International Journal of Automation and Computing 17(2), April 2020, 151-178
DOI: 10.1007/s11633-019-1211-x

Adversarial Attacks and Defenses in Images, Graphs and Text: A Review

Han Xu Yao Ma Hao-Chen Liu Debayan Deb
Hui Liu Ji-Liang Tang Anil K. Jain

Department of Computer Science and Engineering, Michigan State University, Michigan 48823, USA

Abstract: Deep neural networks (DNN) have achieved unprecedented success in numerous machine learning tasks in various domains. However, the existence of adversarial examples raises our concerns in adopting deep learning to safety-critical applications. As a result, we have witnessed increasing interests in studying attack and defense mechanisms for DNN models on different data types, such as images, graphs and text. Thus, it is necessary to provide a systematic and comprehensive overview of the main threats of attacks and the success of corresponding countermeasures. In this survey, we review the state of the art algorithms for generating adversarial examples and the countermeasures against adversarial examples, for three most popular data types, including images, graphs and text.

Keywords: Adversarial example, model safety, robustness, defenses, deep learning.

1 Introduction

Deep neural networks (DNN) have become increasingly popular and successful in many machine learning tasks. They have been deployed in different recognition problems in the domains of images, graphs, text and speech, with remarkable success. In the image recognition domain, they are able to recognize objects with near-human level accuracy[1, 2]. They are also used in speech recognition[3], natural language processing[4] and for playing games[5].

Because of these accomplishments, deep learning techniques are also applied in safety-critical tasks. For example, in autonomous vehicles, deep convolutional neural networks (CNNs) are used to recognize road signs[6]. The machine learning technique used here is required to be highly accurate, stable and reliable. But, what if the CNN model fails to recognize the "STOP" sign by the roadside and the vehicle keeps going? It will be a dangerous situation. Similarly, in financial fraud detection systems, companies frequently use graph convolutional networks (GCNs)[7] to decide whether their customers are trustworthy or not. If there are fraudsters disguising their personal identity information to evade the company's detection, it will cause a huge loss to the company. Therefore, the safety issues of deep neural networks have become a major concern.

In recent years, many works[2, 8, 9] have shown that DNN models are vulnerable to adversarial examples, which can be formally defined as – "Adversarial examples are inputs to machine learning models that an attacker intentionally designed to cause the model to make mistakes". In the image classification domain, these adversarial examples are intentionally synthesized images which look almost exactly the same as the original images (see Fig. 1), but can mislead the classifier to provide wrong prediction outputs. For a well-trained DNN image classifier on the MNIST dataset, almost all the digit samples can be attacked by an imperceptible perturbation, added on the original image. Meanwhile, in other application domains involving graphs, text or audio, similar adversarial attacking schemes also exist to confuse deep learning models. For example, perturbing only a couple of edges can mislead graph neural networks[10], and inserting typos to a sentence can fool text classification or dialogue systems[11]. As a result, the existence of adversarial examples in all application fields has cautioned researchers against directly adopting DNNs in safety-critical machine learning tasks.

To deal with the threat of adversarial examples, stud-

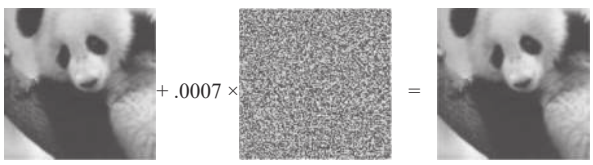


Fig. 1 By adding an unnoticeable perturbation, "panda" is classified as "gibbon" (Image credit: Goodfellow et al.[9])

Review
Manuscript received October 13, 2019; accepted November 11, 2019
Recommended by Associate Editor Hong Qiao

ies have been published with the aim of finding countermeasures to protect deep neural networks. These approaches can be roughly categorized to three main types: 1) Gradient masking[12, 13]: Since most attacking algorithms are based on the gradient information of the classifiers, masking or obfuscating the gradients will confuse the attack mechanisms. 2) Robust optimization[14, 15]: These studies show how to train a robust classifier that can correctly classify the adversarial examples. 3) Adversary detection[16, 17]: The approaches attempt to check whether a sample is benign or adversarial before feeding it to the deep learning models. It can be seen as a method of guarding against adversarial examples. These methods improve DNN′s resistance to adversarial examples.

In addition to building safe and reliable DNN models, studying adversarial examples and their countermeasures is also beneficial for us to understand the nature of DNNs and consequently improve them. For example, adversarial perturbations are perceptually indistinguishable to human eyes but can evade DNN′s detection. This suggests that the DNN′s predictive approach does not align with human reasoning. There are works[9, 18] to explain and interpret the existence of adversarial examples of DNNs, which can help us gain more insight into DNN models.

In this review, we aim to summarize and discuss the main studies dealing with adversarial examples and their countermeasures. We provide a systematic and comprehensive review on the start-of-the-art algorithms from images, graphs and text domain, which gives an overview of the main techniques and contributions to adversarial attacks and defenses.

The main structure of this survey is as follows: In Section 2, we introduce some important definitions and concepts which are frequently used in adversarial attacks and their defenses. It also gives a basic taxonomy of the types of attacks and defenses. In Sections 3 and 4, we discuss main attack and defense techniques in the image classification scenario. We use Section 5 to briefly introduce some studies which try to explain the phenomenon of adversarial examples. Sections 6 and 7 review the studies in graph and text data, respectively.

2 Definitions and notations

In this section, we give a brief introduction to the key components of model attacks and defenses. We hope that our explanations can help our audience to understand the main components of the related works on adversarial attacks and their countermeasures. By answering the following questions, we define the main terminology:

1) Adversary's goal (Section 2.1.1)

What is the goal or purpose of the attacker? Does he want to misguide the classifier′s decision on one sample, or influence the overall performance of the classifier?

2) Adversary's knowledge (Section 2.1.2)

What information is available to the attacker? Does he know the classifier′s structure, its parameters or the training set used for classifier training?

3) Victim models (Section 2.1.3)

What kind of deep learning models do adversaries usually attack? Why are adversaries interested in attacking these models?

4) Security evaluation (Section 2.2)

How can we evaluate the safety of a victim model when faced with adversarial examples? What is the relationship and difference between these security metrics and other model goodness metrics, such as accuracy or risks?

2.1 Threat model

2.1.1 Adversary′s goal

1) Poisoning attack versus evasion attack

Poisoning attacks refer to the attacking algorithms that allow an attacker to insert/modify several fake samples into the training database of a DNN algorithm. These fake samples can cause failures of the trained classifier. They can result in the poor accuracy[19], or wrong prediction on some given test samples[10]. This type of attacks frequently appears in the situation where the adversary has access to the training database. For example, web-based repositories and "honeypots" often collect malware examples for training, which provides an opportunity for adversaries to poison the data.

In evasion attacks, the classifiers are fixed and usually have good performance on benign testing samples. The adversaries do not have authority to change the classifier or its parameters, but they craft some fake samples that the classifier cannot recognize. In other words, the adversaries generate some fraudulent examples to evade detection by the classifier. For example, in autonomous driving vehicles, sticking a few pieces of tapes on the stop signs can confuse the vehicle′s road sign recognizer[20].

2） Targeted attack versus non-targeted attack

In targeted attack, when the victim sample (x, y) is given, where x is feature vector and $y \in \mathcal{Y}$ is the ground truth label of x, the adversary aims to induce the classifier to give a specific label $t \in \mathcal{Y}$ to the perturbed sample x'. For example, a fraudster is likely to attack a financial company′s credit evaluation model to disguise himself as a highly credible client of this company.

If there is no specified target label t for the victim sample x, the attack is called non-targeted attack. The adversary only wants the classifier to predict incorrectly.

2.1.2 Adversary′s knowledge

1) White-box attack

In a white-box setting, the adversary has access to all the information of the target neural network, including its architecture, parameters, gradients, etc. The adversary can make full use of the network information to carefully craft adversarial examples. White-box attacks have been extensively studied because the disclosure of model archi-

tecture and parameters helps people understand the weakness of DNN models clearly and it can be analyzed mathematically. As stated by Tramer et al.[21], security against white-box attacks is the property that we desire machine learning (ML) models to have.

2) Black-box attack

In a black-box attack setting, the inner configuration of DNN models is unavailable to adversaries. Adversaries can only feed the input data and query the outputs of the models. They usually attack the models by keeping feeding samples to the box and observing the output to exploit the model's input-output relationship, and identity its weakness. Compared to white-box attacks, black-box attacks are more practical in applications because model designers usually do not open source their model parameters for proprietary reasons.

3) Semi-white (gray) box attack

In a semi-white box or gray box attack setting, the attacker trains a generative model for producing adversarial examples in a white-box setting. Once the generative model is trained, the attacker does not need victim model anymore, and can craft adversarial examples in a black-box setting.

2.1.3 Victim models

We briefly summarize the machine learning models which are susceptible to adversarial examples, and some popular deep learning architectures used in image, graph and text data domains. In our review, we mainly discuss studies of adversarial examples for deep neural networks.

1) Conventional machine learning models

For conventional machine learning tools, there is a long history of studying safety issues. Biggio et al.[22] attack support vector machine (SVM) classifiers and fully-connected shallow neural networks for the MNIST dataset. Barreno et al.[23] examine the security of SpamBayes, a Bayesian method based spam detection software. In [24], the security of Naive Bayes classifiers is checked. Many of these ideas and strategies have been adopted in the study of adversarial attacks in deep neural networks.

2) Deep neural networks

Different from traditional machine learning techniques which require domain knowledge and manual feature engineering, DNNs are end-to-end learning algorithms. The models use raw data directly as input to the model, and learn objects' underlying structures and attributes. The end-to-end architecture of DNNs makes it easy for adversaries to exploit their weakness, and generate high-quality deceptive inputs (adversarial examples). Moreover, because of the implicit nature of DNNs, some of their properties are still not well understood or interpretable. Therefore, studying the security issues of DNN models is necessary. Next, we will briefly introduce some popular victim deep learning models which are used as "benchmark" models in attack/defense studies.

a) Fully-connected neural networks (FC)

Fully-connected neural networks are composed of layers of artificial neurons. In each layer, the neurons take the input from previous layers, process it with the activation function and send it to the next layer; the input of first layer is sample x, and the (softmax) output of last layer is the score $F(x)$. An m-layer fully connected neural network can be formed as

$$z^{(0)} = x; \quad z^{(l+1)} = \sigma(W^l z^l + b^l).$$

One thing to note is that, the back-propagation algorithm helps calculate $\frac{\partial F(x;\theta)}{\partial \theta}$, which makes gradient descent effective in learning parameters. In adversarial learning, back-propagation also facilitates the calculation of the term: $\frac{\partial F(x;\theta)}{\partial x}$, representing the output's response to a change in input. This term is widely used in the studies to craft adversarial examples.

b) Convolutional neural networks

In computer vision tasks, convolutional neural networks[1] is one of the most widely used models. CNN models aggregate the local features from the image to learn the representations of image objects. CNN models can be viewed as a sparse-version of fully connected neural networks: Most of the weights between layers are zero. Its training algorithm or gradients calculation can also be inherited from fully connected neural networks.

c) Graph convolutional networks (GCN)

The work of graph convolutional networks introduced by Kipf and Welling[7] became a popular node classification model for graph data. The idea of graph convolutional networks is similar to CNN: It aggregates the information from neighbor nodes to learn representations for each node v, and outputs the score $F(v, X)$ for prediction:

$$H^{(0)} = X; \quad H^{(l+1)} = \sigma(\hat{A}H^{(l)}W^l)$$

where X denotes the input graph's feature matrix, and $\hat{A}$ depends on graph degree matrix and adjacency matrix.

d) Recurrent neural networks (RNN)

Recurrent neural networks are very useful for tackling sequential data. As a result, they are widely used in natural language processing. The RNN models, especially long short term memory based models (LSTM)[4], are able to store the previous time information in memory, and exploit useful information from previous sequence for next-step prediction.

2.2 Security evaluation

We also need to evaluate the model's resistance to adversarial examples. "Robustness" and "adversarial risk" are two terms used to describe this resistance of DNN models to one single sample, and the total population, respectively.

2.2.1 Robustness

Definition 1. Minimal perturbation: Given the classi-

fier F and data (x, y), the adversarial perturbation has the least norm (the most unnoticeable perturbation):

$$\delta_{\min} = \arg\min_{\delta} ||\delta|| \quad \text{s.t.} \quad F(x+\delta) \neq y.$$

Here, $||\cdot||$ usually refers to l_p norm.

Definition 2. Robustness: The norm of minimal perturbation:

$$r(x, F) = ||\delta_{\min}||.$$

Definition 3. Global robustness: The expectation of robustness over the whole population D:

$$\rho(F) = \mathop{\mathbb{E}}_{x\sim\mathcal{D}} r(x, F).$$

The minimal perturbation can find the adversarial example which is most similar to x under the model F. Therefore, the larger $r(x, F)$ or $\rho(F)$ is, the adversary needs to sacrifice more similarity to generate adversarial samples, implying that the classifier F is more robust or safe.

2.2.2 Adversarial risk (loss)

Definition 4. Most-adversarial example: Given the classifier F and data x, the sample x_{adv} with the largest loss value in $x's$ ϵ-neighbor ball:

$$x_{adv} = \arg\max_{x'} \mathcal{L}(x', F) \quad \text{s.t.} \quad ||x' - x|| \leq \epsilon.$$

Definition 5. Adversarial loss: The loss value for the most-adversarial example:

$$\mathcal{L}_{adv}(x) = \mathcal{L}(x_{adv}) = \max_{||x'-x||<\epsilon} \mathcal{L}(\theta, x', y).$$

Definition 6. Global adversarial loss: The expectation of the loss value on x_{adv} over the data distribution $\mathcal{D}$:

$$\mathcal{R}_{adv}(F) = \mathop{\mathbb{E}}_{x\sim\mathcal{D}} \max_{||x'-x||<\epsilon} \mathcal{L}(\theta, x', y). \tag{1}$$

The most-adversarial example is the point where the model is most likely to be fooled in the neighborhood of x. A lower loss value $\mathcal{L}_{adv}$ indicates a more robust model F.

2.2.3 Adversarial risk versus risk

The definition of adversarial risk is drawn from the definition of classifier risk (empirical risk):

$$\mathcal{R}(F) = \mathop{\mathbb{E}}_{x\sim\mathcal{D}} \mathcal{L}(\theta, x, y).$$

Risk studies a classifier's performance on samples from natural distribution $\mathcal{D}$. Whereas, adversarial risk from (1) studies a classifier's performance on adversarial example x'. It is important to note that x' may not necessarily follow the distribution $\mathcal{D}$. Thus, the studies on adversarial examples are different from these on model generalization. Moreover, a number of studies reported the relation between these two properties[25–28]. From our clarification, we hope that our audience get the difference and relation between risk and adversarial risk, and the importance of studying adversarial countermeasures.

2.3 Notations

With the aforementioned definitions, Table 1 lists the notations which will be used in the subsequent sections.

Table 1 Notations

Notations	Description
x	Victim data sample
x'	Perturbed data sample
δ	Perturbation
$B_\epsilon(x)$	l_p-distance neighbor ball around x with radius ϵ
$\mathcal{D}$	Natural data distribution
$\|\|\cdot\|\|_p$	l_p norm
y	Sample x's ground truth label
t	Target label t
$\mathcal{Y}$	Set of possible labels. Usually we assume there are m labels
C	Classifier whose output is a label: $C(x) = y$
F	DNN model which outputs a score vector: $F(x) \in [0, 1]^m$
Z	Logits: last layer outputs before softmax: $F(x) = softmax(Z(x))$
σ	Activation function used in neural networks
θ	Parameters of the model F
$\mathcal{L}$	Loss function for training. We simplify $\mathcal{L}(F(x), y)$ in the form $\mathcal{L}(\theta, x, y)$.

