Adversarial Examples: Attacks and Defenses for Deep Learning

Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li

***Abstract*— With rapid progress and significant successes in a wide spectrum of applications, deep learning is being applied in many safety-critical environments. However, deep neural networks (DNNs) have been recently found vulnerable to well-designed input samples called *adversarial examples*. Adversarial perturbations are imperceptible to human but can easily fool DNNs in the testing/deploying stage. The vulnerability to adversarial examples becomes one of the major risks for applying DNNs in safety-critical environments. Therefore, attacks and defenses on adversarial examples draw great attention. In this paper, we review recent findings on adversarial examples for DNNs, summarize the methods for generating adversarial examples, and propose a taxonomy of these methods. Under the taxonomy, applications for adversarial examples are investigated. We further elaborate on countermeasures for adversarial examples. In addition, three major challenges in adversarial examples and the potential solutions are discussed.**

***Index Terms*— Adversarial examples, deep learning (DL), deep neural network (DNN), security.**

Nomenclature

x	Original (clean, unmodified) input data.
l	Label of class in the classification problem, $l = 1, 2, \ldots, m$, where m is the number of classes.
x'	Adversarial example (modified input data).
l'	Label of the adversarial class in targeted adversarial examples.
$f(\cdot)$	Deep learning model (for the image classification task, $f \in F : \mathbb{R}^n \to l$).
θ	Parameters of deep learning model f.
$J_f(\cdot,\cdot)$	Loss function (e.g., cross entropy) of model f.
η	Difference between original and modified input data: $\eta = x' - x$ (the exact same size as the input data).
$\|\cdot\|_p$	ℓ_p norm.
∇	Gradient.
$H(\cdot)$	Hessian, the second-order of derivatives.
$KL(\cdot)$	Kullback–Leibler divergence function.

Manuscript received January 5, 2018; revised July 5, 2018 and October 5, 2018; accepted November 28, 2018. This work was supported by the National Science Foundation under Grant CNS-1842407, Grant CNS-1747783, Grant CNS-1624782, and Grant OAC-1229576. *(Corresponding author: Xiaolin Li).*

The authors are with the National Science Foundation Center for Big Learning, University of Florida, Gainesville, FL 32611, USA (e-mail: chbrian@ufl.edu; pan.he@ufl.edu; valder@ufl.edu, andyli@ece.ufl.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2018.2886017

I. Introduction

Deep learning (DL) has made a significant progress in a wide domain of machine learning (ML): image classification, object recognition [1], [2], object detection [3], [4], speech recognition [5], language translation [6], and voice synthesis [7]. Many DL empowered applications are life crucial, raising great concerns in the field of safety and security. Despite great successes in numerous applications, recent studies find that DL is vulnerable against well-designed input samples. These samples can easily fool a well-performed DL model with little perturbations imperceptible to humans.

Szegedy *et al.* [8] first generated small perturbations on the images for the image classification problem and fooled state-of-the-art deep neural networks (DNNs) with high probability. These misclassified samples were named *adversarial examples*.

Extensive DL-based applications have been used or planned to be deployed in the physical world, especially in the safety-critical environments. In the meanwhile, recent studies show that adversarial examples can be applied to the real world. For instance, an adversary can construct physical adversarial examples and confuse autonomous vehicles by manipulating the stop sign in a traffic sign recognition system [9], [10] or removing the segmentation of pedestrians in an object recognition system [11]. Attackers can generate adversarial commands against automatic speech recognition models and voice controllable system [12], [13], such as Apple Siri [14], Amazon Alexa [15], and Microsoft Cortana [16].

DL is widely regarded as a "black box" technique—we all know that it performs well but with limited knowledge of the reason [17], [18]. Many studies have been proposed to explain and interpret DNNs [19]–[22]. From inspecting adversarial examples, we may gain insights on semantic inner levels of neural networks [23] and find problematic decision boundaries, which in turn helps to increase robustness and performance of neural networks [24] and improve the interpretability [25].

In this paper, we investigate and summarize the approaches for generating adversarial examples, applications for adversarial examples, and corresponding countermeasures. We explore the characteristics and possible causes of adversarial examples. Recent advances in DL revolve around supervised learning, especially in the field of computer vision task. Therefore, most adversarial examples are generated against computer vision models. We mainly discuss adversarial examples for image classification/object recognition tasks in this paper. Adversarial examples for other tasks will be investigated in Section V.

Inspired by [26], we define the *Threat Model* as follows.

1) The adversaries can attack only at the testing/deploying stage. They can tamper only the input data in the testing stage after the victim DL model is trained. Neither the trained model nor the training data set can be modified. The adversaries may have knowledge of trained models (architectures and parameters) but are not allowed to modify models, which is a common assumption for many online ML services. Attacking at the training stage (e.g., training data poisoning) is another interesting topic and has been studied in [27]–[32]. Due to the limitation of space, we do not include this topic in this paper.
2) We focus on attacks against models built with DNNs, due to their great performance achieved. We will discuss adversarial examples against conventional ML [e.g., support vector machine (SVM) and Random Forest) in Section II. Adversarial examples against DNNs proved effective in conventional ML models [33] (see Section VII-A).
3) Adversaries only aim at compromising *integrity*. Integrity is presented by performance metrics (e.g., accuracy, F1 score, and area under curve), which is essential to a DL model. Although other security issues pertaining to confidentiality and privacy have been drawn attention in DL [34]–[36], we focus on the attacks that degrade the performance of DL models, which causes an increase of false positives and false negatives.
4) The rest of the threat model differs in different adversarial attacks. We will categorize them in Section III.

This paper presents the following contributions.

1) To systematically analyze approaches for generating adversarial examples, we taxonomize attack approaches along different axes to provide an accessible and intuitive overview of these approaches.
2) We investigate recent approaches and their variants for generating adversarial examples and compare them using the proposed taxonomy. We show the examples of selected applications from the fields of reinforcement learning, generative modeling, face recognition, object detection, semantic segmentation, natural language processing (NLP), and malware detection. Countermeasures for adversarial examples are also discussed.
3) We outline the main challenges and potential future research directions for adversarial examples based on three main problems: transferability of adversarial examples, existence of adversarial examples, and robustness evaluation of DNNs.

The remaining of this paper is organized as follows. Section II introduces the background of DL techniques, models, and data sets. We discuss the adversarial examples raised in conventional ML in Section II. We propose a taxonomy of approaches for generating adversarial examples in Section III and elaborate on these approaches in Section IV. In Section V, we discuss the applications for adversarial examples. Corresponding countermeasures are investigated in Section VI. We discuss the current challenges and potential solutions in Section VII. Section VIII concludes this paper.

II. Background

In this section, we briefly introduce basic DL techniques and approaches related to adversarial examples. Next, we review adversarial examples in the era of conventional ML and compare the difference between adversarial examples in conventional ML and adversarial examples in DL.

A. Brief Introduction to Deep Learning

This section discusses the main concepts, existed techniques, popular architectures, and standard data sets in DL, which, due to the extensive use and breakthrough successes, have become acknowledged targets of attacks, where adversaries are usually applied to evaluate their attack methods.

1) Main Concepts in Deep Learning: DL is a type of ML methods that makes computers to learn from experience and knowledge without explicit programing and extract useful patterns from raw data. For conventional ML algorithms, it is difficult to extract well-represented features due to limitations, such as curse of dimensionality [37], computational bottleneck [38], and requirement of the domain and expert knowledge. DL solves the problem of representation by building multiple simple features to represent a sophisticated concept. For example, a DL-based image classification system represents an object by describing edges, fabrics, and structures in the hidden layers. With the increasing number of available training data, DL becomes more powerful. DL models have solved many complicated problems with the help of hardware acceleration in computational time.

A neural network layer is composed of a set of perceptrons (artificial neurons). Each perceptron maps a set of inputs to output values with an activation function. The function of a neural network is formed in a chain

$$f(x) = f^{(k)}(\cdots f^{(2)}(f^{(1)}(x))) \tag{1}$$

where $f^{(i)}$ is the function of the ith layer of the network, $i = 1, 2, \cdots k$.

Convolutional neural networks (CNNs) and recurrent neural networks (RNNs) are two most widely used neural networks in recent neural network architectures. CNNs deploy convolution operations on hidden layers to share weights and reduce the number of parameters. CNNs can extract local information from gridlike input data. CNNs have shown incredible successes in computer vision tasks, such as image classification [1], [39], object detection [4], [40], text recognition [41], [42], and semantic segmentation [43], [44]. RNNs are neural networks for processing sequential input data with variable length. RNNs produce outputs at each time step. The hidden neuron at each time step is calculated based on current input data and hidden neurons at the previous time step. Long short-term memory and gated recurrent unit with controllable gates are designed to avoid vanishing/exploding gradients of RNNs in long-term dependence.

2) Architectures of Deep Neural Networks: Several DL architectures are widely used in computer vision tasks: *LeNet* [45], *VGG* [2], *AlexNet* [1], *GoogLeNet* [46]–[48] (Inception V1-V4), and *ResNet* [39], from the simplest (oldest) network to the deepest and the most complex (newest) one. *AlexNet* first showed that DL models can largely surpass conventional ML algorithms in the ImageNet 2012 challenge and led the future study of DL. These architectures made tremendous breakthroughs in the ImageNet challenge and can be seen as milestones in an image classification problem. Attackers usually generate adversarial examples against these baseline architectures.

3) Standard Deep Learning Data Sets: MNIST, CIFAR-10, and ImageNet are three widely used data sets in computer vision tasks. The MNIST data set is for handwritten digits recognition [49]. The CIFAR-10 data set and the ImageNet data set are for image recognition task [50]. The CIFAR-10 data set consists of 60 000 tiny color images (32 × 32) with 10 classes. The ImageNet data set consists of 14 197 122 images with 1000 classes [51]. Because of the large number of images in the ImageNet data set, most adversarial approaches are evaluated on only part of the ImageNet data set. The Street View House Numbers (SVHN) data set, similar to the MNIST data set, consists of 10 digits obtained from real-world house numbers in Google Street View images. The YoutubeDataset data set is gained from Youtube consisting of about 10 million images [45] and used in [8].

B. Adversarial Examples and Countermeasures in Machine Learning

Adversarial examples in conventional ML models have been discussed since decades ago. ML-based systems with handcrafted features are primary targets, such as spam filters, intrusion detection, biometric authentication, and fraud detection [52]. For example, spam emails are often modified by adding characters to avoid detection [53]–[55].

Dalvi *et al.* [53] first discussed adversarial examples and formulated this problem as a game between adversary and classifier (Naïve Bayes), both of which are sensitive to cost. The attack and defense on adversarial examples became an iterative game. Biggio *et al.* [56] first tried a gradient-based approach to generate adversarial examples against a linear classifier, SVM, and neural network. Compared with DL adversarial examples, their methods allow more freedom to modify the data. The MNIST data set was first evaluated under their attack, although a human could easily distinguish the adversarial digit images. Roli *et al.* [28] also reviewed several proactive defenses and discussed reactive approaches to improve the security of ML models.

Barreno *et al.* [52], [57] presented an initial investigation on the security problems of ML. They categorized attacking against ML system into three axes: 1) influence: whether attacks can poison the training data; 2) security violation: whether an adversarial example belongs to false positive or false negative; and 3) specificity: attack is targeted to a particular instance or a wide class. We discuss these axes for DL area in Section III. Barreno *et al.* [52] compared attacks against SpamBayes spam filter and defenses as a study case. However, they mainly focused on a binary classification problem, such as virus detection system, intrusion detection system, and intrusion prevention system.

Adversarial examples in the conventional ML require knowledge of feature extraction, while DL usually needs only raw data input. In the conventional ML, both attacking and defending methods paid great attention to features, even the previous step (data collection) giving less attention to the impact of humans. Then, the target becomes a fully automatic ML system. Inspired by these studies on conventional ML, in this paper, we review the recent security issues in the DL area.

Papernot *et al.* [26] provided a comprehensive overview of security issues in ML and recent findings in DL and established a unifying threat model. A "no free lunch" theorem was introduced: the tradeoff between accuracy and robustness.

Compared with their work, this paper focuses on adversarial examples in DL and has a detailed discussion on recent studies and findings.

For example, adversarial examples in an image classification task can be described as follows. Using a trained image classifier published by a third party, a user inputs one image to get the prediction of class label. Adversarial images are original clean images with small perturbations, often barely recognizable by humans. However, such perturbations misguide the image classifier. The user will get a response of an incorrect image label. Given a trained DL model f and an original input data sample x, generating an adversarial example x' can generally be described as a box-constrained optimization problem

$$\begin{aligned} &\min_{x'} \; \|x' - x\| \\ &\text{s.t.}\; f(x') = l' \\ &\quad\;\; f(x) = l \\ &\quad\;\; l \neq l' \\ &\quad\;\; x' \in [0, 1] \end{aligned} \tag{2}$$

where l and l' denote the output label of x and x', respectively, and $\|\cdot\|$ denotes the distance between two data sample. Let $\eta = x' - x$ be the perturbation added on x. This optimization problem minimizes the perturbation while misclassifying the prediction with a constraint of input data. In the rest of this paper, we will discuss the variants of generating adversarial images and adversarial examples in other tasks.

