Taking Care of the Discretization Problem: A Comprehensive Study of the Discretization Problem and a Black-Box Adversarial Attack in Discrete Integer Domain

Lei Bu, *Member, IEEE*, Zhe Zhao, Yuchao Duan, and Fu Song

Abstract—Neural network based (NN-based) classifiers are known vulnerable against adversarial examples, namely, adding slight perturbations to a benign image cause a classifier to make a false prediction. To evaluate the robustness of NN-based classifiers against adversarial examples, numerous adversarial attacks with high success rates have been proposed recently. NN-based image classifiers usually normalize valid images (e.g., RGB image where the value at each coordinate is an integer between 0 and 255) into a real continuous domain (e.g., 3-dimensional matrix where the value at each coordinate is a real number between 0 and 1) and make classification decisions on the normalized images. However, adversarial examples crafted in a real continuous domain may become benign once they are denormalized back into the corresponding discrete integer domain, known as the discretization problem. This problem has been mentioned in some prior works but received relatively limited attention. In this work, we report the first comprehensive study of existing works to understand the impacts of the discretization problem. By analyzing 35 representative methods and empirically studying 20 representative open source tools, we found 29/35 (theoretically) and 14/20 (empirically) are affected by the discretization problem, e.g., the success rate could dramatically drop from 100 to 10 percent after the domain transformation. As the first step towards addressing this problem in a black-box scenario, we propose a novel derivative-free optimization method, which can directly craft adversarial examples in the discrete integer domain. Experimental results show that the method achieves nearly 100 percent attack success rates for both targeted and untargeted attacks, comparable to the most popular white-box methods (FGSM, BIM and C&W), and significantly outperforms representative black-box methods (ZOO, AutoZOOM, NES-PGD, Bandits, FD, FD-PSO and GenAttack). Our results suggest that the discretization problem should be treated more seriously, and the discrete optimization algorithms show a promising future in crafting effective black-box attacks.

Index Terms—Adversarial examples, deep neural networks, discretization, black-box attacks, derivative-free optimization

1 Introduction

In the past ten years, machine learning algorithms, fueled by massive amounts of data, achieve human-level performance or even better on a number of tasks. Models produced by machine learning algorithms, especially deep neural networks, has been deployed in a variety of applications such as autonomous driving [1], [2], medical diagnostics [3], [4], [5], speech processing [6], [7], computer vision [8], [9], robotics [10], [11], natural language processing [12], [13], and cyber-security [14], [15], [16].

In the early stage of machine learning, people pay more attention to the basic theory and application research, although it is known in 2014 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 examples for the first time in deep neural networks [18]. By adding a subtle perturbation to the input of the deep neural network, it results in a misclassification. Moreover, a relatively large fraction of adversarial examples can be used to attack models that have different architectures and training data. Following these findings, a plethora of studies has 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 [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], [39]. In the literature, there are mainly two types of complementary techniques: testing based [18], [24], [25], [26], [27], [31], [32], [34], [35], [36], [37], [38], [39], [40], [41] and verification based [38], [42], [43], [44], [45], [46], [47], [48], [49], [50] methods for crafting adversarial examples. According to the adversary's knowledge and

- *Lei Bu and Yuchao Duan are with the State Key Laboratory for Novel Software Technology, Department of Computer Science and Technology, Nanjing University, Nanjing, Jiangsu 210093, China. E-mail: bulei@nju.edu.cn, yuchaoduann@gmail.com.*
- *Zhe Zhao is with the School of Information Science and Technology, Shanghai Tech University, Shanghai 201210, China, and with the Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, Shanghai 100049, China, and also with the University of Chinese Academy of Sciences, Beijing 100049, China. E-mail: zhaozhe1@shanghaitech.edu.cn.*
- *Fu Song is with the School of Information Science and Technology, Shanghai Engineering Research Center of Intelligent Vision and Imaging, ShanghaiTech University, Shanghai 201210, China. E-mail: songfu@shanghaitech.edu.cn.*

Manuscript received 26 Apr. 2020; revised 13 May 2021; accepted 4 June 2021. Date of publication 11 June 2021; date of current version 2 Sept. 2022.
(Corresponding author: Fu Song.)
Digital Object Identifier no. 10.1109/TDSC.2021.3088661

capabilities, these techniques also can be categorized into both white-box [18], [24], [25], [25], [31], [32], [40], [42], [43], [44], [45], [46] and black-box [26], [27], [34], [35], [36], [37], [38], [39], where white-box attacks require full white-box access to the target model, which is not always feasible in practice.

However, almost all existing adversarial example attacks target neural networks rather than neural network based classifiers, while a neural network based classifier contains not only a neural network model but also a pre-processor for input data processing. Valid images in computer systems are stored in some format (e.g., png and jpeg) formed as a discrete integer domain (e.g., $\{0, \ldots, 255\}^m$), but will be normalized into some continuous real domain (e.g., $[0,1]^m$) for training and predicting [51]. Therefore, a neural network based image classifier consists of a pre-processor for normalization and a neural network model, the input of the classifier is an image rather than a float array. As a result, adversarial examples crafted by existing attacks against neural networks are in the continuous real domain. Such adversarial examples fool the target neural network, but once denormalized and saved into the discrete integer domain as valid images, may become benign for the neural network based image classifier. This gap was initially considered by Goodfellow *et al.* [19] and Papernot *et al.* [22], and later formally presented by Carlini and Wagner, called *the discretization problem* [21], which mainly refers to the problem that adversarial examples become benign after the rounding of float point numbers. Carlini and Wagner stated that "*This rounding will slightly degrade the quality of the adversarial example*" according to their experimental results on MNIST images. Later on, this problem has received relatively little attention. The discretization problem could cause severe consequences. For instance, attack success rates are widely used to compare the effectiveness of different adversarial attacks and evaluate the robustness of neural network based classifiers. But almost all the existing work calculates the attack success rates without taking into the discretization problem account, resulting in inflated attack success rates. Such results cause readers and authors to misjudge the effectiveness of adversarial attacks and robustness of neural network based classifiers. Actually, the discretization problem already leads to user confusion in practice, e.g., the Github issues [52], [53], [54], where the adversarial examples produced by the attack tools become benign after saving as the valid images. Therefore, it is important to conduct a comprehensive study on the impacts of the discretization problem: e.g., which methods/tools may be affected, to what extent does this problem affect the attack success rate and can it be avoided or alleviated?

To understand the impacts of the discretization problem, in this work, we report the first comprehensive study of existing works for crafting adversarial examples in image classification domain which has a plethora of studies. In the rest of this work, adversarial examples in a continuous real domain will be called *real* adversarial examples and adversarial examples in a discrete integer domain will be called *integer* adversarial examples.

We first discuss the difference of adversarial examples between continuous domain and discrete domain, Then, we theoretically analyze 35 representative methods for crafting adversarial examples. We find that: (1) Almost all of them craft real adversarial examples; (2) 29 methods are affected by the discretization problem; and (3) 23 works do not provide hyper-parameters so that the discretization problem could not be easily and directly avoided.

To understand the impacts of the discretization problem in practice, we carry out an empirical evaluation of 20 representative open source tools. We evaluate the gap between the attack success rates of crafted real adversarial examples and their corresponding integer adversarial examples. Our empirical study shows that: (1) Most of the 20 tools are affected by the discretization problem. In our experiments, there are 8 tools whose gap exceeds 50 percent, 6 tools whose gap exceeds 70 percent, and only 6 tools do not have any gaps. (2) Among the 14 tools that are affected by the discretization problem, only 1 tool (FGSM) can avoid the discretization problem by tuning input parameters, 3 tools can alleviate the discretization problem by tuning input parameters at the cost of attack efficiency or imperceptibility of adversarial examples, and 10 tools can neither avoid nor alleviate the discretization problem by tuning input parameters. *Our study reveals that the discretization problem is far more serious than originally thought and suggests taking it into account seriously when crafting digital adversarial examples and measuring attack success rate.*

According to our comprehensive study, we found there lacks an effective and efficient integer adversarial example attack in the black-box scenario. This is because that white-box attacks are usually efficient than black-box attacks in terms of query times. New adversarial examples can be quickly crafted when "fake" adversarial examples (due to the discretization problem) are produced by white-box attacks, while it would be inefficient for black-box attacks. As the second main contribution of this work, we propose a black-box algorithm that directly crafts adversarial examples in discrete integer domains for both targeted and untargeted attacks. Our method only requires access to the probability distribution of classes for each test input. We formalize the computation of integer adversarial examples as a black-box discrete optimization problem constrained with $\mathbb{L}_\infty$ distance, where $\mathbb{L}_\infty$ is defined in the discrete domain as well. However, this discrete optimization problem cannot be solved using gradient-based methods, as the model is non-continuous. To solve this problem, we propose a novel classification model-based derivative-free discrete optimization method that does not rely on the gradient of the objective function. Instead, it *learns* from samples of the search space and *refines* the search space into small sub-spaces. It is suitable for optimizing non-differentiable functions, with many local minima, or even unknown but only testable.

We demonstrate the effectiveness and efficiency of our method on the MNIST dataset [55] using the LeNet-1 model [56], and the ImageNet dataset [57] using Inception-v3 [58] model. Our method achieves close to 100 percent attack success rates for both targeted and untargeted attacks, comparable to the state-of-the-art white-box attacks: FGSM [19], BIM [26] and C&W [21], and significantly outperforms representative black-box methods: ZOO [33], AutoZOOM [37], NES-PGD [39], Bandits [59], GenAttack [60], substitute model based black-box attacks with FGSM and C&W methods, FD and FD-PSO [61]. In terms of query efficiency, our attack is comparable to (or better than) the black-box attacks: NES-PGD, Bandits, AutoZOOM, and GenAttack, which are specially designed for query-limited scenarios. Moreover, our method is able to break the HGD defense [62], which won the first place of NIPS 2017 competition on defense against adversarial attacks, with 100 percent success rate, and also achieves the so-far best success rate of white-box attacks in the online MNIST Adversarial Examples Challenge [63].

Our contributions in this paper include:

- We report the first comprehensive study of existing works on the discretization problem, including 35 representative methods and 20 representative open source tools.
- Our study sheds light on the impacts of the discretization problem, which is helpful to the community.
- We propose a black-box algorithm for crafting integer adversarial examples for targeted/untargeted attacks by designing a derivative-free discrete optimization method.
- Our attack achieves close to 100 percent attack success rate, comparable to several recent popular white-box attacks, and outperforms several recent popular black-box tools (e.g., ZOO, Bandits, AutoZOOM, GenAttack and NES-PGD) in terms of integer adversarial examples.
- Our attack is able to break the HGD defense [62] with 100 percent success rate, and also achieves the same result as the best white-box attack in MNIST Challenge [63].

To the best of our knowledge, this is the first comprehensive study of the impacts of the discretization problem on adversarial examples and the first black-box attack that directly crafts adversarial examples in discrete integer domain.

2 Related Work

Digital adversarial attacks in the white-box scenario have been widely studied in the literature, to cite a few [18], [24], [25], [25], [31], [32], [40], [42], [43], [44], [45], [46]. In the white-box scenario, an adversary has access to details (e.g., architecture, parameters, training dataset) of the system under attack. This setting is clearly impractical in real-world cases, when the adversary cannot get access to the details. Therefore, in this work, we propose black-box adversarial attacks. In the rest of this section, we mainly discuss existing works on black-box adversarial attacks.

2.1 Digital Adversarial Attack

We classify existing attack methods along three dimensions: substitute model, gradient estimation and heuristic search.

Substitute Model. Papernot *et al.* [35] 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. [64] generalized this idea to attack other machine learning classifiers. However, transferability is not always reliable, other methods such as gradient estimation are explored as alternatives to substitute networks.

Gradient Estimation. Gradient plays an important role in white-box adversarial attacks. Therefore, estimating the gradient to guide the search of adversarial examples is a popular research direction in black-box adversarial attacks. Narodytska and Kasiviswanathan [65] proposed a greedy local search based method to construct numerical approximation to the network gradient, which is then used to construct a small set of pixels in an image to perturb. Chen *et al.* [33] proposed a black-box attack method (named ZOO) with zeroth order optimization. Following ZOO, Tu *et al.* [37] proposed an autoencoder-based method (named AutoZOOM) to improve query efficiency. Similarly, Bhagoji *et al.* [61] proposed a class of black-box attacks (called FD) that approximate FGSM and BIM via gradient estimation. Independently, Ilyas *et al.* [39] proposed an alternative gradient estimation method by leveraging natural evolution strategy (NES) [66], [67] and employing a white-box PGD attack with estimated gradient (named NES-PGD). Based on NES-PGS, Ilyas *et al.* [59] proposed a bandit optimization-based method aimed at enhancing query efficiency. Recently, Zhao *et al.* [68] proposed a method to leverage an alternating direction method of multipliers (ADMM) algorithm for gradient estimation.

Heuristic Search. Instead of gradient estimation, heuristic search-based derivative-free optimization (DFO) methods have been proposed. Hosseini *et al.* [69] proposed a method by iteratively adding Gaussian noise. Liu *et al.* [70] proposed ensemble-based approaches to generating transferable adversarial examples. Brendel *et al.* [27] proposed a decision-based attack (named DBA) with label-only setting, which starts from the target image, moves a small step to the raw image every time, and checks the perturbation cross the decision boundary or not. Su *et al.* [71] proposed a black-box attack for generating one-pixel adversarial images based on differential evolution. Bhagoji *et al.* [61] also proposed a particle swarm optimization (PSO) based DFO method, named FD-PSO. PSO previously was used to find adversarial examples to fool face recognition systems [23]. In a concurrent work, Alzantot *et al.* [60] proposed a genetic algorithm based DFO method (named GenAttack) for generating adversarial images. A genetic algorithm was previously used to find adversarial examples to fool PDF malware classifiers in EvadeML [72]. Co *et al.* [73] proposed a method for generating universal adversarial perturbations (UAPs) in the black-box attack scenario by leveraging Bayesian optimization, it is an interesting new area to generate procedural noise perturbations.