3 Generating adversarial examples

In this section, we introduce main methods for generating adversarial examples in image classification domain. Studying adversarial examples in the image domain is considered to be essential because: 1) Perceptual similarity between fake and benign images is intuitive to observers, and 2) image data and image classifiers have simpler structure than other domains, like graph or audio. Thus, many studies concentrate on attacking image classifiers as a standard case. In this section, we assume the image classifiers refer to fully connected neural networks and convolutional neural networks[1]. The most common datasets used in these studies include 1) handwritten letter images dataset MNIST, 2) CIFAR10 object dataset and 3) ImageNet[29]. Next, we go through some main methods

used to generate adversarial image examples for evasion attack (white-box, black-box, grey-box, physical-world attack), and poisoning attack settings. Note that we also summarize all the attack methods in Table A in Appendix A.

3.1 White-box attacks

Generally, in a white-box attack setting, when the classifier C (model F) and the victim sample (x, y) are given to the attacker, his goal is to synthesize a fake image x' perceptually similar to original image x but that can mislead the classifier C to give wrong prediction results. It can be formulated as

$$\text{find } x' \text{ satisfying } ||x' - x|| \leq \epsilon, \text{ such that } C(x') = t \neq y$$

where $||\cdot||$ measures the dissimilarity between x' and x, which is usually l_p norm. Next, we will go through main methods to realize this formulation.

3.1.1 Biggio's attack

Biggio et al.[22] firstly generates adversarial examples on MNIST data set targeting conventional machine learning classifiers like SVMs and 3-layer fully-connected neural networks.

It optimizes the discriminant function to mislead the classifier. For example, on MNIST dataset, for a linear SVM classifier, its discriminant function $g(x) = \langle w, x\rangle + b$, will mark a sample x with positive value $g(x) > 0$ to be in class "3", and x with $g(x) \leq 0$ to be in class "not 3". An example of this attack is in Fig. 2.

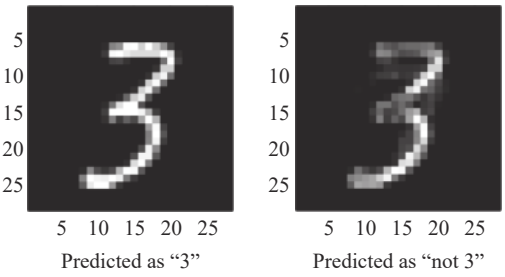


Fig. 2 Biggio's attack on SVM classifier for letter recognition (Image credit: Biggio et al.[22])

Suppose we have a sample x which is correctly classified to be "3". For this model, Biggio's attack crafts a new example x' to minimize the discriminant value $g(x')$ while keeping $||x' - x||_1$ small. If $g(x')$ is negative, the sample is classified as "not 3", but x' is still close to x, so the classifier is fooled. The studies about adversarial examples for conventional machine learning models[19, 22, 24], inspired studies on safety issues of deep learning models.

3.1.2 Szegedy's limited-memory BFGS (L-BFGS) attack

The work of Szegedy et al.[8] is the first to attack deep neural network image classifiers. They formulate their optimization problem as a search for minimal distorted adversarial example x', with the objective:

$$\begin{aligned} &\min \quad ||x - x'||_2^2 \\ &\text{s.t.} \quad C(x') = t \ \text{ and } \ x' \in [0,1]^m. \end{aligned} \tag{2}$$

Szegedy et al. approximately solve this problem by introducing the loss function, which results the following objective:

$$\min c||x - x'||_2^2 + \mathcal{L}(\theta, x', t), \quad \text{s.t.} \quad x' \in [0,1]^m.$$

In the optimization objective of this problem, the first term imposes the similarity between x' and x. The second term encourages the algorithm to find x' which has a small loss value to label t, so the classifier C will be very likely to predict x' as t. By continuously changing the value of constant c, they can find an x' which has minimum distance to x, and at the same time fool the classifier C. To solve this problem, they implement the L-BFGS[30] algorithm.

3.1.3 Fast gradient sign method (FGSM)

Goodfellow et al.[9] introduced an one-step method to fast generate adversarial examples. Their formulation is

$$\begin{aligned} x' &= x + \epsilon \text{sgn}(\nabla_x \mathcal{L}(\theta, x, y)), \quad \text{non-target} \\ x' &= x - \epsilon \text{sgn}(\nabla_x \mathcal{L}(\theta, x, t)), \quad \text{target on } t. \end{aligned}$$

For targeted attack setting, this formulation can be seen as a one-step of gradient descent to solve the problem:

$$\begin{aligned} &\min \quad \mathcal{L}(\theta, x', t) \\ &\text{s.t.} \quad ||x' - x||_\infty \leq \epsilon \ \text{ and } \ x' \in [0,1]^m. \end{aligned} \tag{3}$$

The objective function in (3) searches the point which has the minimum loss value to label t in x's ϵ-neighbor ball, which is the location where model F is most likely to predict it to the target class t. In this way, the one-step generated sample x' is also likely to fool the model. An example of FGSM-generated example on ImageNet is shown in Fig. 1.

Compared to the iterative attack in Section 3.1.2, FGSM is fast in generating adversarial examples, because it only involves calculating one back-propagation step. Thus, FGSM addresses the demands of tasks that need to generate a large amount of adversarial examples. For example, adversarial training[31], uses FGSM to produce adversarial samples for all samples in training set.

3.1.4 DeepFool

In DeepFool[32], the authors study a classifier F's decision boundary around data point x. They try to find a path such that x can go beyond the decision boundary, as shown in Fig. 3, so that the classifier will give a different prediction for x. For example, to attack x_0 (true label is digit 4) to digit class 3, the decision boundary is described as $\mathcal{F}_3 = \{z : F(x)_4 - F(x)_3 = 0\}$. We denote $f(x) = F(x)_4 - F(x)_3$ for short. In each attacking step, it

linearizes the decision boundary hyperplane using Taylor expansion $\mathcal{F}_3' = \{x : f(x) \approx f(x_0) + \langle \nabla_x f(x_0) \cdot (x - x_0) \rangle = 0\}$, and calculates the orthogonal vector ω from x_0 to plane $\mathcal{F}_3'$. This vector ω can be the perturbation that makes x_0 go beyond the decision boundary $\mathcal{F}_3$. By moving along the vector ω, the algorithm is able to find the adversarial example x_0' that is classified to class 3.

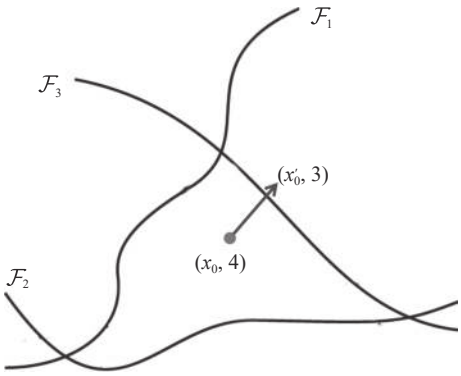


Fig. 3 Decision boundaries: the hyperplane $\mathcal{F}_\infty$ ($\mathcal{F}_\in$ or $\mathcal{F}_\ni$) separates the data points belonging to class 4 and class 1 (class 2 or 3). The sample x_0 crosses the decision boundary $\mathcal{F}_\ni$, so the perturbed data x_0' is classified as class 3. (Image credit: Moosavi-Dezfooli et al.[32])

The experiments of DeepFool[32] shows that for common DNN image classifiers, almost all test samples are very close to their decision boundary. For a well-trained LeNet classifier on MNIST dataset, over 90% of test samples can be attacked by small perturbations whose l_∞ norm is below 0.1 where the total range is [0, 1]. This suggests that the DNN classifiers are not robust to small perturbations.

3.1.5 Jacobian-based saliency map attack

Jacobian-based saliency map attack (JSMA)[33] introduced a method based on calculating the Jacobian matrix of the score function F. It can be viewed as a greedy attack algorithm by iteratively manipulating the pixel which is the most influential to the model output.

The authors used the Jacobian matrix $J_F(x) = \dfrac{\partial F(x)}{\partial x} = \left\{ \dfrac{\partial F_j(x)}{\partial x_i} \right\}_{i \times j}$ to model $F(x)$'s change in response to the change of its input x. For a targeted attack setting where the adversary aims to craft an x' that is classified to the target class t, they repeatedly search and manipulate pixel x_i whose increase (decrease) will cause $F_t(x)$ to increase or decrease $\sum_{j \neq t} F_j(x)$. As a result, for x, the model will give it the largest score to label t.

3.1.6 Basic iterative method (BIM)/Projected gradient descent (PGD) attack

The basic iterative method was first introduced by Kurakin et al.[15, 31] It is an iterative version of the one-step attack FGSM in Section 3.1.3. In a non-targeted setting, it gives an iterative formulation to craft x':

$$x_0 = x; \quad x^{t+1} = Clip_{x,\epsilon}(x^t + \alpha \text{sgn}(\nabla_x \mathcal{L}(\theta, x^t, y))).$$

Here, $Clip$ denotes the function to project its argument to the surface of x's ϵ-neighbor ball $B_\epsilon(x) : \{x' : ||x' - x||_\infty \leq \epsilon\}$. The step size α is usually set to be relatively small (e.g., 1 unit of pixel change for each pixel), and step numbers guarantee that the perturbation can reach the border (e.g., $step = \dfrac{\epsilon}{\alpha} + 10$). This iterative attacking method is also known as projected gradient method (PGD) if the algorithm is added by a random initialization on x, used in work [14].

This BIM (or PGD) attack heuristically searches the samples x' which have the largest loss value in the l_∞ ball around the original sample x. This kind of adversarial examples are called "most-adversarial" examples: They are the sample points which are most aggressive and most-likely to fool the classifiers, when the perturbation intensity (its l_p norm) is limited. Finding these adversarial examples is helpful to find the weaknesses of deep learning models.

3.1.7 Carlini & Wagner's attack

Carlini and Wagner's attack[34] counterattacks the defense strategy[12] which were shown to be successful against FGSM and L-BFGS attacks. C&W's attack aims to solve the same problem as defined in L-BFGS attack (Section 3.1.2), namely trying to find the minimally-distorted perturbation (2).

The authors solve the problem (2) by instead solving:

$$\min \ ||x - x'||_2^2 + c \cdot f(x', t), \quad \text{s.t.} \ \ x' \in [0, 1]^m$$

where f is defined as $f(x', t) = (\max_{i \neq t} Z(x')_i - Z(x')_t)^+$. Minimizing $f(x', t)$ encourages the algorithm to find an x' that has larger score for class t than any other label, so that the classifier will predict x' as class t. Next, applying a line search on constant c, we can find the x' that has the least distance to x.

The function $f(x, y)$ can also be viewed as a loss function for data (x, y): It penalizes the situation where there are some labels i with scores $Z(x)_i$ larger than $Z(x)_y$. It can also be called margin loss function.

The only difference between this formulation and the one in L-BFGS attack (Section 3.1.2) is that C&W's attack uses margin loss $f(x, t)$ instead of cross entropy loss $\mathcal{L}(x, t)$. The benefit of using margin loss is that when $C(x') = t$, the margin loss value $f(x', t) = 0$, the algorithm will directly minimize the distance from x' to x. This procedure is more efficient for finding the minimally distorted adversarial example.

The authors claim their attack is one of the strongest attacks, breaking many defense strategies which were shown to be successful. Thus, their attacking method can be used as a benchmark to examine the safety of DNN classifiers or the quality of other adversarial examples.

3.1.8 Ground truth attack

Attacks and defenses keep improving to defeat each

other. In order to end this stalemate, the work of Carlini et al.[35] tries to find the "provable strongest attack". It can be seen as a method to find the theoretical minimally-distorted adversarial examples.

This attack is based on Reluplex[36], an algorithm for verifying the properties of neural networks. It encodes the model parameters F and data (x, y) as the subjects of a linear-like programming system, and then solve the system to check whether there exists an eligible sample x' in x's neighbor $B_\epsilon(x)$ that can fool the model. If we keep reducing the radius ϵ of search region $B_\epsilon(x)$ until the system determines that there does not exist such an x' that can fool the model, the last found adversarial example is called the ground truth adversarial example, because it has been proved to have least dissimilarity with x.

The ground-truth attack is the first work to seriously calculate the exact robustness (minimal perturbation) of classifiers. However, this method involves using a satisfiability modulo theories (SMT) solver (a complex algorithm to check the satisfiability of a series of theories), which will make it slow and not scalable to large networks. More recent works[37, 38], have improved the efficiency of ground-truth attack.

3.1.9 Other l_p attacks

Previous studies are mostly focused on l_2 or l_∞ norm-constrained perturbations. However, there are other papers which consider other types of l_p attacks.

1) One-pixel attack[39] studies similar problem as in Section 3.1.2, but constrains the perturbation's l_0 norm. Constraining l_0 norm of the perturbation $x' - x$ will limit the number of pixels that are allowed to be changed. Their work shows that: On dataset CIFAR10, for a well-trained CNN classifier (e.g., VGG16, which has 85.5% accuracy on test data), most of the testing samples (63.5%) can be attacked by changing the value of only one pixel in a non-targeted setting. This also demonstrates the poor robustness of deep learning models.

2) EAD: Elastic-net attack[40] also studies a similar problem as in Section 3.1.2, but constrains the perturbations l_1 and l_2 norm together. As shown in their experimental work[41], some strong defense models that aim to reject l_∞ and l_2 norm attacks[14] are still vulnerable to the l_1-based Elastic-net attack.

3.1.10 Universal attack

Previous methods only consider one specific targeted victim sample x. However, the work [42] devises an algorithm that successfully mislead a classifier's decision on almost all testing images. They try to find a perturbation δ satisfying:

1) $||\delta||_p \leq \epsilon$.

2) $\mathop{\mathbb{P}}_{x\sim D(x)}(C(x+\delta) \neq C(x)) \leq 1-\sigma$.

This formulation aims to find a perturbation δ such that the classifier gives wrong decisions on most of the samples. In their experiments, for example, they successfully find a perturbation that can attack 85.4% of the test samples in the ILSVRC 2012[43] dataset under a ResNet-152[2] classifier.

The existence of "universal" adversarial examples reveals a DNN classifier's inherent weakness on all of the input samples. As claimed in work [42], it may suggest the property of geometric correlation among the high-dimensional decision boundary of classifiers.

3.1.11 Spatially transformed attack

Traditional adversarial attack algorithms directly modify the pixel value of an image, which changes the image's color intensity. Spatial attack[44] devises another method, called a spatially transformed attack. They perturb the image by doing slight spatial transformation: They translate, rotate and distort the local image features slightly. The perturbation is small enough to evade human inspection but can fool the classifiers. One example is in Fig. 4.

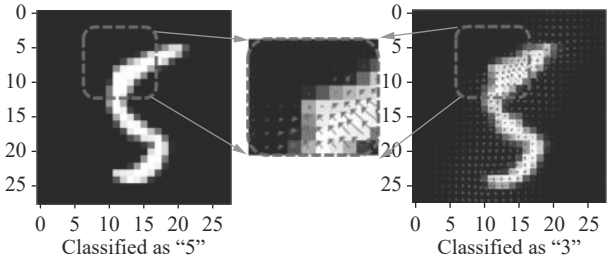


Fig. 4 Top part of digit "5" is perturbed to be "thicker". For the image which was correctly classified as "5", after distortion is now classified as "3".

3.1.12 Unrestricted adversarial examples

Previous attack methods only consider adding unnoticeable perturbations into images. However, the work [45] devised a method to generate unrestricted adversarial examples. These samples do not necessarily look exactly the same as the victim samples, but are still legitimate samples for human eyes and can fool the classifier. Previous successful defense strategies that target perturbation-based attacks fail to recognize them.

In order to attack given classifier C, Odena et al.[46] pretrained an auxiliary classifier generative adversarial network (AC-GAN), so they can generate one legitimate sample x from a noise vector z^0 from class y. Then, to craft an adversarial example, they will find a noise vector z near z^0, but require that the output of AC-GAN generator $\mathcal{G}(z)$ be wrongly classified by victim model C. Because z is near z^0 in latent space of the AC-GAN, its output should belong to the same class y. In this way, the generated sample $\mathcal{G}(z)$ is different from x, misleading classifier F, but it is still a legitimate sample.