III. Taxonomy of Adversarial Examples

To systematically analyze approaches for generating adversarial examples, we analyze the approaches for generating adversarial examples (see details in Section IV) and categorize them along three dimensions: threat model, perturbation, and benchmark.

A. Threat Model

We discuss the threat model in Section I. Based on different scenarios, assumptions, and quality requirements, adversaries decide the attributes they need in adversarial examples and then deploy specific attack approaches. We further decompose the threat model into four aspects: adversarial falsification, adversary's knowledge, adversarial specificity, and attack frequency. For example, if an adversarial example is required to be generated in real time, adversaries should choose a one-time attack instead of an iterative attack in order to complete the task (see attack frequency).

1) *Adversarial Falsification:*
   a) *False positive* attacks generate a negative sample which is misclassified as a positive one (Type I Error). In a malware detection task, a benign software being classified as a malware is a false positive. In an image classification task, a false positive can be an adversarial image unrecognizable to human, while DNNs predict it to a class with a high confidence score.
   b) *False negative* attacks generate a positive sample which is misclassified as a negative one (Type II Error). In a malware detection task, a false negative can be the condition that a malware (usually considered as positive) cannot be identified by the trained model. The false negative attack is also called ML evasion. This error is shown in most adversarial images, where a human can recognize the image, but the neural networks cannot identify it.
2) *Adversary's Knowledge:*
   a) *White-box* attacks assume that the adversary knows everything related to trained neural network models, including training data, model architectures, hyper-parameters, numbers of layers, activation functions, and model weights. Many adversarial examples are generated by calculating model gradients. Since DNNs tend to require only raw input data without handcrafted features and to deploy end-to-end structure, feature selection is not necessary compared with adversarial examples in ML.
   b) *Black-box* attacks assume that the adversary has no access to the trained neural network model. The adversary, acting as a standard user, only knows the output of the model (label or confidence score). This assumption is common for attacking online ML services (e.g., ML on AWS,[1] Google Cloud AI,[2] BigML,[3] Clarifai,[4] Microsoft Azure,[5] IBM Bluemix,[6] and Face++[7]).
   Most adversarial example attacks are *white-box attacks*. However, they can be transferred to attack *black-box* services due to the transferability of adversarial examples proposed by Papernot *et al*. [33]. We will elaborate on it in Section VII-A.
3) *Adversarial Specificity:*
   a) *Targeted* attacks misguide DNNs to a specific class. Targeted attacks usually occur in the multiclass classification problem. For example, an adversary fools an image classifier to predict all adversarial examples as one class. In a face recognition/biometric system, an adversary tries to disguise a face as an authorized user (impersonation) [58]. Targeted attacks usually maximize the probability of targeted adversarial class.
   b) *Nontargeted* attacks do not assign a specific class to the neural network output. The adversarial class of output can be arbitrary except the original one. For example, an adversary makes his/her face misidentified as an arbitrary face in face recognition system (FRS) to evade detection (dodging) [58]. Nontargeted attacks are easier to implement compared with *targeted* attacks since it has more options and space to redirect the output. Nontargeted adversarial examples are usually generated in two ways: 1) running several targeted attacks and taking the one with the smallest perturbation from the results and 2) minimizing the probability of the correct class.
   Some generation approaches [e.g., extended basic iterative method (BIM) and zeroth-order optimization (ZOO)] can be applied to both targeted and nontargeted attacks. For binary classification, *targeted* attacks are equivalent to *nontargeted* attacks.
4) *Attack Frequency:*
   a) *One-time attacks* take only one time to optimize the adversarial examples.
   b) *Iterative attacks* take multiple times to update the adversarial examples.
   Compared with *one-time attacks*, *iterative attacks* usually perform better adversarial examples but require more interactions with victim classifier (more queries) and cost more computational time to generate them. For some computational-intensive tasks (e.g., reinforcement learning), *one-time attacking* may be the only feasible choice.

B. Perturbation

Small perturbation is a fundamental premise for adversarial examples. Adversarial examples are designed to be close to the original samples and imperceptible to a human, which causes the performance degradation of DL models compared with that of a human. We analyze three aspects of perturbation: perturbation scope, perturbation limitation, and perturbation measurement.

1) *Perturbation Scope:*
   a) *Individual* attacks generate different perturbations for each clean input.
   b) *Universal* attacks only create a universal perturbation for the whole data set. This perturbation can be applied to all clean input data.
   Most of the current attacks generate adversarial examples individually. However, *universal* perturbations make it easier to deploy adversary examples in the real world. Adversaries do not require to change the perturbation when the input sample changes.
2) *Perturbation Limitation:*
   a) *Optimized perturbation* sets perturbation as the goal of the optimization problem. These methods aim to

[1]https://aws.amazon.com/machine-learning
[2]https://cloud.google.com/products/machine-learning
[3]https://bigml.com
[4]https://www.clarifai.com
[5]https://azure.microsoft.com/en-us/services/machine-learning
[6]https://console.bluemix.net/catalog/services/machine-learning
[7]https://www.faceplusplus.com

TABLE I
TAXONOMY OF ADVERSARIAL EXAMPLES

Threat Model	Adversarial Falsification	False Negative	[8], [9], [59]–[73]
		False Positive	[74]
	Adversary's Knowledge	White-Box	[8], [9], [59]–[63], [65], [67], [69]–[74]
		Black-Box	[64], [66], [68], [73]
	Adversarial Specificity	Targeted	[8], [59], [61], [63], [64], [66], [67], [69]–[73]
		Non-Targeted	[9], [60], [62], [64]–[66], [68], [69], [73], [74]
	Attack Frequency	One-time	[59], [60]
		Iterative	[8], [9], [61]–[74]
Perturbation	Perturbation Scope	Individual	[8], [9], [59]–[64], [66]–[71], [73], [74]
		Universal	[65]
	Perturbation Limitation	Optimized	[8], [59], [61]–[65], [68], [70], [71]
		Constraint	[59], [66], [67], [69], [73]
		None	[9], [60], [72], [74]
	Perturbation Measurement	Element-wise	[9], [60], [72]
		$\ell_p (p \geq 0)$	ℓ_0: [63], [66], ℓ_1: [70], ℓ_2: [8], [61]–[65], [67]–[69], [71], [73], ℓ_∞: [62], [63], [65], [70], [73],
		PASS	[59]
		None	[74]
Benchmark	Datasets	MNIST	[8], [59]–[63], [68], [70], [71], [74]
		CIFAR-10	[62]–[64], [66]
		ImageNet	[8], [9], [59], [60], [62]–[65], [67], [69], [71]–[74]
		Others	YoutubeDataset: [8] LSUN, SNLI: [68]
	Victim Models	LeNet	[59], [61], [62], [68], [74]
		VGG	[65]–[67], [69]
		AlexNet	[67], [74]
		QuocNet	[8]
		GoogLeNet	[9], [59], [60], [62]–[65], [67]–[69], [72], [73]
		CaffeNet	[62], [65], [67]
		ResNet	[59], [65], [69], [73]
		LSTM	[68]

minimize the perturbation so that humans cannot recognize the perturbation.

b) *Constraint perturbation* sets perturbation as the constraint of the optimization problem. These methods only require the perturbation to be small enough.

3) *Perturbation Measurement:*

a) ℓ_p measures the magnitude of perturbation by p-norm distance

$$\|x\|_p = \left(\sum_{i=1}^{n} \|x_i\|^p\right)^{\frac{1}{p}}. \tag{3}$$

ℓ_0, ℓ_2, and ℓ_∞ are three commonly used ℓ_p metrics. ℓ_0 counts the number of pixels changed in the adversarial examples, ℓ_2 measures the Euclidean distance between the adversarial example and the original sample, and ℓ_∞ denotes the maximum change for all pixels in adversarial examples.

b) Psychometric perceptual adversarial similarity score (PASS) is a new metric introduced in [59], consistent with human perception.

C. Benchmark

Adversaries show the performance of their adversarial attacks based on different data sets and victim models. This inconsistency brings obstacles to evaluate the adversarial attacks and measure the robustness of DL models. Large and high-quality data sets and complex and high-performance DL models usually make adversaries/defenders hard to attack/defend. The diversity of data sets and victim models also makes researchers hard to tell whether the existence of adversarial examples is due to data sets or models. We will discuss this problem in Section VII-C.

1) *Data Sets:* MNIST, CIFAR-10, and ImageNet are the three most widely used image classification data sets to evaluate adversarial attacks. Because MNIST and CIFAR-10 are proved easy to attack and defend due to its simplicity and small size, ImageNet is the best data set to evaluate adversarial attacks so far. A well-designed data set is required to evaluate adversarial attacks.

2) *Victim Models:* Adversaries usually attack several well-known DL models, such as *LeNet*, *VGG*, *AlexNet*, *GoogLeNet*, *CaffeNet*, and *ResNet*.

In Sections IV and V, we will investigate recent studies on adversarial examples according to this taxonomy.

IV. METHODS FOR GENERATING ADVERSARIAL EXAMPLES

In this section, we illustrate several representative approaches for generating adversarial examples. Although many of these approaches are defeated by a countermeasure in later studies, we present these methods to show how the adversarial attacks improved and to what extent state-of-the-art adversarial attacks can achieve. The existence of these methods also requires investigation, which may improve the robustness of DNNs.

Table I summaries the methods for generating adversarial examples in this section based on the proposed taxonomy.

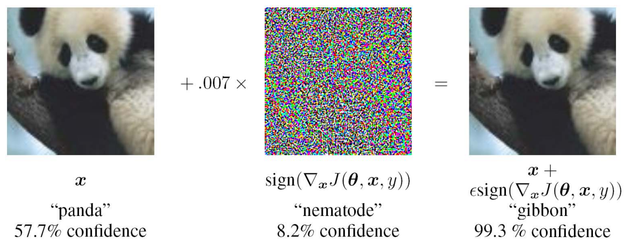
$+.007\times$
=
$\boldsymbol{x}$
"panda"
57.7% confidence
$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"nematode"
8.2% confidence
$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"gibbon"
99.3 % confidence

Fig. 1. Adversarial image generated by FGSM [60]. Left: clean image of a panda. Middle: perturbation. Right: one sample adversarial image, classified as a gibbon.

A. L-BFGS Attack

Szegedy *et al.* [8] first introduced adversarial examples against DNNs in 2014. They generated adversarial examples using an L-BFGS method to solve the general targeted problem

$$\begin{aligned}&\min_{x'} c\|\eta\| + J_\theta(x', l') \\ &\text{s.t. } x' \in [0, 1].\end{aligned} \tag{4}$$

To find a suitable constant c, *L-BFGS attack* calculated approximate values of adversarial examples by line-searching $c > 0$. The authors showed that the generated adversarial examples could also be generalized to different models and different training data sets. They suggested that adversarial examples are never/rarely seen examples in the test data sets.

L-BFGS attack was also used in [71], which implemented a binary search to find the optimal c.

B. Fast Gradient Sign Method

L-BFGS attack used an expensive linear search method to find the optimal value, which was time-consuming and impractical. Goodfellow *et al.* [60] proposed a fast method called fast gradient sign method (FGSM) to generate adversarial examples. They only performed one-step gradient update along the direction of the sign of gradient at each pixel. Their perturbation can be expressed as

$$\eta = \epsilon \text{sign}(\nabla_x J_\theta(x, l)) \tag{5}$$

where ϵ is the magnitude of the perturbation. The generated adversarial example x' is calculated as: $x' = x + \eta$. This perturbation can be computed by using backpropagation. Fig. 1 shows an adversarial example on ImageNet.

They claimed that the linear part of the high-dimensional DNN could not resist adversarial examples, although the linear behavior speeded up training. Regularization approaches are used in DNNs, such as dropout. Pretraining could not improve the robustness of networks.

Rozsa *et al.* [59] proposed a new method, called fast gradient value method, in which they replaced the sign of the gradient with the raw gradient: $\eta = \nabla_x J(\theta, x, l)$. The fast gradient value method has no constraints on each pixel and can generate images with a larger local difference.

According to [72], one-step attack is easy to transfer but also easy to defend (see Section VII-A). Dong *et al.* [73] applied momentum to FGSM to generate adversarial examples more iteratively. The gradients were calculated by

$$\mathbf{g}_{t+1} = \mu \mathbf{g}_t + \frac{\nabla_x J_\theta(x'_t, l)}{\|\nabla_x J_\theta(x'_t, l)\|} \tag{6}$$

and then, the adversarial example is derived by $x'_{t+1} = x'_t + \epsilon \text{sign} \mathbf{g}_{t+1}$. The authors increased the effectiveness of attack by introducing momentum and improved the transferability by applying the one-step attack and the ensembling method.

Kurakin *et al.* [72] extended FGSM to a targeted attack by maximizing the probability of the target class

$$x' = x - \epsilon \text{sign}(\nabla_x J(\theta, x, l')). \tag{7}$$

The authors refer to this attack as one-step target class method.

Tramèr *et al.* [75] found that FGSM with adversarial training is more robust to white-box attacks than to black-box attacks due to *gradient masking*. They proposed a new attack, RAND-FGSM, which added random when updating the adversarial examples to defeat adversarial training

$$\begin{aligned}x_{\text{tmp}} &= x + \alpha \cdot \text{sign}(\mathcal{N}(\mathbf{0}^d, \mathbf{I}^d)) \\ x' &= x_{\text{tmp}} + (\epsilon - \alpha) \cdot \text{sign}(\nabla_{x_{\text{tmp}}} J(x_{\text{tmp}}, l))\end{aligned} \tag{8}$$

where α and ϵ are the parameters, $\alpha < \epsilon$.

