers' original environment. All the parameters we used in our experiments are the default values of the original tools or third-party libraries, such as Cleverhans and FoolBox[4].

Furthermore, we introduce the following metric to evaluate the discretization problem.

Success Rate(SR). We use N_v to denote the number of images considered successful attacked by the corresponding methods and N_a to represent the size of input set. We use N_v/N_a to represent success rate, or in other words, the adversarial vector success rate.

True Success Rate(TSR). We map back the N_v adversarial vector examples to the real discrete image domain and ask the corresponding model to classify them again. We denote the number of success image attack as N_i, and define the ture success rate as N_i/N_a.

GAP We evaluate the gap between SR and TSR as (SR-TSR)/SR

3.2.2 Result and Analysis. Our experimental results shown in Table 4 and Figure 2.

RQ1: As mentioned in previous section, FGSM [24], BIM [40], ILLC [40], MBIM and MILLC [19] can craft adversarial vector examples that are still adversarial after denormalization by using perturbation step sizes corresponding to the magnitudes of the smallest bit of an 8-bit image. Therefore, we conduct experiments using FGSM, BIM and MBIM with the perturbation step size $\epsilon = 0.3$, where 0.3 does not not correspond to the magnitude of the smallest bit of an 8-bit image. The gap of FGSM is small, but the gap of BIM and MBIM is very large as they take multiple small steps and can stop in proper place while adjusting the direction after each step in order to minimize the distortion. We guess that adversarial examples with larger distortions are more robust in terms of the discretization problem. Therefore, perturbation step size does affect attack success rate.

RQ2: The affect of the discretization problem differ in tools. There are ten tools whose gap exceeds 50%, and six tools whose gap exceeds 80%.

On the other hand, the tools that considered the discretization problem, e.g., DeepConcolic, DLV and SafeCV. do not have any gap. One-pixel has no gap as it directly crafts adversarial image examples. Only three tools that did not consider the discretization problem have gap less than 10%.

RQ3: All the tools that did not consider the discretization problem have some gap. These tools implemented gradient-based (such

[3]https://keras.io.
[4]https://foolbox.readthedocs.io/en/latest/index.html

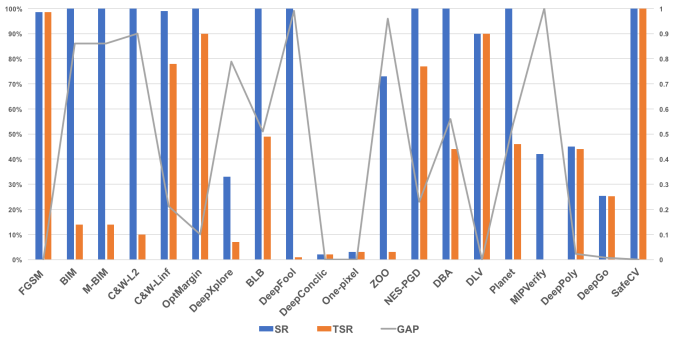

Figure 2: Histogram plot of experiments result

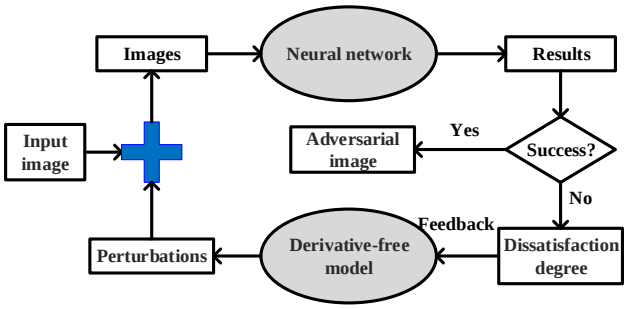

Figure 3: Framework of our approach DFA

as BIM, MBIM, C&W, and DeepXplore) and non-gradient-based (such as BLB, DeepFool and ZOO), and verification-based (such as MIPVerify) methods. Also, some of them are white-box attacks (such as BIM, MBIM, C&W, DeepXplore and MIPVerify) and some of them are black-box attacks (such as ZOO, NES-PGD and DBA) Therefore, the discretization problem is ubiquitous no matter what kind of methods implemented and no matter it is black-box or white-box.

Discussion. Our study reveals that the discretization problem is ubiquitous and more severe than some researchers thought, e.g., [10, 48]. According to our findings, we suggest that attack success rate should be measured using adversarial image examples instead of adversarial vector examples. This can be easily achieved by transforming vector examples into images by denormalization before checking whether the crafted examples are adversarial or not. We suggest that it is better to design and implement methods that directly craft adversarial image examples, which may have better performance in terms of time than that directly craft adversarial vector examples and check them after denormalization. Gradient-based methods can be easily revised by restricting the perturbation step sizes as the magnitudes of the smallest bit of an 8-bit image. However, for non-gradient-based methods, we believe it is non-trivial to adjust due to the following reasons. One reason is that there are too many hyper-parameters which influence the perturbation step size. The another reason is that some tools use third-party solvers to get perturbation step sizes which is much difficult to fix. For verification based tools, adversarial examples should be checked and spurious adversarial examples should be excluded by refinement such as done in ReluVal [76].

4 AN APPROACH FOR BLACK-BOX ATTACK

According to our study in Section 3, all the black-box attacks (except one pixel [69]) downgrade to white-box. Indeed, they assumed that normalization and denormalization can be accessed by the adversary which is not always feasible in practice. While the attack success rate of one pixel [69] is very low compared with other methods.