3.2 Physical world attack

All the previously introduced attack methods are applied digitally, where the adversary supplies input im-

ages directly to the machine learning model. However, this is not always the case for some scenarios, like those that use cameras, microphones or other sensors to receive the signals as input. In this case, can we still attack these systems by generating physical-world adversarial objects? Recent works show such attacks do exist. For example, the work [20] attached stickers to road signs that can severely threaten autonomous car′s sign recognizer. These kinds of adversarial objects are more destructive for deep learning models because they can directly challenge many practical applications of DNN, such as face recognition, autonomous vehicle, etc.

3.2.1 Exploring adversarial examples in physical world

In the work [15], the authors explore the feasibility of crafting physical adversarial objects, by checking whether the generated adversarial images (FGSM, BIM) are "robust" under natural transformation (such as changing viewpoint, lighting, etc). Here, "robust" means the crafted images remain adversarial after the transformation. To apply the transformation, they print out the crafted images, and let test subjects use cellphones to take photos of these printouts. In this process, the shooting angle or lighting environment are not constrained, so the acquired photos are transformed samples from previously generated adversarial examples. The experimental results demonstrate that after transformation, a large portion of these adversarial examples, especially those generated by FGSM, remain adversarial to the classifier. These results suggest the possibility of physical adversarial objects which can fool the sensor under different environments.

3.2.2 Eykholt′s attack on road signs

The work [20], shown in Fig. 5, crafts physical adversarial objects, by "contaminating" road signs to mislead road sign recognizers. They achieve the attack by putting stickers on the stop sign in the desired positions.

The author′s approach consist of: 1) Implement l_1-norm based attack (those attacks that constrain $||x' - x||_1$) on digital images of road signs to roughly find the region to perturb (l_1 attacks render sparse perturbation, which helps to find attack location). These regions will later be the location of stickers. 2) Concentrating on the regions found in step 1, use an l_2-norm based attack to generate the color for the stickers. 3) Print out the perturbation found in Steps 1 and 2, and stick them on road sign. The perturbed stop sign can confuse an autonomous vehicle from any distance and viewpoint.

Fig. 5 Attacker puts some stickers on a road sign to confuse an autonomous vehicle′s road sign recognizer from any viewpoint (Image credit: Eykholt et al.[20])

3.2.3 Athalye′s 3D adversarial object

In the work [47], authors report the first work which successfully crafted physical 3D adversarial objects. As shown in Fig. 6, the authors use 3D-printing to manufacture an "adversarial" turtle. To achieve their goal, they implement a 3D rendering technique. Given a textured 3D object, they first optimize the object′s texture such that the rendering images are adversarial from any viewpoint. In this process, they also ensure that the perturbation remains adversarial under different environments: camera distance, lighting conditions, rotation and background. After finding the perturbation on 3D rendering, they print an instance of the 3D object.

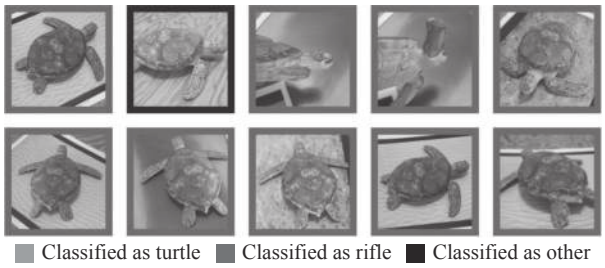


Fig. 6 Image classifier fails to correctly recognize the adversarial object, but the original object can be correctly predicted with 100% accuracy (Image credit: Athalye et al.[47])

3.3 Black-box attacks

3.3.1 Substitute model

The work [48] was the first to introduce an effective algorithm to attack DNN classifiers, under the condition that the adversary has no access to the classifier′s parameters or training set (black-box). An adversary can only feed input x to obtain the output label y from the classifier. Additionally, the adversary may have only partial knowledge about: 1) the classifier′s data domain (e.g., handwritten digits, photographs, human faces) and 2) the architecture of the classifier (e.g., CNN, RNN).

The authors in the work [48] exploits the "transferability" (Section 5.3) property of adversarial examples: a sample x' can attack F_1, it is also likely to attack F_2, which has similar structure to F_1. Thus, the authors introduce a method to train a substitute model F' to imitate the target victim classifier F, and then craft the adversarial example by attacking substitute model F'. The main steps are below:

1) Synthesize substitute training dataset

Make a "replica" training set. For example, to attack a victim classifier for hand-written digits recognition task, make an initial substitute training set by: a) requiring samples from test set; or b) handcrafting samples.

2) Training the substitute model

Feed the substitute training dataset X into the victim classifier to obtain their labels Y. Choose one substitute DNN model to train on (X, Y) to get F'. Based on the attacker′s knowledge, the chosen DNN should have similar structure to the victim model.

3) Dataset augmentation

Augment the dataset (X, Y) and retrain the substitute model F' iteratively. This procedure helps to increase the diversity of the replica training set and improve the accuracy of substitute model F'.

4) Attacking the substitute model

Utilize the previously introduced attack methods, such as FGSM to attack the model F'. The generated adversarial examples are also very likely to mislead the target model F, by the property of "transferability".

What kind of attack algorithm should we choose to attack substitute model? The success of substitute model black-box attack is based on the "transferability" property of adversarial examples. Thus, during black-box attack, we choose attacks that have high transferability, like FGSM, PGD and momentum-based iterative attacks[49].

3.3.2 ZOO: Zeroth order optimization based black-box attack

Different from the work in Section 3.3.1 where an adversary can only obtain the label information from the classifier, the work [50] assume the attacker has access to the prediction confidence (sscore) from the victim classifier′s output. In this case, there is no need to build the substitute training set and substitute model. Chen et al. give an algorithm to "scrape" the gradient information around victim sample x by observing the changes in the prediction confidence $F(x)$ as the pixel values of x are tuned.

Equation (4) shows for each index i of sample x, we add (or subtract) x_i by h. If h is small enough, we can scrape the gradient information from the output of $F(\cdot)$ by

$$\frac{\partial F(x)}{\partial x_i} \approx \frac{F(x + he_i) - F(x - he_i)}{2h}. \tag{4}$$

Utilizing the approximate gradient, we can apply the attack formulations introduced in Sections 3.1.3 and 3.1.7. The attack success rate of ZOO is higher than substitute model (Section 3.3.1) because it can utilize the information of prediction confidence, instead of solely the predicted labels.

3.3.3 Query-efficient black-box attack

Previously introduced black-box attacks require lots of input queries to the classifier, which may be prohibitive in practical applications. There are some studies on improving the efficiency of generating black-box adversarial examples via a limited number of queries. For example, the authors in work [51] introduced a more efficient way to estimate the gradient information from model outputs. They use natural evolutional strategies[52], which sample the model′s output based on the queries around x, and estimate the expectation of gradient of F on x. This procedure requires fewer queries to the model. Moreover, the authors in work [53] apply a genetic algorithm to search the neighbors of benign image for adversarial examples.

3.4 Semi-white (grey) box attack

3.4.1 Using generative adversarial network (GAN) to generate adversarial examples

The work [54] devised a semi-white box attack framework. It first trained a GAN[55], targeting the model of interest. The attacker can then craft adversarial examples directly from the generative network.

The authors believe the advantage of the GAN-based attack is that it accelerates the process of producing adversarial examples, and makes more natural and more undetectable samples. Later, Deb′s grey box attack[56] uses GAN to generate adversarial faces to evade face recognition software. Their crafted face images appear to be more natural and have barely distinguishable difference from target face images.

3.5 Poisoning attacks

The attacks we have discussed so far are evasion attacks, which are launched after the classification model is trained. Some works instead craft adversarial examples before training. These adversarial examples are inserted into the training set in order to undermine the overall accuracy of the learned classifier, or influence its prediction on certain test examples. This process is called a poisoning attack.

Usually, the adversary in a poisoning attack setting has knowledge about the architecture of the model which is later trained on the poisoned dataset. Poisoning attacks frequently applied to attack graph neural network, because of the GNN′s specific transductive learning procedure. Here, we introduce studies that craft image poisoning attacks.

3.5.1 Biggio′s poisoning attack on SVM

The work [19] introduced a method to poison the training set in order to reduce SVM model′s accuracy. In their setting, they try to figure out a poison sample x_c which, when inserted into the training set, will result in the learned SVM model F_{x_c} having a large total loss on the whole validation set. They achieve this by using incremental learning technique for SVMs[57], which can model the influence of training sample on the learned SVM model.

A poisoning attack based on procedure above is quite successful for SVM models. However, for deep learning models, it is not easy to explicitly figure out the influence of training samples on the trained model. Below we introduce some approaches for applying poisoning attacks on DNN models.

3.5.2 Koh's model explanation

Koh and Liang's explanation study[58] introduce a method to interpret deep neural networks: How would the model's predictions change if a training sample were modified? Their model can explicitly quantify the change in the final loss without retraining the model when only one training sample is modified. This work can be naturally adopted to poisoning attacks by finding those training samples that have large influence on model's prediction.

3.5.3 Poison frogs

"Poison frogs"[59] introduced a method to insert an adversarial image with true label to the training set, in order to cause the trained model to wrongly classify a target test sample. In their work, given a target test sample x_t, whose true label is y_t, the attacker first uses a base sample x_b from class y_b. Then, it solves the objective to find x':

$$x' = \arg\min_x ||Z(x) - Z(x_t)||_2^2 + \beta||x - x_b||_2^2.$$

After inserting the poison sample x' into training set, the new model trained on $X_{train} + \{x\}'$ will classify x' as class y_b, because of the small distance between x' and x_b. Using a new trained model to predict x_t, the objective of x' forces the score vector of x_t and x' to be close. Thus, x' and x_t will have the same prediction outcome. In this way, the new trained model will predict the target sample x_t as class y_b.

4 Countermeasures against adversarial examples

In order to protect the security of deep learning models, different strategies have been considered as countermeasures against adversarial examples. There are basically three main categories of these countermeasures:

1) Gradient masking/Obfuscation

Since most attack algorithms are based on the gradient information of the classifier, masking or hiding the gradients will confound the adversaries.

2) Robust optimization

Re-learning a DNN classifier's parameters can increase its robustness. The trained classifier will correctly classify the subsequently generated adversarial examples.

3) Adversarial examples detection

Study the distribution of natural/benign examples, detect adversarial examples and disallow their input into the classifier.

4.1 Gradient masking/Obfuscation

Gradient masking/Obfuscation refers to the strategy where a defender deliberately hides the gradient information of the model, in order to confuse the adversaries, since most attack algorithms are based on the classifier's gradient information.

4.1.1 Defensive distillation

"Distillation", first introduced by Hinton et al.[60], is a training technique to reduce the size of DNN architectures. It fulfills its goal by training a smaller-size DNN model on the logits (outputs of the last layer before softmax).

The work [12] reformulate the procedure of distillation to train a DNN model that can resist adversarial examples, such as FGSM, Szegedy's L-BFGS attack or DeepFool. They design their training process as:

1) Train a network F on the given training set (X, Y) by setting the temperature[1] of the softmax to T.

2) Compute the scores (after softmax) given by $F(X)$, again evaluating the scores at temperature T.

3) Train another network F'_T using softmax at temperature T on the dataset with soft labels $(X, F(X))$. We refer the model F'_T as the distilled model.

4) During prediction on test data X_{test} (or adversarial examples), use the distilled network F'_T but use softmax at temperature 1, which is denoted as F'_1.

Carlini and Wagner[34] explain why this algorithm works: When we train a distilled network F'_T at temperature T and test it at temperature 1, we effectively cause the inputs to the softmax to become larger by a factor of T. Let us say $T = 100$, the logits $Z(\cdot)$ for sample x and its neighbor points x' will be 100 times larger, which will result the softmax function $F_1(\cdot) = \text{softmax}(\text{Z}(\cdot), 1)$ outputting a score vector like $(\epsilon, \epsilon, \cdots, 1-(m-1)\epsilon, \epsilon, \cdots, \epsilon)$, where the target output class has a score extremely close to 1, and all other classes have scores close to 0. In practice, the value of ϵ is so small that its 32-bit floating-point value for computer is rounded to 0. In this way, the computer cannot find the gradient of score function F'_1, which inhibits the gradient-based attacks.

4.1.2 Shattered gradients

Some studies, such as [61, 62], try to protect the model by preprocessing the input data. They add a non-smooth or non-differentiable preprocessor $g(\cdot)$ and then train a DNN model f on $g(X)$. The trained classifier $f(g(\cdot))$ is not differentiable in term of x, causing the failure of adversarial attacks.

For example, Thermometer encoding[61] uses a prepro-

[1]Note that the softmax function at a temperature T means:

$$softmax(x, T)_i = \frac{\mathrm{e}^{\frac{x_i}{T}}}{\sum_j \mathrm{e}^{\frac{x_j}{T}}}, \text{ where } i = 0, 2, \cdots, K-1.$$

cessor to discretize an image's pixel value x_i into a l-dimensional vector $\tau(x_i)$. (e.g., when $l = 10$, $\tau(0.66) = 1\,111\,110\,000$). The vector $\tau(x_i)$ acts as a "thermometer" to record the pixel x_i's value. A DNN model is later trained on these vectors. Another work [62] studies a number of image processing tools, such as image cropping, compressing, total-variance minimization and super-resolution[63], to determine whether these techniques help to protect the model against adversarial examples. All these approaches block up the smooth connection between the model's output and the original input samples, so the attacker cannot easily find the gradient $\frac{\partial F(x)}{\partial x}$ for attacking.

4.1.3 Stochastic/Randomized gradients

Some defense strategies try to randomize the DNN model in order to confound the adversary. For instance, we train a set of classifiers $s = \{F_t : t = 1, 2, \cdots, k\}$. During evaluation on data x, we randomly select one classifier from the set s and predict the label y. Because the adversary has no idea which classifier is used by the prediction model, the attack success rate will be reduced.

Some examples of this strategy include the work [64], who randomly drop some neurons of each layer of the DNN model, and the work [65], who resize the input images to a random size and pad zeros around the input image.

4.1.4 Exploding & vanishing gradients

Both PixelDefend[66] and Defense-GAN[67] suggest using generative models to project a potential adversarial example onto the benign data manifold before classifying them. While PixelDefend uses PixelCNN generative model[68], Defense-GAN uses a GAN architecture[5]. The generative models can be viewed as a purifier that transforms adversarial examples into benign examples.

Both of these methods consider adding a generative network before the classifier DNN, which will cause the final classification model be an extremely deep neural network. The underlying reason that these defenses succeed is because: The cumulative product of partial derivatives from each layer will cause the gradient $\frac{\partial \mathcal{L}(x)}{\partial x}$ to be extremely small or irregularly large, which prevents the attacker accurately estimating the location of adversarial examples.

4.1.5 Gradient masking/Obfuscation methods are not safe

In the work Carlini and Wagner's attack[34], they show the method of "Defensive Distillation" (Section 4.1.1) is still vulnerable to their adversarial examples. In the study [13], the authors devised different attacking algorithms to break gradient masking/obfuscation defending strategies (Sections 4.1.2 – 4.1.4).

The main weakness of the gradient masking strategy is that: It can only "confound" the adversaries; it cannot eliminate the existence of adversarial examples.

4.2 Robust optimization

Robust optimization methods aim to improve the classifier's robustness (Section 2.2) by changing DNN model's manner of learning. They study how to learn model parameters that can give promising predictions on potential adversarial examples. In this field, the works majorly focus on: 1) learning model parameters θ^* to minimize the average adversarial loss: (Section 2.2.2)

$$\theta^* = \arg\min_{\theta\in\Theta} \mathbb{E}_{x\sim\mathcal{D}} \max_{||x'-x||\leq\epsilon} \mathcal{L}(\theta, x', y) \tag{5}$$

or 2) learning model parameters θ^* to maximize the average minimal perturbation distance: (Section 2.2.1)

$$\theta^* = \arg\max_{\theta\in\Theta} \mathbb{E}_{x\sim\mathcal{D}} \min_{C(x')\neq y} ||x' - x||. \tag{6}$$

Typically, a robust optimization algorithm should have a prior knowledge of its potential threat or potential attack (adversarial space $\mathcal{D}$). Then, the defenders build classifiers which are safe against this specific attack. For most of the related works[9, 14, 15], they aim to defend against adversarial examples generated from small l_p (specifically l_∞ and l_2) norm perturbation. Even though there is a chance that these defenses are still vulnerable to attacks from other mechanisms, (e.g., spatial attack[44]), studying the security against l_p attack is fundamental and can be generalized to other attacks.