C. Basic Iterative Method and Iterative Least-Likely Class Method

Previous methods assume that adversarial data can be directly fed into DNNs. However, in many applications, people can only pass data through devices (e.g., cameras and sensors). Kurakin *et al.* [9] applied adversarial examples to the physical world. They extended the FGSM by running a finer optimization (smaller change) for multiple iterations. In each iteration, they clipped pixel values to avoid a large change on each pixel

$$\text{Clip}_{x,\xi}\{x'\} = \min\{255, x + \xi, \max\{0, x - \epsilon, x'\}\} \tag{9}$$

where $\text{Clip}_{x,\xi}\{x'\}$ limits the change of the generated adversarial image in each iteration. The adversarial examples were generated in multiple iterations

$$\begin{aligned}x_0 &= x \\ x_{n+1} &= \text{Clip}_{x,\xi}\{x_n + \epsilon \text{sign}(\nabla_x J(x_n, y))\}.\end{aligned} \tag{10}$$

The authors referred to this method as BIM.

To further attack a specific class, they chose the least-likely class of the prediction and tried to maximize the cross-entropy loss. This method is referred to as iterative least-likely class (ILLC) method

$$\begin{aligned}x_0 &= x \\ y_{LL} &= \arg\min_y\{p(y|x)\} \\ x_{n+1} &= \text{Clip}_{x,\epsilon}\{x_n - \epsilon \text{sign}(\nabla_x J(x_n, y_{LL}))\}.\end{aligned} \tag{11}$$

They successfully fooled the neural network with a crafted image taken from a cellphone camera. They also found that the FGSM is robust to phototransformation, while iterative methods cannot resist phototransformation.

D. Jacobian-Based Saliency Map Attack

Papernot *et al.* [61] designed an efficient saliency adversarial map called Jacobian-based saliency map attack (JSMA). They first computed the Jacobian matrix of given sample x, which is given by

$$J_F(x) = \frac{\partial F(x)}{\partial x} = \left[\frac{\partial F_j(x)}{\partial x_i}\right]_{i\times j}. \tag{12}$$

According to [63], F denotes the second-to-last layer (logits) in [61]. Carlini and Wagner [63] modify this approach by using the output of the softmax layer as F. In this way, they found the input features of x that made most significant changes to the output. A small perturbation was designed to successfully induce large output variations so that a change in a small portion of features could fool the neural network.

Then, the authors defined two adversarial saliency maps to select the feature/pixel to be crafted in each iteration. They achieved a 97% adversarial success rate by modifying only 4.02% input features per sample. However, this method runs very slow due to its significant computational cost.

E. DeepFool

Moosavi-Dezfooli *et al.* [62] proposed *DeepFool* to find the closest distance from the original input to the decision boundary of adversarial examples. To overcome the nonlinearity in high dimension, they performed an iterative attack with a linear approximation. Starting from an affine classifier, they found that the minimal perturbation of an affine classifier is the distance to the separating affine hyperplane $\mathcal{F} = \{x : w^T x + b = 0\}$. The perturbation of an affine classifier f can be $\eta^*(x) = -(f(x)/\|w\|^2)w$.

If f is a binary differentiable classifier, they used an iterative method to approximate the perturbation by considering f that is linearized around x_i at each iteration. The minimal perturbation is computed as

$$\begin{aligned}&\arg\min_{\eta_i} \|\eta_i\|_2\\ &\text{s.t. } f(x_i) + \nabla f(x_i)^T \eta_i = 0.\end{aligned} \tag{13}$$

This result can also be extended to the multiclass classifier by finding the closest hyperplanes. It can also be extended to a more general ℓ_p norm, $p \in [0, \infty)$. *DeepFool* provided less perturbation compared with FGSM and JSMA did. Compared with JSMA, *DeepFool* also reduced the intensity of perturbation instead of the number of selected features.

F. CPPN EA Fool

Nguyen *et al.* [74] discovered a new type of attack, compositional pattern-producing network-encoded evolutionary algorithm (CPPN EA), where adversarial examples are classified by DNNs with high confidence (99%), which is unrecognizable to human. We categorize this kind of attack as a *false-positive* attack. Fig. 2 shows the false-positive adversarial examples.

They used EA algorithm to produce the adversarial examples. To solve a multiclass classification problem using EA algorithms, they applied multidimensional archive of phenotypic elites MAP-Elites [76]. The authors first encoded images with two different methods: direct encoding (grayscale or hue, saturation, value) and indirect encoding (CPPN). Then, in each iteration, MAP-Elites, such as general EA algorithm, chose a random organism, mutated them randomly, and replaced with the current ones if the new ones have higher *fitness* (high certainty for a class of a neural network). In this way, MAP-Elites can find the best individual for each class. As they claimed, for many adversarial images, the CPPN could locate the critical features to change the outputs of DNNs just like JSMA did. Many images from the same evolutionary are found similar in closely related categories. More interestingly, CPPN EA fooling images are accepted by an art contest with a 35.5% acceptance rate.

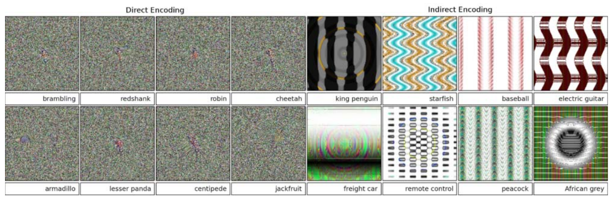

Fig. 2. Unrecognizable examples to humans, but DNNs classify them to a class with high certainty ($\geq 99.6\%$) [74].

G. C&W's Attack

Carlini and Wagner [63] launched a targeted attack to defeat *defensive distillation* (Section VI-A). According to their further study [77], [78], *C&W's Attack* is effective for most of existing adversarial detecting defenses. The authors made several modifications in (2).

They first defined a new objective function g, so that

$$\begin{aligned}&\min_{\eta} \ \|\eta\|_p + c\cdot g(x+\eta)\\ &\text{s.t. } x+\eta \in [0,1]^n\end{aligned} \tag{14}$$

where $g(x') \geq 0$ if and only if $f(x') = l'$. In this way, the distance and the penalty term can be better optimized. The authors listed seven objective function candidates g. One of the effective functions evaluated by their experiments can be

$$g(x') = \max\left(\max_{i\neq l'}(Z(x')_i) - Z(x')_t, -\kappa\right) \tag{15}$$

where Z denotes the softmax function and κ is a constant to control the confidence (κ is set to 0 in [63]).

Second, instead of using box-constrained L-BFGS to find minimal perturbation in L-BFGS Attack method, the authors introduced a new variant w to avoid the box constraint, where w satisfies $\eta = (1/2)(\tanh(w)+1) - x$. General optimizers in DL, such as Adam and stochastic gradient descent, were used to generate adversarial examples and performed 20 iterations of such generation to find an optimal c value by binary searching. However, they found that if the gradients of $\|\eta\|_p$ and $g(x+\eta)$ are not in the same scale, it is hard to find a suitable constant c in all of the iterations of the gradient search and get the optimal result. Due to this reason, two of their proposed functions did not find optimal solutions for adversarial examples.

Third, three distance measurements of perturbation were discussed in this paper: ℓ_0, ℓ_2, and ℓ_∞. The authors provided three kinds of attacks based on the distance metrics: ℓ_0 attack, ℓ_2 attack, and ℓ_∞ attack.

ℓ_2 attack can be described by

$$\min_{w} \|\frac{1}{2}(\tanh(w)+1)\|_2 + c \cdot g\left(\frac{1}{2}\tanh(w)+1\right). \quad (16)$$

The authors showed that the distillation network could not help defend ℓ_2 attack.

The ℓ_0 attack was conducted iteratively since ℓ_0 is not differentiable. In each iteration, a few pixels are considered trivial for generating adversarial examples and removed. The importance of pixels is determined by the gradient of ℓ_2 distance. The iteration stops if the remaining pixels cannot generate an adversarial example.

The ℓ_∞ attack was also an iterative attack, which replaced the ℓ_2 term with a new penalty in each iteration

$$\min\ c \cdot g(x+\eta) + \sum_i [(\eta_i - \tau)^+]. \quad (17)$$

For each iteration, they reduced τ by a factor of 0.9, if all $\eta_i < \tau$. The ℓ_∞ attack considered τ as an estimation of ℓ_∞.

H. Zeroth-Order Optimization

Different from gradient-based adversarial generating approaches, Chen *et al.* [64] proposed a ZOO-based attack. Since this attack does not require gradients, it can be directly deployed in a black-box attack without model transferring. Inspired by [63], the authors modified $g(\cdot)$ in [63] as a new hingelike loss function

$$g(x') = \max\left(\max_{i \neq l'}(\log[f(x)]_i) - \log[f(x)]_{l'}, -\kappa\right) \quad (18)$$

and used symmetric difference quotient to estimate the gradient and Hessian

$$\frac{\partial f(x)}{\partial x_i} \approx \frac{f(x+he_i) - f(x-he_i)}{2h}$$
$$\frac{\partial^2 f(x)}{\partial x_i^2} \approx \frac{f(x+he_i) - 2f(x) + f(x-he_i)}{h^2} \quad (19)$$

where e_i denotes the standard basis vector with the ith component as 1 and h is a small constant.

Through employing the gradient estimation of gradient and Hessian, ZOO does not need the access to the victim DL models. However, it requires expensive computation to query and estimate the gradients. The authors proposed ADAM-like algorithms, ZOO-ADAM, to randomly select a variable and update adversarial examples. Experiments showed that ZOO achieved the comparable performance as *C&W's Attack*.

I. Universal Perturbation

Leveraging their previous method on *DeepFool*, Moosavi-Dezfooli *et al.* [65] developed a universal adversarial attack. The problem they formulated is to find a universal perturbation vector satisfying

$$\|\eta\|_p \le \epsilon$$
$$\mathcal{P}\big(x' \neq f(x)\big) \ge 1 - \delta. \quad (20)$$

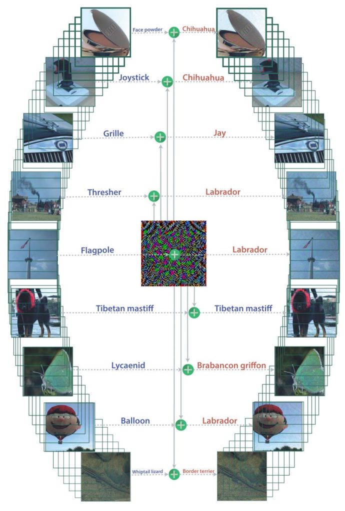

Fig. 3. Universal adversarial example fools the neural network on images. Left images: original labeled natural images. Center image: universal perturbation. Right images: perturbed images with wrong labels. [65].

ϵ limits the size of universal perturbation, and δ controls the failure rate of all the adversarial samples.

For each iteration, they use *DeepFool* method to get a minimal sample perturbation against each input data and update the perturbation to the total perturbation η. This loop will not stop until most data samples are fooled ($\mathcal{P} < 1 - \delta$). From experiments, the universal perturbation can be generated by using a small part of data samples instead of the entire data set. Fig. 3 shows that a universal adversarial example can fool a group of images. The universal perturbations were shown to be generalized well across popular DL architectures (e.g., VGG, CaffeNet, GoogLeNet, and ResNet).

J. One-Pixel Attack

To avoid the problem of measurement of perceptiveness, Su *et al.* [66] generated adversarial examples by only modifying one pixel. The optimization problem becomes

$$\min_{x'}\ J(f(x'), l')$$
$$\text{s.t. } \|\eta\|_0 \le \epsilon_0 \quad (21)$$

where $\epsilon_0 = 1$ for modifying only one pixel. The new constraint made it hard to optimize the problem.

Su *et al.* [66] applied differential evolution (DE), one of the EAs, to find the optimal solution. DE does not require the gradients of the neural networks and can be used in nondifferential objective functions. They evaluated the proposed method on the CIFAR-10 data set using three neural networks: all convolution network (AllConv) [79], network in network [80], and VGG16. Their results showed that 70.97% of images successfully fooled DNNs with at least one target class with confidence of 97.47% on average.

K. Feature Adversary

Sabour *et al.* [67] performed a targeted attack by minimizing the distance of the representation of internal neural network layers instead of the output layer. We refer to this attack as *feature adversary*. The problem can be described by

$$\min_{x'} \|\phi_k(x) - \phi_k(x')\| \quad \text{s.t. } \|x - x'\|_\infty < \delta \tag{22}$$

where ϕ_k denotes a mapping from image input to the output of the kth layer. Instead of finding a minimal perturbation, δ is used as a constraint of perturbation. They claimed that a small fixed value δ is good enough for human perception. Similar to [8], they used L-BFGS-B to solve the optimization problem. The adversarial images are more natural and closer to the targeted images in the internal layers.

L. Hot/Cold

Rozsa *et al*. proposed a *Hot/Cold* method to find multiple adversarial examples for every single image input [59]. They thought small translations and rotations should be allowed as long as they were *imperceptible*.

