Adversarial Examples in Deep Neural Networks: An Overview

Emilio Rafael Balda, Arash Behboodi,, and Rudolf Mathar

Institute for Theoretical Information Technology (TI),
RWTH Aachen University,
ICT cubes, Kopernikusstraße 16, 52074 Aachen
`{balda,behboodi,mathar}@ti.rwth-aachen.de`

Abstract. Deep learning architectures are vulnerable to adversarial perturbations. They are added to the input and alter drastically the output of deep networks. These instances are called adversarial examples. They are observed in various learning tasks from supervised learning to unsupervised and reinforcement learning. In this chapter, we review some of the most important highlights in theory and practice of adversarial examples. The focus is on designing adversarial attacks, theoretical investigation into the nature of adversarial examples, and establishing defenses against adversarial attacks. A common thread in the design of adversarial attacks is the perturbation analysis of learning algorithms. Many existing algorithms rely implicitly on perturbation analysis for generating adversarial examples. The summary of most powerful attacks are presented in this light. We overview various theories behind the existence of adversarial examples as well as theories that consider the relation between the generalization error and adversarial robustness. Finally, various defenses against adversarial examples are also discussed.

Keywords: Adversarial examples, deep learning, classification, regression, perturbation analysis, statistical learning, adversarial training, adversarial defenses

1 Introduction

Artificial intelligence is on the rise and Deep Neural Networks (DNNs) are an important part of it. Whether it is in speech analysis [29] or visual tasks [34,27,69,59], they shine with a performance beyond what was imagined a decade ago. Their success is undeniable, nevertheless a flaw has been spotted in their performance. They are not stable under adversarial perturbations [70]. Adversarial perturbations are intentionally worst case designed noises that aim at changing the output of a DNN to an incorrect one. The perturbations are most of the time so small that an ordinary observer may not even notice it, and even the state-of-the-art DNNs are highly confident in their, wrong, classification of these adversarial examples. This phenomena is depicted in Figure 1, borrowed from [24], where a subtle adversarial perturbation is able to change the classification outcome.

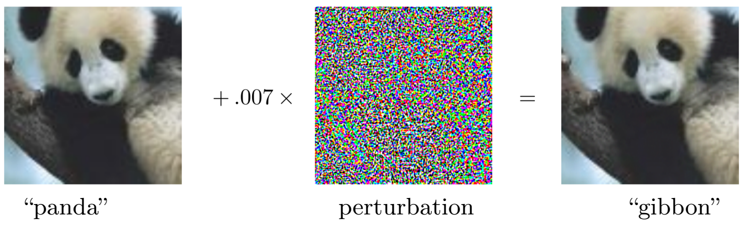


Fig. 1: A demonstration from [24] of adversarial examples generated using the FGSM. By adding an imperceptibly small vector, we can change GoogLeNet's classification of the image.

Robustness to adversarial perturbations is different from robustness to random noise [19], a trait that can be achieved by DNNs. The existence of adversarial perturbations was known for machine learning algorithms [9], however, they were first noticed in deep learning research in [70]. These discoveries generated interest among researchers to understand the instability of DNNs, to explore various attacks and devise multiple defenses. Although it is very difficult to keep up with the pace of results in this area, there are many excellent surveys on the topic. For instance, the surveys [1,79] cover many interesting instances for which adversarial examples exist. In this chapter, we overview as well some of the most important findings regarding adversarial examples for DNNs. However, we adopt a different approach. Instead of creating a catalog of existing attacks and defenses, we present an adequately general framework which can recover many existing attacks. Theoretical findings regarding the nature of adversarial examples are additionally addressed. In this light, we address three problems in this chapter, namely, adversarial attacks, their theoretical explanation and adversarial defenses.

The first question is about generating adversarial examples and designing attacks. This is discussed in the first part of this chapter. Historically these examples were first found for classification tasks and were based on first order approximations of DNNs. These methods require knowledge of model parameters and are therefore sometimes called white-box attacks. We overview some of the most important attacks including iterative and non-iterative methods, as well as single and multiple pixel attacks. Instead of listing different attacks, our goal is to present a unifying framework for generating adversarial examples. The framework, which goes beyond classification problems, is based on a convex optimization formulation of adversarial input generation. We overview, furthermore, black-box attacks where only partial knowledge of model parameters is available for generating adversarial examples. Universal adversarial perturbations and the transferability of adversarial examples are other topics discussed in this part.

The second question is about the nature of adversarial examples. Why are DNNs and other machine learning models vulnerable to adversarial examples? In the second part, we overview some of the attempts to investigate theoretically this question. In many works, the adversarial vulnerability is attributed to some properties of machine learning models. Some examples are linearity of models, curvature of decision boundaries of classifiers and low ℓ_1-norm of weight matrices. After reviewing some of these theories, we discuss statistical learning theoretic approaches that explore the relation between adversarial robustness and generalization capabilities of machine learning models. Out of this study come new guidelines for designing adversarially robust algorithms, which brings us to the third question of this chapter. How can we design effective defenses against adversarial examples?

The defenses take up different approaches from modifying the training process by changing the training set to adding new regularizations or considering new DNN architectures or a combination of preceding approaches. Some of the most recent contributions in this direction are discussed in the last part.

1.1 Notation and Preliminaries

We introduce first the notation used in this chapter and some of the basic definitions needed throughout this chapter. The letters $\mathbf{x}, \mathbf{y}, \dots$ are used for vectors, $\mathbf{A}, \mathbf{B}, \dots$ for matrices and $\mathcal{X}, \mathcal{Y}, \dots$ for sets. We denote the set $\{1, \dots, n\}$ by $[n]$ for $n \in \mathbb{N}$. For any vector $\mathbf{x} = (x_1, \dots, x_n)^\mathsf{T} \in \mathbb{R}^n$ and $p \in \mathbb{N}$, the ℓ_p-norm of $\mathbf{x}$ is defined by

$$\|\mathbf{x}\|_p := \left(\sum_{i=1}^{n} x_i^p \right)^{1/p}.$$

When p tends to zero, the above definition converges to the number of non-zero entries of the vector. This is called, with an abuse of terminology, the ℓ_0-norm. The explicit definition is given as

$$\|\mathbf{x}\|_0 := \sum_{i=1}^{n} \mathbf{1}(x_i \neq 0).$$

The ℓ_0-norm gives the sparsity order of the vector $\mathbf{x}$. The ℓ_∞-norm of a vector $\mathbf{x}$ is obtained when $p \to \infty$. It is defined as

$$\|\mathbf{x}\|_\infty := \max_{i \in [n]} |x_i|.$$

The norm of a matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$ is similarly defined. The Frobenius norm of $\mathbf{X}$ is denoted by $\|\mathbf{X}\|_F$ and defined as:

$$\|\mathbf{X}\|_F := \left(\sum_{i=1}^{m} \sum_{j=1}^{n} X_{ij}^2 \right)^{1/2}.$$

The Shatten p-norm of the matrix $\mathbf{X}$ is equal to the ℓ_p-norm of the singular value vector $(\sigma_1, \dots, \sigma_{\min(m,n)})$ of $\mathbf{X}$, namely:

$$\|\mathbf{X}\|_p := \left(\sum_{i=1}^{\min\{m,n\}} \sigma_i^p \right)^{1/p}.$$

The Frobenius norm of $\mathbf{X}$ is equivalent to the ℓ_2-norm of the singular value vector. The ℓ_1-norm of the singular value vector is called the nuclear norm. The ℓ_0-norm is similarly defined and gives the rank of $\mathbf{X}$.

Consider a function $f : \mathbb{R}^n \to \mathbb{R}^m$ given by $f(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_m(\mathbf{x}))$ for m function $f_i : \mathbb{R}^n \to \mathbb{R}$. The Jacobian of f at $\mathbf{x}$ is denoted by $J_f(\mathbf{x})$ and defined as

$$J_f(\mathbf{x}) := \left(\tfrac{\partial f}{\partial x_1}(\mathbf{x}), \ldots, \tfrac{\partial f}{\partial x_m}(\mathbf{x})\right) = \left[\frac{\partial f_i}{\partial x_j}(\mathbf{x})\right]_{i\in[n],j\in[m]}.$$

2 Adversarial Perturbation Design

Adversarial attacks follow ubiquitously the same pattern. An adversarial attacker is assumed to have access to the system input. This can be the input of DNNs. It applies perturbations to the system inputs under an additional and important constraint. The perturbations should be restricted in some sense. For image-based tasks, this means that an ordinary observer should not be capable of spotting, at least immediately, a significant change in the image and its label. More generally, this constraint makes it hard for the administrator to detect the perturbations. Finally, and most importantly, the system performance, for example its classification accuracy, should be severely degraded. The attacks in [24,48,58] follow similar guidelines. Two categories of adversarial attacks can be envisaged, white-box and black-box attacks. In white-box attacks, the architecture of the target algorithms are known to the attacker, although there are attacks with only partial knowledge of the architecture. In contrast stand black-box attacks, which require no information about the target neural network, see for instance [63].

In the pioneering work of [70], the attack is based on finding adversarial perturbations that maximize the prediction error at the output. The perturbations are approximated by minimizing the ℓ_2-norm of the perturbation. If the multi-class classifier mapping is defined by $f : \mathbb{R}^n \to [K]$, Szegedy et al. in [70] minimize the ℓ_2-norm of the perturbation $\boldsymbol{\eta}$ such that the classifier output is changed to the target label $l \in [K]$, i.e., $f(\mathbf{x} + \boldsymbol{\eta}) = l$. The perturbation $\boldsymbol{\eta}$ is constrained to be inside the box $[0, 1]^n$. The adversarial example is obtained by adding the perturbation $\boldsymbol{\eta}$ to the input vector $\mathbf{x}$. In the next attack, the FGSM in [24], the sign of the gradient of the cost function is used for designing perturbations which were scaled to have bounded ℓ_∞-norm, and therefore to be almost undetectable. If the cost function used for training is given by $c(\mathbf{x})$, the perturbation is given by $\eta = \varepsilon\, \mathrm{sign}(\nabla c(\mathbf{x}))$. The ℓ_∞-norm of the perturbation is ε. An example of the FGSM is shown in Figure 1. Iterative procedures or randomizations can significantly strengthen adversarial attacks. An iterative linearization of the DNN is proposed in the algorithm DeepFool [48] to generate minimal ℓ_p-norm perturbations for $p > 1$. The iterative approach continues to add perturbations with bounded ℓ_p-norm until the classifier's output is altered. An iterative version of FGSM, called Basic Iterative Method (BIM) is proposed in [35]. The Projected Gradient Descent (PGD) attack is an extension of previous techniques, proposed in [43], where randomness is additionally introduced in the computation of adversarial perturbations. The PGD attack can bypass many defenses and is employed in [43] to devise a defense against adversarial examples. An iterative algorithm based on PGD combined with randomization is introduced in [5] and has been used to dismantle many defenses so far [4]. Another popular way of generating adversarial examples is by constraining the ℓ_0-norm of the perturbation. Manipulating only few entries, these types of attacks are known as single pixel attacks [67] and multiple pixel attacks [53].

In what follows, to generate adversarial examples, we provide a unifying framework that incorporates the above techniques. The main ingredient of this framework is perturbation analysis. Given a classifier function, the perturbation analysis of this function quantifies how much its output is perturbed when a known perturbation is applied to its input. An approximation of this output error is usually obtained using a first-order Taylor approximation of the function, under the assumption that the input perturbations are of small norms. Adversarial examples suitably fall into this framework, as they are perturbed versions of original inputs, the perturbations are small and the function at hand comes naturally from the model. Consider, for example, the FGSM given in [24]. The proposed attack aims at maximizing the training loss function that is approximated by its first-order Taylor approximation. Similarly, the authors of [48,24] constructed adversarial examples by maximizing the error, on a relevant function, that occurs as a consequence of input perturbations. Iterative methods like the DeepFool method [48], the BIM [35], the PGD method [43], and the gradient-based norm-constrained method (GNM) [8], maximize the output perturbation using successive first order approximations. A summary about the connections and differences between these methods is provided in [8]. It is based on this framework that we formulate the problem of generating adversarial examples in this section.

Let us first fix the terminology used in this section. The input of classifiers is denoted by $\mathbf{x}$. Then, adversarial examples are constructed by adding an adversarial perturbation $\boldsymbol{\eta}$, of the same dimension as $\mathbf{x}$, to that input. For a multi-class classification with K classes, a classifier maps inputs to the discrete set of labels $[K]$. Classifiers modeled by DNNs based their decision usually on a set of functions, often differentiable, known as *score functions*. These functions can replace the non-differentiable classification function for which a first-order Taylor approximation is not possible because the gradients are not properly defined. The score functions and classification functions are defined below.

Definition 1 (Score functions and classifier functions). *A classifier is defined by the mapping $k : \mathbb{R}^M \to [K]$ that maps an input $\mathbf{x} \in \mathbb{R}^M$ to its estimated class $k(\mathbf{x}) \in [K]$. The mapping $k(\cdot)$ is itself defined by*

$$k(\mathbf{x}) = \underset{l\in[K]}{\operatorname{argmax}} \left\{ f_l(\mathbf{x}) \right\} , \tag{1}$$

where $f_l(\mathbf{x}) : \mathbb{R}^M \to \mathbb{R}$'s represent the probability of class belonging. The function $f(\mathbf{x})$ given by the vector $(f_1(\mathbf{x}), \ldots, f_K(\mathbf{x}))^\mathsf{T}$ is known as score function and can be assumed to be differentiable almost everywhere for many classifiers.