In this section, we concentrate on defense approaches using robustness optimization against l_p attacks. We categorize the related works into three groups: 1) regularization methods, 2) adversarial (re)training and 3) certified defenses.

4.2.1 Regularization methods

Some early studies on defending against adversarial examples focus on exploiting certain properties that a robust DNN should have in order to resist adversarial examples. For example, Szegedy et al.[8] suggest that a robust model should be stable when its inputs are distorted, so they turn to constrain the Lipschitz *constant* to impose this "stability" of model output. Training on these regularizations can sometimes heuristically help the model be more robust.

1) Penalize layer's Lipschitz constant

When Szegydy et al.[8] first claimed the vulnerability of DNN models to adversarial examples, they suggested adding regularization terms on the parameters during training, to force the trained model be stable. It suggested constraining the Lipschitz constant L_k between any two layers:

$$\forall x, \delta, \quad ||h_k(x; W_k) - h_k(x + \delta; W_k)|| \leq L_k||\delta||$$

so that the outcome of each layer will not be easily influenced by the small distortion of its input. The work

Parseval networks[69] formalized this idea, by claiming that the model's adversarial risk (5) is right dependent on this instability L_k:

$$\mathop{\mathbb{E}}_{x\sim\mathcal{D}}\ \mathcal{L}_{adv}(x) \leq \mathop{\mathbb{E}}_{x\sim\mathcal{D}}\ \mathcal{L}(x)+ \mathop{\mathbb{E}}_{x\sim\mathcal{D}}[\max_{||x'-x||\leq\epsilon}|\mathcal{L}(F(x'),y)-\mathcal{L}(F(x),y)|] \leq \mathop{\mathbb{E}}_{x\sim\mathcal{D}}\ \mathcal{L}(x)+\lambda_p\prod_{k=1}^{K}L_k$$

where λ_p is the Lipschitz constant of the loss function. This formula states that during the training process, penalizing the large instability for each hidden layer can help to decrease the adversarial risk of the model, and consequently increase the robustness of model. The idea of constraining instability also appears in the study [70] for semi-supervised, and unsupervised defenses.

2) Penalize layer's partial derivative

The study [71] introduced a deep contractive network algorithm to regularize the training. It was inspired by the contractive autoencoder[72], which was introduced to denoise the encoded representation learning. The deep contractive network suggests adding a penalty on the partial derivatives at each layer into the standard back-propagation framework, so that the change of the input data will not cause large change on the output of each layer. Thus, it becomes difficult for the classifier to give different predictions on perturbed data samples.

4.2.2 Adversarial (re)training

1) Adversarial training with FGSM

Goodfellow's FGSM attack[9] were the first to suggest feeding generated adversarial examples into the training process. By adding the adversarial examples with true label (x', y) into the training set, the training set will tell the classifier that x' belongs to class y, so that the trained model will correctly predict the label of future adversarial examples.

In the work [9], they use non-targeted FGSM (Section 3.1.3) to generate adversarial examples x' for the training dataset:

$$x' = x + \epsilon\text{sgn}(\nabla_x\mathcal{L}(\theta, x, y)).$$

By training on benign samples augmented with adversarial examples, they increase the robustness against adversarial examples generated by FGSM.

The scalaed adversarial training[15] changes the training strategy of this method so that the model can be scaled to larger dataset such as ImageNet. They suggest using batch normalization[73] will improve the efficiency of adversarial training. We give a short sketch of their algorithm in Algorithm 1.

The trained classifier has good robustness on FGSM attacks, but is still vulnerable to iterative attacks. Later, the study [21] argues that this defense is also vulnerable to single-step attacks. Adversarial training with FGSM will cause gradient obfuscation (Section 4.1), where there is an extreme non-smoothness of the trained classifier F near the test sample x. Refer to Fig. 7 as an illustration of the non-smooth property of FGSM trained classifier.

Algorithm 1. Adversarial training with FGSM by batches

Randomly initialize network F

Repeat

1) Read minibatch $B = \{x^1, \cdots, x^m\}$ from training set

2) Generate k adversarial examples $\{x^1_{adv}, \cdots, x^k_{adv}\}$ for corresponding benign examples using current state of the network F.

3) Update $B' = \{x^1_{adv}, \cdots, x^k_{adv}, x^{k+1}, \cdots, x^m\}$

Do one training step of network F using minibatch B'

until training converged.

2) Adversarial training with PGD

The PGD adversarial training[14] suggests using projected gradient descent attack (Section 3.1.6) for adversarial training, instead of using single-step attacks like FGSM. The PGD attacks (Section 3.1.6) can be seen as a heuristic method to find the "most adversarial" example:

$$x_{adv} = \arg\max_{x'\in B_\epsilon(x)}\mathcal{L}(x', F) \tag{7}$$

in the l_∞ ball around x: $B_\epsilon(x)$. Here, the most-adversarial example x_{adv} is the location where the classifier F is most likely to be misled. When training the DNN model on these most-adversarial examples, it actually solves the problem of learning model parameters θ that minimize the adversarial loss (5). If the trained model has small loss value on these most-adversarial examples, the model is safe at everywhere in x's neighbor ball $B_\epsilon(x)$.

One thing to note is: This method trains the model only on adversarial examples, instead of a mix of benign and adversarial examples. The training algorithm is shown Algorithm 2.

The trained model under this method demonstrates good robustness against both single-step and iterative attacks on MNIST and CIFAR10 dataset. However, this method involves an iterative attack for all the training samples. Thus, the time cost of this adversarial training will be k (using k-step PGD) times as large as the time cost for natural training, and as a consequence, it is hard to scale to large datasets such as ImageNet.

3) Ensemble adversarial training

Ensembler adversarial training[21] introduced their adversarial training method which can protect CNN models against single-step attacks and also apply to large datasets such as ImageNet.

Their main approach is to augment the classifier's training set with adversarial examples crafted from other pre-trained classifiers. For example, if we aim to train a robust classifier F, we can first pre-train classifiers F_1, F_2, and F_3 as references. These models have different hyper-parameters with model F. Then, for each sample x, we

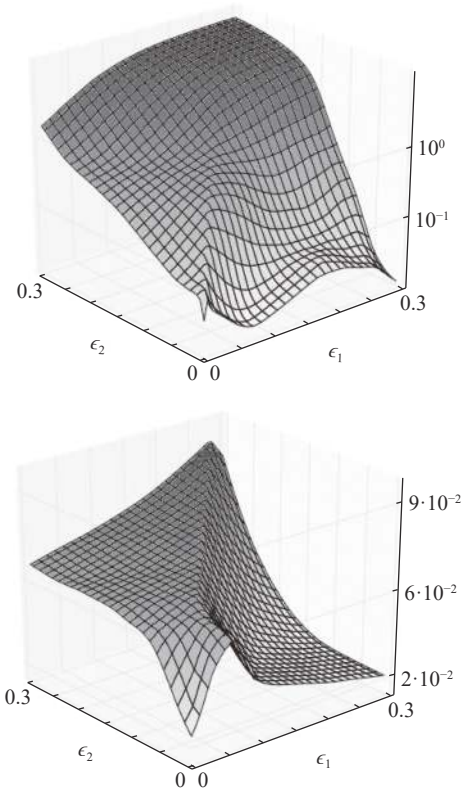


Fig. 7 Illustration of gradient masking for adversarial training via FGSM. It plots the loss function of the trained classifier around x on the grids of gradient direction and another randomly chosen direction. We can see that the gradient poorly approximates the global loss. (Image credit: Tramer et al. [21])

use a single-step attack FGSM to craft adversarial examples on F_1, F_2 and F_3 to get x^1_{adv}, x^2_{adv}, x^3_{adv}. Because of the transferability property (Section 5.3) of the single-step attacks across different models, x^1_{adv}, x^2_{adv}, x^3_{adv} are also likely to mislead the classifier F, which means these samples are a good approximation for the "most adversarial" example (7) for model F on x. Training on these samples together will approximately minimize the adversarial loss in (5).

This ensemble adversarial training algorithm is more time efficient than the methods in Sections 1 and 2, since it decouples the process of model training and generating adversarial examples. The experimental results show that this method can provide robustness against single-step attacks and black-box attacks on ImageNet dataset.

4) Accelerate adversarial training

While it is one of the most promising and reliable defense strategies, adversarial training with PGD attack[14] is generally slow and computationally costly.

The work [74] propose a free adversarial training algorithm which improves the efficiency by reusing the backward pass calculations. In this algorithm, the gradient of the loss to input: $\frac{\partial \mathcal{L}(x+\delta,\theta)}{\partial x}$ and the gradient of the loss to model parameters: $\frac{\partial \mathcal{L}(x+\delta,\theta)}{\partial \theta}$ can be computed together in one back propagation iteration, by sharing the same components of chain rule. Thus, the adversarial training process is highly accelerated. The free adversarial training algorithm is shown in Algorithm 3.

In the work [75], the authors argue that when the model parameters are fixed, the PGD-generated adversarial example is only coupled with the weights of the first layer of DNN. It is based on solving a Pontryagin's maximal principle[76]. Therefore, this work [75] invents an algorithm you only propagate once (YOPO) to reuse the gradient of the loss to the model's first layer output $\frac{\partial \mathcal{L}(x+\delta,\theta)}{\partial Z_1(x)}$ during generating PGD attacks. In this way, YOPO avoids a large amount of times it access the gradient and therefore reduces the computational cost.

Algorithm 2. Adversarial training with PGD

Randomly initialize network F

Repeat

1) Read minibatch $B = \{x^1, \cdots, x^m\}$ from training set

2) Generate m adversarial examples $\{x^1_{adv}, \cdots, x^m_{adv}\}$ by PGD attack using current state of the network F

3) Update $B' = \{x^1_{adv}, \cdots, x^m_{adv}\}$

Do one training step of network F using minibatch B'

until training converged

Algorithm 3. Free adversarial training

Randomly initialize network F

Repeat

1) Read minibatch $B = \{x^1, \cdots, x^m\}$ from training set

2) for $i = 1, \cdots, m$ do

2.1) Update model parameter θ

$g_\theta \leftarrow \mathbb{E}_{(x,y)\in B}[\nabla_\theta \mathcal{L}(x+\delta, y, \theta)]$

$g_{adv} \leftarrow \nabla_x \mathcal{L}(x+\delta, y, \theta)$

$\theta \leftarrow \theta - \alpha g_\theta$

2.2) Generate adversarial examples

$\delta \leftarrow \delta + \epsilon \cdot \mathrm{sgn}(g_{adv})$

$\delta \leftarrow clip(\delta, -\epsilon, \epsilon)$

3) Update minibatch B with adversarial examples $x + \delta$

until training converged

4.2.3 Provable defenses

Adversarial training has been shown to be effective in protecting models against adversarial examples. However, this is still no formal guarantee about the safety of the trained classifiers. We will never know whether there are more aggressive attacks that can break those defenses, so directly applying these adversarial training algorithms in safety-critical tasks would be irresponsible.

As we mentioned in Section 3.1.8, the ground truth attack[35] was the first to introduce a Reluplex algorithm to seriously verify the robustness of DNN models: When the model F is given, the algorithm figures out the exact value of minimal perturbation distance $r(x; F)$. This is to say, the classifier is safe against any perturbations with norm less than this $r(x; F)$. If we apply Reluplex on the whole test set, we can tell what percentage of samples are absolutely safe against perturbations less than norm r_0.

In this way, we gain confidence and reduce the expected risk when building DNN models.

The method of Reluplex seeks to find the exact value of $r(x;F)$ that can verify the model F's robustness on x. Alternately, works such as [77−79], try to find trainable "certificates" $\mathcal{C}(x;F)$ to verify the model robustness. For example, in the work [79], the authors calculate a certificate $\mathcal{C}(x,F)$ for model F on x, which is a lower bound of minimal perturbation distance: $\mathcal{C}(x,F) \leq r(x,F)$. As shown in Fig. 8, the model must be safe against any perturbation with norm limited by $\mathcal{C}(x,F)$. Moreover, these certificates are trainable. Training to optimize these certificates will grant good robustness to the classifier. In this section, we shall briefly introduce some methods to design these certificates.

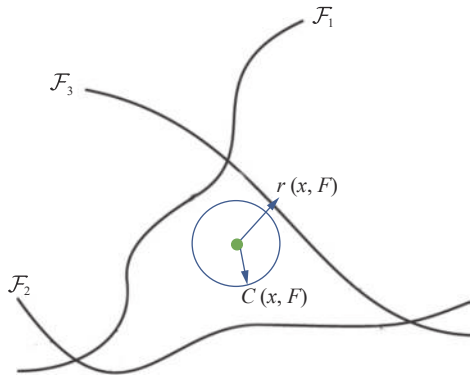


Fig. 8 Derived certificate $\mathcal{C}(\S, \mathcal{F})$ is a lower bound of minimal perturbation distance $\rho(x,F)$. Model is safe in $\mathcal{C}(\S, \mathcal{F})$ ball.

1) Lower bound of minimal perturbation

Hein and Andriushchenko[79] derive a lower bound $\mathcal{C}(x,F)$ for the minimal perturbation distance of F on x based on Cross-Lipschitz theorem:

$$\max_{\epsilon>0} \min\{\min_{i\neq y} \frac{Z_y(x)-Z_i(x)}{\max\limits_{x'\in B_\epsilon(x)} ||\nabla Z_y(x')-\nabla Z_i(x')||}, \epsilon\}.$$

The detailed derivation can be found in their work of [79]. Note that the formulation of $\mathcal{C}(x,F)$ only depends on F and x, and it is easy to calculate for a neural network with one hidden layer. The model F thus can be proved to be safe in the region within distance $\mathcal{C}(x,F)$. Training to maximize this lower bound will make the classifier more robust.

2) Upper bound of adversarial loss

The works proposed by Raghunathan et al.[77] and Wong and Kolter[78] aim to solve the same problem. They try to find an upper bound $\mathcal{U}(x,F)$ which is larger than adversarial loss $\mathcal{L}_{adv}(x,F)$:

$$\mathcal{L}_{adv}(x) = \max_{x'} \{\max_{i\neq y} Z_i(x') - Z_y(x')\}$$
$$\text{s.t.} \quad x' \in B_\epsilon(x). \tag{8}$$

Recall that we introduced in Section 2.2.2, the function $\max_{i\neq y} Z_i(x') - Z_y(x')$ is a type of loss function called margin loss.

The certificate $\mathcal{U}(x,F)$ acts in this way: If $\mathcal{U}(x,F) < 0$, then adversarial loss $\mathcal{L}(x,F) < 0$. Thus, the classifier always gives the largest score to the true label y in the region $B_\epsilon(x)$, and the model is safe in this region. To increase the model's robustness, we should learn parameters that have the smallest $\mathcal{U}$ value, so that more and more data samples will have negative $\mathcal{U}$ values.

The work proposed by Raghunathan et al.[77] uses integration inequalities to derive the certificate and use semi-definite programming (SDP)[80] to solve the certificate. In contrast, the work of Wong and Kolter[78] transforms the problem (8) into a linear programming problem and solves the problem via training an alternative neural network. Both methods only consider neural networks with one hidden layer. There are also studies of Raghunathan et al.[81] and Wong et al.[82], which improved the efficiency and scalability of these algorithms.

Furthermore, distributional adversarial training[83] combine adversarial training and provable defense together. They train the classifier by feeding adversarial examples which are sampled from the distribution of worst-case perturbation, and derive the certificates by studying the Lagrangian duality of adversarial loss.

4.3 Adversarial example detection

Adversarial example detection is another main approach to protect DNN classifier. Instead of predicting the model's input directly, these methods first distinguish whether the input is benign or adversarial. Then, if it can detect the input is adversarial, the DNN classifier will refuse to predict its label. In the work [16], they sort the threat models into 3 categories that the detection techniques should deal with:

1) A zero-knowledge adversary only has access to the classifier F's parameter, and has no knowledge of the detection model D.