They defined a new metric, PASS, to measure the noticeable similarity to humans. *Hot/Cold* neglected the unnoticeable difference based on pixels and replaced widely used ℓ_p distance with PASS. PASS includes two stages: 1) aligning the modified image with the original image and 2) measuring the similarity between the aligned image and the original one.

Let $\phi(x', x)$ be a homography transform from the adversarial example x' to the original example x. $\mathcal{H}$ is the homography matrix, with a size of 3×3. $\mathcal{H}$ is solved by maximizing the enhanced correlation coefficient [81] between x' and x. The optimization function is

$$\arg\min_{\mathcal{H}} \left\| \frac{\overline{x}}{\|\overline{x}\|} - \frac{\overline{\phi(x', x)}}{\|\overline{\phi(x', x)}\|} \right\| \tag{23}$$

where $\bar{\cdot}$ denotes the normalization of an image.

Structural SIMilarity (SSIM) index [82] was adopted to measure the just noticeable difference of images. Rozsa *et al.* [59] leveraged SSIM and defined a new measurement, regional SSIM (RSSIM) index as

$$\text{RSSIM}(x_{i,j}, x'_{i,j}) = L(x_{i,j}, x'_{i,j})^\alpha C(x_{i,j}, x'_{i,j})^\beta S(x_{i,j}, x'_{i,j})^\gamma$$

where α, β, and γ are weights of importance for luminance $(L(\cdot,\cdot))$, contrast $(C(\cdot,\cdot))$, and structure $(S(\cdot,\cdot))$. The SSIM can be calculated by averaging RSSIM

$$\text{SSIM}(x_{i,j}, x'_{i,j}) = \frac{1}{n \times m} \sum_{i,j} \text{RSSIM}(x_{i,j}, x'_{i,j}).$$

PASS is defined by the combination of the alignment and the similarity measurement

$$\text{PASS}(x', x) = \text{SSIM}(\phi^*(x', x), x). \tag{24}$$

The adversarial problem with the new distance is described as

$$\begin{aligned} &\min\ D(x, x') \\ &\text{s.t. } f(x') = y' \\ &\qquad \text{PASS}(x, x') \geq \gamma \end{aligned} \tag{25}$$

where $D(x, x')$ denotes a measure of distance [e.g., $1 - PASS(x, x')$ or $\|x - x'\|_p$].

To generate a diverse set of adversarial examples, the authors defined the targeted label l' as *hot* class and the original label l as *cold* class. In each iteration, they moved toward a target (hot) class while moving away from the original (cold) class. Their results showed that generated adversarial examples are comparable to *FGSM* and with more diversity.

M. Natural GAN

Zhao *et al*. [68] utilized generative adversarial networks (GANs) as part of their approach to generate adversarial examples of images and texts, which made adversarial examples more natural to human. We name this approach *Natural GAN*. The authors first trained a WGAN model on the data set, where the generator $\mathcal{G}$ maps random noise to the input domain. They also trained an "inverter" $\mathcal{I}$ to map input data to dense inner representations. Hence, the adversarial noise was generated by minimizing the distance of the inner representations such as "feature adversary." The adversarial examples were generated using the generator: $x' = \mathcal{G}(z')$

$$\min_{z} \|z - \mathcal{I}(x)\| \quad \text{s.t. } f(\mathcal{G}(z)) \neq f(x). \tag{26}$$

Both the generator $\mathcal{G}$ and the inverter $\mathcal{I}$ were built to make adversarial examples natural. *Natural GAN* was a general framework for many DL fields. Zhao *et al.* [68] applied *Natural GAN* to image classification, textual entailment, and machine translation. Since *Natural GAN* does not require gradients of original neural networks, it can also be applied to *black-box attack*.

N. Model-Based Ensembling Attack

Liu *et al*. [69] conducted a study of transferability (Section VII-A) over DNNs on ImageNet and proposed a *model-based ensembling attack* for targeted adversarial examples. The authors argued that compared with nontargeted adversarial examples, targeted adversarial examples are much harder to transfer over deep models. Using *model-based ensembling attack*, they can generate transferable adversarial examples to attack a black-box model.

The authors generated adversarial examples on multiple DNNs with full knowledge and tested them on a black-box model. *Model-based ensembling attack* was derived by the following optimization problem:

$$\arg\min_{x'} -\log\left(\left(\sum_{i=1}^{k} \alpha_i J_i(x', l')\right)\right) + \lambda\|x' - x\| \tag{27}$$

where k is the number of DNNs in the generation, f_i is the function of each network, and α_i is the ensemble weight $(\sum_i^k \alpha_i = 1)$. The results showed that *model-based ensembling attack* could generate transferable targeted adversarial images, which enhanced the power of adversarial examples for black-box attacks. They also proved that this method performs better in generating nontargeted adversarial examples than the previous methods. The authors successfully conducted a black-box attack against Clarifai.com using *model-based ensembling attack*.

O. Ground-Truth Attack

Formal verification techniques aim to evaluate the robustness of a neural network even against zero-day attacks (Section VI-F). Carlini *et al.* [70] constructed a ground-truth attack, which provided adversarial examples with minimal perturbation (ℓ_1, ℓ_∞). *Network verification* always checks whether an adversarial example violates a property of a DNN and whether there exists an example changes the label within a certain distance. *Ground-truth attack* conducted a binary search and found such an adversarial example with the smallest perturbation by invoking Reluplex [83] iteratively. The initial adversarial example is found using C&W's Attack [63] to improve the performance.

V. Applications for Adversarial Examples

We have investigated adversarial examples for the image classification task. In this section, we review adversarial examples against the other tasks. We mainly focus on three questions. What scenarios are adversarial examples applied in new tasks? How to generate adversarial examples in new tasks? Whether to propose a new method or to translate the problem into the image classification task and solve it by the aforementioned methods? Table II summarizes the applications for adversarial examples in this section.

A. Reinforcement Learning

DNNs have been used in reinforcement learning by training policies on raw input (e.g., images). Huang *et al.* [84] and Kos and Song [85] generated adversarial examples on deep reinforcement learning policies. Both [84] and [85] performed *One-time attack*, due to the intensive computation and real-time response required by reinforcement learning.

Huang *et al.* [84] applied FGSM to attack deep reinforcement learning networks and algorithms: deep Q network (DQN), trust region policy optimization, and asynchronous advantage actor-critic (A3C) [97]. Similar to [60], they added small perturbations on the input of policy by calculating the gradient of the cross-entropy loss function: $\nabla_x J(\theta, x, \ell)$). Since DQN does not have stochastic policy input, softmax of Q-values is considered to calculate the loss function. They evaluated adversarial examples on four Atari 2600 games with three norm constraints ℓ_1, ℓ_2, and ℓ_∞. They found *Huang's Attack* with ℓ_1 norm conducted a successful attack on both *white-box attack* and *black-box attack* (no access to the training algorithms, parameters, and hyperparameters).

Kos and Song [85] used FGSM to attack A3C algorithm and Atari Pong task and found that injecting perturbations in a fraction of frames is sufficient.

B. Generative Modeling

Kos *et al.* [86] and Tabacof *et al.* [87] proposed adversarial examples for generative models. An adversary for autoencoder (AE) can inject perturbations into the input of encoder and generate a targeted class after decoding. Fig. 4 shows a targeted adversarial example for an AE. Adding perturbations on the input image of the encoder can misguide the AE by making decoder to generating a targeted adversarial output image.

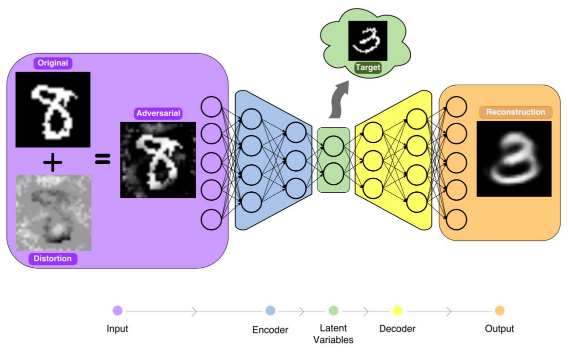

Fig. 4. Adversarial attacks for AEs [87]. Perturbations are added to the input the encoder. After encoding and decoding, the decoder will output an adversarial image presenting an incorrect class.

Kos *et al.* [86] described a scenario to apply adversarial examples against AE. AEs can be used to compress data by an encoder and decompress by a decoder. For example, Toderici *et al.* [98] use RNN-based AE to compress image. Ledig *et al.* [99] used GAN to super-resolve images. Adversaries can leverage AE to reconstruct an adversarial image by adding perturbation to the input of the encoder.

Tabacof *et al.* [87] used *feature adversary* attack against AE and variational AE (VAE). The adversarial examples were formulated as follows:

$$\begin{aligned} &\min_{\eta} \ D(z_{x'}, z_x) + c\|\eta\| \\ &\text{s.t. } x' \in [L, U] \\ &\qquad z_{x'} = \text{Encoder}(x') \\ &\qquad z_x = \text{Encoder}(x) \end{aligned} \tag{28}$$

where $D(\cdot)$ is the distance between latent encoding representations z_x and $z_{x'}$. Tabacof *et al.* [87] chose KL-divergence to measure $D(\cdot)$. They tested their attacks on the MNIST and SVHN data set and found that generating adversarial examples for AE is much harder than for classifiers. VAE is even slightly more robust than deterministic AE.

Kos *et al.* [86] extended Tabacof *et al.*'s work [87] by designing another two kinds of distances. Hence, the adversarial examples can be generated by optimizing

$$\min_{\eta} c\|\eta\| + J(x', l'). \tag{29}$$

The loss function J can be cross-entropy (refer to "Classifier Attack" in [86]), VAE loss function ("L_{VAE} Attck"), and distance between the original latent vector z and the modified encoded vector x' ("Latent Attack," similar to Tabacof *et al.*'s work [87]). They tested VAE and VAE-GAN [100] on the MNIST, SVHN, and CelebA data sets. In their experimental results, "Latent Attack" achieved the best result.

C. Face Recognition

DNN-based FRS and face detection system have been widely deployed in commercial products due to their high performance. Sharif *et al.* [58] first provided a design

TABLE II
SUMMARY OF APPLICATIONS FOR ADVERSARIAL EXAMPLES

Applications	Representative Study	Method	Adversarial Falsification	Adversary's Knowledge	Adversarial Specificity	Perturbation Scope	Perturbation Limitation	Attack Frequency	Perturbation Measurement	Dataset	Architecture
Reinforcement Learning	[84]	FGSM	N/A	White-box & Black-box	Non-Targeted	Individual	N/A	One-time	$\ell_1, \ell_2, \ell_\infty$	Atari	DQN, TRPO, A3C
	[85]	FGSM	N/A	White-box	Non-Targeted	Individual	N/A	One-time	N/A	Atari Pong	A3C
Generative Modeling	[86]	Feature Adversary, C&W	N/A	White-box	Targeted	Individual	Optimized	Iterative	ℓ_2	MNIST, SVHN, CelebA	VAE, VAE-GAN
	[87]	Feature Adversary	N/A	White-box	Targeted	Individual	Optimized	Iterative	ℓ_2	MNIST, SVHN	VAE, AE
Face Recog-nition	[58]	Impersonation & Dodging Attack	False negative	white-box & black-box	Targeted & Non-Targeted	Universal	Optimized	Iterative	Total Variation	LFW,	VGGFace
Object Detection	[11]	DAG	False negative & False positive	White-box & Black-box	Non-Targeted	Individual	N/A	Iterative	N/A	VOC2007, VOC2012	Faster-RCNN
Semantic Segmentation	[11]	DAG	False negative & False positive	White-box & Black-box	Non-Targeted	Individual	N/A	Iterative	N/A	DeepLab	FCN
	[88]	ILLC	False negative	White-box	Targeted	Individual	N/A	Iterative	ℓ_∞	Cityscapes	FCN
	[89]	ILLC	False negative	White-box	Targeted	Universal	N/A	Iterative	N/A	Cityscapes	FCN
Reading Comprehension	[90]	AddSent, AddAny	N/A	Black-box	Non-Targeted	Individual	N/A	One-time & Iterative	N/A	SQuAD	BiDAF, Match-LSTM, and twelve other published models
	[91]	Reinforcement Learning	False negative	White-box	Non-Targeted	Individual	Optimized	Iterative	ℓ_0	TripAdvisor Dataset	Bi-LSTM, memory network
Malware Detection	[92]	JSMA	False negative	White-box	Targeted	Individual	Optimized	Iterative	ℓ_2	DREBIN	2-layer FC
	[93]	Reinforcement Learning	False negative	Black-box	Targeted	Individual	N/A	Iterative	N/A	N/A	Gradient Boosted Decision Tree
	[94]	GAN	False negative	Black-box	Targeted	Individual	N/A	Iterative	N/A	malwr	Multi-layer Perceptron
	[95]	GAN	False negative	Black-box	Targeted	Individual	N/A	Iterative	N/A	Alexa Top 1M	Random Forest
	[96]	Generic Pro-gramming	False negative	Black-box	Targeted	Individual	N/A	Iterative	N/A	Contagio	Random Forest, SVM

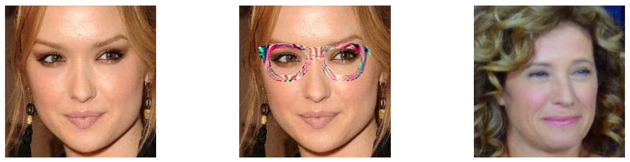

Fig. 5. Example of adversarial eyeglass frame against FRS [58].