Comparison. Our method does not rely on any substitute model or gradient estimation. Different from the above heuristic search based methods, we present a classification model-based DFO method, to distinguish “good” samples from “bad” samples. By learning from the evaluation of the samples, our algorithm iteratively refines large search space into small-subspaces, finally converges to the best solution. To the best of our knowledge, our method is the first one which iteratively refines large search space into small-subspaces during searching adversarial examples. Experimental results show that our method achieves significantly higher success rates in terms of the integer adversarial examples than the state-of-the-art tools from all the above classes, with comparable query times (cf. Section 6).

Although, some of these works (e.g., [27], [69], [70]) for crafting digital adversarial samples add noises onto integer

TABLE 1
Notations Used in This Paper

Notation	Description
w, h, ch	width, height, and number of channels of an image
P	the set of coordinates $w \times h \times ch$
$\mathbb{V}$	**continuous (real)** domain of **real** images $\vec{v}$, e.g., $\mathbb{R}_{[0,1]}^{w\times h\times ch}$
$\mathbb{D}$	**discrete (integer)** domain of **integer** images $\vec{d}$, e.g., $\mathbb{N}_{[0,255]}^{w\times h\times ch}$
$\vec{v},\ \vec{v}^{\mathrm{adv}} \in \mathbb{V}$	continuous **real** (adversarial) image
$\vec{d},\ \vec{d}^{\mathrm{adv}} \in \mathbb{D}$	discrete **integer** (adversarial) image
$\vec{v}[p]$	entity at coordinate p of a **real** image $\vec{v}$
$\vec{d}[p]$	entity at coordinate p of an **integer** image $\vec{d} \in \mathbb{D}$
$\mathbb{T} : \mathbb{D} \rightarrow \mathbb{V}$	normalizer that transforms an **integer** image into a **real** image in the continuous domain $\mathbb{V}$
$\mathbb{T}^{-1} : \mathbb{V} \rightarrow \mathbb{D}$	denormalizer that transforms a **real** image back into an **integer** image such that for all $\vec{d} \in \mathbb{D},\ \mathbb{T}^{-1}(\mathbb{T}(\vec{d})) = \vec{d}$
$\mathbb{C}_t$	set of mutually exclusive classes for the task t

images and clip the value of each pixel into the range of 0 and 255, the noise added to each coordinate could be real numbers and the value of each coordinate is not clipped in the discrete integer domain $\{0, \ldots, 255\}$. Therefore, their methods may craft many useless invalid integer images, reducing efficiency. While our method directly crafts adversarial samples the discrete integer domain, hence avoids to craft useless invalid integer images.

2.2 Physical Adversarial Attack

Thanks to the success of adversarial example attacks in the digital domain, recently, researchers started to study the feasibility of adversarial examples in the physical world. We now discuss recent efforts on physical adversarial examples.

Kurakin showed that printed adversarial examples crafted in the digital domain could be misclassified when viewed through a smartphone camera [26]. Follow-up works proposed methods to improve robustness of physical adversarial examples by synthesizing the digital images to simulate the effect of rotation, brightness and scaling, and digital-to-physical transformation [31], [32], [74], [75], or manually taking physical photos from different viewpoints and distances [31], [76], or adding a scene-independent patch [77]. Furthermore, adversarial example attacks have been applied on road sign images [78], [79], face recognition systems [23], and object detectors [80], [81]. Physical adversarial examples that are printed or showed by devices will not be affected by the discretization problem.

Although these works demonstrated that physical adversarial examples are possible, and integer adversarial images may be damaged by image transformations (e.g., photo, brightness, contrast, etc.) in the physical world [26], it is still very useful to generate effective integer adversarial images.

- First, it can be used in many practical scenarios, e.g., attacking the online image classification systems.
- Second, an attacker who cannot fool a classifier successfully in the digital domain will also struggle to do so physically in practice [23].
- Third, it usually requires relatively expensive manual efforts to directly craft physical adversarial examples. On the other hand, robust digital adversarial examples can survive in physical world [75].

It is interesting to study the impacts of the discretization problem on finding physical adversarial examples. Applying our classification model-based derivative free optimization method in the physical world is also an interesting topic. We leave these topics to future work.

2.3 Other Attacks

Adversarial example attacks against other machine learning based classifiers also have been exhibited, such as malicious PDF files [72], [82], [83], malware [84], [85], malicious websites [86], spam emails [87], and speech recognition [88], [89]. Since each type of machine learning based classifiers has unique characteristics, in general, these existing attacks are orthogonal to our work.

3 Background

In this section, we introduce deep learning based image classifications, adversarial attacks and distance metrics. For convenient reference, we summarize the notations in Table 1.

3.1 Deep Learning Based Image Classification

Valid images are represented as integer images in computer systems. To train a practical image classifier $f_t : \mathbb{D} \rightarrow \mathbb{C}_t$, valid images should first be normalized so that their pixels all lie in the same reasonable range, as integer images come in a form that is difficult for many deep learning architectures to represent [51]. Therefore, as shown in Fig. 1, the classifier f_t is constructed by training an image classifier $g_t : \mathbb{V} \rightarrow \mathbb{C}_t$ in continuous (real) domain aided by a normalizer $\mathbb{T} : \mathbb{D} \rightarrow \mathbb{V}$, which leads to the classifier $f_t = g_t \circ \mathbb{T}$.

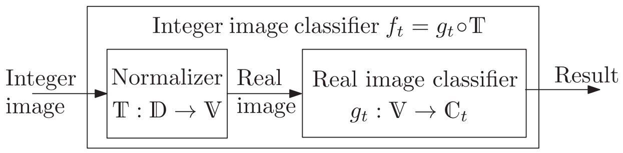

Integer image classifier $f_t = g_t \circ \mathbb{T}$
Integer image
Normalizer $\mathbb{T} : \mathbb{D} \rightarrow \mathbb{V}$
Real image
Real image classifier $g_t : \mathbb{V} \rightarrow \mathbb{C}_t$
Result

Fig. 1. Overview of machine learning based image classifiers.

Neural Networks versus Neural Network Based Classifiers. A neural network consists of an input layer, an output layer and some hidden layers, where each layer consists of some neurons (or nodes) and neurons between layers could be connected, forming a network. The input-output relation of a neural network is a function $g_t : \mathbb{V} \rightarrow \mathbb{C}_t$, where $\mathbb{V}$ denotes the input space and $\mathbb{C}_t$ denotes the output space. As shown in Fig. 1, neural networks are different from neural network based classifiers, where a neural network based classifier $f_t = g_t \circ \mathbb{T}$ contains not only a neural network $g_t : \mathbb{V} \rightarrow \mathbb{C}_t$ but also a normalizer $\mathbb{T} : \mathbb{D} \rightarrow \mathbb{V}$ which transforms inputs from space $\mathbb{D}$ into the input space $\mathbb{V}$ of the neural network.

3.2 Adversarial Attacks

In this work, we consider adversarial attacks using digital adversarial examples instead of physical adversarial examples. We categorize digital adversarial examples into *real* and *integer* ones according to their domains $\mathbb{V}$ and $\mathbb{D}$.

Real and Integer Adversarial Examples. A *real adversarial example* crafted from a real image $\vec{v} \in \mathbb{V}$ is an image $\vec{v}^{\text{adv}} \in \mathbb{V}$ such that the real image classifier g_t misclassifies $\vec{v}^{\text{adv}}$, i.e.,

$$g_t(\vec{v}) \neq g_t(\vec{v}^{\text{adv}}).$$

Likewise, an *integer adversarial example* crafted from an integer image $\vec{d} \in \mathbb{D}$ is an image $\vec{d}^{\text{adv}} \in \mathbb{D}$ such that the integer image classifier f_t misclassifies $\vec{d}^{\text{adv}}$, i.e.,

$$f_t(\vec{d}) \neq f_t(\vec{d}^{\text{adv}}).$$

Untargeted and Targeted Attacks. In the literature, there are two types of adversarial attacks: targeted and untargeted attacks. Untargeted attack aims at crafting an adversarial example that misleads the system being attacked, i.e., $g_t(\vec{v}) \neq g_t(\vec{v}^{\text{adv}})$ for real adversarial examples and $f_t(\vec{d}) \neq f_t(\vec{d}^{\text{adv}})$ for integer adversarial examples. A more powerful but difficult attack, targeted attack, aims at crafting an adversarial example such that the system classifies the adversarial example as the given class c, i.e., $g_t(\vec{v}^{\text{adv}}) = c$ for real adversarial examples and $f_t(\vec{d}^{\text{adv}}) = c$ for integer adversarial examples. It is easy to see that targeted attack can be used to launch untargeted attack by choosing an arbitrary target class.

White-Box and Black-Box Scenarios. Targeted and untargeted attacks have been studied in both white-box and black-box scenarios, according to the knowledge of the target system. In the white-box scenario, the adversary has access to details (e.g., architecture, parameters and training dataset) of the system under attack. This setting is clearly impractical in real-world cases, when the adversary cannot get access to the details. In a more realistic black-box scenario, it is usually assumed that the adversary can only query the system and obtain confidences or probabilities of classes for each input by limited queries.

In the black-box scenario, we emphasize that the adversary has no access to the normalization of the target classification system, otherwise the attack would be a gray-box one. It is also non-trivial to infer the normalization by the adversary in a black-box scenario due to the diversity of normalization. Indeed, there is no standard normalization in literature and they may differ in tools, neural network models and datasets. For instance, let i denote the integer value of a coordinate,

- the Inception-v3 model on ImageNet dataset in ZOO [33] uses the normalization: $v_1 = (i/255 - 0.5)$;
- the Inception-v3 model on ImageNet dataset in Keras [90] uses the normalization: $v_2 = ((2 \times i)/255 - 1)$;
- the VGG and ResNet models on ImageNet dataset in Keras [90] use the normalization: $v_3 = (i - mean)$, where $mean$ denotes the mean value of images in training dataset.

To the best of our knowledge, there is no work on inferring normalization of classifiers. Due to the lack of space, more details refer to [91].

3.3 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 perception of visual difference. In the literature, there are four common distance metrics $\mathbf{L}_0$, $\mathbf{L}_1$, $\mathbf{L}_2$ and $\mathbf{L}_\infty$ which are defined over samples in some continuous domain $\mathbb{V}$. All of them are $\mathbf{L}_n$ norm defined as

$$\|\vec{v} - \vec{v}^{\text{adv}}\|_n = \left(\sum_{p \in P} \left| \vec{v}[p] - \vec{v}^{\text{adv}}[p] \right|^n \right)^{\frac{1}{n}},$$

where $\vec{v}, \vec{v}^{\text{adv}} \in \mathbb{V}$. In more detail, $\mathbf{L}_0$ counts the number of different coordinates, i.e., $\sum_{p \in P} (\vec{v}[p] \neq \vec{v}^{\text{adv}}[p])$; $\mathbf{L}_1$ denotes the sum of absolute differences of each coordinate value, i.e., $\sum_{p \in P} (|\vec{v}[p] - \vec{v}^{\text{adv}}[p]|)$; $\mathbf{L}_2$ denotes euclidean or root-mean-square distance; and $\mathbf{L}_\infty$ measures the largest change introduced. Remark that

$$\lim_{n \rightarrow \infty} \|\vec{v} - \vec{v}^{\text{adv}}\|_n = \max\{ \left| \vec{v}[p] - \vec{v}^{\text{adv}}[p] \right| \mid p \in P \}.$$

However, it seems not reasonable to approximate human's perception of visual difference using distance metrics defined between real images. Instead, it is much better to measure the distance between integer images. For this purpose, we revise distance metrics and introduce $\mathbb{L}_p$ norm which is defined between integer images. Formally, $\mathbb{L}_n$ is defined as follows:

$$\|\vec{d} - \vec{d}^{\text{adv}}\|_n = \left(\sum_{p \in P} \left| \vec{d}[p] - \vec{d}^{\text{adv}}[p] \right|^n \right)^{\frac{1}{n}},$$

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

4 The Discretization Problem

Recall that we categorize digital adversarial examples into real and integer ones according to their domains. There is a gap between adversarial examples in continuous and in discrete domains. In this section, we first formalize the gap as the discretization problem and then report the comprehensive study of the impacts of the discretization problem.

4.1 Formulation of The Discretization Problem

Recall that a practical image classification system f_t is an integer image classifier that consists of both the real image classifier g_t and the normalizer $\mathbb{T}$. Therefore, to attack the system f_t using a real adversarial image $\vec{v}^{\text{adv}} \in \mathbb{V}$ that is crafted by querying g_t, it is necessary to denormalize the real image $\vec{v}^{\text{adv}}$ back into a valid image (i.e, an integer image) $\vec{d}^{\text{adv}} \in \mathbb{D}$, so that it can be fed to the target system f_t. To denormalize $\vec{v}^{\text{adv}}$, a denormalizer $\mathbb{T}^{-1}$ should be implemented according to the knowledge of the normalizer $\mathbb{T}$ such that for any integer image $\vec{d} \in \mathbb{D}$, $\mathbb{T}^{-1}(\mathbb{T}(\vec{d})) = \vec{d}$.