Finding adversarial examples amounts to finding a perturbation that changes the classifier's output. However, since they are imperceptible, such adversarial perturbations should not modify the inputs significantly. The undetectability of adversarial examples can be better understood using image classification tasks as an example. For instance, in Figure 1 we observe that the human eye can not distinguish between the original and adversarial image. A common way to impose this restriction is by constraining adversarial perturbation to belong to a certain set of unnoticeable perturbations. For example, the authors of the FGSM bounded the ℓ_∞-norm of their perturbation, or in the DeepFool method, the norm is incrementally increased until the output classifier changes. Note that DeepFool may produce perceptible perturbations, while the FGSM may not fool the classifier.

Another way of imposing undetectability of adversarial examples is to impose on the input perturbation to preserve the outcome of the ground truth classifier [75], also known as oracle classifier. In many applications, the oracle classifier refers to the human brain. Similar to Definition 1, denote the score function of the oracle classifier as $g : \mathbb{R}^M \to \mathbb{R}^K$, which outputs a vector with entries $g_l : \mathbb{R}^M \to \mathbb{R}$ for $l = 1, \ldots, K$. The adversarial perturbation $\boldsymbol{\eta}$ is said to be undetectable if

$$L_g(\mathbf{x}, \boldsymbol{\eta}) = g_{k(\mathbf{x})}(\mathbf{x} + \boldsymbol{\eta}) - \max_{l \neq k(\mathbf{x})} g_l(\mathbf{x} + \boldsymbol{\eta}) > 0\,. \tag{2}$$

Using this notion, the problem of finding adversarial examples amounts to the following.

Definition 2 (Adversarial Generation Problem). *For a given* $\mathbf{x} \in \mathbb{R}^M$*, the adversarial generation problem consists of finding a perturbation* $\boldsymbol{\eta} \in \mathbb{R}^M$ *to fool the classifier* $k(\cdot)$ *by the adversarial sample* $\hat{\mathbf{x}} = \mathbf{x} + \boldsymbol{\eta}$ *such that* $k(\mathbf{x}) \neq k(\hat{\mathbf{x}})$ *and the oracle classifier is not changed, i.e.,*

$$\begin{aligned} \text{Find}: \quad & \boldsymbol{\eta} \\ \text{s.t.} \quad & L_f(\mathbf{x}, \boldsymbol{\eta}) = f_{k(\mathbf{x})}(\mathbf{x} + \boldsymbol{\eta}) - \max_{l \neq k(\mathbf{x})} f_l(\mathbf{x} + \boldsymbol{\eta}) < 0 \\ & L_g(\mathbf{x}, \boldsymbol{\eta}) = g_{k(\mathbf{x})}(\mathbf{x} + \boldsymbol{\eta}) - \max_{l \neq k(\mathbf{x})} g_l(\mathbf{x} + \boldsymbol{\eta}) > 0 \end{aligned} \tag{3}$$

However, since the oracle classifier is usually unknown, this problem is not interesting for practical purposes. To overcome this issue, it is shown in forthcoming sections how the solution of this problem can be approximated by tractable relaxations.

2.1 White-Box Attacks

The white-box setting corresponds to the scenario when the classification function $f(\cdot)$ and input $\mathbf{x}$ are both known to the attacker. Thus, adversarial perturbations are designed with full knowledge of the target system.

Non-Iterative Methods As discussed above, the constraint on the oracle function of (3) cannot be computed in practice, since the oracle classifier is not available. To address this problem, such constraints are approximated by restricting the set of possible adversarial perturbations to a known subset. The most common choice is to restrict $\boldsymbol{\eta}$ to belong to the set of vectors with bounded ℓ_p-norm for $p \geq 1$. The values of p are restricted to be $p \geq 1$ so that the set $\|\boldsymbol{\eta}\|_p \leq \varepsilon$ is convex for any $\varepsilon > 0$. Note that the choice of p will determine the structure of the obtained adversarial examples . The case of $p = \infty$ has been the focus of research in recent years. Even after replacing the oracle constraint on (3), with a convex one, the problem remains non-convex. For the case of white-box attacks, a similar relaxation can be carried out by approximating $L_f(\mathbf{x}, \cdot)$ with its firs-order Taylor expansion. This is possible since we assume to have full knowledge about $\mathbf{x}$ and the function $f(\cdot)$.

To that end, the first-order Taylor expansion of $L_f(\mathbf{x}, \cdot)$ around $\mathbf{0}$ leads to

$$L_f(\mathbf{x}, \boldsymbol{\eta}) = L_f(\mathbf{x}, \mathbf{0}) + \boldsymbol{\eta}^\mathsf{T} \nabla_{\boldsymbol{\eta}} L_f(\mathbf{x}, \mathbf{0}) + \mathcal{O}(\|\boldsymbol{\eta}\|_2^2),$$

where $\mathcal{O}(\|\boldsymbol{\eta}\|_2^2)$ contains higher order terms. Therefore, by replacing the oracle function constraint in (3) with $\|\boldsymbol{\eta}\|_p \leq \varepsilon$, for sufficiently small $\varepsilon \in \mathbb{R}^+$, we get

$$\begin{aligned} \text{Find:} \quad & \boldsymbol{\eta} \\ \text{s.t.} \quad & L_f(\mathbf{x}, \mathbf{0}) + \boldsymbol{\eta}^\mathsf{T} \nabla_{\boldsymbol{\eta}} L_f(\mathbf{x}, \mathbf{0}) < 0, \quad \|\boldsymbol{\eta}\|_p \leq \varepsilon, \end{aligned} \tag{4}$$

which is a relaxed version of the problem exposed in (3). This formulation of the problem can be used to construct well known existing adversarial attacks from the literature. This will be discussed in detail in Section 2.1. Nevertheless, the following theorem shows that this problem is not always feasible.

Theorem 1. *The optimization problem* (4) *is not feasible if for* $q = \frac{p}{p-1}$

$$\varepsilon \|\nabla_{\boldsymbol{\eta}} L_f(\mathbf{x}, \mathbf{0})\|_q < L_f(\mathbf{x}, \mathbf{0}). \tag{5}$$

The proof can be obtained from the results in [28], as well as in [7].

The theorem points to the insight that there might be no perturbation that is small enough and yet changes the output label. This implication befits the intuition that it should not be expected to fool a classifier for an arbitrarily small ε with the perturbation's norm constraint. This result suggests that a feasible problem can be obtained if we only impose one of the constraints while trying to preserve the other one as much as possible. To that end, a proper objective function that penalizes the deviation from the original constraint is minimized. This gives rise to the following two problems, as feasible counterparts of (4).

First, the norm-constraint in (4) is imposed resulting in the following optimization problem, called GNM in [7]. It minimizes $L_f(\mathbf{x}, \mathbf{0}) + \boldsymbol{\eta}^\mathsf{T} \nabla_{\boldsymbol{\eta}} L_f(\mathbf{x}, \mathbf{0})$ as

$$\min_{\boldsymbol{\eta}} L_f(\mathbf{x}, \mathbf{0}) + \boldsymbol{\eta}^\mathsf{T} \nabla_{\boldsymbol{\eta}} L_f(\mathbf{x}, \mathbf{0}) \quad \text{s.t.} \quad \|\boldsymbol{\eta}\|_p \leq \varepsilon\,. \tag{6}$$

Using this approach we can find the best possible perturbation under the norm-constraint. However, a proper value for ε must be chosen beforehand to guarantee that the perturbations remain unnoticed. Moreover, this problem has a closed form solution which can be computed efficiently, as stated in the following theorem.

Theorem 2. *If* $\nabla_{\boldsymbol{\eta}} L_f(\mathbf{x}, \boldsymbol{\eta}) = (\frac{\partial L_f(\mathbf{x},\boldsymbol{\eta})}{\partial \eta_1}, \dots, \frac{\partial L_f(\mathbf{x},\boldsymbol{\eta})}{\partial \eta_M})$*, the closed form solution to the minimizer of the problem* (6) *is given by*

$$\boldsymbol{\eta} = -\varepsilon \frac{1}{\|\nabla_{\boldsymbol{\eta}} L_f(\mathbf{x}, \mathbf{0})\|_q^{q-1}} \operatorname{sign}(\nabla_{\boldsymbol{\eta}} L_f(\mathbf{x}, \mathbf{0})) \odot |\nabla_{\boldsymbol{\eta}} L_f(\mathbf{x}, \mathbf{0})|^{q-1} \tag{7}$$

for $q = \frac{p}{p-1}$*, where* $\operatorname{sign}(\cdot)$ *and* $|\cdot|^{q-1}$ *are applied element-wise, and* $\odot$ *denotes the element-wise (Hadamard) product. Particularly for* $p = \infty$*, we have* $q = 1$ *and the solution is given by the following*

$$\boldsymbol{\eta} = -\varepsilon \operatorname{sign}(\nabla_{\boldsymbol{\eta}} L_f(\mathbf{x}, \mathbf{0})) \,. \tag{8}$$

The proof can be found in [7]. One advantage of using (6), besides having a closed-form solution, is that additional constraints on the perturbation can be added to the problem. In addition, the solution shown in (7) can be reused for other choices of $L_f(\mathbf{x}, \cdot)$, which can be more suitable depending on the scenario. For instance, the FGSM chooses $L_f(\mathbf{x}, \cdot)$ to be the negative of the loss function used for training, which is often the cross-entropy loss in classification problems. Then, minimizing $L_f(\mathbf{x}, \cdot)$ corresponds to maximizing the loss. A caveat is that using problem (6) ensures perturbations with bounded norms, but such perturbations may not be able to fool the classifier.

A second approach for relaxing (4) into a feasible problem is to keep the constraint regarding $L_f(\mathbf{x}, \cdot)$ and minimize over the norm of $\boldsymbol{\eta}$. Therefore, the problem of (4) is replaced by

$$\min_{\boldsymbol{\eta}} \|\boldsymbol{\eta}\|_p \quad \text{s.t.} \quad L_f(\mathbf{x}, \mathbf{0}) + \boldsymbol{\eta}^\mathsf{T} \nabla_{\boldsymbol{\eta}} L_f(\mathbf{x}, \mathbf{0}) \leq 0 \,. \tag{9}$$

This approach is used by [48] on every iteration of the DeepFool algorithm (more details in Section 2.1). Similarly to (6), this problem has a closed for solution as well, which is given in the following theorem.

Theorem 3. *If* $\nabla_{\boldsymbol{\eta}} L_f(\mathbf{x}, \boldsymbol{\eta}) = (\frac{\partial L_f(\mathbf{x},\boldsymbol{\eta})}{\partial \eta_1}, \dots, \frac{\partial L_f(\mathbf{x},\boldsymbol{\eta})}{\partial \eta_M})$*, the closed form solution to the problem* (9) *is given by*

$$\boldsymbol{\eta} = -\frac{L_f(\mathbf{x}, \mathbf{0})}{\|\nabla_{\boldsymbol{\eta}} L_f(\mathbf{x}, \mathbf{0})\|_q^{q-1}} \operatorname{sign}(\nabla_{\boldsymbol{\eta}} L_f(\mathbf{x}, \mathbf{0})) \odot |\nabla_{\boldsymbol{\eta}} L_f(\mathbf{x}, \mathbf{0})|^{q-1} \tag{10}$$

for $q = \frac{p}{p-1}$.

Observe that the perturbation from Theorem 3, similar to the solution in Theorem 2, is nothing but an adjusted version of the gradient of the classifier with a different norm. The perturbation in (10) might grow unbounded to ensure that the classifier is misled, which makes it perceptible by the oracle. There are other similar methods for computing adversarial examples that depend on a first-order approximation of other performance-related functions. These algorithms are later shown to be slight variations of the methods presented in this section. Furthermore, using the present formulation, we can build iterative procedures by repeating the optimization problem until the classifier output changes. In Section 2.1, we compare different methods, which are formulated as iterative versions of (6) and (9).

There are some methods that rely on adding randomness in the generation process. The PGD attack, from [43], is one well known example. For the PGD attack, the first-order approximation is taken not around $\boldsymbol{\eta} = \mathbf{0}$, but instead, around a random point $\tilde{\boldsymbol{\eta}}$ with its norm bounded by some $\tilde{\varepsilon}$, that is $\tilde{\varepsilon} \triangleq \|\tilde{\boldsymbol{\eta}}\|_p \leq \varepsilon$. In short, the objective function $L_f(\mathbf{x}, \cdot)$ is approximated by its linear counterpart around the point $\tilde{\boldsymbol{\eta}}$, which lies within an $\tilde{\varepsilon}$-radius from $\boldsymbol{\eta} = \mathbf{0}$. The distribution $\tilde{\boldsymbol{\eta}}$ can be arbitrarily chosen as long as the norm constraint is not violated. A common choice is to use the uniform distribution over the set of vectors with bounded ℓ_p norm. We denote this technique as *dithering*. Moreover, displacing the center of the first order approximation from $\mathbf{0}$ to $\tilde{\boldsymbol{\eta}}$ does not lead to solutions which differ from the ones given so far. This is true since $L_f(\mathbf{x}, \boldsymbol{\eta}) \approx L_f(\mathbf{x}, \tilde{\boldsymbol{\eta}}) + (\boldsymbol{\eta} - \tilde{\boldsymbol{\eta}})^\mathsf{T} \nabla_{\boldsymbol{\eta}} L_f(\mathbf{x}, \tilde{\boldsymbol{\eta}})$ leads to the following problem

$$\min_{\boldsymbol{\eta}} L_f(\mathbf{x}, \tilde{\boldsymbol{\eta}}) + (\boldsymbol{\eta} - \tilde{\boldsymbol{\eta}})^\mathsf{T} \nabla_{\boldsymbol{\eta}} L_f(\mathbf{x}, \tilde{\boldsymbol{\eta}}) \quad \text{s.t.} \quad \|\boldsymbol{\eta}\|_p \leq \varepsilon,$$

which corresponds to solving

$$\min_{\boldsymbol{\eta}} \boldsymbol{\eta}^\mathsf{T} \nabla_{\boldsymbol{\eta}} L_f(\mathbf{x}, \tilde{\boldsymbol{\eta}}) \quad \text{s.t.} \quad \|\boldsymbol{\eta}\|_p \leq \varepsilon\,. \tag{11}$$

When training models with adversarial examples, it is advantageous to add randomness to their computation in order to increase the diversity of the adversarial perturbations during the training [73], as done with dithering technique in quantization literature. Further details about training with adversarial examples are discussed in Section 4.1.