To craft adversarial image examples in black-box scenario, we can only query the classifier for input images and get the probabilities of top-k classes for each input image. Then, we have to compute integer perturbations of images in discrete domain, which always results in valid images. As a first step towards addressing this problem, in this section, we propose a black-box algorithm for crafting adversarial image examples for both target and untargeted attacks by leveraging a model-based derivative-free discrete optimization method (DFO) [81]. DFO is widely used to solve complex constrained optimization tasks in a sampling-feedback-style. It does not rely on the gradient of the objective function, but instead, learns from samples of the search space. Therefore, it is suitable for optimizing functions that are non-differentiable, with many local minima, or even unknown but only testable.

In the rest of this section, we first introduce the framework of our approach, then present the formulation, and finally show attack algorithm.

Threat model. In our black-box scenario, we assume that the adversary knows the input format of the target image classifier and has access to the probabilities of top-k classes for each input image, but he/she does not have any access to the normalization, denormalization, architecture, parameters and training data of the classifier. The distortion of adversarial examples is measured by $\mathbb{L}_\infty$ distance metric.

4.1 Framework of DFA

Figure 3 shows the framework of our approach named DFA. Given an image, DFA first randomly generates a set of perturbations from the search space specified by the $\mathbb{L}_\infty$ distance. Then it crafts a set of images by adding these perturbations onto the original image. Next, starting from the set of crafted images, DFA repeats the following procedure until an adversarial image example is found (i.e., an image with minimal dissatisfaction degree) or the number of iterations is reached. During each iteration, DFA queries the black-box classifier to assess the crafted images via the given dissatisfaction degree function which specifies the minimal optimization goal. The set of perturbations is partitioned into two sets in the order of their corresponding images' dissatisfaction degrees: one set consisting of perturbations that yield images with high dissatisfaction degrees and the another set consisting of perturbations that yield images with low dissatisfaction degrees. The search space is refined into a sub-space according these two sets. New perturbations are sampled from the sub-space. From these new perturbations together with old perturbations, a set of best-so-far perturbations is selected by adding them into the original image and querying the black-box program

and the dissatisfaction degree function. Finally, the procedure is repeated again on the set of best-so-far perturbations.

4.2 Formulation

In this section, we formalize the adversarial image example searching problem as a derivative-free optimization problem by defining the dissatisfaction degree functions. We first introduce some notations.

Let us fix a classifier $f_t : \mathbb{D} \to \mathbb{C}_t$ for some image classification task t. Given an image $\vec{d}$, we denote $\mathcal{P}(\vec{d})$ the $|\mathbb{C}_t|$-dimensional probability vector of the classifier on the image $\vec{d}$. For every class $c \in \mathbb{C}_t$, let $\mathcal{P}(\vec{d}, c)$ denote the probability that the image $\vec{d}$ belongs to the class c. For a given integer $1 \le j \le |\mathbb{C}_t|$, we denote $\mathtt{Top}_j(\vec{d})$ the j-th largest probability in $\mathcal{P}(\vec{d})$ and $\mathtt{Top}_j^{\ell}(\vec{d})$ the class whose probability is $\mathtt{Top}_j(\vec{d})$. Obviously, $\mathtt{Top}_1^{\ell}(\vec{d}) = f_t(\vec{d})$.

To craft an adversarial image, we need to locate a perturbation δ which is an image in the discrete domain $\mathbb{D}$. Let us fix the $\mathbb{L}_\infty$ distance ϵ. We define the search space Δ of perturbations as the discrete domain $\mathbb{N}_{[-\epsilon,\epsilon]}^{w \times h \times ch}$.

Given a perturbation $\delta \in \Delta$, we denote by $\mathtt{norm}(\vec{d}+\delta)$, the image after adding the perturbation δ onto the image $\vec{d}$, namely, for every coordinate $p \in P$

$$\mathtt{norm}(\vec{d}+\delta)[p] := \begin{cases} \vec{d}[p] + \delta[p], & \text{if } 0 \le \vec{d}[p] + \delta[p] \le 255; \\ 0, & \text{if } \vec{d}[p] + \delta[p] < 0; \\ 255, & \text{if } \vec{d}[p] + \delta[p] > 255. \end{cases}$$

The *adversarial image example searching problem* is to search some perturbation δ such that:

- $\mathtt{Top}_1^{\ell}(\mathtt{norm}(\vec{d}+\delta)) \neq f_t(\vec{d})$ for untargeted attack;
- $\mathtt{Top}_1^{\ell}(\mathtt{norm}(\vec{d}+\delta)) = c$ for targeted attack with the target class c.

We solve the adversarial image example searching problem by reduction to a model-based derivative-free discrete optimization problem. The reduction is given by defining an optimization goal which is characterized by dissatisfaction-degree functions. We first consider the untargeted case.

The goal of untargeted attack is to search some perturbation δ such that $\mathtt{Top}_1^{\ell}(\mathtt{norm}(\vec{d}+\delta)) \neq f_t(\vec{d})$. To do this, we minimize the probability of $\mathtt{norm}(\vec{d}+\delta)$ being classified as the class $f_t(\vec{d})$ and maximize the probability of $\mathtt{norm}(\vec{d}+\delta)$ being classified as the top-2 class, namely, $\mathtt{Top}_2^{\ell}(\mathtt{norm}(\vec{d}+\delta))$. Therefore, we define the *dissatisfaction-degree function* for untargeted attack as follows:

- $D_{\mathtt{ua}}(\vec{d}, \delta) := 0$ if $\mathtt{Top}_1^{\ell}(\mathtt{norm}(\vec{d}+\delta)) \neq f_t(\vec{d})$;
- $D_{\mathtt{ua}}(\vec{d}, \delta) := \mathtt{Top}_1(\mathtt{norm}(\vec{d}+\delta)) - \mathtt{Top}_2(\mathtt{norm}(\vec{d}+\delta))$, otherwise.