However, after applying the denormalization, $\vec{d}^{\text{adv}}$ may be classified as a class that differs from the one of $\vec{v}^{\text{adv}}$, i.e.,

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

This is so-called *the discretization problem*,[1] which comes from the non-equivalent transformation between continuous real and discrete integer domains, i.e., $\mathbb{T}(\mathbb{T}^{-1}(\vec{v}^{\text{adv}})) \neq \vec{v}^{\text{adv}}$, resulting in

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

In the rest of this work, the maximum error when transforming a real adversarial image back into the discrete domain is called *discretization error*.

Formally, for a neural network based classifier $f_t = g_t \circ \mathbb{T}$ consisting of a neural network $g_t : \mathbb{V} \to \mathbb{C}_t$ and a normalizer $\mathbb{T} : \mathbb{D} \to \mathbb{V}$, *the discretization problem* is the problem that: a real example $\vec{v}$ predicated as the class $g_t(\vec{v})$ by the neural network g_t, will be predicated as another class $f_t(\mathbb{T}^{-1}(\vec{v}))$ (with $f_t(\mathbb{T}^{-1}(\vec{v})) \neq g_t(\vec{v})$) by the neural network based classifier f_t after transforming $\vec{v}$ back into the discrete integer domain $\mathbb{D}$ via the denormalizer $\mathbb{T}^{-1}$.

The discretization problem may result in failure of untargeted and targeted attacks, i.e.,

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

where $\vec{v}^{\text{adv}}$ denotes a real adversarial image crafted from $\mathbb{T}(\vec{d})$ and c denotes the target class.

As stated by Carlini and Wanger [21], the discretization problem slightly degrades the quality of the adversarial example. However, there lacks a comprehensive study of the impacts of the discretization problem. In the rest of this section, we report the first comprehensive study including theoretical analysis of 35 representative methods and empirical study of 20 representative open source tools, in an attempt to understand the impacts of the discretization problem.

4.2 Theoretical Study

We theoretically analyze 35 existing works including 25 testing methods (15 white-box and 10 black-box) and 10 verification methods (9 white-box and 1 black-box), to determine: 1) whether they generate adversarial examples in discrete or continuous domain? 2) if they use some continuous domain, do they consider the discretization problem and how do they deal with it? and 3) if they do not consider it, could the discretization problem be avoided by tuning input parameters? The summary of results is given in Table 2 according to raw papers (primarily) and source code.

1. The term "discretization" comes from Carlini and Wagner[21] which expresses the rounding problem from real numbers to integer numbers. Our definition is more general than theirs.

TABLE 2
Summary of Theoretical Study Results

		Reference	(Un)targeted	Domain	Considered	B2G	Avoid
Testing-based methods	White-box	L-BFGS [18]	Targeted	Continuous	✗	-	✗
		FGSM [19]	Untargeted	Continuous	✓	-	✓
		BIM(ILLC) [26]	Targeted	Discrete	-	-	-
		PGD [92]	Untargeted	Continuous	✗	-	✗
		MBIM [93]	Targeted	Continuous	✗	-	✓
		JSMA [22]	Targeted	Continuous	✗	-	✓
		C&W [21]	Targeted	Continuous	✓	-	✗
		OptMargin [94]	Untargeted	Continuous	✗	-	✗
		EAD [95]	Targeted	Continuous	✗	-	✗
		DeepFool [96]	Untargeted	Continuous	✗	-	✗
		UAP [24]	Untargeted	Continuous	✗	-	✗
		DeepXplore [25]	Untargeted	Continuous	✗	-	✗
		DeepCover [97]	Untargeted	Continuous	✗	-	✗
		DeepGauge [40]	Untargeted	Continuous	✗	-	-
		DeepConcolic [98]	Untargeted	Continuous	✓	-	✗
	Black-box	SModel [35]	Targeted	Continuous	✗	✗	-
		PMG [64]	Untargeted	Continuous	✗	✗	-
		One-pixel [71]	Targeted	Continuous	✗	✗	✗
		ZOO [33]	Targeted	Continuous	✗	✓	✗
		FD [61]	Targeted	Continuous	✗	✗	-
		NES-PGD [39]	Targeted	Continuous	✗	✗	✗
		DBA [27]	Targeted	Continuous	✗	✗	✗
		Bandits [59]	Untargeted	Continuous	✗	✗	✗
		AutoZOOM [37]	Targeted	Continuous	✗	✓	✗
		GenAttack [60]	Targeted	Continuous	✗	✓	✗

		Reference	Complete	Domain	Considered	B2G	Avoid
Verification methods	White-box	BILVNC [99]	✓→ ✗	Continuous	✗	-	✗
		DLV [100]	✗	Continuous	✓	-	✓
		Planet [101]	✓→ ✗	Continuous	✗	-	✗
		MIPVerify [102]	✓→ ✗	Continuous	✗	-	✗
		DeepZ [46]	✗	Continuous	✗	-	✗
		DeepPoly [47]	✗	Continuous	✗	-	✗
		DeepGo [103]	✗	Continuous	✗	-	✗
		ReluVal [104]	✓→ ✗	Continuous	✗	-	✗
		DSGMK [105]	✗	Discrete	-	-	-
	B	SafeCV [38]	✗	Continuous	✓	✓	✓

Note: (un)targeted *column shows the type of attack, once a method could launch targeted attack, we mark it as targeted, as targeted is more powerful than untargeted attack;* Domain *column shows the domain of images;* Considered *column shows whether the method considered the discretization problem;* B2G *column shows whether black-box downgrades to gray-box;* Avoid *column shows whether the discretization problem could be (almost) avoided;* Complete *column shows whether the method is complete, → meaning complete method becomes incomplete due to the discretization problem.*

Discrete or Continuous. After examining the domain of all the 35 works, we found only BIM and DSGMK define the adversarial example searching problem in discrete domains and uses the integer perturbation step sizes. While the other 33 works craft adversarial examples in continuous domains, hence they may be affected by the discretization problem.

Considered or Not. Among 33 works that craft adversarial example in continuous domains, we found only five works (i.e., FGSM, C&W, DeepConcolic, DLV and SafeCV) do consider the discretization problem, while the other 28 works do not, indicating that 28 out of 35 works are affected by the discretization problem.

Specifically, FGSM uses perturbation step sizes that correspond to the magnitude of the smallest bit of an image so that the transformation between continuous and discrete domains is almost equivalent, i.e., the discretization errors are nearly zero. DLV verifies classifiers using discretization such that the crafted real adversarial examples are still adversarial after denormalization. SafeCV limits the perturbation of each pixel to the minimum or maximum values of coordinates. Therefore, the discretization problem in FGSM, DLV and SafeCV are (almost) avoided.

In contrast, C&W and DeepConcolic perform denormalization post-processing before checking crafted real images, and C&W also proposes a greedy algorithm that searches integer adversarial examples on a lattice defined by the discrete solutions by changing one-pixel value at a time. However, discrete solutions are computed by rounding real numbers of coordinates in real adversarial examples to the nearest integers. Therefore, DeepConcolic and C&W either evade or alleviate the discretization problem, but they *cannot* essentially avoid it in theory, as they may craft many useless real adversarial examples.

Avoidable or Not. We further conduct an in-depth analysis of 29 works that craft adversarial example in continuous domains, but do not consider the discretization problem. We investigate whether the discretization problem in these works can be easily and directly avoided by tuning hyper-parameters. We found that only MBIM and JSMA could control the perturbation step sizes directly by hyper-parameters so that the discretization problem could be (almost) avoided by choosing proper perturbation step sizes.

In contrast, 23 out of 29 works do not provide such hyper-parameters so that the discretization problem could not be easily and directly avoided. This is because that

- PGD, DeepXplore, One-pixel, NES-PGD, DBA, Bandits and GenAttack introduce random perturbation step size or random noise, making perturbation step size uncontrollable;
- L-BFGS, OptMargin, EAD, DeepFool, DeepCover, UAP, ZOO and AutoZOOM directly craft perturbations (e.g., from optimizers) in continuous domain;
- BILVNC, Planet, MIPVerify, DeepZ, DeepPoly, DeepGo, ReluVal, and DSGMK do not provide any parameters to constrain real adversarial examples so that the discretization error cannot be minimized.

The remaining 4 methods DeepGauge, SModel, PMG and FD actually leverage other attack methods such as (FGSM, BIM, JSMA, and C&W). Therefore, the impacts of the discretization problem on their methods rely upon other attacks.

Discussion. After an in-depth analysis of 35 existing works, we found that 34 works craft adversarial examples in continuous domains, 29 works are affected by the discretization problem, and 23 works do not provide hyper-parameters to avoid the discretization problem. As aforementioned, real adversarial examples may be damaged when transforming back into valid images, due to the discretization problem, hence fail to launch attacks. Besides this, there are other severe consequences: (1) the black-box methods such as ZOO, AutoZOOM, GenAttack and SafeCV downgrade to gray-box ones, as they directly invoke the normalization of the integer classification systems; (2) the verification methods such as BILVNC, MIPVerify, Planet and ReluVal that are claimed complete are only limited to real image classifiers, and become incomplete on practical image classification systems that are indeed integer image classifiers; and (3) the verification methods such as BILVNC, MIPVerify, Planet, ReluVal, DeepZ, DeepPoly, DeepGo and DSGMK may craft spurious adversarial examples and fail to prove robustness of integer image classifiers.

Moreover, during our study, we found there are lots of Github issues, e.g., [52], [53], [54], asking why adversarial examples are no longer adversarial after saving them in some format (e.g., png). Users may doubt whether the attack was not properly implemented, or images were saved in a wrong way. According to our findings, these issues are caused by the discretization problem.

4.3 Empirical Study

We conduct an empirical study on 20 representative methods in Table 3 whose source code is publicly available, in an attempt to understand the impacts of the discretization problem in practice.

We consider the following two research questions:

RQ1: To what extent does the discretization problem affect the attack success rate?

RQ2: Can the discretization problem be avoided or alleviated by tuning input parameters?

Setting. In our experiments, we use the official implementations of the authors. Due to the diversity of these tools, the dataset and setting may be different. We manage to be consistent with the original environments in their raw papers, attack the target models provided by the tools, and conduct targeted attacks unless the tools are designated for untargeted attacks. For verification tools that cannot directly attack the model, we evaluate them by analyzing the generated counterexamples. Although, we do not change their settings deliberately to get exaggerative results, we should emphasize that the comparison between these tools may be unfair, *our main goal is to understand their own tools.*

Dataset. We use two popular image datasets: MNIST [55] and ImageNet [57]. ImageNet contains over 10000000 images with 1000 classes. We randomly choose 100 classes from which we randomly choose 1 image per class that can be correctly classified by four classifiers in Keras: ResNet50, Inception-v3, VGG16 and VGG19. For MNIST images, the numbers of used images are shown in the last column in Table 3, which depends on the efficiency of the tool under test.

Metrics. We introduce three metrics to evaluate the impacts of the discretization problem. Let N denote the number of input images under test, N_v denote the number of successfully crafted real adversarial examples, and N_i denote the number of integer adversarial examples after the denormalization post-processing,

- Success Rate (SR) is calculated as $\frac{N_v}{N}$,
- True Success Rate (TSR) is calculated as $\frac{N_i}{N}$,
- GAP between SR and TSR is calculated as $\frac{SR-TSR}{SR}$.

To compute N_i, we use the denormalizer provided by the corresponding tools.

4.3.1 RQ1

To answer this research question, we conduct experiments using default input parameters in their raw papers or tools, which have been fine-turned for effectiveness by corresponding authors and widely used by existing works. The results are shown in Table 3.

We can observe that 14 out of 20 tools are affected by the discretization problem. Their gaps range from 0.03 to 100 percent. In more detail, 8 tools have gaps exceeding 50 percent

TABLE 3
Experiment Results on the Discretization Problem

Method	SR	TSR	GAP	Dataset	Model	Default	Note
FGSM [19]	98.61%	98.58%	0.03%	MNIST	LeNet-1$^\sharp$	✓	10000 images
BIM [26]	100%	100%	0%	ImageNet	Inception-v3$^\sharp$	✓	-
MBIM [93]	100%	100%	0%	ImageNet	Inception-v3$^\sharp$	✓	-
JSMA [22]	96%	96%	0%	ImageNet	VGG19$^\sharp$	✓	-
L-BFGS [106]	100%	77%	23%	ImageNet	Inception-v3$^\sharp$	✓	-
C&W-$\mathbf{L}_2$ [21]	100%	10%	**90%**	ImageNet	Inception-v3*	✓	-
DeepFool [96]	100%	23%	**77%**	ImageNet	ResNet34*	✓	-
DeepXplore [25]	65%	28%	**56.92%**	ImageNet	ResNet50, VGG16&19*	✓	Generate examples with 100 seeds
DeepConcolic [98]	2%	2%	0%	MNIST	mnist_complicated.h5*	✓	10000 images with criterion='nc'
ZOO [33]	58%	6%	**89.66%**	ImageNet	Inception-v3*	✓	-
DBA [27]	100%	28%	**72%**	ImageNet	VGG19$^\sharp$	✓	-
NES-PGD [39]	100%	53%	47%	ImageNet	Inception-v3*	✓	-
Bandits [59]	94%	11%	**88.3%**	ImageNet	Inception-v3*	✓	-
GenAttack [60]	100%	91%	9%	ImageNet	Inception-v3*	✓	-
DLV [100]	90%	90%	0%	MNIST	NoName*	✓	20 images
Planet [101]	100%	46%	**54%**	MNIST	testNetworkB.rlv*	✓	Use 'GIVE' model obtain 20 images
MIPVerify [102]	42%	0%	**100%**	MNIST	MNIST.n1*	✓	Quickstart demo with 100 images
DeepPoly [47]	45%	44%	2.22%	MNIST	convBigRELU_DiffAI*	✓	Gap between $\epsilon = 0.3$ and $\epsilon = 76/255$
DeepGo [103]	25.4%	25.2%	0.78%	MNIST	NoName*	✓	Crafted 1000 images from 1 image
SafeCV [38]	100%	100%	0%	MNIST	NoName*	✓	100 images

Note: $*$ means the target model in the corresponding tool; $\sharp$ means that their tools do not have any target models and we choose widely used target models from Tensorflow or Keras; and Default *means default input parameters.*

including white-box testing tools (C&W-$\mathbf{L}_2$, DeepFool, DeepXplore), black-box testing tools (ZOO, DBA and Bandits) and verification tools (Planet and MIPVerify). Among them, 6 tools have gaps exceeding 70 percent. This demonstrates that if attackers do not pay attention to the discretization problem, they will be likely to generate real adversarial examples which will be damaged after transforming them back into the discrete domain.