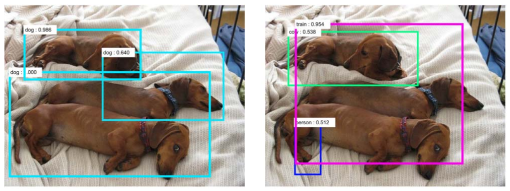

Fig. 6. Adversarial example for object detection task [11]. Left: object detection on a clean image. Right: object detection on an adversarial image.

of eyeglass frames to attack a DNN-based FRS [101], which composes 11 blocks with 38 layers and 1 *triplet loss* function for feature embedding. Based on the *triplet loss* function, [58] designed a *softmaxloss* function

$$J(x) = -\log\left(\frac{e^{\langle h_{c_x}, f(x)\rangle}}{\sum_{c=1}^{m} e^{\langle h_c, f(x)\rangle}}\right) \tag{30}$$

where h_c is a one-hot vector of class c and $\langle\cdot,\cdot\rangle$ denotes inner product. Then, they used *L-BFGS Attack* to generate adversarial examples.

In a further step, [58] implemented adversarial eyeglass frames to achieve attacks in the physical world: the perturbations can only be injected into the area of eyeglass frames. They also enhanced the printability of adversarial images on the frame by adding a penalty of nonprintability score (NPS) to the optimized objective. Similar to *universal perturbation*, they optimize the perturbation to be applied to a set of face images. They successfully dodged (nontargeted attack) against FRS (over 80 % time) and misguided FRS as a specific face (targeted attack) with a high success rate (depending on the target). Fig. 5 shows an example of adversarial eyeglass frames.

Leveraging the approach of printability, [10] proposed an attack algorithm, robust physical perturbations (RP_2), to modify a stop sign as a speed limit sign).[8] They changed the physical road signs by two kinds of attacks: 1) overlaying an adversarial road sign over a physical sign and 2) sticking perturbations on an existing sign. Evtimov *et al.* [10] included an NPS in the optimization objective to improve the printability.

D. Object Detection

The object detection task is to find the proposal of an object (bounding box), which can be viewed as an image classification task for every possible proposal. Xie *et al.* [11] proposed a universal algorithm called dense adversary generation (DAG) to generate adversarial examples for both object detection and semantic segmentation. The authors aimed at making the prediction (detection/segmentation) incorrect (nontargeted). Fig. 6 shows an adversarial example for the object detection task.

[8]This method was shown not effective for standard detectors (YOLO and Faster RCNN) in [102].

Xie *et al.* [11] defined $T = t_1, t_2, \ldots, t_N$ as the recognition targets. For image classification, the classifier only needs one target—entire image ($N = 1$). For semantic segmentation, targets consist of all pixels ($N = \#ofpixels$). For object detection, targets consist of all possible proposals ($N = (\#ofpixels)^2$). Then, the objective function sums up the loss from all targets. Instead of optimizing the loss from all targets, the authors performed an iterative optimization and only updated the loss for the targets correctly predicted in the previous iteration. The final perturbation sums up normalized perturbations in all iterations. To deal with a large number of targets for objective detection problem, the authors used a regional proposal network [4] to generate possible targets, which greatly decreases the computation for targets in object detection. DAG also showed the capability of generating images that are unrecognizable to human but DL could predict (*false positives*).

E. Semantic Segmentation

Image segmentation task can be viewed as an image classification task for every pixel. Since each perturbation is responsible for at least one-pixel segmentation, this makes the space of perturbations for segmentation much smaller than that for image classification [103]. Xie *et al.* [11], Fischer *et al.* [88], and Metzen *et al.* [103] generated adversarial examples against the semantic image segmentation task. However, their attacks are proposed under different scenarios. As we just discussed, [11] performed a nontargeted segmentation. Fischer *et al.* [88] and Metzen *et al.* [103] performed a targeted segmentation and tried to remove a certain class by making DL model to misguide it as background classes.

Fischer *et al.* [88] generated adversarial examples by assigning pixels with the adversarial class that their nearest neighbor belongs to. The success rate was measured by the percentage of pixels of the chosen class to be changed or of the rest classes to be preserved.

Metzen *et al.* [103] presented a method to generate universal adversarial perturbations against semantic image segmentation task. They assigned the primary objective of adversarial examples and hid the objects (e.g., pedestrians) while keeping the rest segmentation unchanged. Metzen *et al.* [103] defined background classes and targeted classes (not targeted adversarial classes). Targeted classes are classes to be removed. Similar to [88], pixels that belong to the targeted classes would be assigned to their nearest background classes

$$\begin{aligned} l_{ij}^{\text{target}} &= l_{i'j'}^{\text{pred}} \quad \forall(i,j) \in I_{\text{targeted}} \\ l_{ij}^{\text{target}} &= l_{ij}^{\text{pred}} \quad \forall(i,j) \in I_{\text{background}} \\ (i',j') &= \underset{(i',j') \in I_{\text{background}}}{\arg\min} \; \|i'-i\| + \|j'-j\| \end{aligned} \tag{31}$$

where $I_{\text{targeted}} = (i,j) | f(x_{ij}) = l^*$ denotes the area to be removed. Fig. 7 shows an adversarial example to hide pedestrians. They used ILLC attack to solve this problem and also extended *universal perturbation* method to get the universal perturbation. Their results showed the existence of universal perturbation for semantic segmentation task.

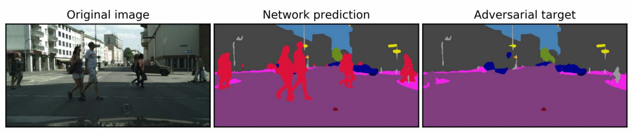

Fig. 7. Adversary examples of hiding pedestrians in the semantic segmentation task [103]. Left: original image. Middle: segmentation of the original image predicted by DNN. Right: segmentation of the adversarial image predicted by DNN.

TABLE III
SUMMARY OF COUNTERMEASURES FOR ADVERSARIAL EXAMPLES

	Defensive Strategies	Representative Studies
Reactive	Adversarial Detecting	[23], [89], [109]–[116]
	Input Reconstruction	[114], [117], [118]
	Network Verification	[83], [119], [120]
Proactive	Network Distillation	[121]
	Adversarial (Re)Training	[24], [25], [60], [72], [75], [122]
	Classifier Robustifying	[123], [124]

F. Natural Language Processing

Many tasks in NLP can be attacked by adversarial examples. People usually generate adversarial examples by adding/deleting words in the sentences.

The task of reading comprehension (also known as question answering) is to read paragraphs and answer questions about the paragraphs. To generate adversarial examples that are consistent with the correct answer and do not confuse human, Jia and Liang [90] added distracting (adversarial) sentences to the end of the paragraph. They found that models for the reading comprehension task are overstable instead of oversensitivity, which means that DL models cannot tell the subtle but critical difference in the paragraphs.

They proposed two kinds of methods to generate adversarial examples: 1) adding grammatical sentences similar to the question but not contradictory to the correct answer (AddSent) and 2) adding a sentence with arbitrary English words (AddAny). Jia and Liang [90] successfully fooled all the models (16 models) they tested on the Stanford Question Answering data set [104]. The adversarial examples also have the capability of transferability and cannot be improved by adversarial training. However, the adversarial sentences require manpower to fix the errors in the sentences.

Li *et al.* [91] aimed to fool a DL-based sentiment classifier by removing the minimum subset of words in the given text. Reinforcement learning was used to find an approximate subset, where the reward function was proposed as $(1/\|D\|)$ when the sentiment label changes and 0 otherwise. $\|D\|$ denotes the number of removing word set D. The reward function also included a regularizer to make sentence contiguous.

The changes in [90] and [91] can easily be recognized by humans. More natural adversarial examples for texture data was proposed by *Natural GAN* [68] (Section IV-M).

G. Malware Detection

DL has been used in static- and behavioral-based malware detection due to its capability of detecting zero-day malware [105]–[108]. Recent studies generated adversarial malware samples to evade DL-based malware detection [92]–[94], [96].

Grosse *et al.* [92] adapted JSMA method to attack Android malware detection model. Xu *et al.* [96] evaded two PDF malware classifier, PDFrate and Hidost, by modifying PDF and parsed the PDF file and changed its object structure using genetic programing. The adversarial PDF file was then packed with new objects.

Anderson *et al.* [95] used GAN to generate adversarial domain names to evade the detection of domain generation algorithms. Hu and Tan [94] proposed a GAN-based algorithm, MalGan, to generate malware examples and evade black-box detection and used a substitute detector to simulate the real detector and leveraged the transferability of adversarial examples to attack the real detector. MalGan was evaluated by 180k programs with application program interface features. However, [94] required the knowledge of features used in the model. Anderson *et al.* [93] used a large number of features (2350) to cover the required feature space of portable executable (PE) files. The features included PE header metadata, section metadata, and import and export table metadata. Anderson *et al.* [93] also defined several modifications to generate malware evading DL detection. The solution was trained by reinforcement learning, where the evasion rate is considered as a reward.

VI. COUNTERMEASURES FOR ADVERSARIAL EXAMPLES

Countermeasures for adversarial examples have two types of defense strategies: 1) *reactive*: detect adversarial examples after DNNs are built and 2) *proactive*: make DNNs more robust before adversaries generate adversarial examples. In this section, we discuss three reactive countermeasures (*adversarial detecting*, *input reconstruction*, and *network verification*) and three proactive countermeasures (*network distillation*, *adversarial (re)training*, and *classifier robustifying*). We will also discuss an ensembling method to prevent adversarial examples. Table III summarizes the countermeasures.

A. Network Distillation

Papernot *et al.* [121] used network distillation to defend DNNs against adversarial examples. Network distillation was originally designed to reduce the size of DNNs by transferring knowledge from a large network to a small one [125], [126] (Fig. 8). The probability of classes produced by the first DNN is used as inputs to train the second DNN. The probability of classes extracts the knowledge learned from the first DNN. Softmax is usually used to normalize the last layer of DNN and produce the probability of classes. The softmax output of the first DNN, also the input of the next DNN, can be described as

$$q_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)} \tag{32}$$

where T is a *temperature* parameter to control the level of knowledge distillation. In DNNs, *temperature* T is set to 1. When T is large, the output of softmax will be vague (when $T \to \infty$, the probability of all classes $\to \frac{1}{m}$). When T is small, only one class is close to 1, while the rest goes to 0.

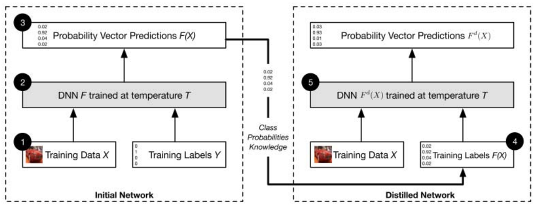

Fig. 8. Network distillation of DNNs [121].

This schema of network distillation can be duplicated several times and connects several DNNs.

In [121], network distillation extracted knowledge from DNNs to improve robustness. The authors found that attacks primarily aimed at the sensitivity of networks and then proved that using high-temperature softmax reduced the model sensitivity to small perturbations. *Network distillation* defense was tested on the MNIST and CIFAR-10 data sets and reduced the success rate of JSMA attack by 0.5% and 5%, respectively. "Network distillation" also improved the generalization of the neural networks.

B. Adversarial (Re)training

Training with adversarial examples is one of the countermeasures to make neural networks more robust. Goodfellow *et al*. [60] and Huang *et al*. [122] included adversarial examples in the training stage. They generated adversarial examples in every step of training and inject them into the training set. Goodfellow *et al*. [60] and Huang *et al*. [122] showed that adversarial training improved the robustness of DNNs. Adversarial training could provide regularization for DNNs [60] and improve the precision as well [24].

Goodfellow *et al*. [60] and Huang *et al*. [122] evaluated only on the MNIST data set. A comprehensive analysis of adversarial training methods on the ImageNet data set was presented in [72]. They used half adversarial examples and half origin examples in each step of training. From the results, *adversarial training* increased the robustness of neural networks for *one-step attacks* (e.g., FGSM) but would not help under *iterative attacks* (e.g., BIM and ILLC methods). Kurakin *et al*. [72] suggested that adversarial training is used for regularization only to avoid overfitting (e.g., the case in [60] with the small MNIST data set).

Tramèr *et al.* [75] found that the adversarial trained models on the MNIST and ImageNet data sets are more robust to white-box adversarial examples than to the transferred examples (black-box).

Dong *et al.* [25] minimized both the cross-entropy loss and the internal representation distance during adversarial training, which can be seen as a defense version of *feature adversary*.

To deal with the transferred black-box model, [75] proposed *ensembling adversarial training* method that trained the model with adversarial examples generated from multiple sources: the models being trained and also pretrained external models.

C. Adversarial Detecting

Many research projects tried to detect adversarial examples in the testing stage [23], [89], [109]–[114], [116].