For targeted attack with the target class c, instead of maximizing the probability of $\mathtt{norm}(\vec{d}+\delta)$ being classified as the top-2 class, we will maximize the probability of $\mathtt{norm}(\vec{d}+\delta)$ being classified as the target class. Therefore, the *dissatisfaction-degree function* for targeted attack is defined as follows:

- $D_{\mathtt{ta}}(\vec{d}, \delta) := 0$ if $\mathtt{Top}_1^{\ell}(\mathtt{norm}(\vec{d}+\delta)) = c$;
- $D_{\mathtt{ta}}(\vec{d}, \delta) := \mathtt{Top}_1(\mathtt{norm}(\vec{d}+\delta)) - \mathcal{P}(\mathtt{norm}(\vec{d}+\delta), c)$, otherwise.

Algorithm 1: A DFO-based algorithm

Input: classifier under attack f_t, original image $\vec{d}$, number of iterations T, ranking threshold k, sample size in each iteration s, search space Δ, dissatisfaction-degree (d.d.) function D

Output: optimized perturbation $\tilde{x}$

1: $B_0 = \{\delta_1, \delta_2, ..., \delta_s, \delta_{s+1}, ..., \delta_{s+k}\}$ randomly sampled from Δ *//initial collection*
2: Compute images $\mathtt{norm}(\vec{d}+\delta_i)$ for $1 \le i \le s+k$
3: Evaluate the dissatisfaction-degree $D(\vec{d}, \delta_i)$
4: $\tilde{x} = \mathtt{argmin}_{\delta \in B_0} D(\vec{d}, \delta)$*//select the best sample so far*
5: **for** $t = 1$ **to** T **do**
6: **if** $D(\vec{d}, \tilde{x}) = 0$ **then**
7: break *//find an adversarial example*
8: **end if**
9: B_{t-1}^{+} =smallest-k solutions in B_{t-1} in terms of d.d.
10: $B_{t-1}^{-} = B_{t-1} - B_{t-1}^{+}$
11: $B = \emptyset$
12: **for** $i = 1$ **to** s **do**
13: Refined search space of Δ by B_{t-1}^{+} and B_{t-1}^{-}
14: *//Refine the value of each coordinate respectively.*
15: Randomly select a perturbation δ' from B_{t-1}^{+}, a subset Y of coordinates of δ', a value v_p from Δ for each coordinate $p \in Y$
16: δ'' = Update δ' with $(v_p)_{p \in Y}$
17: $B = B \cup \{\delta''\}$
18: **end for**
19: Compute images $\mathtt{norm}(\vec{d}+\delta)$ for all $\delta \in B$
20: Evaluate the dissatisfaction-degree $D(\vec{d}, \delta)$ for all $\delta \in B$
21: B_t = smallest-$(s+k)$ solutions in $B \cup B_{t-1}$ in terms of d.d.
22: *//keep the size as* $s+k$
23: $\tilde{x} = \mathtt{argmin}_{\delta \in B_t} D(\vec{d}, \delta)$
24: **end for**
25: **return** $\tilde{x}$

Now, the adversarial image example searching problem is reduced to the minimization problem of the dissatisfaction-degree functions.

4.3 Algorithm

In this section, we present our algorithm (shown in Algorithm 1) for crafting adversarial image examples by leveraging a derivative-free optimization method (DFO) [81].

In detail, we first randomly select $(s+k)$ perturbations (Line 1) from the search space Δ, which is then evaluated by the dissatisfaction-degree function D (Lines 2-3). If the perturbation with the smallest dissatisfaction-degree suffices to craft an adversarial image example, return this perturbation directly (Lines 6-8). Otherwise, we partition the set B_{t-1} of selected perturbations into two sets: "positive" set B_{t-1}^{+} and "negative" set B_{t-1}^{-}, where B_{t-1}^{+} consists of the smallest-k perturbations in terms of the dissatisfaction-degree (Line 9-10).

Our DFO engine refines the search space Δ using B_{t-1}^{+} and B_{t-1}^{-}. Then, we randomly choose: a perturbation δ' from B_{t-1}^{+} as the base,

a subset of the coordinates that will be modified, a random value v_p from the refined search space Δ for each selected coordinate and finally add them onto δ', resulting in a new perturbation δ'' which is stored in B. After repeating this s times, we get s of new perturbations (Lines 12-18). Now, we have $(2s + k)$ perturbations in the set $B \cup B_{t-1}$. From them, we keep the smallest-$(s + k)$ perturbations in terms of the dissatisfaction-degree. We will continue this iteration on B_t until we find an adversarial image example or the number of iterations T is reached.

4.4 Illustrative Example

We illustrate Algorithm 1 procedure through an example, as shown in Figure 4. The original image is classified as *flamingo*. The sample size s is set to 3, the ranking threshold k is set to 2. Consider the first iteration, Algorithm 1 randomly generates 3 perturbations $(\delta_0, \delta_1, \delta_2)$, and add them onto the original image, resulting three new images. Algorithm 1 computes the dissatisfaction degrees of these three new images by querying the classifier. Among these 3 perturbations, δ_0 has the smallest dissatisfaction degree value. After more iterations, the result is shown in Figure 5. We can see that after the 354-th iteration, the class of the crafted image is classified as *hook*, and the crafted images are visually indistinguishable from the original one.

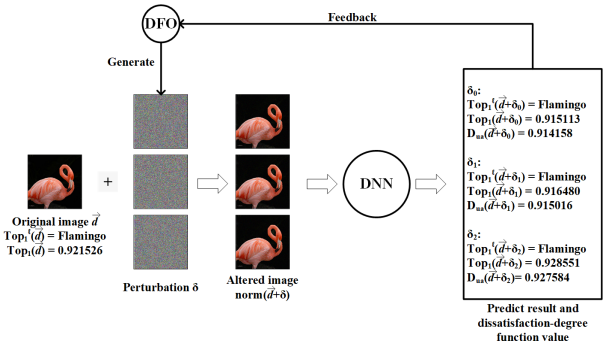

Figure 4: Illustrative example of an untargeted attack on a Flamingo image (In one round).