2) A perfect-knowledge adversary is aware of the model F, and the detection scheme D and its parameters.

3) A limited-knowledge adversary is aware the model F and the detection scheme D, but does not have access to D's parameter. That is, this adversary does not know the model's training set.

In all three of these threat settings, the detection tool is required to correctly classify the adversarial examples, and have low possibility of misclassifying benign examples. Next, we will go through some main methods for adversarial example detection.

4.3.1 An auxiliary model to classify adversarial examples

Some works focus on designing auxiliary models that aim to distinguish adversarial examples from benign examples. The study [84] train a DNN model with

$|\mathcal{Y}| = K + 1$ labels, with an additional label for all adversarial examples, so that network will assign adversarial examples into the $K + 1$ class. Similarly, the work of Gong et al.[85] trains a binary classification model to discriminate all adversarial examples apart from benign samples, and then trains a classifier on recognized benign samples.

The work [86] proposed a detection method to construct an auxiliary neural network D which takes inputs from the values of hidden nodes $\mathcal{H}$ of the natural trained classifier. The trained detection classifier $D : \mathcal{H} \to [0, 1]$ is a binary classification model that distinguishes adversarial examples from benign ones by the hidden layers.

4.3.2 Using statistics to distinguish adversarial examples

Some early works heuristically study the differences in the statistical properties of adversarial examples and benign examples. For example, in the study [87], the authors found adversarial examples place a higher weight on the larger (later) principle components where the natural images have larger weight on early principle components. Thus, they can split them by principled component analysis (PCA).

In the work [84], the authors use a statistical test: maximum mean discrepancy (MMD) test[88], which is used to test whether two datsets are drawn from the same distribution. They use this testing tool to test whether a group of data points are benign or adversarial.

4.3.3 Checking the prediction consistency

Other studies focus on checking the consistency of the sample x's prediction outcome. They usually manipulate the model parameters or the input examples themselves, to check whether the outputs of the classifier have significant changes. These are based on the belief that the classifier will have stable predictions on natural examples under these manipulations.

The work [89] randomizes the classifier using Dropout[90]. If these classifiers give very different prediction outcomes on x after randomization, this sample x is very likely to be an adversarial one.

The work [17] manipulates the input sample itself to check the consistency. For each input sample x, the authors reduce the color depth of the image (e.g., one 8-bit grayscale image with 256 possible values for each pixel becomes a 7-bit with 128 possible values), as shown in Fig. 9. The authors hypothesize that for natural images, reducing the color depth will not change the prediction result, but the prediction on adversarial examples will change. In this way, they can detect adversarial examples. Similar to reducing the color depth, the work [89] also introduced other feature squeezing methods, such as spatial smoothing.

4.3.4 Some attacks which evade adversarial detections

The study [16] bypassed 10 of the detection methods which fall into the three categories above. The feature squeezing methods were broken by Sharma and Chen[91], which introduced a "stronger" adversarial attack.

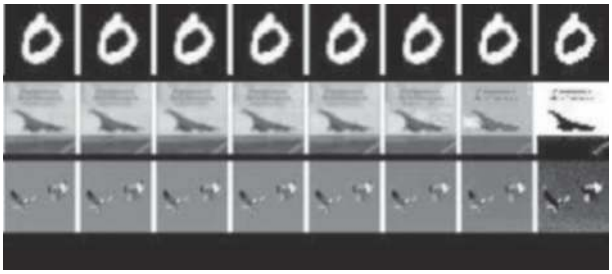


Fig. 9 Images from MNIST and CIFAR10. From left to right, the color depth is reduced from 8-bit, 7-bit, ⋯, 2-bit,1-bit. (Image credit: Xu et al.[17])

The authors in work [16] claim that the properties which are intrinsic to adversarial examples are not very easy to find. They also gave several suggestions on future detection works:

1) Randomization can increase the required attacking distortion.

2) Defenses that directly manipulate on raw pixel values are ineffective.

3) Evaluation should be down on multiple datasets besides MNIST.

4) Report false positive and true positive rates for detection.

5) Evaluate using a strong attack. Simply focusing on white-box attacks is risky.

5 Explanations for the existence of adversarial examples

In addition to crafting adversarial examples and defending them, explaining the reason behind these phenomena is also important. In this section, we briefly introduce the recent works and hypotheses on the key questions of adversarial learning. We hope our introduction will give our audience a basic view on the existing ideas and solutions for these questions.

5.1 Why do adversarial examples exist?

Some original works such as Szegedy's L-BFGS attack[8], state that the existence of adversarial examples is due to the fact that DNN models do not generalize well in low probability space of data. The generalization issue may be caused by the high complexity of DNN model structures.

However, in the work [9], even linear models are also vulnerable to adversarial attacks. Furthermore, in the work [14], they implement experiments to show that an increase in model capacity will improve the model robustness.

Some insight can be gained about the existence of adversarial examples by studying the model's decision boundary. The adversarial examples are almost always

close to decision boundary of a natural trained model, which may be because the decision boundary is too flat[92], too curved[93], or inflexible[94].

Studying the reason behind the existence of adversarial examples is important because it can guide us in designing more robust models, and help us to understand existing deep learning models. However, there is still no consensus on this problem.

5.2 Can we build an optimal classifier?

Many recent works hypothesize that it might be impossible to build optimally robust classifier. For example, the study [95] claim that adversarial examples are inevitable because the distribution of data in each class is not well-concentrated, which leaves room for adversarial examples. In this vein, the work [96] claims that to improve the robustness of a trained model, it is necessary to collect more data. Moreover, the authors in work [25] suggest, even if we can build models with high robustness, it must take cost of some accuracy.

5.3 What is transferability?

Transferability is one of the key properties of adversarial examples. It means that the adversarial examples generated to target one victim model also have a high probability of misleading other models.

Some works compare the transferability between different attacking algorithms. In the work [31], the authors claim that in ImageNet, single step attacks (FGSM) are more likely to transfer between models than iterative attacks (BIM) under same perturbation intensity.

The property of transferability is frequently utilized in attacking techniques in black-box setting[48]. If the model parameters are veiled to attackers, they can turn to attack other substitute models and enjoy the transferability of their generated samples. The property of transferability is also utilized by defending methods as in the work [87]: Since the adversarial examples for model A are also likely to be adversarial for model B, adversarial training using adversarial examples from B will help defend A.

6 Graph adversarial examples

Adversarial examples also exist in graph-structured data[10, 97]. Attackers usually slightly modify the graph structure and node features, in an effort to cause the graph neural networks (GNN) to give wrong prediction for node classification or graph classification tasks. These adversarial attacks therefore raise concerns on the security of applying GNN models. For example, a bank needs to build a reliable credit evaluation system where their model should not be easily attacked by malicious manipulations.

There are some distinct difference between attacking graph models and attacking traditional image classifiers:

1) **Non-independence.** Samples of the graph-structured data are not independent: Changing one′s feature or connection will influence the prediction on others.

2) **Poisoning attacks.** Graph neural networks are usually performed in a transductive learning setting: The test data are also used to train the classifier. This means that we modify the test data, the trained classifier is also changed.

3) **Discreteness.** When modifying the graph structure, the search space for adversarial example is discrete. Previous gradient methods to find adversarial examples may be invalid in this case.

Below are the methods used by some successful works to attack and defend graph neural networks.

6.1 Definitions for graphs and graph models

In this section, the notations and definitions of the graph structured data and graph neural network models are defined below. A graph can be represented as $G = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V}$ is a set of N nodes and $\mathcal{E}$ is a set of M edges. The edges describe the connections between the nodes, which can also be expressed by an adjacency matrix $\boldsymbol{A} \in \{0,1\}^{N\times N}$. Furthermore, a graph G is called an attributed graph if each node in $\mathcal{V}$ is associated with a d-dimensional attribute vector $x_v \in \mathbf{R}^d$. The attributes for all the nodes in the graph can be summarized as a matrix $\boldsymbol{X} \in \mathbf{R}^{N\times d}$, the i-th row of which represents the attribute vector for node v_i.

The goal of node classification is to learn a function $g : \mathcal{V} \to \mathcal{Y}$ that maps each node to one class in $\mathcal{Y}$, based on a group of labeled nodes in G. One of the most successful node classification models is graph convolutional network (GCN)[7]. The GCN model keeps aggregating the information from neighboring nodes to learn representations for each node v,

$$H^{(0)} = X; \quad H^{(l+1)} = \sigma(\hat{A}H^{(l)}W^l)$$

where σ is a non-linear activation function, the matrix $\hat{A}$ is defined as $\hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$, $\tilde{A} = A + I_N$, and $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$. The last layer outputs the score vectors of each node for prediction: $H_v^{(m)} = F(v, X)$.

6.2 Zugner′s greedy method

In the work of Zugner et al.[10], they consider attacking node classification models, graph convolutional networks[7], by modifying the nodes connections or node features (binary). In this setting, an adversary is allowed to add/remove edges between nodes, or flip the feature of nodes with limited number of operations. The goal is to

mislead the GCN model which is trained on the perturbed graph (transductive learning) to give wrong predictions. In their work, they also specify three levels of adversary capabilities: they can manipulate 1) all nodes, 2) a set of nodes $\mathcal{A}$ including the target victim x, and 3) a set of nodes $\mathcal{A}$ which does not include target node x. A sketch is shown in Fig. 10.

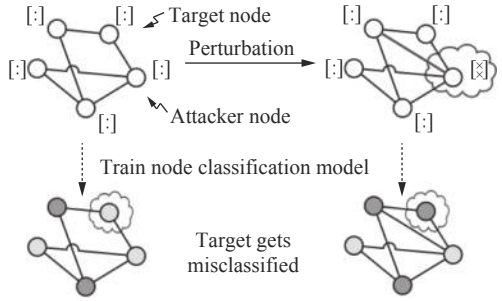


Fig. 10 Adding an edge to alter the prediction of graph convolutional network (Image credit: Zugner et al.[10])

Similar to the objective function in Carlini and Wagner[34] for image data, they formulate the graph attacking problem as a search for a perturbed graph G' such that the learned GCN classifier Z^* has the largest score margin:

$$\max_{i \neq y} \ \ln(Z^*_y(v_0, G')) - \ln(Z^*_i(v_0, G')). \tag{9}$$

The authors solve this objective by finding perturbations on a fixed, linearized substitute GCN classifier G_{sub} which is trained on the clean graph. They use a heuristic algorithm to find the most influential operations on graph G_{sub} (e.g., removing/adding the edge or flipping the feature which can cause largest increase in (9)). The experimental results demonstrate the adversarial operations are also effective on the later trained classifier Z^*.

During the attacking process, the authors also impose two key constraints to ensure the similarity of the perturbed graph to the original one: 1) the degree distribution should be maintained, and 2) two positive features which never happen together in G should also not happen together in G'. Later, some other graph attacking works (e.g., [98]) suggest the eigenvalues/eigenvectors of the graph Laplacian matrix should also be maintained during attacking, otherwise the attacks are easily detected. However, there is still no firm consensus on how to formally define the similarity between graphs and generate unnoticeable perturbation.

6.3 Dai's RL method: RL-S2V

Different from Zugner's greedy method, the work of Dai et al.[97], introduced a reinforcement learning method to attack the graph neural networks. This work only considers adding or removing edges to modify the graph structure.

In the work's setting of [97], a node classifier F trained on the clean graph $G^{(0)} = G$ is given, node classifier F is unknown to the attacker, and the attacker is allowed to modify m edges in total to alter F's prediction on the victim node v_0. The authors formulate this attacking mission as a Q-Learning game[99], with the defined Markov decision process as below:

1) **State.** The state s_t is represented by the tuple $(G^{(t)}, v_0)$, where $G^{(t)}$ is the modified graph with t iterative steps.

2) **Action.** To represent the action to add/remove edges, a single action at time step t is $a_t \in \mathcal{V} \times \mathcal{V}$, which means the edge to be added or removed.

3) **Reward.** In order to encourage actions to fool the classifier, we should give positive reward if v_0's label is altered. Thus, the authors define the reward function as: $r(s_t, a_t) = 0, \forall t = 1, 2, \cdots, m-1$, and for the last step:

$$r(s_m, a_m) = \begin{cases} 1, & \text{if } C(v_0, G^{(m)}) \neq y \\ -1, & \text{if } C(v_0, G^{(m)}) = y. \end{cases}$$

4) **Termination.** The process stops once the agent finishes modifying m edges.

The Q-learning algorithm helps the adversary have knowledge about which actions to take (add/remove which edge) on the given state (current graph structure), in order to get largest reward (change F's output).

6.4 Graph structure poisoning via meta-learning

Previous graph attack works only focus on attacking one single victim node. Meta learning attack[100] attempt to poison the graph so that the global node classification performance of GCN can be undermined and made almost useless. Their approach is based on meta learning[101], which is traditionally used for hyperparameter optimization, few-shot image recognition, and fast reinforcement learning. In the work [100], they use meta learning technique which takes the graph structure as the hyperparameter of the GCN model to optimize. Using their algorithm to perturb 5% edges of a CITESEER graph dataset, they can increase the misclassification rate to over 30%.

6.5 Attack on node embedding

Node embedding attack[102] studies how to perturb the graph structure in order to corrupt the quality of node embedding, and consequently hinder subsequent learning tasks such as node classification or link prediction. Specifically, they study DeepWalk[103] as a random-walk based node embedding learning approach and approxim-

ately find the graph which has the largest loss of the learned node embedding.

6.6 ReWatt: Attacking graph classifier via rewiring

The ReWatt method[98] attempts to attack the graph classification models, where each input of the model is a whole graph. The proposed algorithm can mislead the model by making unnoticeable perturbations on graph.

In their attacking scheme, they utilize reinforcement learning to find a rewiring operation $a = (v_1, v_2, v_3)$ at each step, which is a set of 3 nodes. The first two nodes were connected in the original graph and the edge between them is removed in the first step of the rewiring process. The second step of the rewiring process adds an edge between the nodes v_1 and v_3, where v_3 is constrained to be within 2-hops away from v_1. Some analysis[98] show that the rewiring operation tends to keep the eigenvalues of the graph′s Laplacian matrix, which makes it difficult to detect the attacker.

6.7 Defending graph neural networks

Many works have shown that graph neural networks are vulnerable to adversarial examples, even though there is still no consensus on how to define the unnoticeable perturbation. Some defending works have already appeared. Many of them are inspired by the popular defense methodology in image classification, using adversarial training to protect GNN models[104, 105], which provides moderate robustness.

7 Adversarial examples in audio and text data

Adversarial examples also exist in DNN′s applications in audio and text domains. An adversary can craft fake speech or fake sentences that mislead the machine language processors. Meanwhile, deep learning models on audio/text data have already been widely used in many tasks, such as Apple Siri and Amazon Echo. Therefore, the studies on adversarial examples in audio/text data domain also deserve our attention.

As for text data, the discreteness nature of the inputs makes the gradient-based attack on images not applicable anymore and forces people to craft discrete perturbations on different granularities of text (character-level, word-level, sentence-level, etc). In this section, we introduce the related works in attacking NLP architectures for different tasks.

7.1 Speech recognition attacks

Carlini and Wagner[106] studies to attack state-of-art speech-to-text transcription network, such as DeepSpeech[107]. In their setting, when given any speech waveform x, they can add an inaudible sound perturbation δ that makes the synthesized speech $x + \delta$ be recognized as any targeted desired phrase.

In their attacking work, they limited the maximum decibels (dB) on any time of the added perturbation noise, so that the audio distortion is unnoticeable. Moreover, they inherit the C & W′s attack method[34] on their audio attack setting.

7.2 Text classification attacks

Text classification is one of main tasks in natural language processing. In text classification, the model is devised to understand a sentence and correctly label the sentence. For example, text classification models can be applied on IMDB dataset for characterizing user′s opinion (positive or negative) on the movies, based on their provided reviews. Recent works of adversarial attacks have demonstrated that text classifiers are easily misguided by adversaries slightly modifying the texts' spelling, words or structure.

