On Evaluating Adversarial Robustness

Nicholas Carlini[1], Anish Athalye[2], Nicolas Papernot[1], Wieland Brendel[3], Jonas Rauber[3], Dimitris Tsipras[2], Ian Goodfellow[1], Aleksander Mądry[2], Alexey Kurakin[1] [*]

[1] Google Brain [2] MIT [3] University of Tübingen

[*] List of authors is dynamic and subject to change. Authors are ordered according to the amount of their contribution to the text of the paper.

Please direct correspondence to the GitHub repository
`https://github.com/evaluating-adversarial-robustness/adv-eval-paper`

Last Update: May 14, 2019 (revision `0a2187a`).

Abstract

Correctly evaluating defenses against adversarial examples has proven to be extremely difficult. Despite the significant amount of recent work attempting to design defenses that withstand adaptive attacks, few have succeeded; most papers that propose defenses are quickly shown to be incorrect.

We believe a large contributing factor is the difficulty of performing security evaluations. In this paper, we discuss the methodological foundations, review commonly accepted best practices, and suggest new methods for evaluating defenses to adversarial examples. We hope that both researchers developing defenses as well as readers and reviewers who wish to understand the completeness of an evaluation consider our advice in order to avoid common pitfalls.

1 Introduction

Adversarial examples (Szegedy et al., 2013; Biggio et al., 2013), inputs that are specifically designed by an adversary to force a machine learning system to produce erroneous outputs, have seen significant study in recent years. This long line of research (Dalvi et al., 2004; Lowd & Meek, 2005; Barreno et al., 2006; 2010; Globerson & Roweis, 2006; Kołcz & Teo, 2009; Barreno et al., 2010; Biggio et al., 2010; Šrndic & Laskov, 2013) has recently begun seeing significant study as machine learning becomes more widely used. While attack research (the study of adversarial examples on new domains or under new threat models) has flourished, progress on defense[1] research (i.e., building systems that are robust to adversarial examples) has been comparatively slow.

More concerning than the fact that progress is slow is the fact that most proposed defenses are quickly shown to have performed incorrect or incomplete evaluations (Carlini & Wagner, 2016; 2017c; Brendel & Bethge, 2017; Carlini & Wagner, 2017a; He et al., 2017; Carlini & Wagner, 2017b; Athalye et al., 2018; Engstrom et al., 2018; Athalye & Carlini, 2018; Uesato et al., 2018; Mosbach et al., 2018; He et al., 2018; Sharma & Chen, 2018; Lu et al., 2018a;b; Cornelius, 2019; Carlini, 2019). As a result, navigating the field and identifying genuine progress becomes particularly hard.

Informed by these recent results, this paper provides practical advice for evaluating defenses that are intended to be robust to adversarial examples. This paper is split roughly in two:

- §2: *Principles for performing defense evaluations.* We begin with a discussion of the basic principles and methodologies that should guide defense evaluations.
- §3–§5: *A specific checklist for avoiding common evaluation pitfalls.* We have seen evaluations fail for many reasons; this checklist outlines the most common errors we have seen in defense evaluations so they can be avoided.

We hope this advice will be useful to both those building defenses (by proposing evaluation methodology and suggesting experiments that should be run) as well as readers or reviewers of defense papers (to identify potential oversights in a paper's evaluation).

We intend for this to be a living document. The LaTeX source for the paper is available at `https://github.com/evaluating-adversarial-robustness/adv-eval-paper` and we encourage researchers to participate and further improve this paper.

[1] This paper uses the word "defense" with the understanding that there are non-security motivations for constructing machine learning algorithms that are robust to attacks (see Section 2.1); we use this consistent terminology for simplicity.

2 Principles of Rigorous Evaluations

2.1 Defense Research Motivation

Before we begin discussing our recommendations for performing defense evaluations, it is useful to briefly consider *why* we are performing the evaluation in the first place. While there are many valid reasons to study defenses to adversarial examples, below are the three common reasons why one might be interested in evaluating the robustness of a machine learning model.

- **To defend against an adversary who will attack the system.** Adversarial examples are a security concern. Just like any new technology not designed with security in mind, when deploying a machine learning system in the real-world, there will be adversaries who wish to cause harm as long as there exist incentives (i.e., they benefit from the system misbehaving). Exactly what this harm is and how the adversary will go about causing it depends on the details of the domain and the adversary considered. For example, an attacker may wish to cause a self-driving car to incorrectly recognize road signs[2] (Papernot et al., 2016b), cause an NSFW detector to incorrectly recognize an image as safe-for-work (Bhagoji et al., 2018), cause a malware (or spam) classifier to identify a malicious file (or spam email) as benign (Dahl et al., 2013), cause an ad-blocker to incorrectly identify an advertisement as natural content (Tramèr et al., 2018), or cause a digital assistant to incorrectly recognize commands it is given (Carlini et al., 2016).
- **To test the worst-case robustness of machine learning algorithms.** Many real-world environments have inherent randomness that is difficult to predict. By analyzing the robustness of a model from the perspective of an adversary, we can estimate the *worst-case* robustness in a real-world setting. Through random testing, it can be difficult to distinguish a system that fails one time in a billion from a system that never fails: even when evaluating such a system on a million choices of randomness, there is just under 0.1% chance to detect a failure case.