4.5 Scenario Extension

Our framework is very reflexible and can be adapted to different problems, classifiers and scenarios easily such as one-pixel attack, attack with different lighting condition, occlusion with a single small rectangle, occlusion with multiple tiny black rectangles, more restricted black-box attacks, and so on. These scenario requirements can be easily included into our method by put special restrictions on the search space and/or dissatisfaction-degree functions.

For instance, if the adversary only have access to the class with the largest probability instead of the complete probability vector of all the classes. The dissatisfaction-degree function for untargeted attack can be adapted as follows:

- $D^1_{\mathsf{ua}}(\vec{d}, \delta) := 0$ if $\mathsf{Top}^{\ell}_1(\mathsf{norm}(\vec{d} + \delta)) \neq f_t(\vec{d})$;
- $D^1_{\mathsf{ua}}(\vec{d}, \delta) := \mathsf{Top}_1(\mathsf{norm}(\vec{d} + \delta))$, otherwise.

The dissatisfaction-degree function for targeted attack can be adapted similarly.

For one pixel attack, in our algorithm, we can restrict the underlying DFO engine to change only one pixel in each iteration. Since our method is a black-box method, in general, it can be applied to any scenarios as long as the input and output of the target classifier can be observed. Even a model internally consists of multiple model interactions, e.g., classifiers with defense, we can still treat the whole as a single model and apply our approach.

5 IMPLEMENTATION AND EVALUATION

5.1 Implementation

We implement our methods into a black-box style adversary example generation tool DFA, stands for **D**erivative-**F**ree **A**ttack. We also implement a derivative-free optimization engine in DFA base on RACOS [81], for which we manage to engineer to significantly improve its efficiency and scalability with lots of domain-specific optimizations. Due to the space limitation, the details of the optimizations are omitted here.

In order to process pre-trained networks (e.g., InceptionV3), we also integrate TensorFlow (V1.8.0) [1] and Keras (V2.1.6) [15] DL frameworks in our tool for image prediction. Besides adversarial image example generation for well-known image classification models, our tool can be easily expanded to adapt non-image classification models as well, by customizing the interfaces in our tool, e.g., the dissatisfaction-degree function.

5.2 Dataset & Setting

We use two datasets MNIST and ImageNet in this section.

MNIST. MNIST [42] is a dataset of handwritten digits and contains $28 \times 28 \times 1$ pixel images with 10 classes (0-9). We choose the first 200 images out of 10000 validation images of MNIST as our subjects. We evaluate MNIST images using LeNet-1, a DNN classifier from the LeNet family [41].

ImageNet. ImageNet [18] is a large image dataset. The size of images in ImageNet is around $299 \times 299 \times 3$ or $224 \times 224 \times 3$ pixels. ImageNet contains over 10000000 images with 1000 classes. We choose 100 images, all of them can be correctly classified by four classifiers of Keras (i.e., ResNet50, IncepetionV3, VGG16 and VGG19). We use two well-known pre-trained DNN, ResNet50 [26] and InceptionV3 [72] as the targeted DNN classifiers in our experiments.

We conduct both untargeted attack and targeted attack in the experiments. All experiments are run on a Linux PC running UBUNTU 16.04 LTS with Intel Xeon(R) W-2123 CPU, TITAN Xp COLLECTORS GPU and 64G RAM. Table 5 lists the other settings used.

Besides the SR, TSR, and GAP metrics defined in Section 3, we use ATC to denote the average time cost per adversarial vector example in second for other tools, and the average time cost per adversarial image example in second for our tool.

5.3 Comparison with White-Box methods

Although our method is black-box based, we compare the performance of our tool with two well-known radient-based white-box

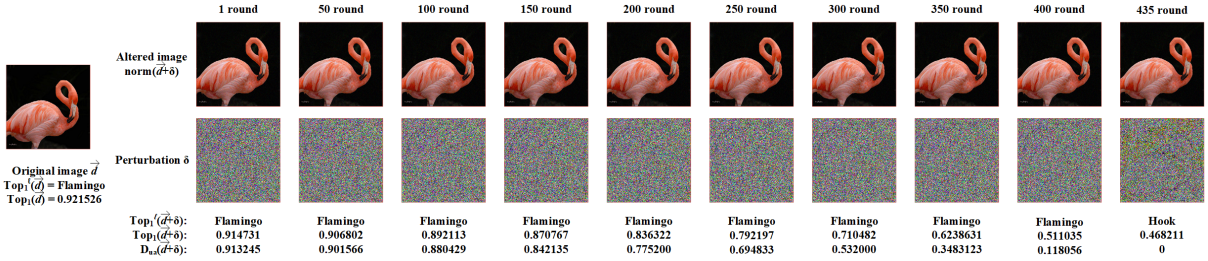

Figure 5: Illustrative example of an untargeted attack on a Flamingo image (Total).

Parameter	Setting
$\mathbb{L}_\infty$ distance	$\epsilon = 64$ for MNIST and $\epsilon = 10$ for ImageNet.
Target class	For MNIST, the class with 4-th high probability. For ImageNet,the class with 11-th high probability.
Sample size s	$s = 3$ for untargeted attack. $s = 10$ for targeted attack.
Ranking Threshold	$k = 2$ in all the experiments.
Coordinate Threshold	The number of coordinates that can be modified in the perturbation during each iteration is about 0.2%, i.e., 2 pixels for MNIST and 600 pixels for ImageNet.
Iteration Threshold	$T = 10000$ in all the experiments.
Timeout Threshold	3 minutes for MNIST . 30 minutes for ImageNet.