7.2.1 Attack word embedding

The work [108] considers to add perturbation on the word embedding[109], so as to fool a LSTM[4] classifier. However, this attack only considers perturbing the word embedding, instead of original input sentence itself.

7.2.2 Manipulate words, letters

The work HotFlip[11] considers to replace a letter in a sentence in order to mislead a character-level text classifier (each letter is encoded to a vector). For example, as shown in Fig. 11, altering a single letter in a sentence alters the model′s prediction on its topic. The attack algorithm manages to achieve this by finding the most-influential letter replacement via gradient information. These adversarial perturbations can be noticed by human readers, but they don't change the content of the text as a whole, nor do they affect human judgments.

South Africa's historic Soweto township marks its 100th birthday on Tuesday in a mood of optimism.
57% **World**
South Africa's historic Soweto township marks its 100th birthday on Tuesday in a moo**P** of optimism.
95% **Sci/Tech**

Fig. 11 Replace one letter in a sentence to alter a text classifier′ s prediction on a sentence′ s topic (Image credit: Ebrahimi et al.[11])

The work [110] considers to manipulate the victim sentence on word, phrase level. They try adding, removing or modifying the words and phrases in the sentences. In their approach, the first step is similar to HotFlip[11]. For each training sample, they find the most-influential letters, called "hot characters". Then, they label the words that have more than 3 "hot characters" as "hot words". "Hot words" composite "hot phrases", which are

most-influential phrases in the sentences. Manipulating these phrases is likely to influence the model's prediction, so these phrases composite a "vocabulary" to guide the attacking. When an adversary is given a sentence, he can use this vocabulary to find the weakness of the sentence, add one hot phrase, remove a hot phrase in the given sentence, or insert a meaningful fact which is composed of hot phrases.

DeepWordBug[111] and TextBugger[112] are black-box attack methods for text classification. The basic idea of the former is to define a scoring strategy to identify the key tokens which will lead to a wrong prediction of the classifier if modified. Then they try four types of "imperceivable" modifications on such tokens: swap, substitution, deletion and insertion, to mislead the classifier. The latter follows the same idea, and improves it by introducing new scoring functions.

The works of Samanta and Mehta[113], Iyyer et al.[114] start to craft adversarial sentences that grammarly correct and maintain the syntax structure of the original sentence. Samanta and Mehta[113] achieve this by using synonyms to replace original words, or adding some words which have different meanings in different context. On the other hand, Iyyer et al.[114] manage to fool the text classifier by paraphrasing the structure of sentences.

Witbrock[115] conducts sentence and word paraphrasing on input texts to craft adversarial examples. In this work, they first build a paraphrasing corpus that contains a lot of word and sentence paraphrases. To find an optimal paraphrase of an input text, a greedy method is adopted to search valid paraphrases for each word or sentence from the corpus. Moreover, they propose a gradient-guided method to improve the efficiency of greedy search. This work also has significant contributions in theory: They formally define the task of discrete adversarial attack as an optimization problem on a set function and they prove that the greedy algorithm ensures a $1-\frac{1}{e}$ approximation factor for CNN and RNN text classifiers.

7.3 Adversarial examples in other NLP tasks

7.3.1 Attack on reading comprehension systems

In the work [116], the authors study whether Reading Comprehension models are vulnerable to adversarial attacks. In reading comprehension tasks, the machine learning model is asked to answer a given question, based on the model's "understanding" from a paragraph of an article. For example, the work [116] concentrates on Stanford Question Answering Dataset (SQuAD), in which systems answer questions about paragraphs from Wikipedia.

The authors successfully degrade the intelligence of the state-of-art reading comprehension models on SQuAD by inserting adversarial sentences. As shown in Fig. 12,

Article: Super Bowl 50
Paragraph: "*Peyton Manning became the first quarterback ever to lead two different teams to multiple Super Bowls. He is also the oldest quarterback ever to play in a Super Bowl at age 39. The past record was held by John Elway, who led the Broncos to victory in Super Bowl XXXIII at age 38 and is currently Denver's Executive Vice President of Football Operations and General Manager. Quarterback Jeff Dean had jersey number 37 in Champ Bowl XXXIV.*"
Question: "*What is the name of the quarterback who was 38 in Super Bowl XXXII?*"
Original Prediction: John Elway
Prediction under adversary: Jeff Dean

Fig. 12 By adding an adversarial sentence which is similar to the answer, the reading comprehension model gives a wrong answer (Image credit: Jia and Liang[116])

the inserted sentence (blue) looks similar to the question, but does not contradict the correct answer. This inserted sentence is understandable for human reader but confuses the machine a lot. As a result, the proposed attacking algorithm reduced the performance of 16 state-of-art reading comprehension models from average 75% F1 score (accuracy) to 36%.

Their proposed algorithm AddSent shows a four-step operation to find adversarial sentence.

1) Fake question: What is the name of the quarterback whose jersey number is 37 in Champ Bowl XXXIV?

2) Fake answer: Jeff Dean.

3) Question to declarative form: Quarterback Jeff Dean is jersey number 37 in Champ Bowl XXXIV.

4) Get grammarly correct: Quarterback Jeff Dean had jersey number 37 in Champ Bowl XXXIV.

7.3.2 Attack on neural machine translation

The work [117] studies the stability of machine learning translation tools when their input sentences are perturbed from natural errors (typos, misspellings, etc) and manually crafted distortions (letter replacement, letter reorder). The experimental results show that the state-of-arts translation models are vulnerable to both two types of errors, and suggest adversarial training to improve the model's robustness.

Seq2Sick[118] tries to attack seq2seq models in neural machine translation and text summarization. In their setting, two goals of attacking are set: to mislead the model to generate an output which has on overlapping with the ground truth, and to lead the model to produce an output with targeted keywords. The model is treated as a while-box and the authors formulate the attacking problem as an optimization problem where they seek to solve a discrete perturbation by minimizing a hinge-like loss function.

7.4 Dialogue generation

Unlike the tasks above where success and failure are clearly defined, in the task of dialogue, there is no unique

appropriate response for a given context. Thus, instead of misleading a well-trained model to produce incorrect outputs, works about attacking dialogue models seek to explore the property of neural dialogue models to be interfered by the perturbations on the inputs, or lead a model to output targeted responses.

In the study [119], the authors explore the over-sensitivity and over-stability of neural dialogue models by using some heuristic techniques to modify original inputs and observe the corresponding outputs. They evaluate the robustness of dialogue models by checking whether the outputs change significantly after the modifications on the inputs but do not consider targeted outputs. They also investigate the effects that take place when retraining the dialogue model using these adversarial examples to improve the robustness and performance of the underlying model.

In the work [120], the authors try to find trigger inputs which can lead a neural dialogue model to generate targeted egregious responses. They design a search-based method to determine the word in the input that maximizes the generative probability of the targeted response. Then, they treat the dialogue model as a white-box and take advantage of the gradient information to narrow the search space. Finally they show that this method works for "normal" targeted responses which are decoding results for some input sentences, but for manually written malicious responses, it hardly succeeds.

The work [121] treats the neural dialogue model as a black-box and adopts a reinforcement learning framework to effectively find trigger inputs for targeted responses. The black-box setting is stricter but more realistic, while the requirements for the generated responses are properly relaxed. The generated responses are expected to be semantically identical to the targeted ones but not necessarily exactly match with them.

8 Adversarial examples in miscellaneous tasks

In this section, we summarize some adversarial attacks in other domains. Some of these domains are safety-critical, so the studies on adversarial examples in these domains are also important.

8.1 Computer vision beyond image classification

1) Face recognition

The work [122] seek to attack face recognition models on both a digital level and physical level. The main victim model is based on the architecture of Parkhi et al.[123], which is a 39-layer DNN model for face recognition tasks. The attack on the digital level is based on traditional attacks, like Szegedy's L-BFGS method (Section 3.1.2).

Beyond digital-level adversarial faces, they also succeed in misleading face recognition models on physical level. They achieve this by asking subjects to wear their 3D printed sunglasses frames. The authors optimize the color of these glasses by attacking the model on a digital level: by considering various adversarial glasses, the most effective adversarial glasses are used for attack. As shown in Fig. 13, an adversary wears the adversarial glasses and successfully fool the detection of victim face recognition system.

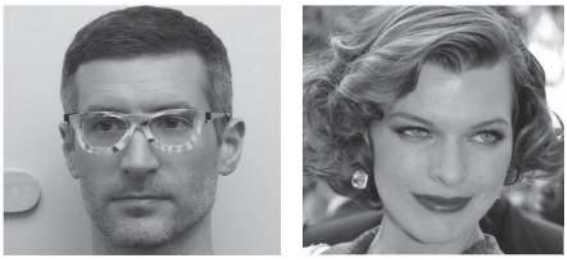


Fig. 13 An adversary (left) wears a pair of adversarial glasses and is recognized as a movie-star, Milla Jovovich (Image credit: Sharif et al.[122])

2) Object detection and semantic segmentation

There are also studies on semantic segmentation and object detection models in computer vision[124, 125]. In both semantic segmentation and object detection tasks, the goal is to learn a model that associates an input image x with a series of labels $\mathcal{Y} = \{y_1, y_2, \cdots, y_N\}$. Semantic segmentation models give each pixel of x a label y_i, so that the image is divided to different segments. Similarly, object detection models label all proposals (regions where the objects lie).

The attacks in [124] can generate an adversarial perturbation on x which can cause the classifier to give wrong prediction on all the output labels of the model, in order to fool either semantic segmentation or object detection models. The attacks[125] finds that there exists universal perturbation for any input image for semantic segmentation models.

8.2 Video adversarial examples

Most works concentrate on attacking static image classification models. However, success on image attacks cannot guarantee that there exist adversarial examples on videos and video classification systems. The work [126] uses GAN[55] to generate a dynamic perturbation on video clips that can mislead the classification of video classifiers.

8.3 Generative models

The work [127] attacks the variational autoencoder (VAE)[128] and VAE-GAN[129]. Both VAE and VAE-GAN use an encoder to project the input image x into a lower-

dimensional latent representation z, and as well a decoder to reconstruct a new image $\hat{x}$ from z. The reconstructed image should maintain the same principle semantics as the original image.

In the setting of attack[127], the authors aim to slightly perturb the input image x fed to encoder, which will cause the decoder to generate image $f_{dec}(f_{enc}(x))$ having different meaning from the input x. For example, in MNIST dataset, the input image is "1", and the reconstructed image is "0".

8.4 Malware detection

The existence of adversarial examples in safety-critical tasks, such as malware detection, should be paid much attention. The work [130] built a DNN model on the DREBIN dataset[131], which contains 120 000 Android application samples, where over 5 000 are malware samples. The trained model has 97% accuracy, but malware samples can evade the classifier if attackers add fake features to them. Some other works, Hu and Tan[132] and Anderson et al.[133], consider using GANs[55] to generate adversarial malware.

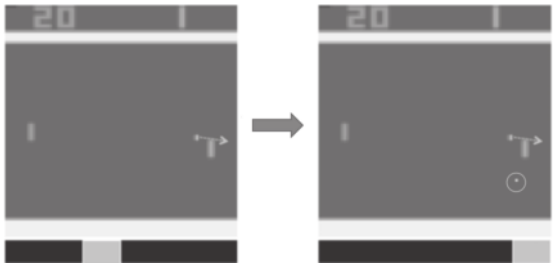
8.5 Fingerprint recognizer attacks

Fingerprint recognition systems are also one of the most safety-critical fields where machine learning models are adopted. While, there are adversarial attacks undermining the reliability of these models. For example, fingerprint spoof attacks copy an authorized person′s fingerprint and replicate it on some special materials such as liquid latex or gelatin. Traditional fingerprint recognition techniques especially minutiae-based models fail to distinguish the fingerprint images generated from different materials. The works of Chugh et al.[134, 135] design a modified CNN to effectively detect this fingerprint spoof attack.

8.6 Reinforcement learning

Different from classification tasks, deep reinforcement learning (RL) aims to learn how to perform some human tasks, such as play Atari 2600 games[99] or play Go[5]. For example, to play an Atari game Pong, (Fig. 14(a)), the trained model takes input from the latest images of game video (state $\sum x$), and output a decision to move up or down (action $\sum y$). The learned model can be viewed as a rule (policy $\sum \pi_\theta$) to win the game (reward $\sum \mathcal{L}(\theta, x, y)$). A simple sketch can be: $x \xrightarrow{\pi_\theta} y$, which is in parallel to classification tasks: $x \xrightarrow{f} y$. The RL algorithms are trained to learn the parameters of π_θ.

The RL attack[137] shows deep reinforcement learning models are also vulnerable to adversarial examples. Their approach is inherited from FGSM[9], to take one-step gradient on the state x (latest images of game video) to craft a fake state x'. The policy′s decision on x' can be totally useless to achieve the reward. Their results show that a slight perturbation on RL models' state, can cause large difference on the models' decision and performance. Their work show Deep Q Learning[99], TRPO[137] and A3C[138] are all vulnerable to their attacks.



(a) Action taken: up
original input

(b) Action taken: down
adversarial input

Fig. 14 Left figure: the brick takes correct actions to go up to catch the ball. Right figure: the current state is perturbed by changing one pixel. The policy gives an incorrect command to go down. (Image credit: Huang et al.[136])

9 Conclusions

In this survey, we give a systemic, categorical and comprehensive overview on the recent works regarding adversarial examples and their countermeasures, in multiple data domains. We summarize the studies from each section in the chronological order as shown in Fig. B in Appendix B, because these works are released with relatively high frequency in response to one another. The current state-of-the-art attacks will likely be neutralized by new defenses, and these defenses will subsequently be circumvented. We hope that our work can shed some light on the main ideas of adversarial learning and related applications in order to encourage progress in this field.

Acknowledgements

This work was supported by National Science Foundation (NSF), USA (Nos. IIS-1845081 and CNS-1815636).