Single Subset Attacks In the field of image recognition, it is also common to model undetectability by restricting the number of pixels that can be altered by an attacker. Single and multiple pixel attacks are introduced to this end. For the case of gray-scale images, altering only one value of the input vector is equivalent to a single pixel attack. This is, however, not a general rule. If inputs are RGB images, each pixel will be defined as a subset of three values.

Since adversarial attacks go beyond image based systems, we allude as single subset attacks to those whose target is only a subset of entries. Given that perturbations belong to $\mathbb{R}^M$, let us partition $[M] = \{1, \ldots, M\}$ into S possible subsets $\mathcal{S}_1, \ldots, \mathcal{S}_S$.

These sets may be of different size, but for the sake of clarity let us assume that they have the same cardinality $Z = M/S$, where $\mathcal{S}_s = \{i_s^1, \dots, i_s^Z\} \subseteq [M]$. Define the mixed zero-$\mathcal{S}$ norm $\|\cdot\|_{0,\mathcal{S}}$ of a vector, for the partition $\mathcal{S} = \{\mathcal{S}_1, \dots, \mathcal{S}_S\}$, as the number of subsets including at least one index related to a non-zero entry of $\mathbf{x}$ [1], that is

$$\|\mathbf{x}\|_{0,\mathcal{S}} = \sum_{i=1}^{S} \mathbf{1}(\|\mathbf{x}_{\mathcal{S}_i}\| \neq 0)\,,$$

where $\mathbf{1}(\cdot)$ denotes the indicator function. Hence, the norm $\|\boldsymbol{\eta}\|_{0,\mathcal{S}}$ counts the number of subsets altered by an attacker. Moreover, we can guarantee that only one subset stays active by including this as an additional constraint in (3), yielding

$$\min_{\boldsymbol{\eta}} L_f(\mathbf{x}, \boldsymbol{\eta}) \quad \text{s.t.} \quad \|\boldsymbol{\eta}\|_\infty \leq \varepsilon\,, \|\boldsymbol{\eta}\|_{0,\mathcal{S}} = 1\,. \tag{12}$$

As a remark, the mixed norm $\|.\|_{0,\mathcal{S}}$ is extensively used in signal processing and compressed sensing to promote group sparsity [58]. In a similar manner as in Section 2.1, we employ the approximation $L_f(\mathbf{x}, \boldsymbol{\eta}) \approx L_f(\mathbf{x}, \tilde{\boldsymbol{\eta}}) + (\boldsymbol{\eta} - \tilde{\boldsymbol{\eta}})^\mathsf{T} \nabla_{\boldsymbol{\eta}} L_f(\mathbf{x}, \tilde{\boldsymbol{\eta}})$ which yields the following linear programming formulation of (12) as

$$\min_{\boldsymbol{\eta}} \boldsymbol{\eta}^\mathsf{T} \nabla_{\boldsymbol{\eta}} L_f(\mathbf{x}, \tilde{\boldsymbol{\eta}}) \quad \text{s.t.} \quad \|\boldsymbol{\eta}\|_\infty \leq \varepsilon\,,\ \|\boldsymbol{\eta}\|_{0,\mathcal{S}} = 1\,. \tag{13}$$

Given a subset $\mathcal{S}_s$ we define $\boldsymbol{\eta}_s$ as

$$\boldsymbol{\eta}_s = \operatorname*{argmin}_{\boldsymbol{\eta}} \nabla_{\boldsymbol{\eta}} L_f(\mathbf{x}, \tilde{\boldsymbol{\eta}})^\mathsf{T} \boldsymbol{\eta} \text{ s.t. } \|\boldsymbol{\eta}\|_\infty \leq \varepsilon\,, (\boldsymbol{\eta})_{i_s^z} = 0\ \forall i_s^z \notin \mathcal{S}_s\,.$$

Note that we have a closed form solution for $\boldsymbol{\eta}_s$, that is

$$\boldsymbol{\eta}_s = -\varepsilon \sum_{z=1}^{Z} \operatorname{sign}((\nabla_{\boldsymbol{\eta}} L_f(\mathbf{x}, \tilde{\boldsymbol{\eta}}))_{i_s^z}) \mathbf{e}_{i_s^z}\,,$$

which implies that $\nabla_{\boldsymbol{\eta}} L_f(\mathbf{x}, \tilde{\boldsymbol{\eta}})^\mathsf{T} \boldsymbol{\eta}_s = -\sum_{z=1}^{Z} \left|(\nabla_{\boldsymbol{\eta}} L_f(\mathbf{x}, \tilde{\boldsymbol{\eta}}))_{i_s^z}\right|$. Then, this problem has the closed form solution given by

$$\boldsymbol{\eta}^* = \boldsymbol{\eta}_{s*}\,, \text{ with } s^* = \operatorname*{argmax}_{s} \sum_{z=1}^{Z} \left|(\nabla_{\boldsymbol{\eta}} L_f(\mathbf{x}, \tilde{\boldsymbol{\eta}}))_{i_s^z}\right|\,. \tag{14}$$

Iterative Methods and Randomization In the previous section, we summarized different versions of the problem of generating adversarial perturbations. An overview of these methods is shown in Table 1. For this section, we will work with these solutions to design adversarial perturbations using iterative approximations. Since the principle behind the approaches in (6) and (9) is the same, we will only focus on (6). Nevertheless, it is trivial to extend the algorithms presented in this section to use (9) instead.

[1] Similar to the so-called ℓ_0-norm, this is not a proper norm.

Type of Attack	Relaxed Problem	Closed-Form Solution
ℓ_2 / ℓ_∞ constrained	(6)	(7)
Single-Subset attack	(13)	(14)

Table 1: Summary of the obtained closed-form solutions.

In Algorithm 1, an iterative method based on (6) is introduced. This iterative version of (6) resembles a gradient descent method for minimizing $L_f(\mathbf{x}, \boldsymbol{\eta})$ over $\boldsymbol{\eta}$ with a fixed number of iterations and steps of equal ℓ_p-norm. For that purpose, a set of parameters $\tilde{\varepsilon}_1, \dots, \tilde{\varepsilon}_T$ is required to control the norm of random noise used for dithering. There is no dithering if $\tilde{\varepsilon}_i$ is set to zero for all $i = 1, \dots, T$. The well known PGD attack uses dithering by applying it at the initial iteration. In this attack $\tilde{\varepsilon}_2 = \cdots = \tilde{\varepsilon}_T = 0$, and random$(\tilde{\varepsilon}_1)$ generates a random vector with a uniform distribution over the ℓ_p-ball of radius $\tilde{\varepsilon}_1$ (centered at $\mathbf{0}$).

Algorithm 1 Iterative extension for ℓ_p constrained methods.

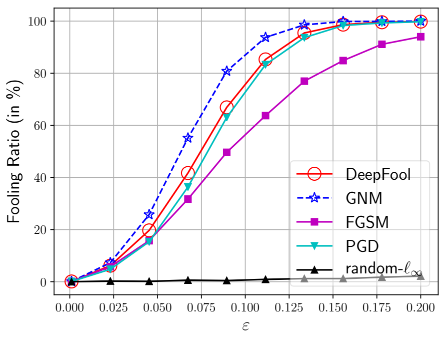
```
**input:** $\mathbf{x}$, $f(\cdot)$, $T$, $\varepsilon$, $\tilde{\varepsilon}_1, \dots, \tilde{\varepsilon}_T$.
**output:** $\boldsymbol{\eta}^*$.
Initialize $\boldsymbol{\eta}_1 \leftarrow \mathbf{0}$.
**for** $t = 1, \dots, T$ **do**
    $\tilde{\boldsymbol{\eta}}_t \leftarrow \boldsymbol{\eta}_t + \text{random}(\tilde{\varepsilon}_t)$
    $\boldsymbol{\eta}_t^* \leftarrow \text{argmin}_{\boldsymbol{\eta}} \boldsymbol{\eta}^\mathsf{T} \nabla_{\boldsymbol{\eta}} L_f(\mathbf{x}, \tilde{\boldsymbol{\eta}}_t)$ s.t. $\|\boldsymbol{\eta}\|_p \leq \varepsilon / T$ (Table 1)
    $\boldsymbol{\eta}_{t+1} \leftarrow \boldsymbol{\eta}_t + \boldsymbol{\eta}_t^*$
**end for**
**return:** $\boldsymbol{\eta}^* \leftarrow \boldsymbol{\eta}_T$
```

Finally, computing $\boldsymbol{\eta}_t^*$ with the additional constraint $\|\boldsymbol{\eta}\|_{0,\mathcal{S}} = 1$ in Algorithm 1 leads to a multiple subset attack. For such an attack one must additionally subtract previously modified subsets from $\mathcal{S}$. This results in a new subset being altered at every iteration. Similarly, given a class label $\bar{l} \in [K]$, changing the objective function to

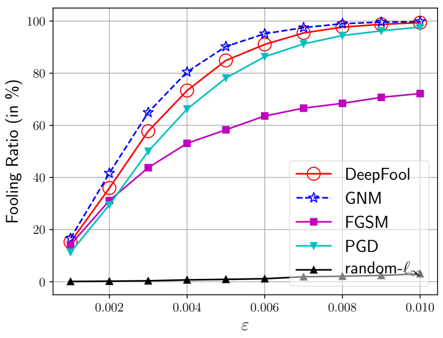
$$L_f(\mathbf{x}, \boldsymbol{\eta}) = f_{k(\mathbf{x})}(\mathbf{x} + \boldsymbol{\eta}) - f_{\bar{l}}(\mathbf{x} + \boldsymbol{\eta})$$

leads to a *targeted* attack, that is when the objective is to apply perturbations such that the outcome of classification is always some “target” class $\bar{l}$.

As we can see, different configurations for Algorithm 1 lead to known adversarial attacks from the literature. A summary of Algorithm 1 configurations with their corresponding attack from the literature is presented in Table 2. In classification, these methods are usually compared using the *fooling ratio*, that is the percentage of correctly classified inputs that are misclassified when adversarial perturbations are added. Visualizing the fooling ratio for different values of ε is often used to empirically asses the performance of an adversarial attack. For



(a) LeNet-5



(b) DenseNet-40

Fig. 2: Fooling ratio, from [7], of different adversarial attacks on vanilla DNNs on the MNIST dataset. (a): 5-layered LeNet architecture from [37], (b): DenseNet architecture from [30] with 40 layers.

example, in Figure 2 we observe the fooling ratio of different attacks on standard DNNs (not trained to resist adversarial attacks).

Algorithm	Objective function L	Iterative	Dithering
FGSM [24]	cross-entropy	×	×
DeepFool [48]	(2) with l chosen using $\hat{\rho}_1(f)$	✓	×
BIM [35]	cross-entropy	✓	×
PGD [43]	cross-entropy	✓	✓
Targeted	(2) with l fixed to the target	✓	✓
GNM [7]	(2)	✓	✓
Regression [8]	ℓ_2-norm of output perturbation	✓	✓

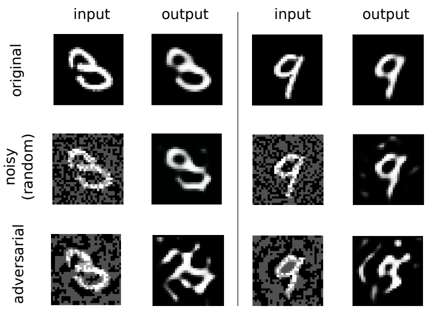
Table 2: Recovering Existing Attacks in Classification using this Framework.

Regression Problems and Other Learning Tasks The objective

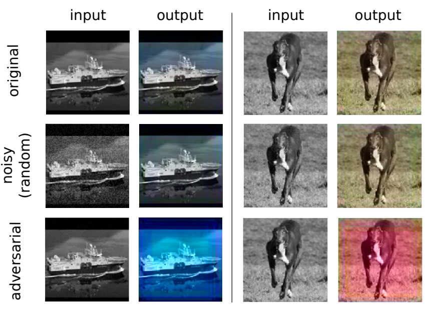
$$L_f(\mathbf{x}, \boldsymbol{\eta}) = -\|f(\mathbf{x}+\boldsymbol{\eta}) - f(\mathbf{x}+\boldsymbol{\eta})\|_2$$

can be used to attack regression problems as well. This possibility is investigated in [8], where the objective is to perturb the output of a regression model as much as possible. Two examples are provided using autoencoders and colorization [2] DNNs in Figure 3. In that figure, we observe how adversarial perturbations heavily distort the outcome of regression. Using the principles explicated in this section, other algorithms have been developed for attacking other types of learning systems.

[2] A colorization model predicts the color values for every pixel in a given gray-scale image.



(a) Autoencoder (96% compression)



(b) Image Colorization

Fig. 3: Adversarial examples for regression [8]. (a): MNIST autoencoder, (b): STL-10 colorization network.