There are only 6 out of 20 tools that do not have any gaps including BIM, MBIM, JSMA, DeepConcolic, DLV and SafeCV. These results are largely consistent with our theoretical study.

Answering RQ1: The results on 20 tools show that most of them are affected by the discretization problem. There are 8 tools whose gap exceeds 50 percent, and 6 tools whose gap exceeds 70 percent, and only 6 tools do not have any gaps.

4.3.2 RQ2

To answer this research question, we propose different strategies to tune input parameters for these 14 tools whose gap is not 0 in RQ1. According to our findings in theoretical study, we distinguish these tools by whether the discretization problem can be easily and directly avoided by tuning input parameters. Remark that we do not investigate how to modify their implementations and methods by taking the discretization problem into account. First, it is a tedious and error-prone process. Second, modifying their implementations may significantly under-estimate their effectiveness and efficiency, as pointed out by Carlini [107], hence less convincing.

Case 1: Empirical study on the tools where the discretization problem can be easily and directly avoided by tuning input parameters in theory. Based on the results in Table 3, we can observe that only FGSM has a non-zero gap and its discretization problem can be easily and directly avoided by tuning input parameters. The default perturbation step size ϵ used in Table 3 is 0.3. Therefore, we revise ϵ to $76/255$ in order to avoid the discretization problem. Then the gap decreases to 0, which confirms our theoretical findings.

To illustrate the importance of controllable perturbation step sizes, we also test the implementations of BIM and MBIM in other toolkits, such as Foolbox [108]. Different from the raw implementation of these tools, Foolbox provides a binary search by default. The binary search is performed between the original clean input and the crafted adversarial image, intending to find the exact adversarial boundary. It has been adopted in recent attacks, e.g., [109], [110]. However, if the binary search is implemented without taking into the discretization problem account such as BIM and MBIM in Foolbox, the perturbation step size will become uncontrollable. We use the same input parameters of BIM and MBIM as in RQ1, except that the binary search is enabled (default in Foolbox). Compared to the results in Table 3, the gaps of both BIM and MBIM increase from 0 to 90 percent. These results show that attackers should pay more attention to input parameters even the discretization problem is avoidable.

Case 2: Empirical study on the tools where the discretization problem cannot be easily and directly avoided by tuning input parameters in theory. Based on the results in Table 3, there remain 13 tools whose gaps are non-zero, and the discretization problem cannot be easily and directly avoided by tuning input parameters. We do our best to fine-turn the input parameters of those tools aimed at increasing TSR and decreasing the gap.

First of all, as discussed in theoretical study, the verification tools (i.e., Planet, MIPVerify, DeepPoly and DeepGo) do not provide any parameters to constrain real adversarial examples so that the discretization error could be minimized, we cannot tune input parameters of those tools. For the other 9 test-based tools (i.e., white-box attacks L-BFGS, C&W, DeepFool and DeepXplore, and black-box attacks ZOO, DBA, NES-PGD, Bandits and GenAttack), we adopt the following three strategies to alleviate the discretization problem:

S1: forbidding adaptive perturbation step size: aims at controlling perturbation step sizes. NES-PGD, DBA, Bandits and GenAttack provide such adaptive mechanism.
S2: increasing overall perturbations: aims at minimizing the ratio of discretization error against the overall perturbations. L-BFGS, DeepFool, DeepXplore and DBA provide input parameters related to this strategy.
S3: enhancing strength/confidence of adversarial examples: aims at enhancing the robustness of real adversarial examples. C&W and ZOO provide input parameters related to confidence.

After tuning input parameters (for details refer to [91]), none of them is able to eliminate the discretization errors absolutely.

In terms of TSR, we found that:

- By applying S1, the TSR of NES-PGD and DBA can increase, but the TSR of Bandits and GenAttack cannot;
- By applying S2, the TSR of DBA can increase, but the TSR of DeepXplore, L-BFGS and DeepFool cannot;
- By applying S3, the TSR of C&W-$\mathbf{L}_2$ can increase, but the TSR of ZOO cannot.

This demonstrates that our strategies are able to increase TSR for 3 tools, but fail to increase TSR for the other 6 tools. However, these strategies also bring some side effects, namely, increasing either overall perturbations in terms of Mean Square Error (MSE) or the number of query times, hence sacrificing attack efficiency and imperceptibility of adversarial samples. Due to limited space, the detailed statistic is given in [91].

Answering RQ2: According to our experiences, among 14 tools that are affected by the discretization problem, only 1 tool, FGSM, can definitely avoid the discretization problem by tuning input parameters, and only 3 tools can alleviate the discretization problem by tuning input parameters at the cost of attack efficiency or imperceptibility.

Discussion. Our empirical study reveals that the discretization problem is more severe than originally thought in practice, in conformance with the results of our theoretical study. According to our experimental results, the attack results in published works may not be as good as those reported in raw papers. For instance, DeepFool assumed that the classifier g_t in continuous domain is the same as the classifier f_t in discrete interger domain which contradicts to our empirical result, e.g., it has gap 77 percent in Table 3. We believe it is important to highlight the potential impacts of the discretization problem, and by no means invalidate existing methods or their importance and contributions.

Compared to adjusting parameters, it may be more effective to change the rounding direction of float point numbers to alleviate the discretization problem. In default, each float point number is round down to the nearest integer. A more promising approach is to round float point numbers guided by the direction of derivatives. Specifically, when transforming a real adversarial example back into the integer domain, the value of a pixel is round up to the nearest integer if its derivative is positive, otherwise round down to the nearest integer. This approach can be seen as performing an additional FGSM-like attack after the original attack. Though it is trivial to obtain derivatives for white-box attacks, it is difficult to compute derivatives for black-box attacks, as estimating the gradient in the black-box scenario often requires a large number of queries, e.g., estimating the gradient of a 299*299*3 image using the finite difference method [61] needs 500,000 queries. Therefore, we implement this approach on the four white-box attacks, L-BFGS, DeepFool, C&W and DeepXplore, where the discretization problem cannot be easily and directly avoided by tuning input parameters in theory. The TSR of L-BFGS, DeepFool, C&W and DeepXplore increases from 77, 23, 10 and 28 to 97, 36, 82 and 37 percent, respectively, demonstrating that this approach is able to alleviate the discretization problem to some extent. However, MSE increases 1.9 times on average, hence sacrificing the imperceptibility of adversarial samples.

It is worth noting that Carlini and Wagner [21] proposed a greedy search based algorithm to alleviate the discretization problem. This approach iteratively modifies the pixel with the largest gradient, thus, can be seen as performing an additional JSMA-like attack when the discretization problem occurs. We conduct an experiment on the greedy search based version of C&W-$\mathbf{L}_2$ which are obtained from Carlini. In our experiment, we use input parameters recommended by Carlini for MNIST images. We found that the greedy search based algorithm significantly improves TSR and reduces gaps without increasing distortions of crafted adversarial examples. This demonstrates that the greedy search based algorithm is a solution to alleviate the discretization problem when one cannot precisely control perturbation step sizes by adjusting input parameters. However, due to the fact that the greedy search based algorithm leverages gradients of targeted networks frequently, it is difficult to integrate it into black-box attacks.

According to our findings, we suggest that: (1) attack success rate should be measured using integer adversarial examples instead of real adversarial examples; (2) it is vital to pay more attention to perturbation step sizes that can be controlled by input parameters; and (3) it is better to revise the implementations of the tools that cannot easily avoid the discretization problem by tuning input parameters if one wants to achieve higher TSR but do not sacrifice the attack efficiency and imperceptibility of adversarial examples.

5 An Approach for Black-Box Attack

According to our study in Section 4, there lacks an effective and efficient *integer* adversarial example attack in black-box scenario. As a first step towards addressing this problem, we propose a novel black-box algorithm for both targeted and untargeted attacks by presenting a *classification model-based* Derivative-Free discrete Optimization (DFO) method. This type of DFO methods has been widely used to solve complex 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, or even unknown but only testable. Furthermore, it was shown by Yu *et al.* [111] that it is not only superior to many state-of-the-art DFO methods (e.g., genetic algorithm, Bayesian optimization and cross-entropy method), but also stable. We refer readers to [111] for the advantages of classification model-based DFO methods.

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

Threat Model. In our black-box scenario, we assume that the adversary does not have any access to any details (e.g., normalization, architecture, parameters and training data) of the target classifier, but he/she knows the input format of the target classifier and has access to the probabilities (or confidences) of all classes for each input image which are widely used assumptions in the black-box scenario [34], [35], [61], [72], [83].

In our attack, the distortion of adversarial examples is measured by the $\mathbb{L}_\infty$ distance metric, which is one of the most widely used metrics to model human perception. The $\mathbb{L}_\infty$ metric imposes the same constraints on the perturbation range of each coordinate. Compared with a benign image, an adversarial example generated under this constraint has a more significant euclidean distance but is relatively uniform and smooth [112], [113], [114]. The $\mathbb{L}_0$ metric limits the number of coordinates that the attacker can modify, and the $\mathbb{L}_2$ metric limits the maximum euclidean distance between an adversarial example and a benign image. The overall perturbation generated by an attack under these two metrics is generally small, but the differences at some specific coordinates may be significant, and thus can be perceived by human-being [115], [116], [117]. Most of the baseline tools we compared in the experimental section support the $\mathbb{L}_\infty$ metric [19], [21], [26], [35], [39], [59], [60], [61]. Furthermore, our attack could be easily revised to use $\mathbb{L}_0$ or $\mathbb{L}_2$ metric by adapting our dissatisfaction-degree function according to [21]. We leave this as future work.

5.1 Framework of DFA

Fig. 2 shows the framework of our approach named DFA, standing for **D**erivative-**F**ree **A**ttack. Given an integer image, DFA directly searches an adversarial image in a (discrete integer) search space specified by the maximum $\mathbb{L}_\infty$ distance.

In principle, DFA first samples some perturbations from the search space and repeats the following procedure until an integer adversarial example is found. During each iteration, DFA queries the target classifier to measure the images (perturbations added onto the input image) via a given *dissatisfaction degree function* which predicates how far is an image from a successful attack. The perturbations are partitioned into two parts w.r.t. *dissatisfaction degrees*: perturbations yielding *high* dissatisfaction degrees and perturbations yielding *low* dissatisfaction degrees. The search space is refined into a *small sub-space* according to the partitions of perturbations. New perturbations are sampled from the refined sub-space. Together with old perturbations, a set of best-so-far perturbations is selected according to their dissatisfaction degrees. Finally, the procedure is repeated on the best-so-far perturbations which will be used to refine the sub-space again.

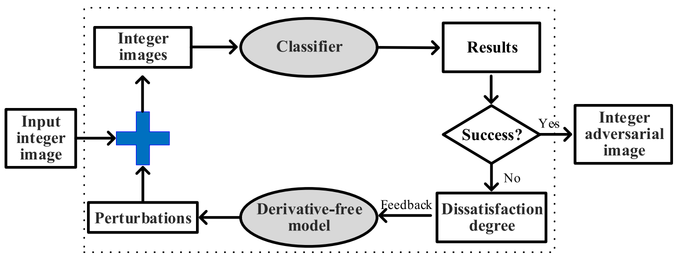


Fig. 2. Framework of our approach DFA.

5.2 Formulation

We formalize the integer adversarial example searching problem as a derivative-free discrete 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 and an integer number ϵ denoting the maximum $\mathbb{L}_\infty$ distance. We denote by $\mathcal{P}(\vec{d})$ the vector of probabilities on the image $\vec{d}$ and by $\mathcal{P}(\vec{d}, c)$ the probability that the image $\vec{d}$ is classified to the class $c \in \mathbb{C}_t$. For a given integer j such taht $1 \leq j \leq |\mathbb{C}_t|$, we denote by $\texttt{Top}_j(\vec{d})$ the jth largest probability in $\mathcal{P}(\vec{d})$ and $\texttt{Top}_j^\ell(\vec{d})$ the class whose probability is $\texttt{Top}_j(\vec{d})$. Obviously, $\texttt{Top}_1^\ell(\vec{d}) = f_t(\vec{d})$.

We define the initial search space Δ of perturbations as a discrete integer domain $\mathbb{N}_{[-\epsilon,\epsilon]}^{w\times h\times ch}$. Specifically, the discrete domain Δ is a two-dimensional array such that for each coordinate $p \in P = w \times h \times ch$, $\Delta[p][\texttt{low}]$ and $\Delta[p][\texttt{high}]$ (such that $\Delta[p][\texttt{high}] \geq \Delta[p][\texttt{low}]$) are integer numbers respectively denoting the lower and upper bound of the value at the coordinate p. Therefore, Δ denotes a set of perturbations such that $\delta \in \Delta$ if and only if $\Delta[p][\texttt{low}] \leq \delta[p] \leq \Delta[p][\texttt{high}]$ for all coordinates $p \in P$. The search space Δ will be refined into small sub-spaces by increasing lower bound $\Delta[p][\texttt{low}]$ or decreasing upper bound $\Delta[p][\texttt{high}]$ for choosing coordinates p in our algorithm.