Lu *et al.* [23], Metzen *et al.* [89], and Gong *et al.* [111] trained DNN-based binary classifiers as detectors to classify the input data as a legitimate (clean) input or an adversarial example. Metzen *et al*. [89] created a detector for adversarial examples as an auxiliary network of the original neural network. The detector is a small and straightforward neural network predicting on binary classification, i.e., the probability of the input being adversarial. SafetyNet [23] extracted the binary threshold of each rectified linear unit (ReLU) layer's output as the features of the adversarial detector and detects adversarial images by an radial basis function (RBF)-SVM classifier. The authors claimed that their method is hard to be defeated by adversaries even when adversaries know the detector because it is difficult for adversaries to find an optimal value, for both adversarial examples and new features of SafetyNet detector. Grosse *et al.* [112] added an outlier class to the original DL model. The model detected the adversarial examples by classifying it as an outlier. They found that the measurement of maximum mean discrepancy and energy distance could distinguish the distribution of adversarial data sets and clean data sets.

Feinman *et al.* [110] provided a Bayesian view of detecting adversarial examples and claimed that the uncertainty of adversarial examples is higher than the clean data. Hence, they deployed a Bayesian neural network to estimate the uncertainty of input data and distinguish adversarial examples and clean input data based on uncertainty estimation.

Similarly, [114] used probability divergence (Jensen–Shannon divergence) as one of its detectors. Hendrycks and Gimpel [113] showed that after whitening by principal component analysis, adversarial examples have different coefficients in low-ranked components.

Song *et al.* [118] trained a PixelCNN neural network [127] and found that the distribution of adversarial examples is different from clean data. They calculated p-value based on the rank of PixelCNN and rejected adversarial examples using the p-values. The results showed that this approach could detect FGSM, BIM, DeepFool, and C&W attack.

Pang *et al.* [115] trained neural networks with "reverse cross entropy" to better distinguish adversarial examples from clean data in the latent layers and then detected adversarial examples using a method called "kernel density" in the testing stage. The "reverse cross entropy" made the DNN to predict with high confidence on the true class and uniform distribution on the other classes. In this way, the DNN was trained to map the clean input close to a low-dimensional manifold in the layer before softmax. This brought great convenience for further detection of adversarial examples.

Lin *et al.* [116] leveraged multiple previous images to predict future input and detect adversarial examples, in the task of reinforcement learning.

However, Carlini and Wagner [77], [78] summarized most of these adversarial detecting methods (see [89], [109]–[113]) and showed that these methods could not defend against their previous attack *C&W's Attack* with slight changes of loss function.

D. Input Reconstruction

Adversarial examples can be transformed to clean data via reconstruction. After transformation, the adversarial examples will not affect the prediction of DL models. Gu and Rigazio [117] proposed a variant of AE network with a penalty, called deep contractive AE, to increase the robustness of neural networks. A denoising AE network is trained to encode adversarial examples to original ones to remove adversarial perturbations. Meng and Chen [114] reconstructed the adversarial examples by: 1) adding Gaussian noise or 2) encoding them with AE as Plan B in MagNet (Section VI-G).

PixelDefend reconstructed the adversarial images back to the training distribution [118] using PixelCNN. *PixelDefend* changed all pixels along each channel to maximize the probability distribution

$$\begin{aligned}&\max_{x'} \ \mathcal{P}_t(x')\\ &\text{s.t.} \ \|x'-x\|_\infty \le \epsilon_{\text{defend}}\end{aligned} \tag{33}$$

where $\mathcal{P}_t$ denotes the training distribution and ϵ_{defend} controls the new changes on the adversarial examples. *PixelDefend* also leveraged adversarial detecting, so that if an adversarial example is not detected as malicious, no change will be made to the adversarial examples ($\epsilon_{\text{defend}} = 0$).

E. Classifier Robustifying

Bradshaw *et al.* [123] and Abbasi and Gagné [124] design robust architectures of DNNs to prevent adversarial examples.

Due to the uncertainty from adversarial examples, Bradshaw *et al.* [123] leveraged Bayesian classifiers to build more robust neural networks. Gaussian processes (GPs) with RBF kernels were used to provide uncertainty estimation. The proposed neural networks were called GP hybrid deep neural networks (GPDNNs). GPs expressed the latent variables as a Gaussian distribution parameterized by the functions of mean and covariance and encoded them with RBF kernels. Bradshaw *et al.* [123] showed that GPDNNs achieved a comparable performance with general DNNs and more robust to adversarial examples. The authors claimed that GPDNNs "know when they do not know."

Abbasi and Gagné [124] observed that adversarial examples usually went into a small subset of incorrect classes and separated the classes into subclasses and ensembled the result from all subclasses by voting to prevent adversarial examples misclassified.

F. Network Verification

Verifying the properties of DNNs is a promising solution to defend adversarial examples, because it may detect the new unseen attacks. *Network verification* checks the properties of a neural network: whether an input violates or satisfies the property.

Katz *et al.* [83] proposed a verification method for neural networks with ReLU activation function called Reluplex. They used satisfiability modulo theory solver to verify the neural networks. The authors showed that within a small perturbation, there was no existing adversarial example to misclassify the neural networks. They also proved that the problem of network verification is NP-complete. Carlini *et al.* [70] extended their assumption of ReLU function by presenting $\max(x, y) = ReLU(x-y)+y$ and $\|x\| = ReLU(2x)-x$. However, Reluplex runs very slow due to the large computation of verifying the networks and only works for DNNs with several hundred nodes [120]. Katz *et al.* [120] proposed two potential solutions: 1) prioritizing the order of checking nodes and 2) sharing information of verification.

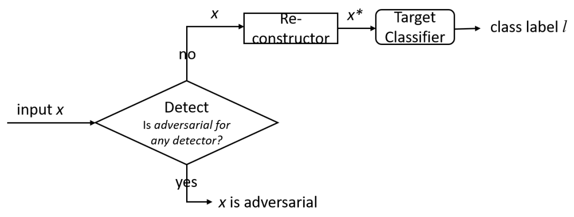

Fig. 9. MagNet workflow: one or more detectors first detects if input x is adversarial; if not, reconstruct x to x^* before feeding it to the classifier (modified from [114]).

Instead of checking each point individually, Gopinath *et al.* [119] proposed *DeepSafe* to provide safe regions of a DNN using Reluplex. They also introduced *targeted robustness* a safe region only regarding a targeted class.

G. Ensembling Defenses

Due to the multifacet of adversarial examples, multiple defense strategies can be performed together (parallel or sequential) to defend adversarial examples.

Aforementioned *PixelDefend* [118] is composed of an adversarial detector and an "input reconstructor" to establish a defense strategy.

MagNet included one or more detectors and a reconstructor ("reformer" in this paper) as Plans A and B [114]. The detectors are used to find the adversarial examples that are far from the boundary of the manifold. In [114], they first measured the distance between input and encoded input and also the probability divergence (Jensen–Shannon divergence) between softmax output of input and encoded input. The adversarial examples were expected a large distance and probability divergence. To deal with the adversarial examples close to the boundary, MagNet used a reconstructor built by neural network-based AEs. The reconstructor will map adversarial examples to legitimate examples. Fig. 9 shows the workflow of the defense of two phases.

After investigating several defensive approaches, [128] showed that the ensemble of those defensive approaches does not make the neural networks strong.

H. Summary

Almost all defenses are shown to be effective only for part of attacks. They tend not to be defensive for some strong (fail to defend) and unseen attacks. Most defenses target adversarial examples in the computer vision task. However, with the development of adversarial examples in other areas, new defenses for these areas, especially for safety-critical systems, are urgently required.

VII. Challenges and Discussion

In this section, we discuss the current challenges and the potential solutions for adversarial examples. Although many methods and theorems have been proposed and developed recently, a lot of fundamental questions need to be well explained and many challenges need to be addressed. The reason for the existence of adversarial examples is interesting and one of the most fundamental problems for both adversaries and researchers, which exploits the vulnerability of neural networks and helps defenders to resist adversarial examples. We will discuss the following questions in this section. Why do adversarial examples transfer? How to stop the transferability? Why are some defenses effective and others not? How to measure the strength of an attack as well as a defense? How to evaluate the robustness of a DNN against seen/unseen adversarial examples?

A. Transferability

Transferability is a common property for adversarial examples. Szegedy *et al.* first found that adversarial examples generated against a neural network can fool the same neural networks trained by different data sets. Papernot *et al.* [33] found that adversarial examples generated against a neural network can fool other neural networks with different architectures, even other classifiers trained by different ML algorithms. Transferability is critical for black-box attacks where the victim DL model and the training data set are not accessible. Attackers can train a substitute neural network model and then generate adversarial examples against a substitute model. Then, the victim model will be vulnerable to these adversarial examples due to transferability. From a defender's view, if we hinder transferability of adversarial examples, we can defend all white-box attackers who need to access the model and require transferability.

We define the transferability of adversarial examples in three levels from easy to hard: 1) transfer among the same neural network architecture trained with different data; 2) transfer among different neural network architectures trained for the same task; and 3) transfer among DNNs for different tasks. To the best of our knowledge, there is no existing solution on the third level yet (for instance, transfer an adversarial image from object detection to semantic segmentation).

Many studies examined transferability to show the ability of adversarial examples [8], [60]. Papernot *et al.* studied the transferability between conventional ML techniques (i.e., logistic regression, SVM, decision tree, and k-nearest neighbors) and DNNs. They found that adversarial examples can be transferred between different parameters, training data set of ML models, and even across different ML techniques.

Liu *et al.* [69] investigated transferability of targeted and nontargeted adversarial examples on complex models and large data sets (e.g., the ImageNet data set). They found that nontargeted adversarial examples are much more transferable than targeted ones. They observed that the decision boundaries of different models aligned well with each other. Thus, they proposed *model-based ensembling attack* to create transferable targeted adversarial examples.

Tramèr *et al.* [129] found that the distance to the model's decision boundary is on average larger than the distance between two models' boundaries in the same direction. This may explain the existence of transferability of adversarial examples. Tramèr *et al.* [129] also claimed that transferability might not be an inherent property of DNNs by showing a counterexample.

B. Existence of Adversarial Examples

The reason for the existence of adversarial examples is still an open question. Are adversarial examples an inherent property of DNNs? Are adversarial examples the "Achilles' heel" of DNNs with high performance? Many hypotheses have been proposed to explain the existence.

1) Data Incompletion: One assumption is that adversarial examples are of low probability and low test coverage of corner cases in the testing data set [8], [130]. From training a PixelCNN, [118] found that the distribution of adversarial examples was different from clean data. Even for a simple Gaussian model, a robust model can be more complicated and requires much more training data than that of a "standard" model [131].

2) Model Capability: Adversarial examples are a phenomenon not only for DNNs but also for all classifiers [33], [132]. Goodfellow *et al.* [60] suggested that adversarial examples are the results of models being too linear in high-dimensional manifolds. Tanay and Griffin [133] showed that in the linear case, the adversarial examples exist when the decision boundary is close to the manifold of the training data.

Contrary to [60], [132] believed that adversarial examples are due to the "low flexibility" of the classifier for certain tasks. Linearity is not an "obvious explanation" [67]. Tabacof and Valle [71] blamed adversarial examples for the sparse and discontinuous manifold that makes classifier erratic.

3) No Robust Model: Dong *et al.* [25] suggested that the decision boundaries of DNNs are inherently incorrect, which do not detect semantic objects. Fawzi *et al.* [134] showed that if a data set is generated by a smooth generative model with large latent space, there is no robust classifier to adversarial examples. Similarly, [135] proved that if a model is trained on a sphere data set and misclassifies a small part of the data set, then there exist adversarial examples with a small perturbation.

In addition to adversarial examples for the image classification task, as discussed in Section V, adversarial examples have been generated in various applications. Many of them deployed utterly different methods. Some applications can use the same method used in the image classification task. However, some of them need to propose a novel method. Current studies on adversarial examples mainly focus on the image classification task. No existing paper explains the relationship among different applications and existence of a universal attacking/defending method to be applied to all the applications.

C. Robustness Evaluation

The competition between attacks and defenses for adversarial examples becomes an "arms race": a defensive method that was proposed to prevent existing attacks was later shown to be vulnerable to some new attacks and vice versa [70], [112]. Some defenses showed that they could defend a particular attack but later failed with a slight change of the attack [109], [110]. Hence, the evaluation on the robustness of a DNN is necessary. For example, [136] provided an upper bound of robustness for linear classifier and quadratic classifier. The following problems for robustness evaluation of DNNs require further exploration.

1) Methodology for Evaluation on the Robustness of Deep Neural Networks: Many DNNs are planned to be deployed in safety-critical settings. Defending only existing attacks is not sufficient. Zero-day (new) attacks would be more harmful to DNNs. A methodology for evaluating the robustness of DNNs is required, especially for zero-day attacks, which helps people understand the confidence of model prediction and how much we can rely on them in the real world. Carlini *et al.* [70], Katz *et al.* [83], Bastani *et al.* [137], and Huang *et al.* [138] conducted initial studies on the evaluation. Moreover, this problem lies not only in the performance of DNN models but also in the confidentiality and privacy.