In the field of computer vision, [45] constructed an attack on image segmentation, while [77] designed attacks for object detection. The Houdini attack [12] aims at distorting speech recognition systems. In addition, [54] tailored an attack for recurrent neural networks, and [40] for reinforcement learning. Adversarial examples exist for probabilistic methods as well. For instance, [33] showed the existence of adversarial examples for generative models. For regression problems, [71] designed an attack that specifically targets variational autoencoders.

Robustness metrics Going back to the definitions of Section 2.1, Theorem 1 shows that given a vector $\mathbf{x}$ and a score function $f(\cdot)$, the adversarial perturbation should have at least ℓ_p-norm equal to $\frac{L_f(\mathbf{x},\mathbf{0})}{\|\nabla_{\boldsymbol{\eta}} L_f(\mathbf{x},\mathbf{0})\|_q}$ to fool the linearized version of $f(\cdot)$. In other words if the ratio $\frac{L_f(\mathbf{x},\mathbf{0})}{\|\nabla_{\boldsymbol{\eta}} L_f(\mathbf{x},\mathbf{0})\|_q}$ is small, then it is easier to fool the network with ℓ_p-attacks. In that sense, Theorem 1 provides an insight into the stability of classifiers. Therefore, regularizing the loss function with $\frac{L_f(\mathbf{x},\mathbf{0})}{\|\nabla_{\boldsymbol{\eta}} L_f(\mathbf{x},\mathbf{0})\|_q}$ may lead to adversarial robustness. Moreover, one can also include dithering by regularizing with $\frac{L_f(\mathbf{x},\tilde{\boldsymbol{\eta}})}{\|\nabla_{\boldsymbol{\eta}} L_f(\mathbf{x},\tilde{\boldsymbol{\eta}})\|_q}$ with some randomly chosen $\tilde{\boldsymbol{\eta}}$.

In [48], the authors suggest that the robustness of the classifiers can be measured as

$$\hat{\rho}_1(f) = \frac{1}{|\mathcal{D}|} \sum_{\mathbf{x}\in\mathcal{D}} \frac{\|\hat{\mathbf{r}}(\mathbf{x})\|_p}{\|\mathbf{x}\|_p},$$

where $\mathcal{D}$ denotes the test set and $\hat{\mathbf{r}}(\mathbf{x})$ is the minimum perturbation required to change the classifier's output. Proposition 1 suggests that one can also use the following as the measure of robustness

$$\hat{\rho}_2(f) = \frac{1}{|\mathcal{D}|} \sum_{\mathbf{x}\in\mathcal{D}} \frac{L_f(\mathbf{x},\mathbf{0})}{\|\nabla_{\boldsymbol{\eta}} L_f(\mathbf{x},\mathbf{0})\|_q}.$$

The lower $\hat{\rho}_2(f)$, the easier it gets to fool the classifier and therefore it becomes less robust to adversarial examples. According to the experiments in [7], shown in Table 3, these two robustness metrics seem to be coherent when measuring the robustness of non-adversarialy trained DNNs.

	Test error	$\hat{\rho}_1(f)$ [48]	$\hat{\rho}_2(f)$ [7]	fooled >99%
FCNN (MNIST)	1.7%	0.036	0.034	ε =0.076
LeNet-5 (MNIST)	0.9%	**0.077**	**0.061**	ε =**0.164**
NIN (CIFAR-10)	13.8%	**0.012**	**0.004**	ε =**0.018**
DenseNet (CIFAR-10)	5.2%	0.006	0.002	ε =0.010

Table 3: Experiment from [7] showing the robustness measures for different DNNs on the MNIST and CIFAR-10 datasets. The acronyms FCNN denotes a standard fully connected neural network, while NIN refers to the network-in-network architecture from [39]. LeNet-5 and DenseNet are the same architectures used in Figure 2.

2.2 Black-Box Attacks and Universal Adversarial Perturbations

So far we have assumed that the adversarial attacker has perfect knowledge of the target classifier function $f(\cdot)$ as well as the input $\mathbf{x}$. By loosening of these requirements, into more realistic assumptions, new types of algorithms arise, namely

- **black-box attacks:** these methods correspond to the settings where the target classifier $f(\cdot)$ is unknown but the input $\mathbf{x}$ may still be known to the attacker,
- **universal adversarial perturbations:** these perturbations are designed to work regardless of the input $\mathbf{x}$, which is assumed to be unknown. Nevertheless, the classifier $f(\cdot)$ may be available to the attacker.

If both the target model $f(\cdot)$ and input $\mathbf{x}$ are unknown to the attacker, the adversarial attack would be a black-box as well as universal adversarial perturbation. These types of attacks are still possible by assuming partial or indirect knowledge about the input $\mathbf{x}$ and the classifier $f(\cdot)$. For example, the attacker may have access to a set of independent realizations $\{\mathbf{x}_1, \mathbf{x}_2, \dots\}$ of the input, which provides knowledge about the input distribution $P_{\mathbf{x}}$. Similarly, implicit information about the classifier can be inferred by observing the independent realizations of the pairs $\{(\mathbf{x}_1, f(\mathbf{x}_1)), (\mathbf{x}_2, f(\mathbf{x}_2)), \dots\}$. Finally, we may also have knowledge about the structure (number of layers, types of connections, activation functions, etc.) of the used classifier.

It is probably unexpected that some adversarial perturbations produce the same effect over different inputs and different DNNs architectures, although they are generated for a particular model. These universal adversarial perturbations are reported in [58,24] where the authors show the existence of such perturbations for various datasets and DNNs. This phenomena suggests that there exist certain common properties shared by adversarial perturbations that account for most of the success when attacking a system. This can explain why certain perturbations are able simultaneously fool a target DNN on different inputs. Adversarial examples are indeed transferable. In [73] the authors construct an attack such that adversarial examples can transfer from one random instance of a neural network to another. Surprisingly, these methods were proved to be effective against well known DNNs. Since no explicit knowledge about the DNN weights is required to compute these perturbations, they can be thought of as black-box attacks. Moreover, the authors showed that including such black-box adversarial examples into the training set significantly enhances the robustness of neural networks. Finally, the authors in [46] showed that there exist adversarial examples that are both universal and black-box, that is perturbations that are independent from target DNN and input.

Black-Box Attacks As discussed, in the black-box setting the classifier function $f(\cdot)$ is unknown, thus we cannot compute the gradient necessary for Algorithm 1. A common approach to circumvent this issue is to estimate the gradient by choosing a substitute model $\tilde{f}$ which is hoped to behave in a similar way as the unknown $f(\cdot)$. This concept is introduced in [52] under the assumption that the input $\mathbf{x}$ is known, as well as n independent realizations of $(\mathbf{x}, f(\mathbf{x}))$ denoted as $(\mathbf{x}_1, f(\mathbf{x}_1)), \ldots, (\mathbf{x}_n, f(\mathbf{x}_n))$. This method consists on the following two steps:

1. Train a substitute model $\tilde{f}$ that predicts $f(\mathbf{x})$, thus it resembles the target classifier.
2. Perform a white-box attack on the substitute model $\tilde{f}$ and hope it transfers to the target model $f(\cdot)$.

This concept is later extended in [41], where the authors make use of several substitute models, that is $\tilde{f}_1, \ldots, \tilde{f}_r$ for $r > 1$. In that work, adversarial perturbations are computed by approximately solving the following optimization for an ensemble of loss functions:

$$\min_{\boldsymbol{\eta}} -\log\left(\sum_{i=1}^{r} \alpha_i \mathcal{L}_{\tilde{f}_i}(\mathbf{x}, \boldsymbol{\eta})\right) + \lambda \|\boldsymbol{\eta}\|_p \,,$$

where $\lambda > 0$, $p > 1$, $0 < \alpha_i < 1$, $\sum_i \alpha_i = 1$ and $\mathcal{L}_f(\mathbf{x}, \cdot)$ is some positive loss function like the cross-entropy loss of $f(\cdot)$ at the point $(\mathbf{x} + \boldsymbol{\eta})$. The key idea of this method is that a perturbation $\boldsymbol{\eta}$ that is able to fool the classifiers $\tilde{f}_1, \ldots, \tilde{f}_r$ will most likely fool the unknown classifier $\tilde{f}_{r+1} \triangleq f$ as well.Note that it is also possible to generate norm-constrained versions of this method by approximately solving

$$\min_{\boldsymbol{\eta}} -\log\left(\sum_{i=1}^{r} \alpha_i \mathcal{L}_{\tilde{f}_i}(\mathbf{x}, \boldsymbol{\eta})\right) \text{ s.t. } \|\boldsymbol{\eta}\|_p \leq \varepsilon \qquad (15)$$

using the same methods described in Table 1.

Universal Adversarial Perturbations Given $0 < \delta < 1$, the paradigm of designing universal adversarial perturbations $\mathbf{u}$ can be summarized as follows

$$\begin{aligned} \text{Find :} \quad & \mathbf{u} \\ \text{s.t.} & \|\mathbf{u}\|_p \leq \varepsilon \\ & P_{\mathbf{x}}(k(\mathbf{x}+\mathbf{u}) \neq k(\mathbf{x})) \geq 1 - \delta \,. \end{aligned}$$

Note that in order to approximately solve this problem one needs information about the distribution of the input. A common assumption when designing universal perturbations is that the attacker has perfect knowledge of the classifier $k(\cdot)$, but only partial knowledge about $P_{\mathbf{x}}$ in the form of n independent realizations $\mathcal{X}_n = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ of $\mathbf{x} \sim P_{\mathbf{x}}$.

This problem is first approached in [46] by iteratively aggregating the perturbations that move $\mathbf{x}_1, \dots, \mathbf{x}_n$ to their corresponding decision boundaries. Given an input $\mathbf{x}_i$, such perturbations are computed by iteratively solving (9) in the same manner as in Algorithm 1. Then, in order to preserve the ℓ_p-norm constraint, these perturbations are projected into the ℓ_p ball of radius ε. A summary of this method is shown in Algorithm 2. In addition, some example pictures showing the effectiveness of this algorithm are shown in Figure 4. Note that this algorithm does not converge for an arbitrary choice of δ, thus additional stopping criteria are needed.

Algorithm 2 Universal adversarial perturbations with constrained ℓ_p-norm.

```
input: $\mathbf{x}_1, \dots, \mathbf{x}_n$, $k(\cdot)$, $\varepsilon$, $\delta$.
output: $\mathbf{u}^*$.
Initialize $\mathbf{u}^* \leftarrow \mathbf{0}$.
while $P_{\mathbf{x} \in \mathcal{X}_n}(k(\mathbf{x}+\mathbf{u}) \neq k(\mathbf{x})) \geq 1 - \delta$ do
  Shuffle $\mathcal{X}_n$
  for $i = 1, \dots, n$ do
    if $k(\mathbf{x}_i + \mathbf{u}^*) = k(\mathbf{x}_i)$ then
      Compute the minimal perturbation that sends $\mathbf{x}_i$ to the decision boundary:
      $\boldsymbol{\eta}^* \leftarrow \text{argmin}_{\boldsymbol{\eta}} \|\boldsymbol{\eta}\|_2$ s.t. $k(\mathbf{x}_i + \mathbf{u}^*) \neq k(\mathbf{x}_i)$
      Project the perturbation $\boldsymbol{\eta}^*$ into the $\ell_p$ ball of radius $\varepsilon$:
      $\mathbf{u}^* \leftarrow \text{argmin}_{\mathbf{u}} \|\mathbf{u}^* + \boldsymbol{\eta}^* - \mathbf{u}\|_2$ s.t. $\|\mathbf{u}\|_p \leq \varepsilon$
    end if
  end for
end while
return: $\mathbf{u}^*$
```

In a similar fashion as in (15), a universal adversarial perturbation can be obtained by minimizing and ensemble of objective functions, that is

$$\min_{\boldsymbol{\eta}} \sum_{i=1}^{n} L_f(\mathbf{x}_i, \boldsymbol{\eta}) \text{ s.t. } \|\boldsymbol{\eta}\|_p \leq \varepsilon,$$

where $L_f(\mathbf{x}, \cdot)$ is some objective function as in (2). Choosing L_f to be $L_f(\mathbf{x}, \boldsymbol{\eta}) = \|f(\mathbf{x}) - f(\mathbf{x}+\boldsymbol{\eta})\|_p^p$ and using the approximation $\|f(\mathbf{x}) - f(\mathbf{x}+\boldsymbol{\eta})\|_p \approx \|J_f(\mathbf{x})\boldsymbol{\eta}\|_p$, where $J_f(\mathbf{x})$ denotes the Jacobian matrix of $f(\cdot)$ at $\mathbf{x}$, the method proposed in [32] is obtained. This method is based on the insight that a perturbation that manages to fool several known inputs will most likely fool an unknown one as well.

3 Theoretical Explanations of the Nature of Adversarial Examples

Among various theories regarding the nature of adversarial examples, two directions can be singled out. One line of research focuses on local properties of classifiers, for example, decision boundaries of classifiers and their geometric properties. A notable example is the linearity hypothesis, proposed by the authors in [24], where the existence of adversarial images is attributed to the approximate linearity of classifiers. Another line of research tend to explain such phenomena by means of global properties of classifiers such as the topological dimension of their feature spaces or the sparsity of weight matrices in DNNs. We present some of the most important results along these two lines.

There is, however, another theoretical question raised in the literature. After some experimental results witnessed a seemingly opposing relation between adversarial robustness and generalization, researchers discussed formally the connection between generalization properties of DNNs and their adversarial robustness. The central question is whether adversarial robustness is realized only at the cost of worse generalization. This question is the subject of the last part of this section.