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

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

The *integer adversarial example searching problem* with respect to the maximum $\mathbb{L}_\infty$ distance ϵ is to find some perturbation $\delta \in \Delta$ such that:

- $\texttt{Top}_1^\ell(\vec{d} \oplus \delta) \neq f_t(\vec{d})$ for untargeted attack;
- $\texttt{Top}_1^\ell(\vec{d} \oplus \delta) = c$ for targeted attack with a target class c.

We solve the integer adversarial example searching problem by reduction to a 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 find some perturbation $\delta \in \Delta$ such that $\mathtt{Top}_1^{\ell}(\vec{d} \oplus \delta) \neq f_t(\vec{d})$. To do this, we maximize the current probability of the image $\vec{d} \oplus \delta$ being classified as the class $\mathtt{Top}_2^{\ell}(\vec{d} \oplus \delta)$ (i.e., the current class with second largest probability, which may change w.r.t. different δ) until the image is able to successfully mislead the classifier. Therefore, we define the *dissatisfaction-degree function* for untargeted attack, denoted by $D_{\mathtt{ua}}(\cdot,\cdot)$, as follows:

- $D_{\mathtt{ua}}(\vec{d}, \delta) := 0$, if $\mathtt{Top}_1^{\ell}(\vec{d} \oplus \delta) \neq f_t(\vec{d})$;
- $D_{\mathtt{ua}}(\vec{d}, \delta) := 1 - \mathtt{Top}_2(\vec{d} \oplus \delta)$, otherwise.

In this function, if the attack has succeeded, the perturbation δ is "satisfying", then the value of the dissatisfaction-degree becomes 0. Otherwise, we return the distance between 1 and the currently reported second largest probability, which is in the range of [0,1], indicating how far it is from 1. Clearly, in this case, the distance is definitely positive. To this end, our goal is to find a perturbation δ such that the dissatisfaction-degree is 0.

For targeted attack with the target class c, instead of maximizing the probability of $\vec{d} \oplus \delta$ being classified as the class $\mathtt{Top}_2^{\ell}(\vec{d} \oplus \delta)$, we maximize the probability of the image $\vec{d} \oplus \delta$ being classified as c. Hence, the *dissatisfaction-degree function*, denoted by $D_{\mathtt{ta}}(\cdot,\cdot)$, is defined as follows:

- $D_{\mathtt{ta}}(\vec{d}, \delta) := 0$, if $\mathtt{Top}_1^{\ell}(\vec{d} \oplus \delta) = c$;
- $D_{\mathtt{ta}}(\vec{d}, \delta) := 1 - \mathcal{P}((\vec{d} \oplus \delta), c)$, otherwise.

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

5.3 Algorithm

Instead of using heuristic search methods, e.g., genetic programming, particle swarm optimization, simulated annealing, to solve the minimization problem of the dissatisfaction-degree functions, we propose a classification model-based DFO method (shown in Algorithm 1). Unlike heuristic search based methods, our method maintains a classification model during the search to distinguish "good" samples from "bad" samples. Then, the algorithm will refine the search space Δ by learning from the samples to converge to the best solution.

In detail, Algorithm 1 first initializes the search space Δ according to the given maximum $\mathbb{L}_\infty$ distance ϵ (Line 1). Then, it randomly selects $(s+k)$ perturbations (stored as the set B_0) from the search space Δ (Line 2), where s denotes the sample size during each iteration and k denotes the ranking threshold. Next, it computes $(s+k)$ valid images by adding the perturbations onto the source integer image $\vec{d}$ and evaluates the dissatisfaction-degree (d.d.) of these images using the dissatisfaction-degree function D (Line 3). The perturbation $\tilde{x}$ with the smallest dissatisfaction-degree is selected from set B_0 (Line 4). After that, Algorithm 1 repeats the following procedure.

For each iteration $t \geq 1$, if the perturbation $\tilde{x}$ suffices to craft an integer adversarial example, return $\tilde{x}$ (Lines 6-7). Otherwise, the set B_{t-1} of perturbations is partitioned 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 (Lines 8-9).

Algorithm 1. A DFO-Based Algorithm

```
Input: classifier under attack $f_t : \mathbb{D} \to \mathbb{C}_t$, integer image $\vec{d} \in \mathbb{D}$,
    number of iterations $T \in \mathbb{N}$,
    ranking threshold $k \in \mathbb{N}$, sample size $s \in \mathbb{N}$, maximum $\mathbb{L}_\infty$
    distance $\epsilon \in \mathbb{N}$,
    the number of coordinates to be changed in each refinement process $u \in \mathbb{N}$,
    dissatisfaction-degree (d.d.) function $D$
Output: optimized perturbation $\tilde{x}$
1:  $\Delta = \mathbb{N}_{[-\epsilon,\epsilon]}^{w \times h \times ch}$;
2:  $B_0 = \{\delta_1, \ldots, \delta_{s+k}\}$ sampled from $\Delta$; // initial collection
3:  Evaluate the dissatisfaction-degree $D(\vec{d}, \delta_i)$ for
    $1 \leq i \leq s+k$;
4:  $\tilde{x} = \mathtt{argmin}_{\delta \in B_0} D(\vec{d}, \delta)$; // select the best-so-far sample
5:  for $t = 1$ to $T$ do
6:      if $D(\vec{d}, \tilde{x}) = 0$ then
7:          break; // find an adversarial example
8:      $B_{t-1}^+ =$ smallest-k solutions in $B_{t-1}$ in terms of d.d.;
9:      $B_{t-1}^- = B_{t-1} - B_{t-1}^+$;
10:     $B = \emptyset$;
11:     for $i = 1$ to $s$ do
12:         // Refine the space $\Delta$ into a small one by $B_{t-1}^+$ and $B_{t-1}^-$
13:         Randomly select a sample $b^+$ from the positive set
            $B_{t-1}^+$;
14:         $Y = \emptyset$;
15:         for $j = 1$ to $u$ do
16:             Randomly select a coordinate $p$ from
                $P = w \times h \times ch$;
17:             $Y = Y \cup \{p\}$;
18:             $\mathtt{ge} := \{b \in B_{t-1}^- \mid b[p] > b^+[p]\}$;
19:             $\mathtt{le} := \{b \in B_{t-1}^- \mid b[p] < b^+[p]\}$;
20:             if $|\mathtt{ge}| > |\mathtt{le}|$ then
21:                 $\mathtt{minVal} = \mathtt{min}_{b \in \mathtt{ge}} b[p]$;
22:                 Randomly select an integer $r$ from $\mathtt{minVal}$ to $b^+[p]$;
23:                 $\Delta[p][\mathtt{high}] = r$; // decrease the upper bound at $p$
24:             else
25:                 $\mathtt{maxVal} = \mathtt{max}_{b \in \mathtt{le}} b[p]$;
26:                 Randomly select an integer $r$ from $b^+[p]$ to $\mathtt{maxVal}$;
27:                 $\Delta[p][\mathtt{low}] = r$; // increase the lower bound at $p$
28:         $b' = \mathtt{Copy\ of}\ b^+$;
29:         for $p \in Y$ do
30:             // Sample in the refined the search space $\Delta$
31:             Randomly select an integer $r$ from $\Delta[p][\mathtt{low}]$ to
                $\Delta[p][\mathtt{high}]$;
32:             $b'[p] = r$;
33:         $B = B \cup \{b'\}$;
34:         $\Delta = \mathbb{N}_{[-\epsilon,\epsilon]}^{w \times h \times ch}$; // Reset $\Delta$ for next sample to avoid over
            fitting
35:     Evaluate the dissatisfaction-degree $D(\vec{d}, \delta)$ for all $\delta \in B$;
36:     $B_t =$ smallest-$(s+k)$ solutions in $B \cup B_{t-1}$ in terms of
        dissatisfaction-degree // keep the size as $s+k$;
37:     $\tilde{x} = \mathtt{argmin}_{\delta \in B_t} D(\vec{d}, \delta)$;
38: return $\tilde{x}$;
```

Based on B_{t-1}^+ and B_{t-1}^-, Algorithm 1 refines the search space Δ into a small sub-space (Lines 11-34) as follows. It first randomly selects a sample b^+ from the positive set B_{t-1}^+ (Line 13) and randomly selects u coordinates to refine (Lines 15-27). For each selected coordinate p, it compares the number of perturbations in B_{t-1}^- whose value is larger than the value of b^+ at the coordinate p against the number of perturbations in B_{t-1}^- whose value is smaller than the value of b^+ at the coordinate p. If the majority of perturbations in B^{-}_{t-1} are larger than b^{+} at the coordinate p, we decrease the upper bound of the search space Δ (Lines 20-23), otherwise increase the lower bound (Lines 25-27), at the coordinate p. Once u coordinates have been processed, we craft a new image b' from b^{+} by reassigning the value of each coordinate $p \in Y$ with the random integer r from $\Delta[p][\texttt{low}]$ to $\Delta[p][\texttt{high}]$ (Lines 29-33). The new perturbation b' is added into the set B. Then, the search space Δ is reset to the original size. We remark that the refining procedure for the next sample will be conducted on the original search space to avoid over fitting.

When the search space Δ has been refined s times, we get s new perturbations (i.e., set B), resulting in $(2s + k)$ perturbations in the set $B \cup B_{t-1}$. From them, we choose the smallest-$(s + k)$ perturbations in terms of the dissatisfaction-degree (Line 37). Algorithm 1 continues the above procedure on B_t until an integer adversarial example is found or the number of iterations T is reached.

Dimensionality Reduction. Algorithm 1 depicts the workflow of our approach, which solves the integer adversarial example searching problem by a classification model-based DFO method. It can be further optimized by a dimensionality reduction technique, which reduces the search space Δ into a lower dimensional space to improve query efficiency. Dimensionality reduction has been adopted in recent attacks, e.g., AutoZOOM [37] and GenAttack [60]. Instead of searching in the large search space $\Delta = \mathbb{N}^{w\times h\times ch}_{[-\epsilon,\epsilon]}$, we can first search a perturbation δ_r in a small search space $\Delta_r = \mathbb{N}^{w_r\times h_r\times ch}_{[-\epsilon,\epsilon]}$ for $w_r \leq w$ and $h_r \leq h$, and scale δ_r up to δ_o with the same size as input (i.e., the search space Δ) by applying resizing methods (e.g., bilinear resizing), resulting in the valid image $\vec{d} \oplus \delta_o$ in the original size. (Please refer to [37] and [60] for more details about dimensionality reduction.) By doing so, the query efficiency of our method can be improved while maintaining the attack success rate under the $\mathbb{L}_\infty$ constraint.

5.4 Illustrative Example

We illustrate Algorithm 1 through an example, as shown in Fig. 3. The original integer image $\vec{d}$ is an image from the ImageNet dataset and it is classified as the class *flamingo* by the target classifier Inception-v3. To launch an untargeted attack using this image, we set the sample size s as 3 and the ranking threshold k as 2. Consider the first iteration, Algorithm 1 samples 5 perturbations $(\delta_i)_{1\leq i\leq 5}$ from $\Delta = \mathbb{N}^{w\times h\times ch}_{[-20,20]}$ and adds them onto the original image, resulting in five new images (shown in Fig. 3). Then, it computes the dissatisfaction degrees of these five new images $(\vec{d} \oplus \delta_i)_{1\leq i\leq 5}$ by querying the classifier and the dissatisfaction-degree function D_{ua}. Among these 5 perturbations, $(\vec{d} \oplus \delta_1)$ has the smallest dissatisfaction degree, hence δ_1 is the best-so-far perturbation. After more iterations, the results are shown in Fig. 4. We can see that after 388 iterations, the image with the smallest dissatisfaction degree is classified as the class *hook*, but is visually indistinguishable from the original one.

5.5 Scenario Extension

Our framework is very reflexible and could be potentially adapted to other scenarios such as: (1) target classifiers that only output *top-1 class* and *its probability*, and (2) target classifiers that are integrated with defenses, by restricting the search space or modifying dissatisfaction-degree functions.

For instance, if the adversary only has access to the top-1 class and its probability, the dissatisfaction-degree function for untargeted attack can be adapted as follows:

- $D^1_{\texttt{ua}}(\vec{d}, \delta) := 0$, if $\texttt{Top}^\ell_1(\vec{d} \oplus \delta) \neq f_t(\vec{d})$;
- $D^1_{\texttt{ua}}(\vec{d}, \delta) := \texttt{Top}_1(\vec{d} \oplus \delta)$, otherwise.

The dissatisfaction-degree function for targeted attack could be adapted accordingly. Remark that it is different from label-only attacks in which the adversary has access to the top-1 class, but *not* its probability. We leave this to future work.

6 Implementation and Evaluation

We implement our classification model-based DFO method in DFA based on the framework of RACOS [111], for which we implement our new algorithm and manage to engineer to significantly improve its efficiency and scalability with lots of domain-specific optimizations. Hereafter, we report experimental results compared with state-of-the-art white-box and black-box attacks.

6.1 Dataset & Setting

Dataset. We use two standard datasets MNIST [55] and ImageNet [57]. MNIST is a dataset of handwritten digits with 10 classes (0-9). We choose the first 200 images out of 10000 validation images of MNIST as our subjects.

We use the same 100 ImageNet images as in Section 4.3. (Recall that we randomly choose 100 classes from which we randomly choose 1 image per class that can be correctly classified by four classifiers in Keras: ResNet50, Inception-v3, VGG16 and VGG19.)

Target Model. For MNIST images, we use a DNN classifier LeNet-1 from the LeNet family [56]. LeNet-1 is a popular target model for MNIST images, e.g., [25], [40], [118], [119], [120]. For ImageNet images, we use a pre-trained DNN classifier Inception-v3 [58] which is a widely used target model for ImageNet images, e.g., [21], [33], [37], [39], [60].