  However, analyzing the worst-case robustness can discover a difference. If a powerful adversary who is intentionally trying to cause a system to misbehave (according to some definition) cannot succeed, then we have strong evidence that the system will not misbehave due to *any* unforeseen randomness.
- **To measure progress of machine learning algorithms towards human-level abilities.** To advance machine learning algorithms it is important to understand where they fail. In terms of performance, the gap between humans and machines is quite small on many widely studied problem domains, including reinforcement learning (e.g., Go and Chess (Silver et al., 2016)) or natural image classification (Krizhevsky et al., 2012). In terms of adversarial robustness, however, the gap between humans and machines is astonishingly large: even in settings where machine learning achieves super-human accuracy, an adversary can often introduce perturbations that reduce their accuracy to levels of random guessing and far below the accuracy of even the most uninformed human.[3] This suggests a fundamental difference of the decision-making process of humans and machines. From this point of view, adversarial robustness is a measure of progress in machine learning that is orthogonal to performance.

The motivation for why the research was conducted informs the methodology through which it should be evaluated: a paper that sets out to prevent a real-world adversary from fooling a specific spam detector assuming the adversary can not directly access the underlying model will have a very different evaluation than one that sets out to measure the worst-case robustness of a self-driving car's vision system.

[2] While this threat model is often repeated in the literature, it may have limited impact for real-world adversaries, who in practice may have have little financial motivation to cause harm to self-driving cars.

[3] Note that time-limited humans appear vulnerable to some forms of adversarial examples (Elsayed et al., 2018).

This paper therefore does not (and could not) set out to provide a definitive answer for how all evaluations should be performed. Rather, we discuss methodology that we believe is common to most evaluations. Whenever we provide recommendations that may not apply to some class of evaluations, we state this fact explicitly. Similarly, for advice we believe holds true universally, we discuss why this is the case, especially when it may not be obvious at first.

The remainder of this section provides an overview of the basic methodology for a defense evaluation.

2.2 Threat Models

A threat model specifies the conditions under which a defense is designed to be secure and the precise security guarantees provided; it is an integral component of the defense itself.

Why is it important to have a threat model? In the context of a defense where the purpose is motivated by security, the threat model outlines what type of actual attacker the defense intends to defend against, guiding the evaluation of the defense.

However, even in the context of a defense motivated by reasons beyond security, a threat model is necessary for evaluating the performance of the defense. One of the defining properties of scientific research is that it is *falsifiable*: there must exist an experiment that can contradict its claims. Without a threat model, defense proposals are often either not falsifiable or trivially falsifiable.

Typically, a threat model includes a set of assumptions about the adversary's *goals*, *knowledge*, and *capabilities*. Next, we briefly describe each.

2.2.1 Adversary goals

How should we define an **adversarial example**? At a high level, adversarial examples can be defined as inputs specifically designed to force a machine learning system to produce erroneous outputs. However, the precise goal of an adversary can vary significantly across different settings.

For example, in some cases the adversary's goal may be to simply cause misclassification—any input being misclassified represents a successful attack. Alternatively, the adversary may be interested in having the model misclassify certain examples from a *source* class into a *target* class of their choice. This has been referred to a *source-target* misclassification attack (Papernot et al., 2016b) or *targeted* attack (Carlini & Wagner, 2017c).

In other settings, only specific types of misclassification may be interesting. In the space of malware detection, defenders may only care about the specific source-target class pair where an adversary causes a malicious program to be misclassified as benign; causing a benign program to be misclassified as malware may be uninteresting.

2.2.2 Adversarial capabilities

In order to build meaningful defenses, we need to impose reasonable constraints to the attacker. An unconstrained attacker who wished to cause harm may, for example, cause bit-flips on the weights of the neural network, cause errors in the data processing pipeline, backdoor the machine learning model, or (perhaps more relevant) introduce large perturbations to an image that would alter its semantics. Since such attacks are outside the scope of defenses adversarial examples, restricting the adversary is necessary for designing defenses that are not trivially bypassed by unconstrained adversaries.

To date, most defenses to adversarial examples typically restrict the adversary to making "small" changes to inputs from the data-generating distribution (e.g. inputs from the test set). Formally, for some natural input x and similarity metric $\mathcal{D}$, x' is considered a valid adversarial example if $\mathcal{D}(x, x') \leq \epsilon$ for some small ϵ and x' is misclassified[4]. This definition is motivated by the assumption that small changes under the metric $\mathcal{D}$ do not change the true class of the input and thus should not cause the classifier to predict an erroneous class.

A common choice for $\mathcal{D}$, especially for the case of image classification, is defining it as the ℓ_p-norm between two inputs for some p. (For instance, an ℓ_∞-norm constraint of ϵ for image classification implies that the adversary cannot modify any individual pixel by more than ϵ.) However, a suitable choice of $\mathcal{D}$ and ϵ may vary significantly based on the particular task. For example, for a task with binary features one may wish to study ℓ_0-bounded adversarial examples more closely than ℓ_∞-bounded ones. Moreover, restricting adversarial perturbations to be small may not always be important: in the case of malware detection, what is required is that the adversarial program preserves the malware behavior while evading ML detection.