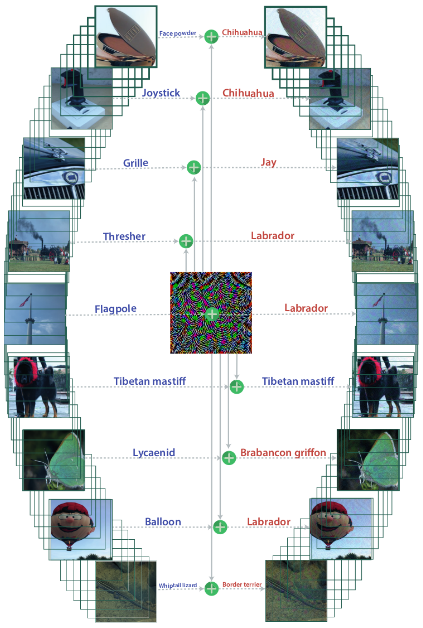
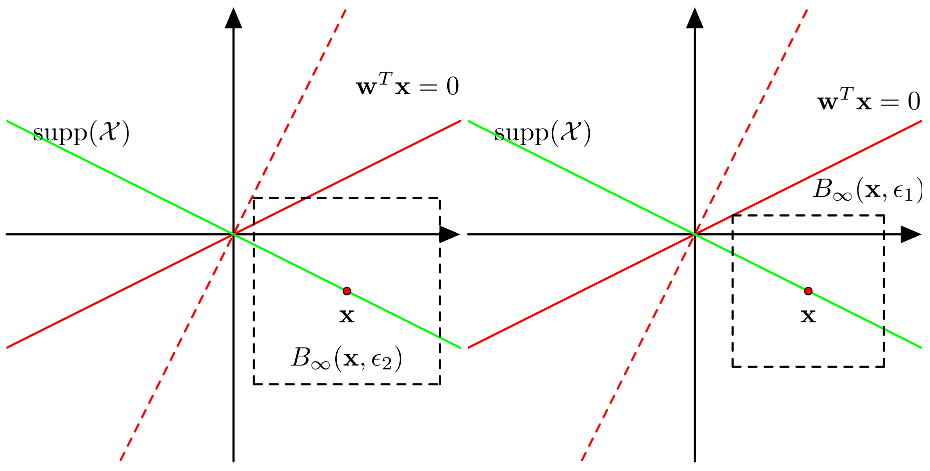


Fig. 4: The authors in [46] add a universal perturbation (center image) that is able to mislead the classification of several images.

3.1 Linearity Hypothesis and Curvature of Decision Boundaries

How can the existence of adversarial examples be explained? The experimental results showed that adversarial examples are also misclassified by neural networks


$\mathbf{w}^T\mathbf{x} = 0$
supp($\mathcal{X}$)
$\mathbf{x}$
$B_\infty(\mathbf{x}, \epsilon_2)$
$\mathbf{w}^T\mathbf{x} = 0$
supp($\mathcal{X}$)
$B_\infty(\mathbf{x}, \epsilon_1)$
$\mathbf{x}$

(a) Not-tilted boundary (b) Tilted boundary

Fig. 5: Adversarial robustness of tilted boundaries: (a) the dashed line is the ground truth linear classifier for the data supported on $\mathcal{X}$, (b) the solid line, a tilted boundary, yields the same risk as the ground truth but is fooled with smaller ℓ_∞-perturbations

trained either on a different dataset or with different hyper-parameters. Therefore, this phenomenon cannot be attributed to overfitting to a particular model. It is first conjectured in [24] that, the adversarial examples exists because neural networks are well approximated, locally, by linear classifiers. This hypothesis, known as the linearity hypothesis, is supported by easy generations of adversarial examples using first order approximation of neural networks.

For deep neural networks, this claim is mainly experimentally substantiated. The attacks applied to the linear approximation of neural networks around an instance manage to effectively generate misclassified examples, and therefore, the effectiveness of first-order approximation attacks testifies, according to [24], to the linearity of these models.

To elucidate this claim, consider a linear classifier for binary classification tasks with the parameter $\mathbf{w} \in \mathbb{R}^M$. The classification rule is given simply by $\text{sign}(\mathbf{w}^\mathsf{T}\mathbf{x})$. In this linear setting, the FGSM provides the best ℓ_∞-bounded adversarial perturbation, which is given by $\boldsymbol{\eta} = -\varepsilon\text{sign}(\mathbf{w})$. The total perturbation caused at the output is $\boldsymbol{\eta}^\mathsf{T}\mathbf{w}$, which is equal to $-\varepsilon\|\mathbf{w}\|_1$. The value of $\|\mathbf{w}\|_1$ can be as large as $\sqrt{M}$ for $\mathbf{w}$'s with unit norm. Therefore, a small ℓ_∞-perturbation can be blown up by $\sqrt{M}$. Two conclusions can initially be drawn from this example. First, small input perturbations can incur large output perturbations for high dimensional linear classifiers. The authors in [24] argue accordingly for the existence of adversarial examples. High dimensional approximately linear classifiers can blow up small perturbations at their output. It is, according to [24], the approximate linearity of Convolutional Neural Networks (CNNs) that explains the existence of adversarial examples for the ImageNet classification problem. Although CNNs have many non-linearities, the parameters of CNNs after training are chosen so that the non-linearity of the architecture is diminished.

The second conclusion points to the ℓ_1-norm of the parameter $\mathbf{w}$ as the key to control the effectiveness of ℓ_∞-attacks, which can be used to design robust neural networks as we will see later. In general the ℓ_q-norm of $\mathbf{w}$ controls the output perturbation for ℓ_p-attacks where (p, q) are dual to each other[3]. However, this example only shows that for some linear classifiers, small input perturbations changes significantly the output.

The linearity hypothesis did not remain unchallenged. After all, the non-linear machine learning algorithms were equally vulnerable to adversarial examples. In [60], adversarial examples were generated by imposing a similarity constraint between the hidden representation of the perturbed image in DNN and the hidden representation of an image from a different class. An optimization problem is used to minimize the difference between hidden representations with constraints on the perturbation. The method yielded adversarial images which differed from other adversarial images in that they did not rely, even implicitly, on linear approximations of the model. They could not be generated from linear approximations. The linearity hypothesis cannot explain the existence of these adversarial images.

For ImageNet classification problem with CNNs, the authors in [42] examined the linearity hypothesis by comparing $f(\boldsymbol{\eta})$ and $f(\mathbf{x} + \boldsymbol{\eta}) - f(\mathbf{x})$. These two values are equal for linear classifiers, thus it is used to measure the linearity of CNNs. The conclusion of this study goes against the linearity hypothesis. The experimental studies of these values do not indicate any linear structure for CNNs. According to the modified linearity hypothesis, proposed in [42], CNNs are locally linear around the objects recognized by the model. The locality assumption is crucial here. Deep neural networks are non-linear in general and cannot be replaced completely by a linear classifier. However, the local linearity hypothesis claims that these models in a neighborhood of an instance can be approximated by a linear classifier. Additionally, the CNNs can be non-linear around those instances that are not recognized by the model.

The local linearity hypothesis implies that the decision boundaries around an instance can be approximated by a linear boundary. The geometric notion for characterizing the linearity of a surface is its curvature, as decision boundaries for binary classifiers are surfaces on higher dimension. The differential geometric notions of curvature are complex to characterize for DNNs. Therefore, in [19] the authors came up with an alternative, yet related definition of curvature. Consider the decision boundary for a binary classification problem [4] with the classifier $f(\cdot)$

[3] We call the pair (p, q) dual if the corresponding norms are dual. In particular $1/p + 1/q = 1$.

[4] In this section, we focus mainly on binary classification examples assuming that the results can be extended without particular difficulty to multi-class classification problems.

defined as

$$\mathcal{B} = \{\mathbf{x} : f(\mathbf{x}) = 0\},$$

and decision regions given by

$$\mathcal{R}_1 = \{\mathbf{x} : f(\mathbf{x}) > 0\} \quad \text{and} \quad \mathcal{R}_{-1} = \{\mathbf{x} : f(\mathbf{x}) < 0\}.$$

The curvature of the decision boundary $\mathcal{B}$ with respect to ℓ_q-norm is defined by

$$\kappa_q(\mathcal{B}) = \frac{1}{r_{\min}} \quad \text{where}$$
$$r_{\min} = \inf_{\mathbf{x}\in\mathcal{B}} \min_{i\in\{-1,1\}} \sup_{\mathbf{x}_o\in\mathbb{R}^M} \{\|\mathbf{x}_o - \mathbf{x}\|_q : \mathbf{B}_q(\mathbf{x}_o, \|\mathbf{x}_o - \mathbf{x}\|_q) \subseteq \mathcal{R}_i\}$$

and $B_q(\mathbf{x}, \varepsilon)$ denotes the ℓ_q-ball of radius ε centered at $\mathbf{x}$. In other words $r_{\min}$ is obtained by first finding at each point $\mathbf{x}$, on the decision boundary, the largest radius of ℓ_q-balls that contain $\mathbf{x}$ while being contained in $\mathcal{R}_1$ and $\mathcal{R}_{-1}$. The radius is infinity for all $q \geq 1$ when the decision boundary is flat, which means that the local curvature is equal to zero at this point. The minimum of such radii for all $\mathbf{x}$, that is $r_{\min}$, points at the most curved portion of the surface. The global curvature of the surface $\mathcal{B}$ is the inverse of $r_{\min}$. A linear classifier yields decision boundaries with zero curvature, and a small curvature surface might be completely flat in most of its points. It turns out that the ℓ_q-curvature $\kappa_q(\mathcal{B})$ determines the robustness against ℓ_p-attacks with (p, q) as a dual pair.

It is based on this notion of curvature and for ℓ_2-attacks that the authors in [19] compare the robustness of classifiers to random noise and adversarial noise and characterize it according to the curvature of decision boundaries[5]. The random noise is modeled as a random direction and the robustness against random noise at $\mathbf{x}$, denoted by $\rho_M(\mathbf{x})$, is defined by the minimum ℓ_2-norm of a random vector required to change the label of $\mathbf{x}$. The adversarial robustness, denoted by $\rho(\mathbf{x})$, is the minimum ℓ_2-norm of a perturbation particularly designed to change the label.

Theorem 4 ([19, Theorem 2]). *Suppose that for a binary classifier the curvature* $\kappa_2(\mathcal{B})$ *satisfies*

$$\kappa_2(\mathcal{B}) \leq \frac{0.2}{\zeta_2(\delta) M \rho(\mathbf{x})},$$

then with probability at least $1 - 4\delta$ *it holds*

$$\left(1 - 0.625 M\rho(\mathbf{x})\kappa_2(\mathcal{B})\zeta_2(\delta)\right)\sqrt{M\zeta_1(\delta)} \leq \frac{\rho_M(\mathbf{x})}{\rho(\mathbf{x})} \leq \left(1 + 2.25 M\rho(\mathbf{x})\kappa_2(\mathcal{B})\zeta_2(\delta)\right)\sqrt{M\zeta_2(\delta)},$$

where

$$\zeta_1(\delta) = \left(1 + 2\sqrt{\ln(1/\delta)} + 2\ln(1/\delta)\right)^{-1},$$
$$\zeta_2(\delta) = \left(\max\left((1/e)\delta^2, 1 - \sqrt{2(1-\delta^2)}\right)\right)^{-1}.$$

[5] They consider semi-random noise as well, however, we restrict ourselves to simple random noise.

The theorem implies that if the curvature of decision boundaries are small enough, the robustness of random noise is scaled with $1/\sqrt{M}$ of the adversarial perturbation. In other words, in higher dimensions, classifiers with flat boundaries can be robust to random noise even if they are not robust to adversarial examples. Note that the curvature of non-smooth boundaries can be huge, rendering the above theorem non-informative. The curvature for multi-class classification, however, is characterized by the curvature of pairwise boundaries which does not include high curvature junctions. The intersection of these boundaries might have high curvature, but this issue does not matter in the above theorem where the pairwise boundaries are considered. Although only for ℓ_2-attacks, this result is extended to ℓ_p-attacks in [21], where the small curvature condition is replaced by a condition called locally approximately flat decision boundaries.

The geometric properties of decision boundaries are further investigated in [47] for universal adversarial perturbations. Universal adversarial perturbations exist for both flat and curved decision boundaries. The essential to the existence of universal adversarial perturbations are shared directions along which the surface is positively curved. The above results compare random noise and adversarial perturbations. The result rely on low curvature assumptions of decision boundaries. The conclusion can be put with a flavor of blessing of dimensionality. That is, the locally flat classifiers are more robust to random noise particularly in higher dimensions.

The local linearity assumption is further refined in [16]. The flatness of decision boundaries can be violated in some directions, and nevertheless, the adversarial vulnerability persists. The authors show that the adversarial examples can exist if the boundaries are flat along most of the directions and highly curved only around few directions. This claim is additionally supported by [56,20] where the curvature profile of deep networks are numerically characterized and is shown to be highly sparse, which implies that the boundaries are not flat overall but effectively only along most directions.

It is worth to finish the discussion around the linearity hypothesis by referring to a recent result that sheds some doubts and raises some questions about the role of linearity in adversarial robustness. Contrary to all above claims, the work [47] shows that the adversarial training, a powerful and consistent defense against adversarial attacks, leads to significant decrease in the curvature of loss functions. The connection runs in both directions as training DNNs with curvature regularization tends to improve the adversarial robustness. As long as the curvature of loss functions affects the curvature of decision boundaries, the result stands out as a strong counter-argument for linearity hypothesis and opens new challenges for it.

3.2 Boundary tilting and other explanations

The linear hypothesis is not the only available theory. Other theories attribute adversarial robustness to other features of classifiers. A simple intuition already emerged from our discussion of linear classifiers. The ℓ_∞-attacks for linear classifiers with unit-norm parameters $\mathbf{w}$ generated a perturbation equal to $-\varepsilon\|\mathbf{w}\|_1$.