2) Benchmark Platform for Attacks and Defenses: Most attacks and defenses described their methods without publicly available code, not to mention the parameters used in their methods. This brings difficulties for other researchers to reproduce their solutions and provide the corresponding attacks/defenses. For example, Carlini tried his best to "find the best possible defense parameters + random initialization".[9] Some researchers even drew different conclusions because of different settings in their experiments. If there exists any benchmark, where both adversaries and defenders conduct experiments in a uniform way (i.e., the same threat model, data set, classifier, and attacking/defending approach), we can make a more precise comparison between different attacking and defending techniques.

Cleverhans [139] and Foolbox [140] are open-source libraries to benchmark the vulnerability of DNNs against adversarial images. They build frameworks to evaluate the attacks. However, defensive strategies are missing in both the tools. Providing a data set of adversarial examples generated by different methods will make it easy for finding the blind point of DNNs and developing new defense strategies. This problem also occurs in other areas in DL.

Google Brain organized three competitions in NIPS 2017 competition track, including targeted adversarial attack, nontargeted adversarial attack, and defense against adversarial attack [141]. The data set in the competition consisted of a set of images never used before and manually labeled the images, 1000 images for development and 5000 images for final testing. The submitted attacks and competitions are used as the benchmarks to evaluate themselves. The adversarial attacks and defenses are scored by the number of runs to fool the defenses/correctly classify images.

We present the workflow of a benchmark platform for attackers and defenders (Fig. 10).

[9]Code repository used in [77]: https://github.com/carlini/nn_breaking_detection.

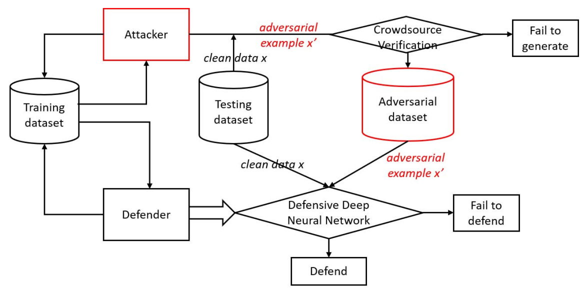

Fig. 10. Workflow of a benchmark platform for attackers and defenders: 1) attackers and defenders update/train their strategies on training data set; 2) attackers generate adversarial examples on the clean data; 3) the adversarial examples are verified by crowdsourcing whether recognizable to human; 4) defenders generate a DNN as a defensive strategy; and 5) evaluate the defensive strategy.

3) Various Applications for Robustness Evaluation: Similar to the existence of adversarial examples for various applications, a wide range of applications make it hard to evaluate the robustness of a DNN architecture. How to compare methods generating adversarial example under different threat models? Do we have a universal methodology to evaluate the robustness under all scenarios? Tackling these unsolved problems is a future direction.

VIII. Conclusion

In this paper, we reviewed the recent findings of adversarial examples in DNNs. We investigated the existing methods for generating adversarial examples.[10] A taxonomy of adversarial examples was proposed. We also explored the applications and countermeasures for adversarial examples.

This paper attempted to cover the state-of-the-art studies for adversarial examples in the DL domain. Compared with recent work on adversarial examples, we analyzed and discussed the current challenges and potential solutions in adversarial examples.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[2] K. Simonyan and A. Zisserman. (2014). "Very deep convolutional networks for large-scale image recognition." [Online]. Available: https://arxiv.org/abs/1409.1556
[3] J. Redmon and A. Farhadi. (2016). "YOLO9000: Better, faster, stronger." [Online]. Available: https://arxiv.org/abs/1612.08242
[4] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks." in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 91–99.
[5] G. Saon, H.-K. J. Kuo, S. Rennie, and M. Picheny. (2015). "The IBM 2015 English conversational telephone speech recognition system." [Online]. Available: https://arxiv.org/abs/1505.05899
[6] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3104–3112.
[7] A. V. D. Oord *et al.* (2016). "WaveNet: A generative model for raw audio." [Online]. Available: https://arxiv.org/abs/1609.03499
[8] C. Szegedy *et al.* (2013). "Intriguing properties of neural networks." [Online]. Available: https://arxiv.org/abs/1312.6199
[9] A. Kurakin, I. Goodfellow, and S. Bengio. (2016). "Adversarial examples in the physical world." [Online]. Available: https://arxiv.org/abs/1607.02533

[10]Due to the rapid development of adversarial examples (attacks and defenses), we only considered the papers published before November 2017. We will update the survey with new methodologies and papers in our future work.

[10] I. Evtimov *et al.* (2017). "Robust physical-world attacks on deep learning models." [Online]. Available: https://arxiv.org/abs/1707.08945

[11] C. Xie, J. Wang, Z. Zhang, Y. Zhou, L. Xie, and A. Yuille, "Adversarial examples for semantic segmentation and object detection," in *Proc. Int. Conf. Comput. Vis.*, Oct. 2017, pp. 1378–1387.

[12] N. Carlini *et al.*, "Hidden voice commands," in *Proc. USENIX Security Symp.*, 2016, pp. 513–530.

[13] G. Zhang, C. Yan, X. Ji, T. Zhang, T. Zhang, and W. Xu. (2017). "DolphinAtack: Inaudible voice commands." [Online]. Available: https://arxiv.org/abs/1708.09537

[14] *Apple Siri*. Accessed: 2018. [Online]. Available: https://www.apple.com/siri/

[15] *Amazon Alexa*. Accessed: 2018. [Online]. Available: https://developer.amazon.com/alexa

[16] *Cortana*. Accessed: 2018. [Online]. Available: https://www.microsoft.com/en-us/cortana

[17] W. Knight, *The Dark Secret at the Heart of AI*. Cambridge, MA, USA: MIT Technology Review, 2017.

[18] D. Castelvecchi, "Can we open the black box of AI?" *Nature News*, vol. 538, no. 7623, p. 20, 2016.

[19] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2017, pp. 1–11.

[20] Z. C. Lipton, "The mythos of model interpretability," in *Proc. Int. Conf. Mach. Learn. (ICML) Workshop*, 2016, pp. 1–9.

[21] R. Shwartz-Ziv and N. Tishby. (2017). "Opening the black box of deep neural networks via information." [Online]. Available: https://arxiv.org/abs/1703.00810

[22] R. R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, and D. Batra. (2016). "Grad-CAM: Visual explanations from deep networks via gradient-based localization." [Online]. Available: https://arxiv.org/abs/1610.02391

[23] J. Lu, T. Issaranon, and D. Forsyth, "SafetyNet: Detecting and rejecting adversarial examples robustly," in *Proc. ICCV*, 2017, pp. 1–9.

[24] Y. Wu, D. Bamman, and S. Russell, "Adversarial training for relation extraction," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2017, pp. 1779–1784.

[25] Y. Dong, H. Su, J. Zhu, and F. Bao. (2017). "Towards interpretable deep neural networks by leveraging adversarial examples." [Online]. Available: https://arxiv.org/abs/1708.05493

[26] N. Papernot, P. McDaniel, A. Sinha, and M. Wellman. (2016). "Towards the science of security and privacy in machine learning." [Online]. Available: https://arxiv.org/abs/1611.03814

[27] B. Biggio, B. Nelson, and P. Laskov. (2012). "Poisoning attacks against support vector machines." [Online]. Available: https://arxiv.org/abs/1206.6389

[28] F. Roli, B. Biggio, and G. Fumera, "Pattern recognition systems under attack," in *Proc. Iberoamerican Congr. Pattern Recognit.* Springer, 2013, pp. 1–8.

[29] H. Xiao, B. Biggio, G. Brown, G. Fumera, C. Eckert, and F. Roli, "Is feature selection secure against training data poisoning?" in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1689–1698.

[30] M. Mozaffari-Kermani, S. Sur-Kolay, A. Raghunathan, and N. K. Jha, "Systematic poisoning attacks on and defenses for machine learning in healthcare," *IEEE J. Biomed. Health Informat.*, vol. 19, no. 6, pp. 1893–1905, Nov. 2015.

[31] A. Beatson, Z. Wang, and H. Liu, "Blind attacks on machine learners," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2397–2405.

[32] S. Alfeld, X. Zhu, and P. Barford, "Data poisoning attacks against autoregressive models," in *Proc. AAAI*, 2016, pp. 1452–1458.

[33] N. Papernot, P. D. McDaniel, and I. J. Goodfellow. (2016). "Transferability in machine learning: From phenomena to black-box attacks using adversarial samples." [Online]. Available: https://arxiv.org/abs/1605.07277

[34] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, 2015, pp. 1322–1333.

[35] M. Abadi *et al.*, "Deep learning with differential privacy," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 308–318.

[36] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 3–18.

[37] Y. Bengio and Y. LeCun, "Scaling learning algorithms towards AI," *Large-Scale Kernel Mach.*, vol. 34, no. 5, pp. 1–41, 2007.

[38] D. Storcheus, A. Rostamizadeh, and S. Kumar, "A survey of modern questions and challenges in feature extraction," in *Proc. NIPS Workshop*, Dec. 2015, pp. 1–18.

[39] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1026–1034.

[40] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 779–788.

[41] P. He, W. Huang, T. He, Q. Zhu, Y. Qiao, and X. Li, "Single shot text detector with regional attention," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 3066–3074.

[42] C. Yan, H. Xie, J. Chen, Z. Zha, X. Hao, and Y. Zhang, "A fast uyghur text detector for complex background images," *IEEE Trans. Multimedia*, vol. 20, no. 12, pp. 3389–3398, Dec. 2018.

[43] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 3431–3440.

[44] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. (2017). "Rethinking Atrous convolution for semantic image segmentation." [Online]. Available: https://arxiv.org/abs/1706.05587

[45] Q. V. Le, "Building high-level features using large scale unsupervised learning," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2013, pp. 8595–8598.

[46] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. pattern Recognit.*, Jun. 2015, pp. 1–9.

[47] 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.*, Jul. 2016, pp. 2818–2826.

[48] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-ResNet and the impact of residual connections on learning," in *Proc. AAAI*, 2017, pp. 4278–4284.

[49] 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.

[50] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[51] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[52] M. Barreno, B. Nelson, A. D. Joseph, and J. D. Tygar, "The security of machine learning," *Mach. Learn.*, vol. 81, no. 2, pp. 121–148, 2010.

[53] N. Dalvi, P. Domingos, S. Sanghai, and D. Verma, "Adversarial classification," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2004, pp. 99–108.

[54] D. Lowd and C. Meek, "Adversarial learning," in *Proc. 11th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2005, pp. 641–647.

[55] B. Biggio, G. Fumera, and F. Roli, "Multiple classifier systems for robust classifier design in adversarial environments," *Int. J. Mach. Learn. Cybern.*, vol. 1, nos. 1–4, pp. 27–41, 2010.

[56] B. Biggio *et al.*, "Evasion attacks against machine learning at test time," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*. Springer, 2013, pp. 387–402.

[57] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar, "Can machine learning be secure?" in *Proc. ACM Symp. Inf., Comput. Commun. Secur.*, 2006, pp. 16–25.

[58] 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.

[59] A. Rozsa, E. M. Rudd, and T. E. Boult, "Adversarial diversity and hard positive generation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR) Workshops*, Jun. 2016, pp. 25–32.

[60] I. J. Goodfellow, J. Shlens, and C. Szegedy. (2014). "Explaining and harnessing adversarial examples." [Online]. Available: https://arxiv.org/abs/1412.6572

[61] N. Papernot, P. 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 (EuroS&P)*, Mar. 2016, pp. 372–387.

[62] 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. (CVPR)*, Jun. 2016, pp. 2574–2582.

[63] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. IEEE Symp. Secur. Privacy (S&P)*, Mar. 2017, pp. 39–57.

[64] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh. (2017). "ZOO: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models." [Online]. Available: https://arxiv.org/abs/1708.03999

[65] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 86–94.

[66] J. Su, D. V. Vargas, and S. Kouichi. (2017). "One pixel attack for fooling deep neural networks." [Online]. Available: https://arxiv.org/abs/1710.08864

[67] S. Sabour, Y. Cao, F. Faghri, and D. J. Fleet, "Adversarial manipulation of deep representations," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2016.

[68] Z. Zhao, D. Dua, and S. Singh. (2017). "Generating natural adversarial examples." [Online]. Available: https://arxiv.org/abs/1710.11342

[69] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017, pp.'1–14.

[70] N. Carlini, G. Katz, C. Barrett, and D. L. Dill. (2017). "Provably minimally-distorted adversarial examples." [Online]. Available: https://arxiv.org/abs/1709.10207

[71] P. Tabacof and E. Valle, "Exploring the space of adversarial images," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2016, pp. 426–433.

[72] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017, pp. 1–17.

[73] Y. Dong *et al.* (2017). "Boosting adversarial attacks with momentum." [Online]. Available: https://arxiv.org/abs/1710.06081

[74] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 427–436.

[75] F. Tramèr, A. Kurakin, N. Papernot, D. Boneh, and P. McDaniel. (2017). "Ensemble adversarial training: Attacks and defenses." [Online]. Available: https://arxiv.org/abs/1705.07204

[76] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret, "Robots that can adapt like animals," *Nature*, vol. 521, no. 7553, pp. 503–507, May 2015.

[77] N. Carlini and D. Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," in *Proc. AISEC*, 2017, pp. 3–14.