Table 5: Experiment Setting

Table 6: Results of white-box untargeted attacks

Dataset	DNN	Method	SR	TSR	GAP	ATC
MNIST	LeNet-1	C&W	100%	85%	15%	14.3
		FGSM	96.7%	96.7%	0%	0
		DFA	100%	**100%**	0%	1.2
ImageNet	ResNet50	C&W	99%	94%	5.05%	8.56
		FGSM	95%	**95%**	0%	0.08
		DFA	90%	90%	0%	65
ImageNet	InceptionV3	C&W	100%	68%	32%	1.41
		FGSM	79%	79%	0%	0.15
		DFA	93%	**93%**	0%	106.8

tools: FGSM [24] and C&W [10]. The $\mathbb{L}_\infty$ distances are transformed into their $\mathbf{L}_\infty$ distance setting accordingly.

The results are shown in Table 6 and Table 7 for untargeted and targeted attacks, respectively. In terms of traditional attack success rate (SR), DFA is comparable with FGSM [24] and C&W [10]. In terms of attack true success rate (TSR), our tool DFA outperforms others in most of the cases. It is not surprising that our black-box tool DFA requires more time than other white-box methods.

Similar to the results given in Table 4, we can see that C&W has a huge gap of targeted attacks on InceptionV3 in $\mathbf{L}_\infty$ setting, as its TSR is only 24% compared with 100% SR. Thus, although C&W outperforms DFA in terms of SR, DFA outperforms C&W in most of the cases in terms of TSR.

Table 7: Results of white-box targeted attacks

Dataset	DNN	Method	SR	TSR	GAP	ATC
MNIST	LeNet-1	C&W	100%	70%	0%	24.5
		FGSM	3%	3%	0%	0
		DFA	100%	**100%**	0%	3
ImageNet	ResNet50	C&W	99%	62%	37.37%	19.85
		FGSM	0%	-	-	-
		DFA	90%	**90%**	0%	118
ImageNet	InceptionV3	C&W	100%	24%	76%	14.88
		FGSM	0%	-	-	-
		DFA	72%	**72%**	0%	375

5.4 Comparison with Black-Box methods

We compare DFA with several well-known black-box methods: substitute model-based attacks, ZOO [12], NES-PGD [32] and DBA [8]. In order to avoid the side-effect of the selected white-box methods when evaluating substitution model, we use both the FGSM and C&W methods respectively. For ImageNet, we use ResNet50 as the substitute model for InceptionV3 and vice versa. The substitute model for LeNet-1 is from the evaluation of ZOO [12]. Since our tool uses $\mathbb{L}_\infty$ distance, DBA and ZOO use $\mathbf{L}_2$ distance. We map the $\mathbb{L}_\infty$ distance used in DFA to the corresponding approximated $\mathbf{L}_2$ value in the experiments. We remark that not all these tools provide related parameters for certain problems, in these cases, we do not give their experimental results.

The results of untargeted and targeted attacks are given in Table 8 and Table 9 respectively. We can see that our tool DFA significantly outperforms all the other tools in terms of TSR no matter white-box or black-box attacks. Substitution model based attacks can achieve at most 38% TSR under untargeted setting, and 3% TSR under targeted setting.

Compared with ZOO [12], NES-PGD [32] and DBA [59], our tool DFA can achieve similar or higher attack success rate in almost all the cases in terms of SR. However, all these methods suffer from the discretization problem, they usually have gap between SR and TSR. For instance, the attack success rate of NES-PGD is dramatically dropped from 73% to 3% on InceptionV3 in targeted setting.

5.5 Efficiency Analysis

Time Usage. From Table 6 to Table 9, we can see the time usage of DFA varies in different cases. On small models and images from MNIST, the average time cost of DFA is very small, around $1 - -3$ seconds; while on larger models and images from ImageNet,

Table 8: Results of black-box untargeted attacks

Dataset	DNN	Method	SR	TSR	GAP	ATC
MNIST	LeNet-1	SModel+C&W	2.5%	2.5%	0%	28
		SModel+FGSM	20%	20%	0%	1
		DBA	99.5%	84%	15.58%	11.89
		ZOO	100%	89.5%	10.5%	15.9
		DFA	100%	**100%**	0%	1.2
ImageNet	ResNet50	SModel+C&W	2%	1%	50%	1.29
		SModel+FGSM	24%	24%	0%	0.12
		DBA	100%	44%	56%	137.79
		DFA	97%	**97%**	0%	150.66
ImageNet	InceptionV3	SModel+C&W	6%	6%	0%	24.32
		SModel+FGSM	38%	38%	0%	0.37
		DBA	100%	34%	66%	254.2
		ZOO	73%	3%	95.79%	113
		NES-PGD	100%	77%	23%	24
		DFA	93%	**93%**	0%	106.8

Table 9: Results of black-box targeted attacks

Dataset	DNN	Method	SR	TSR	GAP	ATC
MNIST	LeNet-1	SModel+C&W	1.5%	1.5%	0%	22
		SModel+FGSM	3%	3%	0%	1
		DBA	95.5%	79%	17.28%	12.17
		ZOO	100%	94.5%	5.5%	17.4
		DFA	100%	**100%**	0%	3
ImageNet	ResNet50	SModel+C&W	2%	2%	0%	27.21
		SModel+FGSM	0%	-	-	-
		DBA	95%	21%	77.89%	335
		DFA	87%	**87%**	0%	211.32
ImageNet	InceptionV3	SModel+C&W	1%	1%	0%	41.66
		SModel+FGSM	0%	-	-	-
		DBA	87%	25%	71.3%	544
		ZOO	62%	6%	90.32%	716.04
		NES-PGD	100%	47%	53%	56
		DFA	72%	**72%**	0%	375

the time usage is increased to at most 375 seconds. After a deep investigation, we find out that our algorithm changes the values of two coordinates during each sample for MNIST, while it changes the values of 600 coordinates during each sample for ImageNet. Thus, it spends more time to handle these modifications, and the time usage is significantly increased on ImageNet. We remark that on black-box attacks, time usage of our method is on the same level as other non-substitution model-based tools in most of the cases.