Therefore, among all unit-norm $\mathbf{w}$'s, the most robust classifiers are those with smallest ℓ_1-norm, which are also the sparsest possible vectors.

For binary classification problems, the authors in [25] showed theoretically that the adversarial robustness decreases when ℓ_1-norm of $\mathbf{w}$ increases. We introduce some definitions before stating the theorem. Suppose that the instances and labels $(\mathbf{x}, y)$ follow a distribution $P_{\mathbf{x},y}$. The adversarial robustness for this probabilistic model is defined as

$$\rho_\infty = P_{\mathbf{x},y}[y \neq \operatorname{sign}(\mathbf{w}^\mathsf{T}\mathbf{x}_{\text{adv}})],$$

where $\mathbf{x}_{\text{adv}}$ is the ℓ_∞-perturbed instance and given by $\mathbf{x}_{\text{adv}} = \mathbf{x} - \varepsilon y \operatorname{sign}(\mathbf{w})$. Let us define $\boldsymbol{\mu}_k$ for $k \in \{1, -1\}$ as follows

$$\boldsymbol{\mu}_k = \mathbb{E}(\mathbf{x}|y = k, \operatorname{sign}(\mathbf{w}^\mathsf{T}\mathbf{x}) = k).$$

We can now state the theorem.

Theorem 5 ([25, Theorem 3.1]). *For a binary classification problem with uniformly distributed labels, if the accuracy of a classifier is given by t then the adversarial robustness ρ_∞ against ε bounded ℓ_∞ attacks is given by*

$$\rho_\infty \leq \frac{t\mathbf{w}^\mathsf{T}(\boldsymbol{\mu}_{+1} - \boldsymbol{\mu}_{-1})}{2\varepsilon\|\mathbf{w}\|_1}.$$

The denominator of the bound on the right hand side contains the ℓ_1-norm of $\mathbf{w}$. Therefore, as the authors in [25] maintain, among those linear classifiers with a similar discriminatory capability, those with the smallest ℓ_1-norm perform better under ℓ_∞-attacks. The small ℓ_1-norm implies a larger ρ_∞, which means better robustness.

The theorem, however, provides only an upper bound, and, although it can characterize the negative effect of large ℓ_1-norm on robustness, it cannot necessarily guarantee that the small ℓ_1-norm promotes robustness nor that small ℓ_1-norms necessarily lead to sparse $\mathbf{w}$. The claim, however, seems to hold, as experimental findings seem to support the idea that the sparsity of weights promote adversarial robustness. Besides, since the difference $\boldsymbol{\mu}_{+1} - \boldsymbol{\mu}_{-1}$ is independent of the norm of $\mathbf{w}$, the inner product $\mathbf{w}^\mathsf{T}(\boldsymbol{\mu}_{+1} - \boldsymbol{\mu}_{-1})$ scales with the norm of $\mathbf{w}$. In this light, another reading of this theorem suggests that among all unit ℓ_1-norm $\mathbf{w}$'s, the one with smallest $\mathbf{w}^\mathsf{T}(\boldsymbol{\mu}_{+1} - \boldsymbol{\mu}_{-1})$ restricts robustness the least. To summarize the theorem, a first step toward robustness of linear classifiers is to find the smallest ℓ_1-norm $\mathbf{w}$ for which the inner product $\mathbf{w}^\mathsf{T}(\boldsymbol{\mu}_{+1} - \boldsymbol{\mu}_{-1})$ is high enough.

Another explanation of adversarial examples, introduced in [72], starts from the assumption that the data lies on a low-dimensional manifold in higher dimensional space, and many classifiers exist with similar accuracy. This is shown in Figure 5 using a simple example of linear manifolds and linear classifiers. We assume that the data lies on a linear subspace and the dashed line represents the boundary of an optimal Bayes linear classifier for the data distribution with zero error. However, the rotated versions of this linear classifier, for example the one with the solid line as the boundary, yield the same accuracy. The main difference between these classifiers is their robustness to adversarial examples. If the linear boundary is tilted so that it lies close to the data subspace, the smaller ℓ_∞-norm perturbation can fool the classifier. This can be seen in Figure 5 as the ℓ_∞-ball touching the tilted classifier is smaller that the original not-tilted classifier. This is known under boundary tilted hypothesis. Under this hypothesis, the adversarial vulnerability of classifiers arises from the tilted classification boundary close to the data manifold. A further exploration of the linear classifier example can reveal that some of the tilted boundaries can indeed improve the robustness.

3.3 Feature Selection and No Free Lunch Theorems for Adversarial Robustness

Many classifier functions can be clearly decomposed into feature extraction and classification parts. In [75], adversarial robustness is shown to be affected by the feature selection part of the model. The results of [75] rely on the assumption that there is an oracle classifier function $g(\mathbf{x})$ that generates the ground truth labels. For image classification problems, it is simply the human eye. Classifiers $f(\cdot)$, in particular $g(\cdot)$, are decomposed into a feature extraction part $e_f(\cdot)$ and a classifier part $c_f(\cdot)$. The feature spaces of a classifier is the image of the domain set $\mathcal{X}$ under the feature extraction $e_f(\cdot)$. Feature spaces are assumed to be metric spaces. Denote the oracle feature space by $(\mathcal{X}_g, d_g)$ where d_g is the respective metric, and $(\mathcal{X}_f, d_f)$ similarly for a classifier $f(\cdot)$.

Adversarial perturbations do not change the oracle decision, i.e., $g(\mathbf{x}) = g(\mathbf{x} + \boldsymbol{\eta})$ nor the feature extraction:

$$d_g(e_g(\mathbf{x}), e_g(\mathbf{x} + \boldsymbol{\eta})) < \delta.$$

However, the classifier is fooled ($f(\mathbf{x}) \neq f(\mathbf{x} + \boldsymbol{\eta})$). A classifier is called (ε, δ)-robust if for all $\mathbf{x}, \mathbf{y} \in \mathcal{X}$ for which $g(\mathbf{x}) = g(\mathbf{y})$ and $d_g(e_g(\mathbf{x}), e_g(\mathbf{y})) < \varepsilon$ then with probability at least $1 - \delta$ it holds that $f(\mathbf{x}) \neq f(\mathbf{y})$.

Theorem 6 ([75, Theorem 3.2-3.4]). *Let a classifier $f(\cdot)$ be continuous almost everywhere and $g(\cdot)$ be the oracle classifier. The classifier $f(\cdot)$ is (ε, δ)-robust to adversarial examples if and only if the topology of the feature space $(\mathcal{X}_f, d_f)$ is finer than the topology of the oracle feature space $(\mathcal{X}_g, d_g)$.*

As a direct corollary of the above theorem, when the two features spaces are Euclidean spaces of dimension n_g and n_f, then the classifier $h(\cdot)$ is robust if and only if $n_f < n_g$. The above theorem implies first that the selection of features and feature spaces is crucial for adversarial robustness. Although the assumption of an oracle function and a unique suitable feature space can be contested, the theorem applies to any two classifiers and states that if a perturbation does not fool $g(\cdot)$ then it does not fool $f(\cdot)$. So among the classifiers those with feature spaces of finer topology, or lower dimension in Euclidean spaces, are favored for adversarial robustness.

The importance of selecting proper features is addressed in other works such as [17,18]. A toy example is used in [17,18] to show that linear classifiers are unable to use more robust features of an image for adversarial robust classification, unlike quadratic classifiers that are more robust in that example. For ℓ_2-attacks, the authors point out that the adversarial robustness is directly related to the so-called distinguishability measure of classes and the risk of the classifier. The distinguishability measure can be seen as low flexibility of classifiers in general compared to the difficulty of the classification task. We state a simplified version of their theorem for linear classifiers using the definition of $\rho(\mathbf{x})$ from Theorem 4.

Theorem 7 ([17, Theorem 4.1]). *For a binary classification task with uniformly distributed labels and $\|\mathbf{x}\|_2 \leq B$ a.e., the adversarial robustness of a linear classifier* $\operatorname{sign}(\mathbf{w}^\mathsf{T}\mathbf{x})$ *with accuracy t satisfies*

$$\mathbb{E}(\rho(\mathbf{x})) \leq \frac{1}{2}\|\mathbb{E}_{P_{\mathbf{x}|y=+1}}(\mathbf{x}) - \mathbb{E}_{P_{\mathbf{x}|y=-1}}(\mathbf{x})\|_2 + 2Bt.$$

The distinguishability measure $\|\mathbb{E}_{P_{\mathbf{x}|y=+1}}(\mathbf{x}) - \mathbb{E}_{P_{\mathbf{x}|y=-1}}(\mathbf{x})\|_2$ is a feature of classification problem and not dependent on the classifier. However, an unexpected conclusion of the theorem is that if the classification task is difficult, in that the distinguishability measure is small, the risk of the classifier becomes dominant in the upper bound and inversely related with the robustness. Therefore, low risk classifiers have less adversarial robustness for difficult classification tasks.

The inverse connection of risk and robustness is further explored in [74] through a binary classification example. An instance of data is given by $\mathbf{x} = (x_1, \ldots, x_n, x_{n+1})^\mathsf{T}$ and it is related to its label y randomly as follows. The first entry is a Bernoulli random variable with $P(x_1 = y) = p$ and the other entries, x_i, are normal distributed random variables with mean value ξy and unit variance. A linear classifier with $\mathbf{w} = (0, 1/n, \ldots, 1/n)$ can be shown to achieve more than 0,99 accuracy if $\xi = \Theta(1/\sqrt{M})$. However, this classifier can achieve an adversarial accuracy at most 0.01 under the ℓ_∞-attack with $\varepsilon = 2\xi$. However, if one uses only the first feature x_1 for the classification both standard and adversarial accuracies are 0.7. The data consists of, on the one hand, robust features with less accuracy and, on the other hand, informative and non-robust features. We might ask whether this tension can be circumvented using a smart combination of features so that the adversarial robustness does not come at the price of accuracy. The authors of [74] answer negatively by stating a no free-lunch theorem for adversarial robustness.

Theorem 8 ([74, Theorem 2.1]). *Any classifier with standard accuracy at least $1 - \delta$ on the above problem cannot achieve adversarial accuracy more that $\frac{1-p}{p}\delta$ against ℓ_∞-bounded perturbations with $\|\boldsymbol{\eta}\|_\infty \geq 2\xi$.*

The importance of feature selection for adversarial robustness is highlighted already in [74]. A similar result is obtained in [15] for a class of data distributions satisfying W_2-Talagrand transportation-cost inequality. The condition is intuitively related to the curvature of decision regions and matches the previously mentioned intuition that low curvature decision boundaries entail adversarial vulnerability. From another perspective, ε-bounded ℓ_p-adversarial attacks manage to alter the label of instances that are included in the ε ℓ_p-boundary of decision regions. The adversarial problem so formulated, naturally, can be cast as the study of blowing-up property of decision regions– a problem well studied by concentration results and isoperimetric inequalities. A similar approach is followed in [44,22,14] for instances following Gaussian distribution and uniform-distribution over hypercubes.

3.4 Generalization Bounds for Adversarial Examples

The no-free-lunch-theorem states that adversarial robustness of machine learning algorithms does not align in general with their risk. Adversarial training of DNNs, indeed, confirm the same point that adversarial robustness is obtained at the cost of degraded generalization. According to an example in [43], the adversarial accuracy of 96 percent for ResNet dataset trained on CIFAR-10 comes with a test accuracy of 47 percent. In [68], the authors attribute this trade-off to the definition of adversarial robustness. They propose another definition of adversarial robustness for which no trade-off is observed between adversarial robustness and accuracy.

From these indications, therefore, emerge questions regarding generalization properties of adversarially robust algorithms. We summarize some of the progresses in that direction that use statistical learning theory as the framework for studying generalization properties of learning algorithms. An interested reader can refer to the excellent manuals [2,65] for introduction to fundamental notions of statistical learning theory.

Statistical learning theory explains generalization properties of many classical learning algorithms using notions like VC-dimension, Rademacher complexity and uniform convergence. A similar approach, however, cannot be directly applied to understanding generalization for neural networks. The large number of parameters in DNNs renders many of these bounds ultimately useless as these models are capable of fitting arbitrary large number of random instances with random labeling [80]. A solution is to use a suitable normalization. It has been shown that the margin based normalization can be used to obtain tight generalization bounds by Rademacher complexity or PAC-Bayesian methods that match experimental results [50,23,3,10].

We already discussed some works relating adversarial robustness and accuracy, all of them derived for a class of data distributions. In this section, we focus on sample complexity bounds for adversarially robust generalization and see whether the available bounds attest to difficulty of training adversarially robust and yet accurate models. A first indication in this direction can be traced to [64] where it is shown that for Gaussian model of data, the sample complexity of robust learning for M-dimensional data is $\Theta(\sqrt{M})$ times larger that standard learning, hence, more difficulty of the former. The gap is information theoretic. PAC-learning for the adversarial setting is an open problem, although some bounds exist for binary linear classifiers obtained in terms of VC-dimension [13]. The former result, however, show no negative effect of robust training on generalization, counter to the above intuition. A generalization bound is also obtained in [6] when the set of adversarial perturbations is finite. The sample complexity of binary classification depends $k\log(k)\mathrm{VC}(\mathcal{H})$ where $\mathrm{VC}(\mathcal{H})$ is the VC-dimension of the hypothesis class $\mathcal{H}$ and k is the number of different adversarial perturbations. The result is not directly applicable to standard adversarial attacks where the set of possible perturbations is not finite, however, it points to the larger sample complexity of robust learning.