Setting. As shown in Section 4.3, the discretization problem can be avoided or alleviated by tuning input parameters for some tools, at the cost of attack efficiency or quality of adversarial examples, except for FGSM and C&W+GS. Therefore, to maximize their TSRs as done in Section 4.3, we choose proper step sizes for FGSM and enable greedy search for C&W+GS with parameters recommended by Carlini on MNIST images. For other tools, we use the parameters as in their raw papers which are already fine-tuned by the authors for effectiveness and efficiency. A discussion on turning input parameters refers to Section 4.3.2. Furthermore, some tools only provide implementations for attacking under some specific settings. If this issue happens, we may not modify their implementations as it may greatly under-estimate their effectiveness and efficiency, as pointed out by Carlini [107], hence some attacks are not evaluated in all settings.

We conduct both untargeted attack and targeted attack on a Linux PC running UBUNTU 16.04 LTS with Intel Xeon (R) W-2123 CPU, TITAN Xp COLLECTORS GPU and 64G RAM. Table 4 lists the other experiment settings.

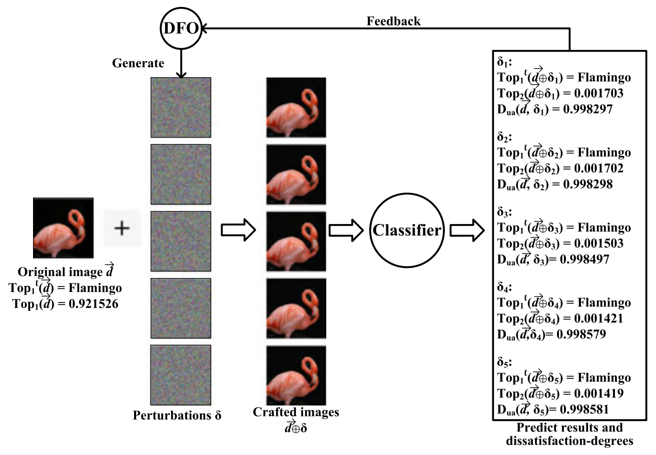

DFO
Feedback
Generate
+
Original image $\vec{d}$
$Top_1^t(\vec{d})$ = Flamingo
$Top_1(\vec{d})$ = 0.921526
Perturbations δ
Crafted images $\vec{d}\oplus\delta$
Classifier
δ_1: $Top_1^t(\vec{d}\oplus\delta_1)$ = Flamingo $Top_2(\vec{d}\oplus\delta_1)$ = 0.001703 $D_{ua}(\vec{d},\delta_1)$ = 0.998297
δ_2: $Top_1^t(\vec{d}\oplus\delta_2)$ = Flamingo $Top_2(\vec{d}\oplus\delta_2)$ = 0.001702 $D_{ua}(\vec{d},\delta_2)$ = 0.998298
δ_3: $Top_1^t(\vec{d}\oplus\delta_3)$ = Flamingo $Top_2(\vec{d}\oplus\delta_3)$ = 0.001503 $D_{ua}(\vec{d},\delta_3)$ = 0.998497
δ_4: $Top_1^t(\vec{d}\oplus\delta_4)$ = Flamingo $Top_2(\vec{d}\oplus\delta_4)$ = 0.001421 $D_{ua}(\vec{d},\delta_4)$ = 0.998579
δ_5: $Top_1^t(\vec{d}\oplus\delta_5)$ = Flamingo $Top_2(\vec{d}\oplus\delta_5)$ = 0.001419 $D_{ua}(\vec{d},\delta_5)$ = 0.998581
Predict results and dissatisfaction-degrees

Fig. 3. Untargeted attack on a Flamingo image (the first iteration).

6.2 Comparison With White-Box Methods

Although our method is a black-box one, we compare the performance with four well-known white-box tools: FGSM, BIM, C&W and C&W+GS, where the implementations are by their authors. Since FGSM has nearly no ability to handle targeted attack, we use one-step target class method (denoted by FGSM-1) of [113], which can be regarded as the targeted version of FGSM. The maximum $\mathbb{L}_\infty$ distances are transformed into their maximum $\mathbf{L}_\infty$ distances accordingly.

The results are shown in Tables 5 and 6 for untargeted and targeted attacks, respectively. Notice that C&W+GS only implements attacks for MNIST images, hence is not applied to ImageNet images.

Overall, our attack DFA achieves close to 100 percent attack success rates for both targeted and untargeted attacks. In terms of SR, our tool DFA outperforms FGSM/FGSM-1 and is comparable to the other tools. In terms of TSR, DFA is comparable to BIM and outperforms FGSM, FGSM-1 and C&W in most cases.

Specifically, FGSM, FGSM-1, BIM and C&W+GS do not have any gap due to the tuning of step sizes and the greedy search based algorithm. It is easy to observe that C&W has a relatively larger gap in targeted attacks on Inception-v3 in $\mathbf{L}_\infty$ norm setting, as its TSR is only 24 percent compared with 100 percent SR. Thus, although C&W outperforms DFA in terms of SR, DFA outperforms C&W in most cases in terms of TSR. We remark that the gap of C&W is slightly different from the one given in Table 3, as C&W-$\mathbf{L}_2$ has an input parameter κ which can control the confidence. By increasing κ, the confidence of real adversarial examples as well as the TSR of C&W-$\mathbf{L}_2$ increase, and the gap can be minimized. Whereas C&W-$\mathbf{L}_\infty$ does not have this parameter.

6.3 Comparison With Black-Box Methods

We compare DFA with well-known recent black-box methods: substitute model based attacks, ZOO, NES-PGD, FD, FD-PSO, and also three concurrent works Bandits, AutoZOOM and GenAttack, representing all the classes of existing black-box attacks (cf. Section 2), where the implementations are by their authors.

Recall that it is very difficult to tune input parameters for those tools without loss of attack efficiency or quality of adversarial examples, hence we use the parameters as in their raw papers which are already fine-tuned by the authors. When evaluating substitute model, we use FGSM/FGSM-1 and C&W methods, and use ResNet50 [121] as the substitute model for Inception-v3, the model in ZOO as the substitute model for LeNet-1. Since ZOO and AutoZOOM use $\mathbf{L}_2$ distance, we map our maximum $\mathbb{L}_\infty$ distances into maximum $\mathbf{L}_2$ distances by considering the worst case of $\mathbb{L}_\infty$, namely, all the pixels are modified by the maximum $\mathbb{L}_\infty$ distance. For instance, the $\mathbb{L}_\infty$ distance 10 is approximated by $\mathbf{L}_2$ distance $\sqrt{(10/255)^2 \times (299\times 299\times 3)} \approx 20$ for $299\times 299\times 3$ images. Remark that this is not a rigorous mapping, ZOO and AutoZOOM under $\mathbf{L}_2$ would be easier to find an adversarial example, as the corresponding $\mathbf{L}_2$ distances are less restricted.

The results of untargeted and targeted attacks are given in Tables 7 and 8. We can see that our attack DFA achieves close to 100 percent attack success rates for both targeted and untargeted attacks and outperforms all the other tools in terms of TSR no matter targeted or untargeted attacks. In terms of SR, our tool is also comparable (or better) to the other tools. One may notice that substitute models perform poorly. This may be due to the difference between training data and architectures of the substitute model and the target model, as the larger gap between the substitute model and the target model is, the less effective of transferability of adversarial samples is.

6.4 Query Comparison

In many black-box scenarios, the attacker has a limited number of queries to the classifier. Therefore, we report the average number of queries of the black-box attacks in Table 9, where substitute model based attack is excluded due to its low SR. We remark that ZOO is regarded as baseline, the others are state-of-the-art query-efficient tools.

On Attack Against Inception-v3, our tool DFA outperforms all the other tools for targeted attacks, except for GenAttack, which is slightly better than DFA. Recall that GenAttack downgrades to gray-box one, as they directly invoke the

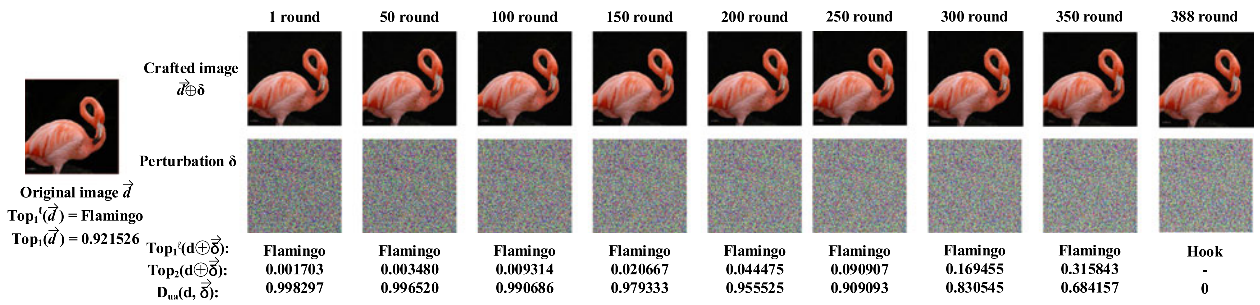

1 round
50 round
100 round
150 round
200 round
250 round
300 round
350 round
388 round
Crafted image $\vec{d}\oplus\delta$
Perturbation δ
Original image $\vec{d}$
$Top_1^t(\vec{d})$ = Flamingo
$Top_1(\vec{d})$ = 0.921526
$Top_1^t(d\oplus\vec{\delta})$: Flamingo Flamingo Flamingo Flamingo Flamingo Flamingo Flamingo Flamingo Hook
$Top_2(d\oplus\vec{\delta})$: 0.001703 0.003480 0.009314 0.020667 0.044475 0.090907 0.169455 0.315843 -
$D_{ua}(d,\vec{\delta})$: 0.998297 0.996520 0.990686 0.979333 0.955525 0.909093 0.830545 0.684157 0

Fig. 4. Illustrative example of an untargeted attack on a *Flamingo* image.

TABLE 4
Experiment Settings

Parameter	Setting
Max. $\mathbb{L}_\infty$ distance ϵ	$\epsilon = 64$ for MNIST and $\epsilon = 10$ for ImageNet.
Target class	For MNIST images, the class with 4th largest probability is chosen as the target class. For ImageNet images, the class with 11th largest probability is chosen as the target class.
Sample size s	$s = 3$ in all the experiments.
Ranking Threshold k	$k = 2$ in all the experiments.
Coordinate Threshold u	$u = 2$ pixels for MNIST images. $u = 10$ pixels for ImageNet images.
Iteration Threshold T	$T = 30000$ in all the experiments.
Timeout Threshold	3 minutes for MNIST images. 30 minutes for ImageNet images.
Resized Space Δ_r	No resize for MNIST images. $100 \times 100 \times 3$ for ImageNet images.

normalization of the integer classification systems, while our attack is black-box one. For untargeted attacks, our tool DFA outperforms the baseline tool ZOO and comparable to other tools. Recall that our tool DFA outperforms all these tools in terms of TSR.

We remark that ZOO and AutoZOOM are tested under the $\mathbf{L}_2$ distance 20, which is less restricted than the $\mathbb{L}_\infty$ distance 10 used for the other tools. Indeed, in untargeted attack setting, the average $\mathbf{L}_2$ distance of our tool is 8.33. Whereas the average query times of AutoZOOM becomes 4971 (worse than ours) if $\mathbf{L}_2 = 12$.

On attack against LeNet-1, our tool DFA outperforms both of them in almost all cases, except that FD uses less query times than DFA for targeted attacks. Note that our tool DFA achieves a higher attack success attack rate than FD and FD-PSO in terms of both SR/TSR. One may notice that the query times of FD and FD-PSO are the same between untargeted and targeted attacks. This is due to the implementations of FD and FD-PSO (confirmed by some authors of [61]).

Furthermore, we also report the average Mean Square Error (MSE) of the adversarial examples in Table 9. We can observe that our tool DFA outperforms most of the other tools on attacks against Inception-v3. FD and FD-PSO are slightly better than DFA on attacks against LeNet-1, at the same order of magnitude. ZOO outperforms all the other tools in terms of MSE against Inception-v3, but at the cost of a huge number of queries.

TABLE 5
Results of White-Box Untargeted Attacks

Dataset & DNN	Method	SR	TSR	GAP
MNIST LeNet-1	FGSM	97%	97%	0%
	BIM	100%	**100%**	0%
	C&W	100%	88%	12%
	C&W+GS	100%	**100%**	0%
	DFA	100%	**100%**	0%
ImageNet Inception-v3	FGSM	79%	79%	0%
	BIM	100%	**100%**	0%
	C&W	100%	68%	32%
	DFA	99%	**99%**	0%

TABLE 6
Results of White-Box Targeted Attacks

Dataset & DNN	Method	SR	TSR	GAP
MNIST LeNet-1	FGSM-1	84%	84%	0%
	BIM	100%	**100%**	0%
	C&W	100%	75%	25%
	C&W+GS	100%	**100%**	0%
	DFA	100%	**100%**	0%
ImageNet Inception-v3	FGSM-1	9%	9%	0%
	BIM	99%	**99%**	0%
	C&W	100%	24%	76%
	DFA	96%	**96%**	0%

TABLE 7
Results of Black-Box Untargeted Attacks

Dataset & DNN	Method	SR	TSR	GAP
MNIST LeNet-1	SModel+C&W	2.5%	2.5%	0%
	SModel+FGSM	20%	20%	0%
	FD	94.5%	94.5%	0%
	FD-PSO	46.5%	46.5%	0%
	DFA	100%	**100%**	0%
ImageNet Inception-v3	SModel+C&W	6%	6%	0%
	SModel+FGSM	38%	38%	0%
	ZOO	89%	5%	94.3%
	AutoZOOM	100%	57%	43%
	NES-PGD	100%	77%	23%
	Bandits	100%	12%	88%
	GenAttack	100%	93%	7%
	DFA	99%	**99%**	0%

Note: FD and FD-PSO do not provide attacks against ImageNet images. ZOO and AutoZOOM do not provide attacks under $\mathbb{L}_\infty$ distance, so we only compare our tool with them on ImageNet images. NES-PGS and Bandits do not provide attacks against MNIST images. Meanwhile, the version of GenAttack's attack against MNIST is buggy.