Appendix

A. Dichotomy of attacks

Table A Dichotomy of attacks

Attack	Publication	Similarity	Attacking capability	Algorithm	Apply domain
L-BFGS	[8]	l_2	White-box	Iterative	Image classification
FGSM	[9]	l_∞,l_2	White-box	Single-step	Image classification
Deepfool	[32]	l_2	White-box	Iterative	Image classification
JSMA	[33]	l_2	White-box	Iterative	Image classification
BIM	[31]	l_∞	White-box	Iterative	Image classification
C&W	[34]	l_2	White-box	Iterative	Image classification
Ground truth	[35]	l_0	White-box	SMT solver	Image classification
Spatial	[44]	Total variation	White-box	Iterative	Image classification
Universal	[125]	l_∞, l_2	White-box	Iterative	Image classification
One-Pixel	[39]	l_0	White-box	Iterative	Image classification
EAD	[40]	$l_1 + l_2, l_2$	White-box	Iterative	Image classification
Substitute	[48]	l_p	Black-box	Iterative	Image classification
ZOO	[50]	l_p	Black-box	Iterative	Image classification
Biggio	[19]	l_2	Poisoning	Iterative	Image classification
Explanation	[58]	l_p	Poisoning	Iterative	Image classification
Zugner's	[10]	Degree distribution, coocurrence	Poisoning	Greedy	Node classification
Dai's	[97]	Edges	Black-box	RL	Node & Graph classification
Meta	[100]	Edges	Black-box	RL	Node classification
C&W	[106]	max dB	White-box	Iterative	Speech recognition
Word embedding	[108]	l_p	White-box	One-step	Text classification
HotFlip	[11]	letters	White-box	Greedy	Text classification
Jia & Liang	[116]	letters	Black-box	Greedy	Reading comprehension
Face recognition	[122]	physical	White-box	Iterative	Face recognition
RL attack	[137]	l_p	White-box	RL	

B. Dichotomy of defenses

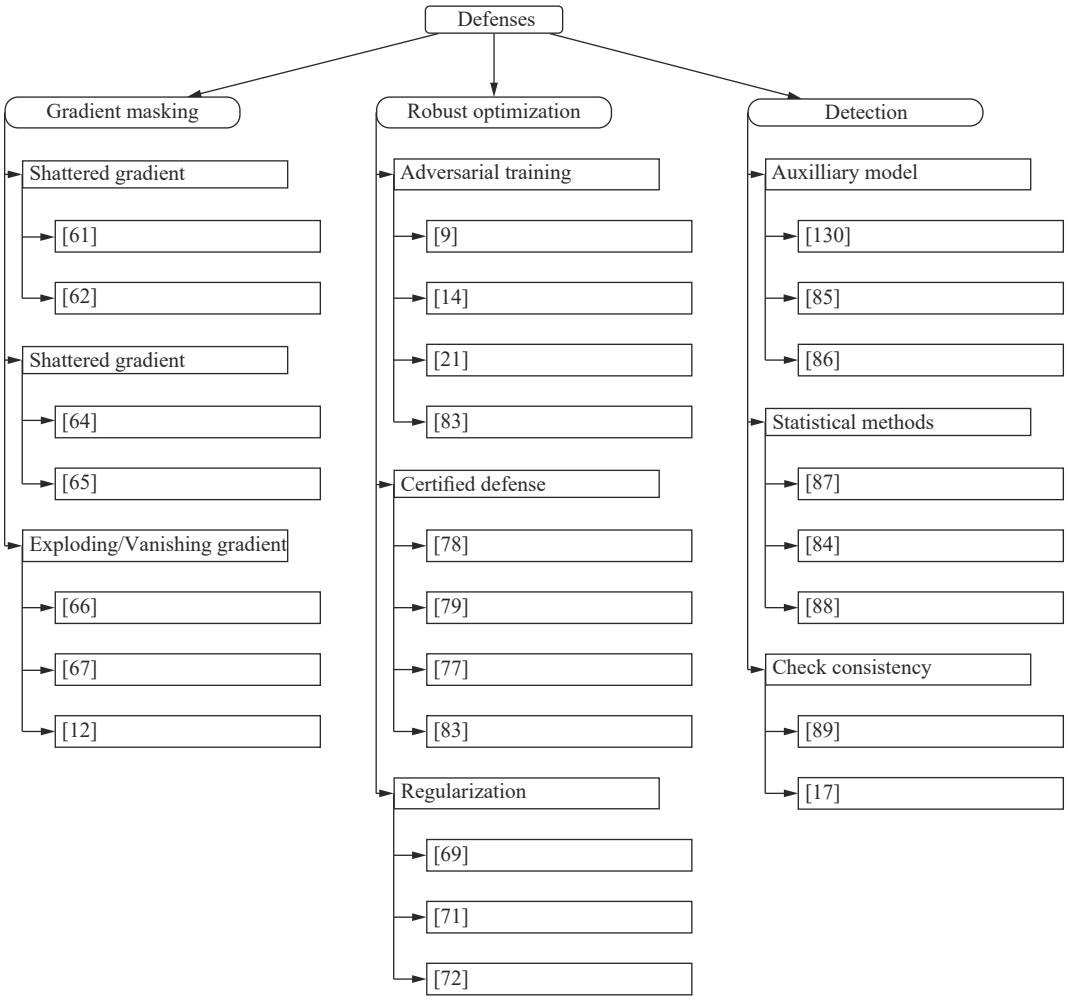


Fig. B Dichotomy of defenses

References

[1] A. Krizhevsky, I. Sutskever, G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems*, Curran Associates Inc., Lake Tahoe, USA, pp. 1097–1105, 2012.

[2] K. M. He, X. Y. Zhang, S. Q. Ren, J. Sun. Deep residual learning for image recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, Las Vegas, USA, pp. 770–778, 2016. DOI: 10.1109/CVPR.2016.90.

[3] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012. DOI: 10.1109/MSP.2012.2205597.

[4] S. Hochreiter, J. Schmidhuber. Long short-term memory. *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. DOI: 10.1162/neco.1997.9.8.1735.

[5] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, D. Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, vol. 529, no. 7587, pp. 484–489, 2016. DOI: 10.1038/nature16961.

[6] D. Cireşan, U. Meier, J. Masci, J. Schmidhuber. Multi-column deep neural network for traffic sign classification. *Neural Networks*, vol. 32, pp. 333–338, 2012. DOI: 10.1016/j.neunet.2012.02.023.

[7] T. N. Kipf, M. Welling. Semi-supervised classification with graph convolutional networks. ArXiv: 1609.02907, 2016.

[8] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, R. Fergus. Intriguing properties of neural networks. ArXiv: 1312.6199, 2013.

[9] I. J. Goodfellow, J. Shlens, C. Szegedy. Explaining and harnessing adversarial examples. ArXiv: 1412.6572, 2014.

[10] D. Zügner, A. Akbarnejad, S. Günnemann. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference*

on Knowledge Discovery & Data Mining, ACM, London, UK, pp. 2847–2856, 2018. DOI: 10.1145/3219819.3220078.

[11] J. Ebrahimi, A. Y. Rao, D. Lowd, D. J. Dou. HotFlip: White-box adversarial examples for text classification. ArXiv: 1712.06751, 2017.

[12] N. Papernot, P. McDaniel, X. Wu, S. Jha, A. Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *Proceedings of IEEE Symposium on Security and Privacy*, IEEE, San Jose, USA, pp. 582–597, 2016. DOI: 10.1109/SP.2016.41.

[13] A. Athalye, N. Carlini, D. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. ArXiv: 1802.00420, 2018.

[14] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, A. Vladu. Towards deep learning models resistant to adversarial attacks. ArXiv: 1706.06083, 2017.

[15] A. Kurakin, I. Goodfellow, S. Bengio. Adversarial examples in the physical world. ArXiv: 1607.02533, 2016.

[16] N. Carlini, D. Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, ACM, Dallas, USA, pp. 3–14, 2017. DOI: 10.1145/3128572.3140444.

[17] W. L. Xu, D. Evans, Y. J. Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. ArXiv: 1704.01155, 2017.

[18] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, A. Madry. Adversarial examples are not bugs, they are features. ArXiv: 1905.02175, 2019.

[19] B. Biggio, B. Nelson, P. Laskov. Poisoning attacks against support vector machines. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, Omnipress, Edinburgh, UK, 2012.

[20] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. W. Xiao, A. Prakash, T. Kohno, D. Song. Robust physical-world attacks on deep learning models. ArXiv: 1707.08945, 2017.

[21] F. Tramer, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, P. McDaniel. Ensemble adversarial training: Attacks and defenses. ArXiv: 1705.07204, 2017.

[22] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, F. Roli. Evasion attacks against machine learning at test time. In *Proceedings of European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, Prague, Czech Republic, pp. 387–402, 2013. DOI: 10.1007/978-3-642-40994-3_25.

[23] M. Barreno, B. Nelson, A. D. Joseph, J. D. Tygar. The security of machine learning. *Machine Learning*, vol. 81, no. 2, pp. 121–148, 2010. DOI: 10.1007/s10994-010-5188-5.

[24] N. Dalvi, P. Domingos, Mausam, S. Sanghai, D. Verma. Adversarial classification. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, Seattle, USA, pp. 99–108, 2004. DOI: 10.1145/1014052.1014066.

[25] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, A. Madry. Robustness may be at odds with accuracy. ArXiv: 1805.12152, 2018.

[26] D. Su, H. Zhang, H. G. Chen, J. F. Yi, P. Y. Chen, Y. P. Gao. Is robustness the cost of accuracy? – A comprehensive study on the robustness of 18 deep image classification models. In *Proceedings of the 15th European Conference on Computer Vision*, Springer, Munich, Germany, pp. 644–661, 2018. DOI: 10.1007/978-3-030-01258-8_39.

[27] D. Stutz, M. Hein, B. Schiele. Disentangling adversarial robustness and generalization. In *Proceedings of the 32nd IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, Piscataway, USA, pp. 6976–6987, 2019.

[28] H. Y. Zhang, Y. D. Yu, J. T. Jiao, E. P. Xing, L. El Ghaoui, M. I. Jordan. Theoretically principled trade-off between robustness and accuracy. ArXiv: 1901.08573, 2019.

[29] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, F. F. Li. Imagenet: A large-scale hierarchical image database. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, Miami, USA, pp. 248–255, 2009. DOI: 10.1109/CVPR.2009.5206848.

[30] D. C. Liu, J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, vol. 45, no. 1–3, pp. 503–528, 1989. DOI: 10.1007/BF01589116.

[31] A. Kurakin, I. Goodfellow, S. Bengio. Adversarial machine learning at scale. ArXiv: 1611.01236, 2016.

[32] S. M. Moosavi-Dezfooli, A. Fawzi, P. Frossard. DeepFool: A simple and accurate method to fool deep neural networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, Las Vegas, USA, pp. 2574–2582, 2016. DOI: 10.1109/CVPR.2016.282.

[33] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, A. Swami. The limitations of deep learning in adversarial settings. In *Proceedings of IEEE European Symposium on Security and Privacy*, IEEE, Saarbrucken, Germany, pp. 372–387, 2016. DOI: 10.1109/EuroSP.2016.36.

[34] N. Carlini, D. Wagner. Towards evaluating the robustness of neural networks. In *Proceedings of IEEE Symposium on Security and Privacy*, IEEE, San Jose, USA, pp. 39–57, 2017. DOI: 10.1109/SP.2017.49.

[35] N. Carlini, G. Katz, C. Barrett, D. L Dill. Provably minimally-distorted adversarial examples. ArXiv: 1709.10207, 2017.

[36] G. Katz, C. Barrett, D. L. Dill, K. Julian, M. J. Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. In *Proceedings of the 29th International Conference on Computer Aided Verification*, Springer, Heidelberg, Germany, pp. 97–117, 2017. DOI: 10.1007/978-3-319-63387-9_5.

[37] V. Tjeng, K. Xiao, R. Tedrake. Evaluating robustness of neural networks with mixed integer programming. ArXiv: 1711.07356, 2017.

[38] K. Y. Xiao, V. Tjeng, N. M. Shafiullah, A. Madry. Training for faster adversarial robustness verification via inducing ReLU stability. ArXiv: 1809.03008, 2018.

[39] J. W. Su, D. V. Vargas, K. Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 828–841, 2019. DOI: 10.1109/TEVC.2019.2890858.

[40] P. Y. Chen, Y. Sharma, H. Zhang, J. F. Yi, C. J. Hsieh. EAD: Elastic-net attacks to deep neural networks via ad-

versarial examples. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, 2018.

[41] Y. Sharma, P. Y. Chen. Attacking the madry defense model with L_1-based adversarial examples. ArXiv: 1710.10733, 2017.

[42] S. M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, P. Frossard. Universal adversarial perturbations. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, Honolulu, USA, pp. 86–94, 2017. DOI: 10.1109/CVPR.2017.17.

[43] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. A. Ma, Z. H. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, F. F. Li. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015. DOI: 10.1007/s11263-015-0816-y.

[44] C. W. Xiao, J. Y. Zhu, B. Li, W. He, M. Y. Liu, D. Song. Spatially transformed adversarial examples. ArXiv: 1801.02612, 2018.

[45] Y. Song, R. Shu, N. Kushman, S. Ermon. Constructing unrestricted adversarial examples with generative models. In *Proceedings of the 32nd Conference on Neural Information Processing Systems*, Montréal, Canada, pp. 8312–8323, 2018.

[46] A. Odena, C. Olah, J. Shlens. Conditional image synthesis with auxiliary classifier GANs. In *Proceedings of the 34th International Conference on Machine Learning*, Sydney, Australia, pp. 2642–2651, 2017.

[47] A. Athalye, L. Engstrom, A. Ilyas, K. Kwok. Synthesizing robust adversarial examples. ArXiv: 1707.07397, 2017.

[48] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, A. Swami. Practical black-box attacks against machine learning. In *Proceedings of ACM on Asia Conference on Computer and Communications Security*, ACM, Abu Dhabi, United Arab Emirates, pp. 506–519, 2017. DOI: 10.1145/3052973.3053009.

[49] Y. P. Dong, F. Z. Liao, T. Y. Pang, H. Su, J. Zhu, X. L. Hu, J. G. Li. Boosting adversarial attacks with momentum. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, Salt Lake City, USA, pp. 9185–9193, 2018. DOI: 10.1109/CVPR.2018.00957.

[50] P. Y. Chen, H. Zhang, Y. Sharma, J. F Yi, C. J. Hsieh. ZOO: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, ACM, Dallas, USA, pp. 15–26, 2017. DOI: 10.1145/3128572.3140448.

[51] A. Ilyas, L. Engstrom, A. Athalye, J. Lin. Black-box adversarial attacks with limited queries and information. ArXiv: 1804.08598, 2018.

[52] D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, J. Peters, J. Schmidhuber. Natural evolution strategies. Natural evolution strategies. *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 949–980, 2014.

[53] M. Alzantot, Y. Sharma, S. Chakraborty, M. Srivastava. Genattack: Practical black-box attacks with gradient-free optimization. ArXiv: 1805.11090, 2018.

[54] C. W. Xiao, B. Li, J. Y. Zhu, W. He, M. Y. Liu, D. Song. Generating adversarial examples with adversarial networks. ArXiv: 1801.02610, 2018.

[55] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, MIT Press, Montreal, Canada, pp. 2672–2680, 2014.

[56] D. Deb, J. B. Zhang, A. K. Jain. Advfaces: Adversarial face synthesis. ArXiv: 1908.05008, 2019.

[57] G. Cauwenberghs, T. Poggio. Incremental and decremental support vector machine learning. In *Proceedings of the 13th International Conference on Neural Information Processing Systems*, MIT Press, Denver, USA, pp. 388–394, 2000.

[58] P. W. Koh, P. Liang. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning*, Sydney, Australia, pp. 1885–1894, 2017.

[59] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, T. Goldstein. Poison frogs! Targeted clean-label poisoning attacks on neural networks. In *Proceedings of the 32nd Conference on Neural Information Processing Systems*, Montréal, Canada, pp. 6103–6113, 2018.

[60] G. Hinton, O. Vinyals, J. Dean. Distilling the knowledge in a neural network. ArXiv: 1503.02531, 2015.

[61] J. Buckman, A. Roy, C. Raffel, I. Goodfellow. Thermometer encoding: One hot way to resist adversarial examples. In *Proceedings of the 6th International Conference on Learning Representations*, Vancouver, Canada, 2018.

[62] C. Guo, M. Rana, M. Cisse, L. van der Maaten. Countering adversarial images using input transformations. ArXiv: 1711.00117, 2017.

[63] V. K. Ha, J. C. Ren, X. Y. Xu, S. Zhao, G. Xie, V. M. Vargas. Deep learning based single image super-resolution: A survey. In *Proceedings of the 9th International Conference on Brain Inspired Cognitive Systems*, Springer, Xi′an, China, pp. 106–119, 2018. DOI: 10.1007/978-3-030-00563-4_11.

[64] G. S. Dhillon, K. Azizzadenesheli, Z. C. Lipton, J. Bernstein, J. Kossaifi, A. Khanna, A. Anandkumar. Stochastic activation pruning for robust adversarial defense. ArXiv: 1803.01442, 2018.

[65] C. H. Xie, J. Y. Wang, Z. S. Zhang, Z. Ren, A. Yuille. Mitigating adversarial effects through randomization. ArXiv: 1711.01991, 2017.

[66] Y. Song, T. Kim, S. Nowozin, S. Ermon, N. Kushman. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. ArXiv: 1710.10766, 2017.

[67] P. Samangouei, M. Kabkab, R. Chellappa. Defense-GAN: Protecting classifiers against adversarial attacks using generative models. ArXiv: 1805.06605, 2018.

[68] A. van den Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, K. Kavukcuoglu. Conditional image generation with PixelCNN decoders. In *Proceedings of the 30th Conference on Neural Information Processing Systems*, Curran Associates Inc., Barcelona, Spain, pp. 4790–4798, 2016.

[69] M. Cisse, P. Bojanowski, E. Grave, Y. Dauphin, N. Usunier. Parseval networks: Improving robustness to adversarial examples. In *Proceedings of the 34th International Conference on Machine Learning*, Sydney, Aus-

tralia, pp. 854–863, 2017.