There is a difficulty with VC-dimension bounds when they are applied to DNNs. As it is explained in [78], VC-dimension bounds depend usually on the number of model parameters, which is very large for DNNs. The corresponding sample complexity bound becomes unreasonably large beyond typical available datasets. Rademacher complexity bounds, on the other hand, depend mostly on inherently smaller quantities like the norm of weight matrices and, therefore, are more appropriate for establishing generalization property of DNNs.

Rademacher complexity bounds for adversarial robustness is obtained in [31,78]. They use different techniques for deriving their bounds and have different scope of applicability. Nevertheless, both works contain Rademacher complexity bounds for binary and multi-class classification and are applicable to neural networks. Both works use surrogate adversarial loss. In particular, the authors in [78] build on semidefinite programming (SDP) relaxation techniques of [57]. We do not expand on the technical results and, instead, state qualitatively some implications of these results.

As we discussed above shortly, Rademacher complexity bounds depend mostly on the norm of weight matrices. However, the lower bound on Rademacher complexity of neural networks for robust training in [78] has additional dependence on the dimension for the ℓ_∞-attacks. The dependence disappears only if the weight matrix of the first layer has bounded ℓ_1-norm. In absence of this assumption, this bound confirms the hypothesis that robust training is more difficult than standard training. The technique employed in [31], however, yields upper bounds in which the effect of adversarial perturbations appears as an additive term in the generalization bound. The authors, therefore, conclude that it should not be impossible to obtain both high adversarial robustness and high accuracy. Although a final verdict seems to be far reaching at the moment, new regularization techniques arises from these generalization bounds that can be used during training for robust learning with high accuracy.

4 Defenses Against Adversarial Attacks

There exist several types of defenses against adversarial examples, as well as subsequent methods for bypassing them. It is difficult to point out, at the time of writing this chapter, a consensus on the effective defense against adversarial examples with the possible exception of adversarial training. For instance, the authors in [11] proposed three attacks to bypass defensive distillation of the adversarial perturbations [55]. Moreover, the attacks from [5], bypassed 7 out of 9 non-certified defenses of ICLR 2018 that claimed to be white-box secure. Adversarial training, however, adds adversarial examples to the training set and is the most commonly accepted defense against adversarial attacks. In what follows, we discuss some difficulties of adversarial training as well as those methods that try to promote robustness merely through regularization techniques.

4.1 Obfuscated Gradients and Adversarial Training

The rise of adversarial perturbations in computer vision has motivated further research on defending against such perturbations. To that end several defenses against adversarial examples, such as [38,61,66], have been designed. Since many adversarial attacks make use of the classifier's gradient with respect to some objective function, as in Algorithm 1, initial works on adversarial defenses rely on distorting and hiding the information about that gradient. In [5], these techniques were said to *obfuscate* the gradient. More precisely, obfuscating the gradient may be done in either of the following manners.

- **Shattered Gradients** appear when the defense mechanism is not differentiable, numerically unstable, or intentionally has misleading gradients.
- **Stochastic Gradients** occur when the defense method is based on introducing randomness into the prediction. Such randomness is added to prevent the attacker from estimating the gradients.
- **Exploding Gradients** happen when the defense algorithm consists on recursive evaluations of the DNN function. In other words, the output of one DNN evaluation is the input of the next one. This type of computation implicitly transforms the original DNN into a extremely deep neural network, which may lead gradients to explode (or vanish) during inference.

Despite the apparent success of this type of defenses, in [5] it is shown that such mechanisms give a false sense of security. The main reason behind this phenomena is that obfuscating the gradients of model does not necessarily increases its robustness against adversarial perturbations, instead it prevents specific algorithms to find them. In other words, for such models adversarial perturbations still exist; they are just harder to find using certain methods. Therefore, although a model with obfuscated gradients can be robust against a specific adversarial attack, it may still be vulnerable to others. Using this idea, the authors in [5] provide the following conditions to identify models that exhibit this problem.

- **One-step attacks perform better than iterative attacks.**
- **Black-box attacks perform better than white-box attacks.**
- **Attacks with large ε do not reach 100% fooling ratio.**
- **Random sampling finds adversarial examples, while adversarial attacks don't.**

If a model satisfies one of the above conditions, it suffers from the obfuscated gradient problem. Using these guidelines, the authors identified that 7 out of

9 defenses accepted to ICLR 2018, that were deemed to be white-box secure, suffered from this issue. In addition, the authors fooled those defenses using customized attacks.

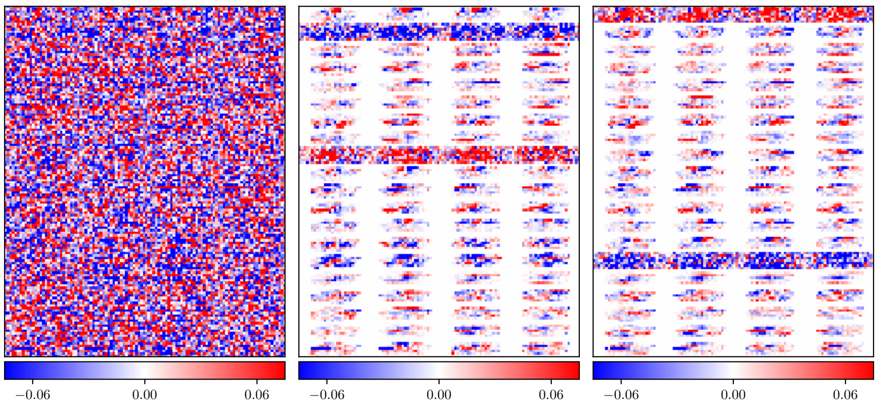
So far, the most successful defenses against adversarial attacks consist of adding adversarial examples to the training set. This is known as adversarial training. Initial attempts to perform adversarial training using the FGSM proved to suffer from obfuscated gradients. This occurs since a DNN trained solely with the FGSM learns to shatter the gradients that are in a close vicinity to the data samples, such that the gradients used for the FGSM point into misleading directions. While this process may mislead the FGSM it is still vulnerable to other perturbations, for instance black-box attacks. To overcome this issue the dithering mechanism proposed for the PGD attack is employed in [43], along with large ε values [6] during adversarial training. This approach provided diverse sets of random adversarial examples, which prevented DNNs from obtaining low fooling ratios by shattering the gradients around the data samples. In other words, the randomness of the starting point in the PGD attack prevents the model from overfitting the perturbations.

From these initial findings it is concluded that *diversity* in the adversarial examples used for training is necessary in order to prevent DNNs for overfitting to specific types of perturbations. To that end, in [73] the authors include black-box perturbations into the training set. This is carried out using substitute models, as well as ensembles of these models, in the objective function of white-box attacks (such as PGD) as described in Section 2.2.

4.2 Robust Regularization

Despite the success of adversarial training on promoting robustness, these methods either suffer from obfuscated gradients (e.g., when the FGSM is employed) or are deemed to be computationally expensive, since iterative methods require several evaluations of the DNN function to compute a single adversarial example. In [43] it is observed that adversarial training induces sparsity on the weights of the first convolutional filters of CNNs trained with the MNIST dataset. Similarly, the authors in [36] observe that adversarial training induces low-rank structures in the weight matrices of the DNN, as well sparsity. As an example the authors provide a visualization of the weights in the first layer of a DNN, shown in Figure 6, where the simultaneously low-rank and sparse structure of such weights is clearly visible. This is additionally confirmed by looking at the mutual information between the input and the layers. Information theoretically, increasing adversarial robustness coincides with decreasing the mutual information that indicates more compression of the input in hidden layers. These results serve as motivation for aiming research towards finding the key properties that lead to robustness of DNNs. The idea is to propose a metric for robustness and promote it during training. A common technique for promoting specific properties during training

[6] In that work, the ℓ_∞-constraint $\|\boldsymbol{\eta}\|_\infty \leq \varepsilon = 0.3$ is employed to train models where the input values were between 0 and 1.



(a) Natural training **(b)** FGSM training **(c)** PGD training

Fig. 6: Reshaped input weight matrix $\mathbf{W}^1 \in \mathbb{R}^{20\times 784}$ of a DNN, from [36], after natural training as well as adversarial training with $\varepsilon = 0.05$. A simultaneously low-rank and sparse structure is observed in the weights after adversarial training.

is to add a penalty term in the loss function, known as regularization term, that penalizes undesired properties of the classifier function. Here are some examples of robust regularization.

- **Sparsity:** In [26,76] the authors argue that sparsity of the weight matrices of a DNN promotes robustness against adversarial examples. They propose to add a regularization term with the sum of the ℓ_1-norm of the weight matrices involved, which is known to promote approximately sparse solutions. In addition, the authors make use of *pruning* [7] to impose arbitrary sparsity levels.
- **Low-Rankness:** In [62,36] it is observed that adversarial training induces low-rank structures on the weight matrices of DNNs. Motivated by this phenomena, low-rank regularization techniques are proposed. In [62] the authors explicitly constrain the rank of weight matrices in the optimization algorithm used for training. On the other hand in [36] the nuclear norm of the weight matrices is employed as a regularization term in the training loss. The nuclear norm of a matrix can be written as the ℓ_1-norm of its vector of singular values, thus using it as a regularization term promotes sparsity in that vector of singular values (i.e., low-rankness).
- **Norm of the network's Jacobian:** In [51], the authors aim at minimizing the ℓ_2-norm of the output perturbation, that is $\|f(\mathbf{x}) - f(\mathbf{x}+\boldsymbol{\eta})\|_2$. Assuming

[7] Pruning consists of setting to zero smallest weights (in absolute value) of the a given weight matrix, thus enforcing a certain level of sparsity. The amount of weights to be set to zero is arbitrarily chosen. Usually pruning requires an extra phase of retraining (fine-tunning of the remaining non-zero weights) to compensate for the performance degradation caused by the initial manipulation of the weights.

$\|\boldsymbol{\eta}\|_2 \leq \varepsilon$, upper-bounding an approximate of this functional yields

$$\|f(\mathbf{x}) - f(\mathbf{x}+\boldsymbol{\eta})\|_2 \approx \|\mathbf{J}_f(\mathbf{x})\boldsymbol{\eta}\|_2 \leq \varepsilon \|\mathbf{J}_f(\mathbf{x})\|_{\mathrm{F}} .$$

Motivated by this result, the authors propose using the Frobenius norm of the Jacobian $\|\mathbf{J}_f(\mathbf{x})\|_{\mathrm{F}}$ as regularization term to promote robustness. The Frobenius norm is an upper bound on the ℓ_2-norm of the output perturbation. If it is limited during training by proper regularization, it can restrict the ℓ_2-perturbations.

– **Curvature:** In Section 3.1 it is argued that low curvature in the decision boundaries, as well as in the loss function, are desired properties for robustness. Motivated by that discussion, the authors of [49] proposed penalizing solutions with high curvature of the loss function around the training data.

5 Future Directions

Adversarial examples appear as a potential obstacle for widespread employment of DNNs particularly in safety critical applications. An ultimate solution, yet, seems to be out of reach even for a simple task of MNIST image classification. Adversarial training is the best known defense in this situation that comes with two additional problems. It is computationally costly and degrades the generalization of DNNs. Future works can see whether the latter problem can be solved by different training techniques, or the generalization degradation cost is to be paid inevitably for more robustness. There is no consensus on the nature of adversarial examples and which features of classifiers play the central role in adversarial robustness. There are many indications that occasionally align with experimental results. Recent statistical learning theory approaches, however, provide a promising path to address generalization and robustness simultaneously. The ultimate goal of an adequate account of adversarial robustness, although not attained so far, constitutes an exciting field of research in coming years.

6 Acknowledgment

The authors would like to thank the reviewers for their fruitful feedbacks.

References

1. Akhtar, N., Mian, A.: Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey. IEEE Access 6, 14410–14430 (2018)
2. Anthony, M., Bartlett, P.L.: Neural network learning: theoretical foundations. Cambridge Univ. Press, Cambridge (2009)
3. Arora, S., Ge, R., Neyshabur, B., Zhang, Y.: Stronger generalization bounds for deep nets via a compression approach (Feb 2018), `http://arxiv.org/abs/1802.05296`
4. Athalye, A., Carlini, N.: On the Robustness of the CVPR 2018 White-Box Adversarial Example Defenses. arXiv:1804.03286 [cs, stat] (Apr 2018), `http://arxiv.org/abs/1804.03286`