TABLE 8
Results of Black-Box Targeted Attacks

Dataset & DNN	Method	SR	TSR	GAP
MNIST LeNet-1	SModel+C&W	1.5%	1.5%	0%
	SModel+FGSM-1	5%	5%	0%
	FD	72%	72%	0%
	FD-PSO	6.5%	6.5%	0%
	DFA	100%	**100%**	0%
ImageNet Inception-v3	SModel+C&W	1%	1%	0%
	SModel+FGSM-1	2%	2%	0%
	ZOO	69%	5%	92.7%
	AutoZOOM	95%	43%	54.7%
	NES-PGD	100%	47%	53%
	GenAttack	100%	84%	16%
	DFA	96%	**96%**	0%

Note: Bandits does not support targeted attack.

TABLE 9
Comparison With Average Query Times and Corresponding MSE

Dataset & DNN	Method	Untargeted		Targeted	
		Query	MSE	Query	MSE
MNIST LeNet-1	FD	1568	3.4e-2	**1568**	3.5e-2
	FD-PSO	10000	**2.5e-2**	10000	**2.5e-2**
	DFA	**817**	**3.7e-2**	**1593**	**5.1e-2**
ImageNet Inception-v3	ZOO	85368	**1.7e-5**	203683	**3.6e-5**
	AutoZOOM	**2224**	9.2e-4	14322	1.2e-3
	NES-PGD	4741	8.4e-4	13421	9.0e-4
	Bandits	4595	1.4e-3	-	-
	GenAttack	4008	6.3e-4	**12369**	9.2e-4
	DFA	**4746**	**2.6e-4**	**12740**	**3.4e-4**

Note: the queries of our tool is computed on integer adversarial examples, while it is computed on real adversarial examples for the others.

6.5 Attack Classifiers With Defense

To show the effectiveness of our approach, we use our tool to attack the HGD defense [62], which won the first place on defense against adversarial attacks in NIPS 2017 competition. HGD defense is a typical denoising based defense method for image classification. The whole classification system is an ensemble of 4 independent models and their denoiser (ResNet, ResNext, InceptionV3, inceptionResNetV2). We conduct untargeted attacks against this model using the same 100 ImageNet images and parameters as previously, except that the $\mathbb{L}_\infty$ distance ϵ is 32 according to the NIPS 2017 competition. Our tool achieves 100 percent TSR in the experiments, indicating the effectiveness of DFA. This benefits from the advantage of our classification model-based derivative-free optimization method, which does not rely on the gradient of the objective function, but instead, learns from samples of the search space, hence suitable for attack systems that are non-differentiable or even unknown but only testable.

MNIST Adversarial Examples Challenge [63] is another widely recognized attack problem. It uses adversarial training as a defense method. We use the same 200 MNIST images as previously on the attack of this problem. Our tool DFA achieves 10.5 percent TSR, the same as the current best white-box attack “interval attacks”, which is publicly reported on the webpage of the challenge. The images on which the attacks succeed by both methods are exactly same, and the time costs of both tools are also similar.

7 Conclusion and Future Work

We conducted the first comprehensive study of 35 methods and 20 open source tools for crafting adversarial examples, in an attempt to understand the impacts of the discretization problem. Our study revealed that most of these methods and tools are affected by this problem and researchers should pay more attention when designing adversarial example attacks and measuring attack success rate. We also proposed strategies to avoid or alleviate the discretization problem, which can improve TSR of some tools, at the cost of attack efficiency or imperceptibility of adversarial examples.

We proposed a black-box method by designing a classification model-based derivative-free optimization method. Our method directly crafts adversarial examples in discrete integer domains, hence it does not have the discretization problem and is able to attack a wide range of classifiers including non-differentiable ones. Our attack method 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 implemented our method into tool DFA, and conducted an intensive set of experiments on MNIST and ImageNet in both untargeted and targeted scenarios. The experimental results show that our method achieved close to 100 percent attack success rate, comparable to the white-box methods (FGSM, BIM and C&W) and outperformed the state-of-the-art black-box methods. Moreover, our method achieved 100 percent success rate on the winner of NIPS 2017 competition on defense, and achieved the same result as the best white-box attack in MNIST Challenge. Our results suggest that classification model-based derivative-free discrete optimization opens up a promising research direction into effective black-box attacks. Our method could serve as a test for designing robust networks.

In the future, we plan to lift our generic method to other neural network based systems such as face recognition systems [23] and speech recognition [88], [89]. It is also worth investigating how to intergrade gradient estimation techniques into our sampling. This may improve query efficiency of our method.

Acknowledgments

The authors would like to thank anonymous reviewers for their valuable comments and suggestions. The work of Lei Bu and Yuchao Duan was supported in part by the Leading-edge Technology Program of Jiangsu Natural Science Fundation under Grant BK20202001, and in part by the National Natural Science Foundation of China (NSFC) under Grant 62032010. The work of Zhe Zhao and Fu Song was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 62072309. Lei Bu and Zhe Zhao are co-first authors ordered alphabetically.

References

[1] Apollo, “An open, reliable and secure software platform for autonomous driving systems,” 2018. [Online]. Available: http://apollo.auto
[2] Waymo, “A self-driving technology development company,” 2009. [Online]. Available: https://waymo.com/

[3] D. C. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, "Deep neural networks segment neuronal membranes in electron microscopy images," in *Proc. 26th Annu. Conf. Neural Inf. Process. Syst.*, 2012, pp. 2843–2851.
[4] D. Shen, G. Wu, and H.-I. Suk, "Deep learning in medical image analysis," *Annu. Rev. Biomed. Eng.*, vol. 19, pp. 221–248, 2017.
[5] T. Parag, D. C. Ciresan, and A. Giusti, "Efficient classifier training to minimize false merges in electron microscopy segmentation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 657–665.
[6] G. Hinton *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, Nov. 2012.
[7] G. Chen *et al.*, "Who is real Bob? Adversarial attacks on speaker recognition systems," in *Proc. 42nd IEEE Symp. Secur. Privacy*, 2021.
[8] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 1725–1732.
[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, 2017.
[10] F. Zhang, J. Leitner, M. Milford, B. Upcroft, and P. I. Corke, "Towards vision-based deep reinforcement learning for robotic motion control," 2015, *arXiv:1511.03791*.
[11] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *Int. J. Robot. Res.*, vol. 37, no. 4–5, pp. 421–436, 2018.
[12] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global vectors for word representation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2014, pp. 1532–1543.
[13] D. Andor *et al.*, "Globally normalized transition-based neural networks," in *Proc. 54th Annu. Meet. Assoc. Comput. Linguistics*, 2016, pp. 2442–2452.
[14] W. Song, H. Yin, C. Liu, and D. Song, "DeepMem: Learning graph neural network models for fast and robust memory forensic analysis," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 606–618.
[15] L. De La Rosa, S. Kilgallon, T. Vanderbruggen, and J. Cavazos, "Efficient characterization and classification of malware using deep learning," in *Proc. Resilience Week*, 2018, pp. 77–83.
[16] Y. Lei, S. Chen, L. Fan, F. Song, and Y. Liu, "Advanced evasion attacks and mitigations on practical ML-based phishing website classifiers," 2020, *arXiv:2004.06954*.
[17] N. N. Dalvi, P. M. Domingos, Mausam, S. K. Sanghai, and D. Verma, "Adversarial classification," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2004, pp. 99–108.
[18] C. Szegedy *et al.*, "Intriguing properties of neural networks," in *Proc. Int. Conf. Learn. Representations*, 2014.
[19] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. Int. Conf. Learn. Representations*, 2014.
[20] N. Carlini and D. A. Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," in *Proc. 10th ACM Workshop Artif. Intell. Secur.*, 2017, pp. 3–14.
[21] N. Carlini and D. A. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. IEEE Symp. Secur. Privacy*, 2017, pp. 39–57.
[22] N. Papernot, P. D. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Proc. IEEE Eur. Symp. Secur. Privacy*, 2016, pp. 372–387.
[23] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, "Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 1528–1540.
[24] S.-M. Moosavi-Dezfooli , A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 86–94.
[25] K. Pei, Y. Cao, J. Yang, and S. Jana, "DeepXplore: Automated whitebox testing of deep learning systems," in *Proc. 26th Symp. Operating Syst. Princ.*, 2017, pp. 1–18.
[26] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *Proc. Int. Conf. Learn. Representations*, 2017.
[27] W. Brendel, J. Rauber, and M. Bethge, "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models," in *Proc. Int. Conf. Learn. Representations*, 2018.
[28] C. Xiao, J.-Y. Zhu, B. Li, W. He, M. Liu, and D. Song, "Spatially transformed adversarial examples," in *Proc. Int. Conf. Learn. Representations*, 2018.
[29] Z. Zhao, D. Dua, and S. Singh, "Generating natural adversarial examples," in *Proc. Int. Conf. Learn. Representations*, 2018.
[30] J. Kos, I. Fischer, and D. Song, "Adversarial examples for generative models," in *Proc.IEEE Secur. Privacy Workshops*, 2018, pp. 36–42.
[31] K. Eykholt *et al.*, "Robust physical-world attacks on deep learning visual classification," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 1625–1634.
[32] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok, "Synthesizing robust adversarial examples," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 284–293.
[33] P. Chen, H. Zhang, Y. Sharma, J. Yi, and C. Hsieh, "ZOO: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *Proc. 10th ACM Workshop Artif. Intell. Secur.*, 2017, pp. 15–26.
[34] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin, "Query-efficient black-box adversarial examples," 2017, *arXiv:1712.07113*.
[35] N. Papernot, P. D. McDaniel, I. J. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proc.ACM Asia Conf. Comput. Commun. Secur.*, 2017, pp. 506–519.
[36] A. N. Bhagoji, W. He, B. Li, and D. Song, "Exploring the space of black-box attacks on deep neural networks," 2017, *arXiv:1712.09491*.
[37] C. Tu *et al.*, "Autozoom: Autoencoder-based zeroth order optimization method for attacking black-box neural networks," in *Proc. 33rd AAAI Conf. Artif. Intell.*, 2019, pp. 742–749.
[38] M. Wicker, X. Huang, and M. Kwiatkowska, "Feature-guided black-box safety testing of deep neural networks," in *Proc. 24th Int. Conf. Tools Algorithms Construction Anal. Syst.*, 2018, pp. 408–426.
[39] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin, "Black-box adversarial attacks with limited queries and information," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 2142–2151.
[40] L. Ma *et al.*, "DeepGauge: Multi-granularity testing criteria for deep learning systems," in *Proc. 33rd ACM/IEEE Int. Conf. Automated Softw. Eng.*, 2018, pp. 120–131.
[41] Z. Zhao, G. Chen, J. Wang, Y. Yang, F. Song, and J. Sun, "Attack as defense: Characterizing adversarial examples using robustness," 2021, *arXiv:2103.07633*.
[42] G. Katz, C. W. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An efficient SMT solver for verifying deep neural networks," in *Proc. 29th Int. Conf. Comput. Aided Verification*, 2017, pp. 97–117.
[43] L. Pulina and A. Tacchella, "An abstraction-refinement approach to verification of artificial neural networks," in *Proc. 22nd Int. Conf. Comput. Aided Verification*, 2010, pp. 243–257.
[44] T. Gehr, M. Mirman, D. Drachsler-Cohen , P. Tsankov, S. Chaudhuri, and M. T. Vechev, "AI2: Safety and robustness certification of neural networks with abstract interpretation," in *Proc. IEEE Symp. Secur. Privacy*, 2018, pp. 3–18.
[45] D. Gopinath, G. Katz, C. S. Pasareanu, and C. Barrett, "DeepSafe: A data-driven approach for assessing robustness of neural networks," in *Proc. 16th Int. Symp. Automated Technol. Verification Anal.*, 2018, pp. 3–19.
[46] G. Singh, T. Gehr, M. Mirman, M. Püschel, and M. T. Vechev, "Fast and effective robustness certification," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 10825–10836.
[47] G. Singh, T. Gehr, M. Püschel, and M. Vechev, "An abstract domain for certifying neural networks," *Proc. ACM Program. Lang.*, vol. 3, 2019, Art. no. 41.
[48] W. Wan, Z. Zhang, Y. Zhu, M. Zhang, and F. Song, "Accelerating robustness verification of deep neural networks guided by target labels," 2020, *arXiv:2007.08520*.
[49] W.-W. Liu, F. Song, T.-H.-R. Zhang, and J. Wang, "Verifying ReLU neural networks from a model checking perspective," *J. Comput. Sci. Technol.*, vol. 35, no. 6, pp. 1365–1381, 2020.
[50] Y. Zhang, Z. Zhao, G. Chen, F. Song, and T. Chen, "BDD4BNN: A BDD-based quantitative analysis framework for binarized neural networks," 2021, *arXiv:2103.07224*.
[51] I. J. Goodfellow, Y. Bengio, and A. C. Courville, *Deep Learning* (Adaptive Computation and Machine Learning). Cambridge, MA, USA: MIT Press, 2016.
[52] Adversarial example classified correctly, 2019. [Online]. Available: https://github.com/peikexin9/deepxplore/issues/20
[53] Different Prediction for adversarial images when saved to disk, 2019. [Online]. Available: https://github.com/bethgelab/foolbox/issues/264

[54] Kaggle user reports a bug in FGSM when using uint8 images, 2019. [Online]. Available: https://github.com/tensorflow/cleverhans/issues/265

[55] Y. LeCun, C. Cortes, and C. J. Burges, "The MNIST database of handwritten digits," 1998. [Online]. Available: http://yann.lecun.com/exdb/mnist

[56] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[57] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei , "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 248–255.