Query Efficiency. In many black-box scenarios, there may exist a limitation of query times to avoid detection. Therefore, in the context of black-box attacks, query efficiency should be considered. We compare the query time of DFA with two of the most query-efficient methods so far: NES-PGD and Bandit [33]. NES-PGD and Bandit are black-box attack tools which focus on query-limited scenarios. Bandit is an optimized version of NES-PGD by using bandit optimization, while NES-PGD is natural evolutionary-based tool.

We use InceptionV3 as the target model and conduct untargeted and targeted attacks using NES-PGD, Bandit[5] and our tool DFA. The average query time per successful attack used by each tool is given in Table 10. We can see that DFA outperforms the other two tools in terms for query times.

[5]The parameter setting of Bandit for targeted attack is not given therein, therefore conduct untargeted attacks.

Table 10: Comparison with NES-PGD and Bandit with average query times of success cases

Dataset	DNN	Method	Untargeted attack	Targeted attack
ImageNet	InceptionV3	NES-PGD	4741	13421
		Bandit	3255	-
		Our tool	**2660**	**11364**

5.6 Attack Classifiers with Defense

To show the effectiveness of our approach, we first use our tool to attack the HGD defense [45], which won the first place of NIPS 2017 competition on defense against adversarial attacks.

In this experiment, we conduct untargeted attacks on this model using the same 100 images from ImageNet as previously. The $\mathbb{L}_\infty$ distance ϵ is 32 according to the competition's setting. The coordinate threshold is set to 200. Our tool achieves 72% attack success rate with time 30 minute limitation for each attack. The attack success rate significantly increases to 95% with 200 minutes time limitation. This confirms that DFA can handle difficult problems with enough computation resource.

On MNIST Adversarial Examples Challenge[6], another widely recognized attack problem, we use the same 200 images. Our tool DFA achieves 10.5% attack success rate, the same as the current best white-box attack "interval attacks", which is publicly reported on the webpage of the challenge. Indeed, the images crafted by both methods are exactly the same, and the time usage of both tools are also nearly the same.

6 CONCLUSION

Adversarial example attack is a severe problem for deep learning network that by adding a subtle perturbation to the input, the network may misclassify the input to a wrong category. Many research studies have been devoted to this problem and made huge progress recently. No matter white-box or black-box based, most of the existing works make the perturbation on the vector level of the image, in the continuous real domain. However, if the "successful" attack is mapped back to concrete image domain, many of them become benign. This discretization problem has not attracted enough attention so far. We make a comprehensive study of this problem theoretically and empirically. From the study, we can see the majority of existing works, 26 out of 34 methods and 16 out of 20 tools, are affected by the discretization problem significantly.

To solve this problem, we propose a derivative-free optimization based method to conduct a black-box adversarial attack on the image level. Our attack only requires access to the probability distribution of classes for each test input and does not rely on the gradient of the objective function, but instead, learns from samples of the search space. We implement our method into tool DFA, and conduct an intensive set of experiments on MNIST and ImageNet in both nontargeted and targeted modes. The result shows our tool outperforms the competitors in the aspect of success rate significantly with satisfactory efficiency. Meanwhile, we also apply

[6]https://github.com/MadryLab/mnist_challenge

DFA to attack defense model and achieve a very high success rate with 95% which also confirms our argument that our "real" black-box method can handle any difficult model that is not differentiable.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*. 265–283.

[2] Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally Normalized Transition-Based Neural Networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

[3] Apollo. 2018. An open, reliable and secure software platform for autonomous driving systems. http://apollo.auto. (2018).

[4] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. 2018. Synthesizing Robust Adversarial Examples. In *Proceedings of the 35th International Conference on Machine Learning*. 284–293.

[5] Osbert Bastani, Yani Ioannou, Leonidas Lampropoulos, Dimitrios Vytiniotis, Aditya V. Nori, and Antonio Criminisi. 2016. Measuring Neural Net Robustness with Constraints. In *NIPS*. 2613–2621.

[6] Arjun Nitin Bhagoji, Warren He, Bo Li, and Dawn Song. 2017. Exploring the Space of Black-box Attacks on Deep Neural Networks. *CoRR* abs/1712.09491 (2017).

[7] Arjun Nitin Bhagoji, Warren He, Bo Li, and Dawn Song. 2018. Practical Black-Box Attacks on Deep Neural Networks Using Efficient Query Mechanisms. In *Proceedings of the 15th European Conference on Computer Vision (ECCV)*. 158–174.

[8] Wieland Brendel, Jonas Rauber, and Matthias Bethge. 2018. Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models. In *International Conference on Learning Representations*.

[9] Nicholas Carlini and David A. Wagner. 2017. Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. 3–14.

[10] Nicholas Carlini and David A. Wagner. 2017. Towards Evaluating the Robustness of Neural Networks. In *2017 IEEE Symposium on Security and Privacy*. 39–57.

[11] Pin-Yu Chen, Yash Sharma, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh. 2018. EAD: Elastic-Net Attacks to Deep Neural Networks via Adversarial Examples. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI)*. 10–17.

[12] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. 2017. ZOO: Zeroth Order Optimization Based Black-box Attacks to Deep Neural Networks without Training Substitute Models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. 15–26.

[13] Sen Chen, Minhui Xue, Lingling Fan, Shuang Hao, Lihua Xu, Haojin Zhu, and Bo Li. 2018. Automated poisoning attacks and defenses in malware detection systems: An adversarial machine learning approach. *Computers & Security* 73 (2018), 326–344.

[14] Minhao Cheng, Thong Le, Pin-Yu Chen, Jinfeng Yi, Huan Zhang, and Cho-Jui Hsieh. 2018. Query-Efficient Hard-label Black-box Attack: An Optimization-based Approach. *CoRR* abs/1807.04457 (2018).