[78] N. Carlini and D. Wagner. (2017). "MagNet and 'efficient defenses against adversarial attacks' are not robust to adversarial examples." [Online]. Available: https://arxiv.org/abs/1711.08478

[79] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. (2014). "Striving for simplicity: The all convolutional net." [Online]. Available: https://arxiv.org/abs/1412.6806

[80] M. Lin, Q. Chen, and S. Yan. (2013). "Network in network." [Online]. Available: https://arxiv.org/abs/1312.4400

[81] G. D. Evangelidis and E. Z. Psarakis, "Parametric image alignment using enhanced correlation coefficient maximization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 10, pp. 1858–1865, Oct. 2008.

[82] J. R. Flynn, S. Ward, J. Abich, and D. Poole, "Image quality assessment using the SSIM and the just noticeable difference paradigm," in *Proc. Int. Conf. Eng. Psychol. Cognit. Ergonom.* Springer, 2013, pp. 23–30.

[83] G. Katz, C. Barrett, D. Dill, K. Julian, and M. Kochenderfer. (2017). "Reluplex: An efficient SMT solver for verifying deep neural networks." [Online]. Available: https://arxiv.org/abs/1702.01135

[84] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel. (2017). "Adversarial attacks on neural network policies." [Online]. Available: https://arxiv.org/abs/1702.02284

[85] J. Kos and D. Song. Delving into adversarial attacks on deep policies," in *Proc. Int. Conf. Learn. Represent. (ICLR) Workshop*, 2017. [Online]. Available: https://openreview.net/pdf?id=BJcib5mFe

[86] J. Kos, I. Fischer, and D. Song. (2017). "Adversarial examples for generative models." [Online]. Available: https://arxiv.org/abs/1702.06832

[87] P. Tabacof, J. Tavares, and E. Valle, "Adversarial images for variational autoencoders," in *Proc. NIPS Workshop*, 2016.

[88] V. Fischer, M. C. Kumar, J. H. Metzen, and T. Brox, "Adversarial examples for semantic image segmentation," in *Proc. ICLR Workshop*, 2017. [Online]. Available: https://openreview.net/pdf?id=S1SED1MYe

[89] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, "On detecting adversarial perturbations," in *Proc. ICLR*, 2017. [Online]. Available: https://openreview.net/pdf?id=SJzCSf9xg

[90] R. Jia and P. Liang, "Adversarial examples for evaluating reading comprehension systems," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2017, pp. 1–11.

[91] 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.

[92] 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.

[93] H. S. Anderson, A. Kharkar, B. Filar, and P. Roth, "Evading machine learning malware detection," in *Proc. Black Hat*, 2017. [Online]. Available: https://www.blackhat.com/docs/us-17/thursday/us-17-Anderson-Bot-Vs-Bot-Evading-Machine-Learning-Malware-Detection-wp.pdf

[94] W. Hu and Y. Tan. (2017). "Generating adversarial malware examples for black-box attacks based on GAN." [Online]. Available: https://arxiv.org/abs/1702.05983

[95] H. S. Anderson, J. Woodbridge, and B. Filar, "DeepDGA: Adversarially-tuned domain generation and detection," in *Proc. ACM Workshop Artif. Intell. Secur. (AISec)*, 2016, pp. 13–21.

[96] W. Xu, Y. Qi, and D. Evans, "Automatically evading classifiers," in *Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, 2016, pp. 1–15.

[97] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1928–1937.

[98] G. Toderici *et al.* (2016). "Full resolution image compression with recurrent neural networks." [Online]. Available: https://arxiv.org/abs/1608.05148

[99] C. Ledig *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1–10.

[100] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther. (2015). "Autoencoding beyond pixels using a learned similarity metric." [Online]. Available: https://arxiv.org/abs/1512.09300

[101] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," in *Proc. Brit. Mach. Vis. Conf.*, 2015, vol. 1. no. 3, p. 6.

[102] J. Lu, H. Sibai, E. Fabry, and D. Forsyth. (2017). "Standard detectors aren't (currently) fooled by physical adversarial stop signs." [Online]. Available: https://arxiv.org/abs/1710.03337

[103] J. H. Metzen, M. C. Kumar, T. Brox, and V. Fischer, "Universal adversarial perturbations against semantic image segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2017, pp. 2755–2764.

[104] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. (2016). "SQuAD: 100,000+ questions for machine comprehension of text." [Online]. Available: https://arxiv.org/abs/1606.05250

[105] Z. Yuan, Y. Lu, Z. Wang, and Y. Xue, "Droid-Sec: Deep learning in android malware detection," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 371–372, 2014.

[106] G. E. Dahl, J. W. Stokes, L. Deng, and D. Yu, "Large-scale malware classification using random projections and neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2013, pp. 3422–3426.

[107] J. Saxe and K. Berlin, "Deep neural network based malware detection using two dimensional binary program features," in *Proc. 10th Int. Conf. Malicious Unwanted Softw. (MALWARE)*, Oct. 2015, pp. 11–20.

[108] R. Sun *et al.* (2017). "Learning fast and slow: PROPEDEUTICA for real-time malware detection." [Online]. Available: https://arxiv.org/abs/1712.01145

[109] A. N. Bhagoji, D. Cullina, C. Sitawarin, and P. Mittal. (2017). "Enhancing robustness of machine learning systems via data transformations," [Online]. Available: https://arxiv.org/abs/1704.02654

[110] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner. (2017). "Detecting adversarial samples from artifacts." [Online]. Available: https://arxiv.org/abs/1703.00410

[111] Z. Gong, W. Wang, and W.-S. Ku. (2017). "Adversarial and clean data are not twins." [Online]. Available: https://arxiv.org/abs/1704.04960

[112] K. Grosse, P. Manoharan, N. Papernot, M. Backes, and P. McDaniel. (2017). "On the (statistical) detection of adversarial examples." [Online]. Available: https://arxiv.org/abs/1702.06280

[113] D. Hendrycks and K. Gimpel, "Early methods for detecting adversarial images," in *Proc. ICLR Workshop*, 2017. [Online]. Available: https://openreview.net/pdf?id=B1dexpDug

[114] D. Meng and H. Chen, "MagNet: A two-pronged defense against adversarial examples," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, New York, NY, USA, pp. 135–147.

[115] T. Pang, C. Du, Y. Dong, and J. Zhu. (2017). "Towards robust detection of adversarial examples." [Online]. Available: https://arxiv.org/abs/1706.00633

[116] Y.-C. Lin, M.-Y. Liu, M. Sun, and J.-B. Huang. (2017). "Detecting adversarial attacks on neural network policies with visual foresight." [Online]. Available: https://arxiv.org/abs/1710.00814

[117] S. Gu and L. Rigazio, "Towards deep neural network architectures robust to adversarial examples," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2015, pp. 1–9.

[118] Y. Song, T. Kim, S. Nowozin, S. Ermon, and N. Kushman. (2017). "PixelDefend: Leveraging generative models to understand and defend against adversarial examples." [Online]. Available: https://arxiv.org/abs/1710.10766

[119] D. Gopinath, G. Katz, C. S. Pasareanu, and C. Barrett. (2017). "DeepSafe: A data-driven approach for checking adversarial robustness in neural networks." [Online]. Available: https://arxiv.org/abs/1710.00486

[120] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer. (2017). "Towards proving the adversarial robustness of deep neural networks." [Online]. Available: https://arxiv.org/abs/1709.02802

[121] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2016, pp. 582–597.

[122] R. Huang, B. Xu, D. Schuurmans, and C. Szepesvári. (2015). "Learning with a strong adversary." [Online]. Available: https://arxiv.org/abs/1511.03034

[123] J. Bradshaw, A. G. de G. Matthews, and Z. Ghahramani. (2017). "Adversarial examples, uncertainty, and transfer testing robustness in Gaussian process hybrid deep networks." [Online]. Available: https://arxiv.org/abs/1707.02476

[124] M. Abbasi and C. Gagné. (2017). "Robustness to adversarial examples through an ensemble of specialists." [Online]. Available: https://arxiv.org/abs/1702.06856

[125] J. Ba and R. Caruana, "Do deep nets really need to be deep?" in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2654–2662.

[126] G. Hinton, O. Vinyals, and J. Dean. (2015). "Distilling the knowledge in a neural network." [Online]. Available: https://arxiv.org/abs/1503.02531

[127] T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma, "PixelCNN++: Improving the PixelCNN with discretized logistic mixture likelihood and other modifications," in *Proc. ICLR Poster*, 2017, pp. 1–10.

[128] W. He, J. Wei, X. Chen, N. Carlini, and D. Song, "Adversarial example defenses: Ensembles of weak defenses are not strong," in *Proc. 11th USENIX Workshop Offensive Technol. (WOOT)*. Vancouver, BC, Canada: USENIX Association, 2017.

[129] F. Tramèr, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel. (2017). "The space of transferable adversarial examples." [Online]. Available: https://arxiv.org/abs/1704.03453

[130] K. Pei, Y. Cao, J. Yang, and S. Jana, "DeepXplore: Automated whitebox testing of deep learning systems," in *Proc. ACM SIGOPS 26th Symp. Oper. Syst. Principles*, 2017, pp. 1–18.

[131] L. Schmidt, S. Santurkar, D. Tsipras, K. Talwar, and A. Mądry. (2018). "Adversarially robust generalization requires more data." [Online]. Available: https://arxiv.org/abs/1804.11285

[132] A. Fawzi, O. Fawzi, and P. Frossard, "Fundamental limits on adversarial robustness," in *Proc. ICML Workshop Deep Learn.*, 2015, pp. 1–7.

[133] T. Tanay and L. Griffin. (2016). "A boundary tilting persepective on the phenomenon of adversarial examples." [Online]. Available: https://arxiv.org/abs/1608.07690

[134] A. Fawzi, H. Fawzi, and O. Fawzi. (2018). "Adversarial vulnerability for any classifier." [Online]. Available: https://arxiv.org/abs/1802.08686

[135] J. Gilmer *et al.* (2018). "Adversarial spheres." [Online]. Available: https://arxiv.org/abs/1801.02774

[136] A. Fawzi, O. Fawzi, and P. Frossard. (2015). "Analysis of classifiers' robustness to adversarial perturbations." [Online]. Available: https://arxiv.org/abs/1502.02590

[137] 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.

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

[139] I. Goodfellow *et al.* (2017). "CleverHans v2.1.0 adversarial examples library." [Online]. Available: https://arxiv.org/abs/1610.00768

[140] J. Rauber, W. Brendel, and M. Bethge. (2017). "Foolbox: A Python toolbox to benchmark the robustness of machine learning models." [Online]. Available: http://arxiv.org/abs/1707.04131

[141] A. Kurakin *et al.* (2018). "Adversarial attacks and defences competition." [Online]. Available: https://arxiv.org/abs/1804.00097

Xiaoyong Yuan received the B.S. degree in mathematics from Fudan University, Shanghai, China, in 2012, and the M.S. degree in software engineering from Peking University, Beijing, China, in 2015. He is currently pursuing the Ph.D. degree with the Department of Computer and Information Science and Engineering, University of Florida, Gainesville, FL, USA.

His current research interests include deep learning and security.

Pan He received the B.E. degree from the Department of Software Engineering, Sichuan University, Chengdu, China, in 2015. He is currently pursuing the Ph.D. degree with the Department of Computer and Information Science and Engineering, University of Florida, Gainesville, FL, USA.

His current research interests include deep learning and computer vision. As an enthusiastic researcher, his goal is to combine state-of-the-art computer vision algorithms with real-life industrial problems.

Qile Zhu received the B.E. degree from Zhejiang University, Hangzhou, China, in 2013, and the master's degree from the University of Florida, Gainesville, FL, USA, in 2015, where he is currently pursuing the Ph.D. degree with the Department of Computer and Information Science and Engineering.

His current research interests include deep learning and natural language processing.

Xiaolin (Andy) Li received the Ph.D. degree in computer engineering from Rutgers University, New Brunswick, NJ, USA.

He is currently a Professor and a University Term Professor with the Department of Electrical and Computer Engineering and the Department of Computer and Information Science and Engineering, University of Florida (UF), Gainesville, FL, USA. He is also the Founding Director of the National Science Foundation Center for Big Learning, the first NSF center on deep learning, with UF (lead), Carnegie Mellon University, University of Oregon, and University of Missouri-Kansas City. He is also the Director of the Large-scale Intelligent Systems Laboratory. He has published over 120 peer-reviewed papers in journals and conference proceedings and five books. He holds four patents (three licensees). His team has created many software systems and tools. His current research interests include big data, machine learning, deep learning, cloud computing, intelligent platforms, high-performance computing, and security and privacy for health, precision medicine, Internet of Things, computer vision, natural language processing, robotics, genomics, science, engineering, and business.

Dr. Li was a recipient of the National Science Foundation CAREER Award, the Internet2 Innovative Application Award, the NSF I-Corps Top Team Award, Top Team (DeepBipolar) in the CAGI Challenge on detecting bipolar disorder, and the best paper awards (IEEE International Conference on Machine Learning and Applications 2016, IEEE International Conference on Sensing, Communication and Networking 2016, ACM Cloud and Autonomic Computing Conference 2013, and IEEE International Symposium on Ubisafe Computing 2007). For more information, http://www.andyli.ece.ufl.edu and http://nsfcbl.org.