5. Athalye, A., Carlini, N., Wagner, D.: Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. In: International Conference on Machine Learning (2018)
6. Attias, I., Kontorovich, A., Mansour, Y.: Improved generalization bounds for robust learning. arXiv:1810.02180 [cs, stat] (Oct 2018), `http://arxiv.org/abs/1810.02180`
7. Balda, E.R., Behboodi, A., Mathar, R.: On generation of adversarial examples using convex programming. In: 52-th Asilomar Conference on Signals, Systems, and Computers. pp. 1–6. Pacific Grove, California, USA (Oct 2018)
8. Balda, E.R., Behboodi, A., Mathar, R.: Perturbation analysis of learning algorithms: A unifying perspective on generation of adversarial examples (2018)
9. Barreno, M., Nelson, B., Joseph, A.D., Tygar, J.D.: The security of machine learning. Machine Learning 81(2), 121–148 (Nov 2010)
10. Bartlett, P.L., Foster, D.J., Telgarsky, M.J.: Spectrally-normalized margin bounds for neural networks. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30, pp. 6240–6249. Curran Associates, Inc. (2017), `http://papers.nips.cc/paper/7204-spectrally-normalized-margin-bounds-for-neural-networks.pdf`
11. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: Security and Privacy (SP), 2017 IEEE Symposium on. pp. 39–57. IEEE (2017)
12. Cisse, M., Adi, Y., Neverova, N., Keshet, J.: Houdini: Fooling deep structured prediction models. arXiv preprint arXiv:1707.05373 (2017)
13. Cullina, D., Bhagoji, A.N., Mittal, P.: PAC-learning in the presence of adversaries. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems 31, pp. 228–239. Curran Associates, Inc. (2018), `http://papers.nips.cc/paper/7307-pac-learning-in-the-presence-of-adversaries.pdf`
14. Diochnos, D.I., Mahloujifar, S., Mahmoody, M.: Adversarial Risk and Robustness: General Definitions and Implications for the Uniform Distribution. arXiv:1810.12272 (Oct 2018), `http://arxiv.org/abs/1810.12272`, arXiv: 1810.12272
15. Dohmatob, E.: Limitations of adversarial robustness: strong No Free Lunch Theorem. arXiv:1810.04065 [cs, stat] (Oct 2018), `http://arxiv.org/abs/1810.04065`
16. Fawzi, A., Moosavi-Dezfooli, S.M., Frossard, P.: The Robustness of Deep Networks: A Geometrical Perspective. IEEE Signal Processing Magazine 34(6), 50–62 (Nov 2017)
17. Fawzi, A., Fawzi, O., Frossard, P.: Fundamental limits on adversarial robustness. Proceedings of ICML, Workshop on Deep Learning (2015)
18. Fawzi, A., Fawzi, O., Frossard, P.: Analysis of classifiers' robustness to adversarial perturbations. Machine Learning 107(3), 481–508 (2018)
19. Fawzi, A., Moosavi-Dezfooli, S.M., Frossard, P.: Robustness of classifiers: from adversarial to random noise. In: Advances in Neural Information Processing Systems 29, pp. 1632–1640. Curran Associates, Inc. (2016)
20. Fawzi, A., Moosavi-Dezfooli, S.M., Frossard, P., Soatto, S.: Classification regions of deep neural networks. arXiv:1705.09552 [cs, stat] (May 2017), `http://arxiv.org/abs/1705.09552`
21. Franceschi, J.Y., Fawzi, A., Fawzi, O.: Robustness of classifiers to uniform ℓ_p and Gaussian noise. In: 21st International Conference on Artificial Intelligence and Statistics (AISTATS). vol. 84, p. 9. Lanzarote, Spain (2018)

22. Gilmer, J., Metz, L., Faghri, F., Schoenholz, S.S., Raghu, M., Wattenberg, M., Goodfellow, I.: Adversarial Spheres. arXiv:1801.02774 [cs] (Jan 2018), `http://arxiv.org/abs/1801.02774`
23. Golowich, N., Rakhlin, A., Shamir, O.: Size-Independent Sample Complexity of Neural Networks. In: Bubeck, S., Perchet, V., Rigollet, P. (eds.) Proceedings of the 31st Conference On Learning Theory. Proceedings of Machine Learning Research, vol. 75, pp. 297–299. PMLR (Jul 2018), `http://proceedings.mlr.press/v75/golowich18a.html`
24. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and Harnessing Adversarial Examples. In: International Conference on Learning Representations (Dec 2014)
25. Guo, Y., Zhang, C., Zhang, C., Chen, Y.: Sparse DNNs with Improved Adversarial Robustness. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems 31, pp. 240–249. Curran Associates, Inc. (2018), `http://papers.nips.cc/paper/7308-sparse-dnns-with-improved-adversarial-robustness.pdf`
26. Guo, Y., Zhang, C., Zhang, C., Chen, Y.: Sparse dnns with improved adversarial robustness. In: NeurIPS (2018)
27. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 770–778 (Jun 2016)
28. Hein, M., Andriushchenko, M.: Formal guarantees on the robustness of a classifier against adversarial manipulation. In: NIPS (2017)
29. Hinton, G., Deng, L., Yu, D., Dahl, G.E., Mohamed, A.r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T.N., Kingsbury, B.: Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. IEEE Signal Processing Magazine 29(6), 82–97 (Nov 2012)
30. Huang, G., Liu, Z., Weinberger, K.Q., van der Maaten, L.: Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition (2017)
31. Khim, J., Loh, P.L.: Adversarial Risk Bounds via Function Transformation. arXiv:1810.09519 (Oct 2018), `http://arxiv.org/abs/1810.09519`
32. Khrulkov, V., Oseledets, I.V.: Art of singular vectors and universal adversarial perturbations. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition pp. 8562–8570 (2018)
33. Kos, J., Fischer, I., Song, D.: Adversarial Examples for Generative Models. In: 2018 IEEE Security and Privacy Workshops (SPW). pp. 36–42 (May 2018)
34. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)
35. Kurakin, A., Goodfellow, I., Bengio, S.: Adversarial examples in the physical world. arXiv preprint arXiv:1607.02533 (2016)
36. Langenberg, P., Balda, E.R., Behboodi, A., Mathar, R.: On the effect of low-rank weights on adversarial robustness of neural networks (2019)
37. LeCun, Y., Haffner, P., Bottou, L., Bengio, Y.: Object recognition with gradient-based learning. In: Shape, contour and grouping in computer vision, pp. 319–345. Springer (1999)
38. Liao, F., Liang, M., Dong, Y., Pang, T., Hu, X., Zhu, J.: Defense against adversarial attacks using high-level representation guided denoiser. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2018)
39. Lin, M., Chen, Q., Yan, S.: Network in network. arXiv preprint arXiv:1312.4400 (2013)

40. Lin, Y.C., Hong, Z.W., Liao, Y.H., Shih, M.L., Liu, M.Y., Sun, M.: Tactics of Adversarial Attack on Deep Reinforcement Learning Agents. In: Proceedings of the 26th International Joint Conference on Artificial Intelligence. pp. 3756–3762. IJCAI'17, AAAI Press, Melbourne, Australia (2017)
41. Liu, Y., Chen, X., Liu, C., Song, D.X.: Delving into transferable adversarial examples and black-box attacks. CoRR abs/1611.02770 (2016)
42. Luo, Y., Boix, X., Roig, G., Poggio, T., Zhao, Q.: Foveation-based Mechanisms Alleviate Adversarial Examples. arXiv:1511.06292 (Nov 2015), `http://arxiv.org/abs/1511.06292`
43. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards Deep Learning Models Resistant to Adversarial Attacks. In: International Conference on Learning Representations (2018)
44. Mahloujifar, S., Diochnos, D.I., Mahmoody, M.: The Curse of Concentration in Robust Learning: Evasion and Poisoning Attacks from Concentration of Measure. In: Thirty-Third AAAI Conference on Artificial Intelligence (AAAI) (2019), `http://arxiv.org/abs/1809.03063`
45. Metzen, J.H., Kumar, M.C., Brox, T., Fischer, V.: Universal adversarial perturbations against semantic image segmentation. stat 1050, 19 (2017)
46. Moosavi-Dezfooli, S.M., Fawzi, A., Fawzi, O., Frossard, P.: Universal adversarial perturbations. arXiv preprint (2017)
47. Moosavi-Dezfooli, S.M., Fawzi, A., Fawzi, O., Frossard, P., Soatto, S.: Robustness of Classifiers to Universal Perturbations: A Geometric Perspective. In: International Conference on Learning Representations (2018), `https://openreview.net/forum?id=ByrZyglCb`
48. Moosavi Dezfooli, S.M., Fawzi, A., Frossard, P.: Deepfool: a simple and accurate method to fool deep neural networks. In: Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
49. Moosavi-Dezfooli, S.M., Fawzi, A., Uesato, J., Frossard, P.: Robustness via curvature regularization, and vice versa. arXiv:1811.09716 [cs, stat] (Nov 2018), `http://arxiv.org/abs/1811.09716`
50. Neyshabur, B., Bhojanapalli, S., Srebro, N.: A PAC-Bayesian Approach to Spectrally-Normalized Margin Bounds for Neural Networks. In: International Conference on Learning Representations (2018), `https://openreview.net/forum?id=Skz_WfbCZ`
51. Novak, R., Bahri, Y., Abolafia, D.A., Pennington, J., Sohl-Dickstein, J.: Sensitivity and generalization in neural networks: an empirical study. CoRR abs/1802.08760 (2018)
52. Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z.B., Swami, A.: Practical black-box attacks against machine learning. In: Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security. pp. 506–519. ACM (2017)
53. Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z.B., Swami, A.: The limitations of deep learning in adversarial settings. In: Security and Privacy (EuroS&P), 2016 IEEE European Symposium on. pp. 372–387. IEEE (2016)
54. Papernot, N., McDaniel, P., Swami, A., Harang, R.: Crafting adversarial input sequences for recurrent neural networks. In: Military Communications Conference, MILCOM 2016-2016 IEEE. pp. 49–54. IEEE (2016)
55. Papernot, N., McDaniel, P., Wu, X., Jha, S., Swami, A.: Distillation as a defense to adversarial perturbations against deep neural networks. In: Security and Privacy (SP), 2016 IEEE Symposium on. pp. 582–597. IEEE (2016)

56. Poole, B., Lahiri, S., Raghu, M., Sohl-Dickstein, J., Ganguli, S.: Exponential expressivity in deep neural networks through transient chaos. In: Lee, D.D., Sugiyama, M., Luxburg, U.V., Guyon, I., Garnett, R. (eds.) Advances in Neural Information Processing Systems 29, pp. 3360–3368. Curran Associates, Inc. (2016), http://papers.nips.cc/paper/6322-exponential-expressivity-in-deep-neural-networks-through-transient-chaos.pdf
57. Raghunathan, A., Steinhardt, J., Liang, P.: Certified Defenses against Adversarial Examples. In: International Conference on Learning Representations (2018)
58. Rao, N., Recht, B., Nowak, R.: Universal Measurement Bounds for Structured Sparse Signal Recovery. In: Artificial Intelligence and Statistics. pp. 942–950 (Mar 2012)
59. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence 39(6), 1137–1149 (Jun 2017)
60. Sabour, S., Cao, Y., Faghri, F., Fleet, D.J.: Adversarial Manipulation of Deep Representations. In: ICLR 2016 (2016), http://arxiv.org/abs/1511.05122
61. Samangouei, P., Kabkab, M., Chellappa, R.: Defense-GAN: Protecting classifiers against adversarial attacks using generative models. In: International Conference on Learning Representations (2018), https://openreview.net/forum?id=BkJ3ibb0-
62. Sanyal, A., Kanade, V., Torr, P.H.S.: Intriguing properties of learned representations (2018)
63. Sarkar, S., Bansal, A., Mahbub, U., Chellappa, R.: Upset and angri: Breaking high performance image classifiers. arXiv preprint arXiv:1707.01159 (2017)
64. Schmidt, L., Santurkar, S., Tsipras, D., Talwar, K., Mądry, A.: Adversarially Robust Generalization Requires More Data. arXiv:1804.11285 [cs, stat] (Apr 2018), http://arxiv.org/abs/1804.11285
65. Shalev-Shwartz, S., Ben-David, S.: Understanding machine learning: from theory to algorithms. Cambridge University Press, New York, NY, USA (2014)
66. Song, Y., Kim, T., Nowozin, S., Ermon, S., Kushman, N.: Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. In: International Conference on Learning Representations (2018), https://openreview.net/forum?id=rJUYGxbCW
67. Su, J., Vargas, D.V., Kouichi, S.: One pixel attack for fooling deep neural networks. arXiv preprint arXiv:1710.08864 (2017)
68. Suggala, A.S., Prasad, A., Nagarajan, V., Ravikumar, P.: Revisiting Adversarial Risk. arXiv:1806.02924 [cs, stat] (Jun 2018), http://arxiv.org/abs/1806.02924
69. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1–9 (2015)
70. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. International Conference on Learning Representations (2014)
71. Tabacof, P., Tavares, J., Valle, E.: Adversarial images for variational autoencoders. arXiv preprint arXiv:1612.00155 (2016)
72. Tanay, T., Griffin, L.: A Boundary Tilting Persepective on the Phenomenon of Adversarial Examples. arXiv:1608.07690 [cs, stat] (August 2016), http://arxiv.org/abs/1608.07690
73. Tramèr, F., Kurakin, A., Papernot, N., Goodfellow, I., Boneh, D., McDaniel, P.: Ensemble Adversarial Training: Attacks and Defenses. In: International Conference on Learning Representations (2018), https://openreview.net/forum?id=rkZvSe-RZ

74. Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., Madry, A.: Robustness May Be at Odds with Accuracy. In: International Conference on Learning Representations (2019), http://arxiv.org/abs/1805.12152
75. Wang, B., Gao, J., Qi, Y.: A Theoretical Framework for Robustness of (Deep) Classifiers against Adversarial Examples. In: International Conference on Learning Representations (2017)
76. Wang, L., Ding, G.W., Huang, R., Cao, Y., Lui, Y.C.: Adversarial robustness of pruned neural networks (2018), https://openreview.net/forum?id=SJGrAisIz
77. Xie, C., Wang, J., Zhang, Z., Zhou, Y., Xie, L., Yuille, A.: Adversarial examples for semantic segmentation and object detection. In: International Conference on Computer Vision. IEEE (2017)
78. Yin, D., Ramchandran, K., Bartlett, P.: Rademacher Complexity for Adversarially Robust Generalization. arXiv:1810.11914 [cs, stat] (Oct 2018), http://arxiv.org/abs/1810.11914
79. Yuan, X., He, P., Zhu, Q., Li, X.: Adversarial Examples: Attacks and Defenses for Deep Learning. IEEE Transactions on Neural Networks and Learning Systems pp. 1–20 (2019)
80. Zhang, C., Bengio, S., Hardt, M., Recht, B., Vinyals, O.: Understanding deep learning requires rethinking generalization. In: ICLR 2018 (2017), https://openreview.net/forum?id=Sy8gdB9xx