[15] François Chollet et al. 2015. Keras. https://keras.io. (2015).

[16] Dan C. Ciresan, Alessandro Giusti, Luca Maria Gambardella, and Jürgen Schmidhuber. 2012. Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images. In *Proceedings of the 26th Annual Conference on Neural Information Processing Systems*. 2852–2860.

[17] Nilesh N. Dalvi, Pedro M. Domingos, Mausam, Sumit K. Sanghai, and Deepak Verma. 2004. Adversarial classification. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 99–108.

[18] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. 248–255.

[19] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. 2018. Boosting Adversarial Attacks With Momentum. In *Proceedings of 2018 IEEE Conference on Computer Vision and Pattern Recognition*. 9185–9193.

[20] Krishnamurthy Dvijotham, Robert Stanforth, Sven Gowal, Timothy A. Mann, and Pushmeet Kohli. A Dual Approach to Scalable Verification of Deep Networks. *CoRR* abs/1803.06567 (????).

[21] Rüdiger Ehlers. 2017. Formal Verification of Piece-Wise Linear Feed-Forward Neural Networks. In *Proceedings of the 15th International Symposium on Automated Technology for Verification and Analysis*. 269–286.

[22] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. 2018. Robust Physical-World Attacks on Deep Learning Visual Classification. In *Proceedings of 2018 IEEE Conference on Computer Vision and Pattern Recognition*. 1625–1634.

[23] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin T. Vechev. 2018. AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation. In *Proceedings of the 2018 IEEE Symposium on Security and Privacy*. 3–18.

[24] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*.

[25] Divya Gopinath, Guy Katz, Corina S. Pasareanu, and Clark Barrett. 2018. DeepSafe: A Data-Driven Approach for Assessing Robustness of Neural Networks. In *Proceedings of the 16th International Symposium on Automated Technology for Verification and Analysis*. 3–19.

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 770–778.

[27] Warren He, Bo Li, and Dawn Song. 2018. Decision boundary analysis of adversarial examples. In *Proceedings of International Conference on Learning Representations*.

[28] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury. 2012. Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. *IEEE Signal Process. Mag.* 29, 6 (2012), 82–97.

[29] Peter Holley. 2018. Texas becomes the latest state to get a self-driving car service. https://www.washingtonpost.com/news/innovations/wp/2018/05/07/texas-becomes-the-latest-state-to-get-a-self-driving-car-service. (May 2018).

[30] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. 2017. Safety Verification of Deep Neural Networks. In *Proceedings of the 29th International Conference on Computer Aided Verification*. 3–29.

[31] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. 2017. Query-Efficient Black-box Adversarial Examples. *arXiv preprint arXiv:1712.07113* (2017).

[32] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. 2018. Black-box Adversarial Attacks with Limited Queries and Information. In *Proceedings of the 35th International Conference on Machine Learning*. 2142–2151.

[33] Andrew Ilyas, Logan Engstrom, and Aleksander Madry. 2018. Prior Convictions: Black-Box Adversarial Attacks with Bandits and Priors. *ICLR 2019* (2018).

[34] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Fei-Fei Li. 2014. Large-Scale Video Classification with Convolutional Neural Networks. In *Proceedings of 2014 IEEE Conference on Computer Vision and Pattern Recognition*. 1725–1732.

[35] Guy Katz, Clark W. Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. 2017. Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks. In *Proceedings of the 29th International Conference on Computer Aided Verification*. 97–117.

[36] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations (ICLR)*.

[37] Jernej Kos, Ian Fischer, and Dawn Song. 2018. Adversarial Examples for Generative Models. In *Proceedings of 2018 IEEE Security and Privacy Workshops*. 36–42.

[38] Alex Krizhevsky. 2009. *Learning multiple layers of features from tiny images*. Technical Report.

[39] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2017. ImageNet classification with deep convolutional neural networks. *Commun. ACM* 60, 6 (2017), 84–90.

[40] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2017. Adversarial examples in the physical world. In *Proceedings of International Conference on Learning Representations*.

[41] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[42] Yann LeCun, Corinna Cortes, and Christopher JC Burges. 1998. The mnist database of handwritten digits. (1998).

[43] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. 2018. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *I. J. Robotics Res.* 37, 4-5 (2018), 421–436.

[44] Bo Li and Yevgeniy Vorobeychik. 2014. Feature Cross-Substitution in Adversarial Classification. In *Proceedings of Advances in Neural Information Processing Systems*. 2087–2095.

[45] Fangzhou Liao, Ming Liang, Yinpeng Dong, Tianyu Pang, Xiaolin Hu, and Jun Zhu. 2018. Defense Against Adversarial Attacks Using High-Level Representation Guided Denoiser. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*. 1778–1787.

[46] Lei Ma, Felix Juefei-Xu, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Chunyang Chen, Ting Su, Li Li, Yang Liu, Jianjun Zhao, and Yadong Wang. 2018. DeepGauge: multi-granularity testing criteria for deep learning systems. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*. 120–131.

[47] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *Proceedings of International Conference on Learning Representations*.

[48] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition*. 2574–2582.

[49] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. 2017. Universal Adversarial Perturbations. In *Proceedings of 2017 IEEE Conference on Computer Vision and Pattern Recognition*. 86–94.

[50] Nina Narodytska and Shiva Prasad Kasiviswanathan. 2017. Simple Black-Box Adversarial Attacks on Deep Neural Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 1310–1318.

[51] Anh Mai Nguyen, Jason Yosinski, and Jeff Clune. 2015. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of 2015 IEEE Conference on Computer Vision and Pattern Recognition*. 427–436.

[52] Nicolas Papernot, Patrick D. McDaniel, and Ian J. Goodfellow. 2016. Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples. *CoRR* abs/1605.07277 (2016).