[58] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2818–2826.

[59] A. Ilyas, L. Engstrom, and A. Madry, "Prior convictions: Black-box adversarial attacks with bandits and priors," in *Proc. 7th Int. Conf. Learn. Representations*, 2019.

[60] M. Alzantot, Y. Sharma, S. Chakraborty, H. Zhang, C. Hsieh, and M. B. Srivastava,"GenAttack: Practical black-box attacks with gradient-free optimization," in *Proc. Genet. Evol. Comput. Conf.*, 2019, pp. 1111–1119.

[61] A. N. Bhagoji, W. He, B. Li, and D. Song, "Practical black-box attacks on deep neural networks using efficient query mechanisms," in *Proc. 15th Eur. Conf. Comput. Vis.*, 2018, pp. 158–174.

[62] F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu, and J. Zhu, "Defense against adversarial attacks using high-level representation guided denoiser," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 1778–1787.

[63] M. Lab, "MNIST adversarial examples challenge," Accessed: May 2019. [Online]. Available: https://github.com/MadryLab/mnist_challenge

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

[65] N. Narodytska and S. P. Kasiviswanathan, "Simple black-box adversarial attacks on deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2017, pp. 1310–1318.

[66] T. Salimans, J. Ho, X. Chen, and I. Sutskever, "Evolution strategies as a scalable alternative to reinforcement learning," 2017, *arXiv:1703.03864*.

[67] D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, J. Peters, and J. Schmidhuber, "Natural evolution strategies," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 949–980, 2014.

[68] P. Zhao *et al.*, "On the design of black-box adversarial examples by leveraging gradient-free optimization and operator splitting method," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 121–130.

[69] H. Hosseini, B. Xiao, and R. Poovendran, "Google's cloud vision API is not robust to noise," in *Proc. 16th IEEE Int. Conf. Mach. Learn. Appl.*, 2017, pp. 101–105.

[70] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," in *Proc. 5th Int. Conf. Learn. Representations*, 2017.

[71] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Trans. Evol. Comput.*, vol. 23, no. 5, pp. 828–841, Oct. 2019.

[72] W. Xu, Y. Qi, and D. Evans, "Automatically evading classifiers: A case study on PDF malware classifiers," in *Proc. Netw. Distrib. Syst. Symp.*, 2016, pp. 21–24.

[73] K. T. Co, L. Mu ñoz-González, S. de Maupeou, and E. C. Lupu, "Procedural noise adversarial examples for black-box attacks on deep convolutional networks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2019, pp. 275–289.

[74] J. Lu, H. Sibai, and E. Fabry, "Adversarial examples that fool detectors," 2017, *arXiv:1712.02494*.

[75] S. T. K. Jan, J. Messou, Y. Lin, J. Huang, and G. Wang, "Connecting the digital and physical world: Improving the robustness of adversarial attacks," in *Proc. 33rd AAAI Conf. Artif. Intell.*, 2019, pp. 962–969.

[76] K. Eykholt *et al.*, "Note on attacking object detectors with adversarial stickers," 2017, *arXiv:1712.08062*.

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

[78] J. Lu, H. Sibai, E. Fabry, and D. A. Forsyth, "NO need to worry about adversarial examples in object detection in autonomous vehicles," 2017, *arXiv:1707.03501*.

[79] C. Sitawarin, A. N. Bhagoji, A. Mosenia, M. Chiang, and P. Mittal, "DARTS: Deceiving autonomous cars with toxic signs," 2018, *arXiv:1802.06430*.

[80] S.-T. Chen, C. Cornelius, J. Martin, and D. H. Chau, "ShapeShifter: Robust physical adversarial attack on faster R-CNN object detector," 2018, *arXiv:1804.05810*.

[81] K. Eykholt *et al.*, "Physical adversarial examples for object detectors," in *Proc. 12th USENIX Workshop Offensive Technol.*, 2018.

[82] D. Maiorca, I. Corona, and G. Giacinto, "Looking at the bag is not enough to find the bomb: An evasion of structural methods for malicious PDF files detection," in *Proc. 8th ACM SIGSAC Symp. Inf., Comput. Commun. Secur.*, 2013, pp. 119–130.

[83] N. Srndic and P. Laskov, "Practical evasion of a learning-based classifier: A case study," in *Proc. IEEE Symp. Secur. Privacy*, 2014, pp. 197–211.

[84] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel , "Adversarial examples for malware detection," in *Proc. Eur. Symp. Res. Comput. Secur.*, 2017, pp. 62–79.

[85] A. Demontis *et al.*, "Yes, machine learning can be more secure! A case study on android malware detection," *IEEE Trans. Dependable Secure Comput.*, vol. 16, no. 4, pp. 711–724, Jul./Aug. 2019.

[86] L. Xu, Z. Zhan, S. Xu, and K. Ye, "An evasion and counter-evasion study in malicious websites detection," in *Proc. IEEE Conf. Commun. Netw. Secur.*, 2014, pp. 265–273.

[87] D. Lowd and C. Meek, "Good word attacks on statistical spam filters," in *Proc. 2nd Conf. Email Anti-Spam*, 2005.

[88] X. Yuan *et al.*, "Commandersong: A systematic approach for practical adversarial voice recognition," in *Proc. 27th USENIX Secur. Symp., USENIX Secur.*, 2018, pp. 49–64.

[89] N. Carlini *et al.*, "Hidden voice commands," in *Proc. 25th USENIX Secur. Symp.*, 2016, pp. 513–530.

[90] F. Chollet *et al.*, "Keras," 2015. [Online]. Available: https://keras.io

[91] Y. Duan, Z. Zhao, L. Bu, and F. Song, "Things you may not know about adversarial example: A black-box adversarial image attack," 2019, *arXiv:1905.07672*.

[92] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *Proc. Int. Conf. Learn. Representations*, 2018.

[93] Y. Dong *et al.*, "Boosting adversarial attacks with momentum," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 9185–9193.

[94] W. He, B. Li, and D. Song, "Decision boundary analysis of adversarial examples," in *Proc. Int. Conf. Learn. Representations*, 2018.

[95] P. Chen, Y. Sharma, H. Zhang, J. Yi, and C. Hsieh, "EAD: Elastic-net attacks to deep neural networks via adversarial examples," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 10–17.

[96] S.-M. Moosavi-Dezfooli , A. Fawzi, and P. Frossard, "DeepFool: A simple and accurate method to fool deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2574–2582.

[97] Y. Sun, X. Huang, and D. Kroening, "Testing deep neural networks," 2018, *arXiv:1803.04792*.

[98] Y. Sun, M. Wu, W. Ruan, X. Huang, M. Kwiatkowska, and D. Kroening, "Concolic testing for deep neural networks," in *Proc. 33rd ACM/IEEE Int. Conf. Automated Softw. Eng.*, 2018, pp. 109–119.

[99] O. Bastani, Y. Ioannou, L. Lampropoulos, D. Vytiniotis, A. V. Nori, and A. Criminisi, "Measuring neural net robustness with constraints," in *Proc. Adv. neural. Inf. Process. Syst.*, 2016, pp. 2613–2621.

[100] X. Huang, M. Kwiatkowska, S. Wang, and M. Wu, "Safety verification of deep neural networks," in *Proc. 29th Int. Conf. Comput. Aided Verification*, 2017, pp. 3–29.

[101] R. Ehlers, "Formal verification of piece-wise linear feed-forward neural networks," in *Proc. 15th Int. Symp. Automated Technol. Verification Anal.*, 2017, pp. 269–286.

[102] V. Tjeng, K. Xiao, and R. Tedrake, "Evaluating robustness of neural networks with mixed integer programming," in *Proc. 7th Int. Conf. Learn. Representations*, 2019.

[103] W. Ruan, X. Huang, and M. Kwiatkowska, "Reachability analysis of deep neural networks with provable guarantees," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, 2018, pp. 2651–2659.

[104] S. Wang, K. Pei, J. Whitehouse, J. Yang, and S. Jana, “Formal security analysis of neural networks using symbolic intervals,” in *Proc. 27th USENIX Secur. Symp. Secur.*, 2018, pp. 1599–1614.

[105] K. Dvijotham, R. Stanforth, S. Gowal, T. A. Mann, and P. Kohli, “A dual approach to scalable verification of deep networks,” 2018, *arXiv:1803.06567*.

[106] P. Tabacof and E. Valle, “Exploring the space of adversarial images,” in *Proc. Int. Joint Conf. Neural Netw.*, 2016, pp. 426–433.

[107] N. Carlini, “A critique of the DEEPSEC platform for security analysis of deep learning models,” 2019, *arXiv:1905.07112*.

[108] J. Rauber, W. Brendel, and M. Bethge, “Foolbox: A python toolbox to benchmark the robustness of machine learning models,” 2017, *arXiv:1707.04131*.

[109] W. Brendel, J. Rauber, M. Kümmerer, I. Ustyuzhaninov, and M. Bethge, “Accurate, reliable and fast robustness evaluation,” 2019, *arXiv:1907.01003*.

[110] Y. Shi, S. Wang, and Y. Han, “Curls & Whey: Boosting black-box adversarial attacks,” in *Proc. Comput. Vis. Pattern Recognit.*, 2019, pp. 6519–6527.

[111] Y. Yu, H. Qian, and Y.-Q. Hu, “Derivative-free optimization via classification,” in *Proc. 13th AAAI Conf. Artif. Intell.*, 2016, pp. 2286–2292.

[112] M. Jordan, N. Manoj, S. Goel, and A. G. Dimakis, “Quantifying perceptual distortion of adversarial examples,” 2019, *arXiv:1902.08265*.

[113] A. Kurakin, I. J. Goodfellow, and S. Bengio, “Adversarial machine learning at scale,” in *Proc. 5th Int. Conf. Learn. Representations*, 2017.

[114] N. Carlini and D. Wagner, “Audio adversarial examples: Targeted attacks on speech-to-text,” in *Proc. IEEE Secur. Privacy Workshops*, 2018, pp. 1–7.

[115] M. Sharif, L. Bauer, and M. K. Reiter, “On the suitability of lp-norms for creating and preventing adversarial examples,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops*, 2018, pp. 1605–1613.

[116] Z. Wang, M. Song, S. Zheng, Z. Zhang, Y. Song, and Q. Wang, “Invisible adversarial attack against deep neural networks: An adaptive penalization approach,” *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 3, pp. 1474–1488, May/Jun. 2021.

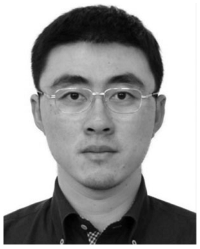
[117] B. Luo, Y. Liu, L. Wei, and Q. Xu, “Towards imperceptible and robust adversarial example attacks against neural networks,” in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 1652–1659.

[118] J. Sekhon and C. Fleming, “Towards improved testing for deep learning,” in *Proc. 41st Int. Conf. Softw. Eng.: New Ideas Emerg. Results*, 2019, pp. 85–88.

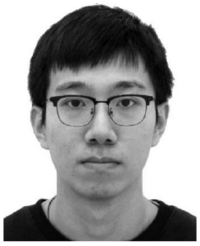
[119] J. Guo, Y. Jiang, Y. Zhao, Q. Chen, and J. Sun, “DLFuzz: Differential fuzzing testing of deep learning systems,” in *Proc. ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, 2018, pp. 739–743.

[120] X. Xie *et al.*, “DeepHunter: A coverage-guided fuzz testing framework for deep neural networks,” in *Proc. 28th ACM SIGSOFT Int. Symp. Softw. Testing Anal.*, 2019, pp. 146–157.

[121] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.



Lei Bu (Member, IEEE) received the BS and PhD degrees in computer science from Nanjing University, in 2004 and 2010, respectively. He is currently a professor with the State Key Laboratory for Novel Software Technology and Department of Computer Science and Technology, Nanjing University. He has authored or coauthored more than 50 research papers in major peer-reviewed international journals and conference proceedings. His research interests include formal method, model checking, with a focus on verification of hybrid system and cyber-physical system. He is a member of the IEEE and the ACM.



Zhe Zhao received the BS degree from the Ocean University of China, Tsingtao, China, in 2016. He is currently working toward the PhD degree with the School of Information Science and Technology, ShanghaiTech University. From 2016 to 2018, he was a software engineer with Hewlett-Packard Company. His research interests include software engineering and testing. His research focuses on artificial intelligence. He is supervised by Dr. Song.

Yuchao Duan received the BS and master’s degrees from Nanjing University, Nanjing, China, in 2017 and 2020, respectively. His research interest focuses on software engineering. He is supervised by Dr. Bu.

Fu Song received the BS degree from Ningbo University, Ningbo, China, in 2006, the MS degree from East China Normal University, Shanghai, China, in 2009, and the PhD degree in computer science from University Paris-Diderot, Paris, France, in 2013. From 2013 to 2016, he was a lecturer and an associate research professor with East China Normal University. Since August 2016, he has been an assistant professor with ShanghaiTech University, Shanghai, China. His research interests include formal methods and computer or AI security, with a focus on automata, logic, model checking, and program analysis. He was the recipient of the EASST Best Paper Award at ETAPS 2012.