[70] T. Miyato, S. I. Maeda, M. Koyama, K. Nakae, S. Ishii. Distributional smoothing with virtual adversarial training. ArXiv: 1507.00677, 2015.

[71] S. X. Gu, L. Rigazio. Towards deep neural network architectures robust to adversarial examples. ArXiv: 1412.5068, 2014.

[72] S. Rifai, P. Vincent, X. Muller, X. Glorot, Y. Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, Omnipress, Bellevue, USA, pp. 833–840, 2011.

[73] S. Ioffe, C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. ArXiv: 1502.03167, 2015.

[74] A. Shafahi, M. Najibi, A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L. S. Davis, G. Taylor, T. Goldstein. Adversarial training for free! ArXiv: 1904.12843, 2019.

[75] D. H. Zhang, T. Y. Zhang, Y. P. Lu, Z. X. Zhu, B. Dong. You only propagate once: Accelerating adversarial training via maximal principle. ArXiv: 1905.00877, 2019.

[76] L. S. Pontryagin. *Mathematical Theory of Optimal Processes*, London, UK: Routledge, 2018.

[77] A. Raghunathan, J. Steinhardt, P. Liang. Certified defenses against adversarial examples. ArXiv: 1801.09344, 2018.

[78] E. Wong, J. Z. Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. ArXiv: 1711.00851, 2017.

[79] M. Hein, M. Andriushchenko. Formal guarantees on the robustness of a classifier against adversarial manipulation. In *Proceedings of the 31st Conference on Neural Information Processing Systems*, Long Beach, USA, pp. 2266–2276, 2017.

[80] L. Vandenberghe, S. Boyd. Semidefinite programming. Semidefinite programming. *SIAM Review*, vol. 38, no. 1, pp. 49–95, 1996. DOI: 10.1137/1038003.

[81] A. Raghunathan, J. Steinhardt, P. S. Liang. Semidefinite relaxations for certifying robustness to adversarial examples. In *Proceedings of the 32nd Conference on Neural Information Processing Systems*, Montréal, Canada, pp. 10877–10887, 2018.

[82] E. Wong, F. Schmidt, J. H. Metzen, J. Z. Kolter. Scaling provable adversarial defenses. In *Proceedings of the 32nd Conference on Neural Information Processing Systems*, Montréal, Canada, pp. 8400–8409, 2018.

[83] A. Sinha, H. Namkoong, J. Duchi. Certifying some distributional robustness with principled adversarial training. ArXiv: 1710.10571, 2017.

[84] K. Grosse, P. Manoharan, N. Papernot, M. Backes, P. McDaniel. On the (statistical) detection of adversarial examples. ArXiv: 1702.06280, 2017.

[85] Z. T. Gong, W. L. Wang, W. S. Ku. Adversarial and clean data are not twins. ArXiv: 1704.04960, 2017.

[86] J. H. Metzen, T. Genewein, V. Fischer, B. Bischoff. On detecting adversarial perturbations. ArXiv: 1702.04267, 2017.

[87] D. Hendrycks, K. Gimpel. Early methods for detecting adversarial images. ArXiv: 1608.00530, 2016.

[88] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, A. Smola. A kernel two-sample test. A kernel two-sample test. *Journal of Machine Learning Research*, vol. 13, pp. 723–773, 2012.

[89] R. Feinman, R. R. Curtin, S. Shintre, A. B. Gardner. Detecting adversarial samples from artifacts. ArXiv: 1703.00410, 2017.

[90] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[91] Y. Sharma, P. Y. Chen. Bypassing feature squeezing by increasing adversary strength. ArXiv: 1803.09868, 2018.

[92] A. Fawzi, S. M. Moosavi-Dezfooli, P. Frossard. Robustness of classifiers: From adversarial to random noise. In *Proceedings of the 30th Conference on Neural Information Processing Systems*, Barcelona, Spain, pp. 1632–1640, 2016.

[93] S. M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, P. Frossard, S. Soatto. Analysis of universal adversarial perturbations. ArXiv: 1705.09554, 2017.

[94] A. Fawzi, O. Fawzi, P. Frossard. Analysis of classifiers′ robustness to adversarial perturbations. *Machine Learning*, vol. 107, no. 3, pp. 481–508, 2018. DOI: 10.1007/s10994-017-5663-3.

[95] A. Shafahi, W. R. Huang, C. Studer, S. Feizi, T. Goldstein. Are adversarial examples inevitable? ArXiv: 1809.02104, 2018.

[96] L. Schmidt, S. Santurkar, D. Tsipras, K. Talwar, A. Madry. Adversarially robust generalization requires more data. In *Proceedings of the 32nd Conference on Neural Information Processing Systems*, Montréal, Canada, pp. 5014–5026, 2018.

[97] H. J. Dai, H. Li, T. Tian, X. Huang, L. Wang, J. Zhu, L. Song. Adversarial attack on graph structured data. ArXiv: 1806.02371, 2018.

[98] Y. Ma, S. H. Wang, T. Derr, L. F. Wu, J. L. Tang. Attacking graph convolutional networks via rewiring. ArXiv: 1906.03750, 2019.

[99] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller. Playing Atari with deep reinforcement learning. ArXiv: 1312.5602, 2013.

[100] D. Zügner, S. Günnemann. Adversarial attacks on graph neural networks via meta learning. ArXiv: 1902.08412, 2019.

[101] C. Finn, P. Abbeel, S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, JMLR.org, Sydney, Australia, pp. 1126–1135, 2017.

[102] A. Bojchevski, S. Günnemann. Adversarial attacks on node embeddings via graph poisoning. ArXiv: 1809.01093, 2018.

[103] B. Perozzi, R. Al-Rfou, S. Skiena. DeepWalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, New York, USA, pp. 701–710, 2014. DOI: 10.1145/2623330.2623732.

[104] F. L. Feng, X. N. He, J. Tang, T. S. Chua. Graph adversarial training: Dynamically regularizing based on graph structure. ArXiv: 1902.08226, 2019.

[105] K. D. Xu, H. G. Chen, S. J. Liu, P. Y. Chen, T. W. Weng, M. Y. Hong, X. Lin. Topology attack and defense for graph neural networks: An optimization perspective. ArXiv: 1906.04214, 2019.

[106] N. Carlini, D. Wagner. Audio adversarial examples: Targeted attacks on speech-to-text. In *Proceedings of IEEE Security and Privacy Workshops*, IEEE, San Francisco, USA, pp. 1–7, 2018. DOI: 10.1109/SPW.2018.00009.

[107] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, A. Y. Ng. Deep speech: Scaling up end-to-end speech recognition. ArXiv: 1412.5567, 2014.

[108] T. Miyato, A. M. Dai, I. Goodfellow. Adversarial training methods for semi-supervised text classification. ArXiv: 1605.07725, 2016.

[109] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, J. Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, Curran Associates Inc., Lake Tahoe, USA, pp. 3111–3119, 2013.

[110] B. Liang, H. C. Li, M. Q. Su, P. Bian, X. R. Li, W. C. Shi. Deep text classification can be fooled. ArXiv: 1704.08006, 2017.

[111] J. Gao, J. Lanchantin, M. L. Soffa, Y. J. Qi. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *Proceedings of IEEE Security and Privacy Workshops*, IEEE, San Francisco, USA, pp. 50–56, 2018. DOI: 10.1109/SPW.2018.00016.

[112] J. F. Li, S. L. Ji, T. Y. Du, B. Li, T. Wang. Textbugger: Generating adversarial text against real-world applications. ArXiv: 1812.05271, 2018.

[113] S. Samanta, S. Mehta. Towards crafting text adversarial samples. ArXiv: 1707.02812, 2017.

[114] M. Iyyer, J. Wieting, K. Gimpel, L. Zettlemoyer. Adversarial example generation with syntactically controlled paraphrase networks. ArXiv: 1804.06059, 2018.

[115] Q. Lei, L. F. Wu, P. Y. Chen, A. G. Dimakis, I. S. Dhillon, M. Witbrock. Discrete attacks and submodular optimization with applications to text classification. ArXiv: 1812.00151, 2018.

[116] R. Jia, P. Liang. Adversarial examples for evaluating reading comprehension systems. ArXiv: 1707.07328, 2017.

[117] Y. Belinkov, Y. Bisk. Synthetic and natural noise both break neural machine translation. ArXiv: 1711.02173, 2017.

[118] M. H. Cheng, J. F. Yi, H. Zhang, P. Y. Chen, C. J. Hsieh. Seq2Sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples. ArXiv: 1803.01128, 2018.

[119] T. Niu, M. Bansal. Adversarial over-sensitivity and over-stability strategies for dialogue models. ArXiv: 1809.02079, 2018.

[120] T. X. He, J. Glass. Detecting egregious responses in neural sequence-to-sequence models. ArXiv: 1809.04113, 2018.

[121] H. C. Liu, T. Derr, Z. T. Liu, J. L Tang. Say what I want: Towards the dark side of neural dialogue models. ArXiv: 1909.06044, 2019.

[122] M. Sharif, S. Bhagavatula, L. Bauer, M. K. Reiter. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, ACM, Vienna, Austria, pp. 1528–1540, 2016. DOI: 10.1145/2976749.2978392.

[123] O. M. Parkhi, A. Vedaldi, A. Zisserman. Deep face recognition. *Machine Learning* 2015.

[124] C. H. Xie, J. Y. Wang, Z. S. Zhang, Y. Y. Zhou, L. X. Xie, A. Yuille. Adversarial examples for semantic segmentation and object detection. In *Proceedings of IEEE International Conference on Computer Vision*, IEEE, Venice, Italy, pp. 1378–1387, 2017. DOI: 10.1109/ICCV.2017.153.

[125] J. H. Metzen, M. C. Kumar, T. Brox, V. Fischer. Universal adversarial perturbations against semantic image segmentation. In *Proceedings of IEEE International Conference on Computer Vision*, IEEE, Venice, Italy, pp. 2774–2783, 2017. DOI: 10.1109/ICCV.2017.300.

[126] S. S. Li, A. Neupane, S. Paul, C. Y. Song, S. V. Krishnamurthy, A. K. R. Chowdhury, A. Swami. Adversarial perturbations against real-time video classification systems. ArXiv: 1807.00458, 2018.

[127] J. Kos, I. Fischer, D. Song. Adversarial examples for generative models. In *Proceedings of IEEE Security and Privacy Workshops*, IEEE, San Francisco, USA, pp. 36–42, 2018. DOI: 10.1109/SPW.2018.00014.

[128] D. P. Kingma, M. Welling. Auto-encoding variational Bayes. ArXiv: 1312.6114, 2013.

[129] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, O. Winther. Autoencoding beyond pixels using a learned similarity metric. ArXiv: 1512.09300, 2015.

[130] K. Grosse, N. Papernot, P. Manoharan, M. Backes, P. McDaniel. Adversarial perturbations against deep neural networks for malware classification. ArXiv: 1606.04435, 2016.

[131] D. Arp, M. Spreitzenbarth, H. Gascon, K. Rieck. DREBIN: Effective and explainable detection of android malware in your pocket. *In Proceedings of Symposium Network Distributed System Security*, Internet Society, San Diego, USA, 2014.

[132] W. W. Hu, Y. Tan. Generating adversarial malware examples for black-box attacks based on GAN. ArXiv: 1702.05983, 2017.

[133] H. S. Anderson, J. Woodbridge, B. Filar. DeepDGA: Adversarially-tuned domain generation and detection. In *Proceedings of ACM Workshop on Artificial Intelligence and Security*, ACM, Vienna, Austria, pp. 13–21, 2016. DOI: 10.1145/2996758.2996767.

[134] T. Chugh, A. K. Jain. Fingerprint presentation attack detection: Generalization and efficiency. ArXiv: 1812.11574, 2018.

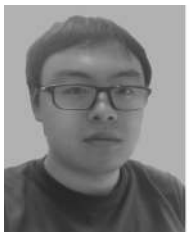
[135] T. Chugh, K. Cao, A. K. Jain. Fingerprint spoof buster: Use of minutiae-centered patches. *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 9, pp. 2190–2202, 2018. DOI: 10.1109/TIFS.2018.2812193.

[136] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, P. Ab-

beel. Adversarial attacks on neural network policies. ArXiv: 1702.02284, 2017.

[137] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, P. Abbeel. Trust region policy optimization. In *Proceedings of the 31st International Conference on Machine Learning*, JMLR, Lille, France, pp. 1889–1897, 2015.

[138] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Harley, T. P. Lillicrap, D. Silver, K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33rd International conference on Machine Learning*, PMLR, New York, USA, pp. 1928–1937, 2016.



Han Xu is a second year Ph. D. student of computer science in DSE Lab, Michigan State University, USA. He is under supervision by Dr. Ji-Liang Tang.

His research interests include deep learning safety and robustness, especially studying the problems related to adversarial examples.

E-mail: xuhan1@msu.edu (Corresponding author)

ORCID iD: 0000-0002-4016-6748

Yao Ma received the B. Sc. degree in applied mathematics at Zhejiang University, China in 2015, the M. Sc. degree in statistics, probabilities and operation research at Eindhoven University of Technology, the Netherlands in 2016. He is now a Ph. D. degree candidate of Department of Computer Science and Engineering, Michigan State University, USA. His Ph. D. advisor is Dr. Jiliang Tang.

His research interests include graph neural networks and their related safety issues.

E-mail: mayao4@msu.edu

Hao-Chen Liu is currently a Ph. D. student at the Department of Computer Science and Engineering at Michigan State University, under the supervision of Dr. Jiliang Tang. He is a member of Data Science and Engineering (DSE) Lab.

His research interests include natural language processing problems, especially in the robustness, fairness of dialogue systems.

E-mail: liuhaoc@msu.edu

Debayan Deb is a Ph. D. degree candidate in the Biometrics Lab, Michigan State University, USA under the supervision of Dr. Anil K. Jain. Before joining the Biometrics Lab of MSU, he graduated from Michigan State University with a Bachelor Degree of Computer Science and Engineering.

His research interests include face recognition and computer vision tasks.

E-mail: debdebay@msu.edu

Hui Liu is a research associate at Michigan State University. Before joining MSU, she received her Ph. D. degree of Electrical Engineering in Southern Methodist University, USA under the supervision by Dr. Dinesh Rajen.

Her research interests include signal processing, wireless communication, and deep learning related topics.

E-mail: liuhui7@msu.edu

Ji-Liang Tang is an assistant professor in the computer science and engineering department at Michigan State University since Fall 2016. Before that, he was a research scientist in Yahoo Research and received his Ph. D. degree from Arizona State University in 2015. He was the recipients of 2019 NSF Career Award, the 2015 KDD Best Dissertation runner up and 6 Best Paper Awards (or runner-ups) including WSDM 2018 and KDD 2016. He serves as conference organizers (e.g., KDD, WSDM and SDM) and journal editors (e.g., TKDD). He has published his research in highly ranked journals and top conference proceedings, which received thousands of citations and extensive media coverage.

His research interests include social computing, data mining and machine learning and their applications in education.

E-mail: tangjili@msu.edu

Anil K. Jain (Ph. D., 1973, Ohio State University; B. Tech., IIT Kanpur) is a University Distinguished Professor at Michigan State University where he conducts research in pattern recognition, machine learning, computer vision, and biometrics recognition. He was a member of the United States Defense Science Board and Forensics Science Standards Board. His prizes include Guggenheim, Humboldt, Fulbright, and King-Sun Fu Prize. For advancing pattern recognition, Jain was awarded Doctor Honoris Causa by Universidad Autónoma de Madrid. He was Editor-in-Chief of the IEEE Transactions on Pattern Analysis and Machine Intelligence and is a Fellow of ACM, IEEE, AAAS, and SPIE. Jain has been assigned 8 U.S. and Korean patents and is active in technology transfer for which he was elected to the National Academy of Inventors. Jain is a member of the U.S. National Academy of Engineering (NAE), foreign member of the Indian National Academy of Engineering (INAE), a member of The World Academy of Science (TWAS) and a foreign member of the Chinese Academy of Sciences (CAS).

His research interests include pattern recognition, machine learning, computer vision, and biometrics recognition.

E-mail: jain@egr.msu.edu