[53] Nicolas Papernot, Patrick D. McDaniel, Ian J. Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. 2017. Practical Black-Box Attacks against Machine Learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*. 506–519.

[54] Nicolas Papernot, Patrick D. McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. 2016. The Limitations of Deep Learning in Adversarial Settings. In *Proceedings of IEEE European Symposium on Security and Privacy*. 372–387.

[55] Toufiq Parag, Dan C. Ciresan, and Alessandro Giusti. 2015. Efficient Classifier Training to Minimize False Merges in Electron Microscopy Segmentation. In *Proceedings of 2015 IEEE International Conference on Computer Vision*. 657–665.

[56] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. 2017. DeepXplore: Automated Whitebox Testing of Deep Learning Systems. In *Proceedings of the 26th Symposium on Operating Systems Principles*. 1–18.

[57] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. 1532–1543.

[58] Luca Pulina and Armando Tacchella. 2010. An Abstraction-Refinement Approach to Verification of Artificial Neural Networks. In *Proceedings of the 22nd International Conference on Computer Aided Verification (CAV)*. 243–257.

[59] Jonas Rauber, Wieland Brendel, and Matthias Bethge. 2017. Foolbox: A Python toolbox to benchmark the robustness of machine learning models. *arXiv preprint arXiv:1707.04131* (2017).

[60] L. D. L. Rosa, S. Kilgallon, T. Vanderbruggen, and J. Cavazos. 2018. Efficient Characterization and Classification of Malware Using Deep Learning. In *2018 Resilience Week (RWS)*. 77–83.

[61] Wenjie Ruan, Xiaowei Huang, and Marta Kwiatkowska. 2018. Reachability Analysis of Deep Neural Networks with Provable Guarantees. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*. 2651–2659.

[62] Tim Salimans, Jonathan Ho, Xi Chen, and Ilya Sutskever. 2017. Evolution Strategies as a Scalable Alternative to Reinforcement Learning. *CoRR* abs/1703.03864 (2017). arXiv:1703.03864 http://arxiv.org/abs/1703.03864

[63] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K. Reiter. 2016. Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. 1528–1540.

[64] Dinggang Shen, Guorong Wu, , and Heung-Il Suk. 2017. Deep learning in medical image analysis. *Annual Review of Biomedical Engineering* 19 (2017), 221–248.

[65] Eui Chul Richard Shin, Dawn Song, and Reza Moazzezi. 2015. Recognizing Functions in Binaries with Neural Networks. In *Proceedings of the 24th USENIX Security Symposium, USENIX Security*. 611–626.

[66] Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin T. Vechev. 2018. Fast and Effective Robustness Certification. In *Advances in Neural Information Processing Systems*. 10825–10836.

[67] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin Vechev. 2019. An Abstract Domain for Certifying Neural Networks. In *POPL*.

[68] Wei Song, Heng Yin, Chang Liu, and Dawn Song. 2018. DeepMem: Learning Graph Neural Network Models for Fast and Robust Memory Forensic Analysis. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 606–618.

[69] J. Su, D. V. Vargas, and K. Sakurai. 2019. One Pixel Attack for Fooling Deep Neural Networks. *IEEE Transactions on Evolutionary Computation* (2019).

[70] Youcheng Sun, Xiaowei Huang, and Daniel Kroening. 2018. Testing Deep Neural Networks. *CoRR* abs/1803.04792 (2018).

[71] Youcheng Sun, Min Wu, Wenjie Ruan, Xiaowei Huang, Marta Kwiatkowska, and Daniel Kroening. 2018. Concolic testing for deep neural networks. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineerin*. 109–119.

[72] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2016. Rethinking the Inception Architecture for Computer Vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, (CVPR)*. 2818–2826.

[73] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *Proceedings of International Conference on Learning Representations*.

[74] Vincent Tjeng, Kai Xiao, and Russ Tedrake. 2019. Evaluating Robustness Of Neural Networks With Mixed Integer Programming. *ICLR*.

[75] Chun-Chen Tu, Pai-Shun Ting, Pin-Yu Chen, Sijia Liu, Huan Zhang, Jinfeng Yi, Cho-Jui Hsieh, and Shin-Ming Cheng. 2018. AutoZOOM: Autoencoder-based Zeroth Order Optimization Method for Attacking Black-box Neural Networks. *CoRR* abs/1805.11770 (2018).

[76] Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. 2018. Formal Security Analysis of Neural Networks using Symbolic Intervals. In *Proceedings of the 27th USENIX Security Symposium on Security*. 1599–1614.

[77] Waymo. 2009. A self-driving technology development company. https://waymo.com/. (2009).

[78] Matthew Wicker, Xiaowei Huang, and Marta Kwiatkowska. 2018. Feature-Guided Black-Box Safety Testing of Deep Neural Networks. In *Proceedings of the 24th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. 408–426.

[79] Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. 2014. Natural evolution strategies. *Journal of Machine Learning Research* 15, 1 (2014), 949–980. http://dl.acm.org/citation.cfm?id=2638566

[80] Chaowei Xiao, Jun-Yan Zhu, Bo Li, Warren He, Mingyan Liu, and Dawn Song. 2018. Spatially Transformed Adversarial Examples. In *International Conference on Learning Representations*.

[81] Yang Yu, Hong Qian, and Yi-Qi Hu. 2016. Derivative-free optimization via classification. In *Thirtieth AAAI Conference on Artificial Intelligence*.

[82] Fangyi Zhang, Jürgen Leitner, Michael Milford, Ben Upcroft, and Peter I. Corke. 2015. Towards Vision-Based Deep Reinforcement Learning for Robotic Motion Control. *CoRR* abs/1511.03791 (2015).

[83] Zhengli Zhao, Dheeru Dua, and Sameer Singh. 2018. Generating Natural Adversarial Examples. In *International Conference on Learning Representations*.