Nevertheless, such a rigorous and precise definition of the adversary's capability, leads to well-defined measures of adversarial robustness that are, in principle, computable. For example, given a model $f(\cdot)$, one common way to define robustness is the worst-case loss L for a given perturbation budget,

$$\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{X}}\left[\max_{x':\mathcal{D}(x,x')<\epsilon} L\big(f(x'), y\big)\right].$$

Another commonly adopted definition is the average (or median) minimum-distance of the adversarial perturbation,

$$\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{X}}\left[\min_{x'\in A_{x,y}} \mathcal{D}(x, x')\right],$$

where $A_{x,y}$ depends on the definition of *adversarial example*, e.g. $A_{x,y} = \{x' \mid f(x') \neq y\}$ for misclassification or $A_{x,y} = \{x \mid f(x') = t\}$ for some target class t.

A key challenge of security evaluations is that while this *adversarial risk* (Madry et al., 2017; Uesato et al., 2018) is often computable in theory (e.g. with optimal attacks or brute force enumeration of the considered perturbations), it is usually intractable to compute exactly, and therefore in practice we must approximate this quantity. This difficulty is at the heart of why evaluating worst-case robustness is difficult: while evaluating average-case robustness is often as simple as sampling a few hundred (or thousand) times from the distribution and computing the mean, such an approach is not possible for worst-case robustness.

It is customary for defenses not to impose any computational bounds on an attacker (i.e., the above definitions of adversarial risk consider only the *existence* of adversarial examples, and not the difficulty of finding them). We believe that restricting an adversary's computational power could be interesting if we could make formal statements about the computational cost of finding adversarial examples (e.g., via a reduction to a concrete hardness assumption as is done in cryptography). We are not aware of any defense that currently achieves this. Stronger restrictions (e.g., the adversary is limited to attacks with 100 iterations) are usually uninteresting in that they cannot be meaningfully enforced and there is often no economic reason for an adversary not to spend a little more time to succeed.

Finally, a common, often implicit, assumption in adversarial example research is that the adversary has direct access to the model's input features: e.g., in the image domain, the adversary directly manipulates the image pixels. However, in certain domains, such as malware detection or language modeling, these features can be difficult to reverse-engineer. As a result, different assumptions on the capabilities of the adversary can significantly impact the evaluation of a defense's effectiveness.

Comment on ℓ_p-norm-constrained threat models. A large body of work studies a threat model where the adversary is constrained to ℓ_p-bounded perturbations. This threat model is highly limited and does not perfectly match real-world threats (Engstrom et al.,

[4] It is often required that the original input x is classified correctly, but this requirement can vary across papers. Some papers consider x' an adversarial example as long as it is classified *differently* from x.

2017; Gilmer et al., 2018). However, the well-defined nature of this threat model is helpful for performing principled work towards building strong defenses. While ℓ_p-robustness does not imply robustness in more realistic threat models, it is almost certainly the case that lack of robustness against ℓ_p-bounded perturbation will imply lack of robustness in more realistic threat models. Thus, working towards solving robustness for these well-defined ℓ_p-bounded threat models is a useful exercise.

2.2.3 Adversary knowledge.

A threat model clearly describes what knowledge the adversary is assumed to have. Typically, works assume either white-box (complete knowledge of the model and its parameters) or black-box access (no knowledge of the model) with varying degrees of black-box access (e.g., a limited number of queries to the model, access to the predicted probabilities or just the predicted class, or access to the training data).

In general, the guiding principle of a defense's threat model is to assume that the adversary has complete knowledge of the inner workings of the defense. It is not reasonable to assume the defense algorithm can be held secret, even in black-box threat models. This widely-held principle is known in the field of security as Kerckhoffs' principle (Kerckhoffs, 1883), and the opposite is known as "security through obscurity". The open design of security mechanisms is a cornerstone of the field of cryptography (Saltzer & Schroeder, 1975). This paper discusses only how to perform white-box evaluations, which implies robustness to black-box adversaries, but not the other way around.

Holding Data Secret. While it can be acceptable to hold some limited amount of information secret, the defining characteristic of a white-box evaluation (as we discuss in this paper) is that the threat model assumes the attacker has **full knowledge** of the underlying system.

That does not mean that all information has to be available to the adversary—it can be acceptable for the defender to hold a small amount of information secret. The field of cryptography, for example, is built around the idea that one *can* keep secret the encryption keys, but the underlying algorithm is be assumed to be public.

A defense that holds values secret should justify that it is reasonable to do so. In particular, secret information generally satisfies at least the following two properties:

1. *The secret must be easily replaceable.* That is, there should be an efficient algorithm to generate a new secret if the prior one happened to be leaked.
2. *The secret must be nonextractable.* An adversary who is allowed to query the system should not be able to extract any information about the secret.

For example, a defense that includes randomness (chosen fresh) at inference time is using secret information not available to the adversary. As long as the distribution is known, this follows Kerckhoffs' principle. On the other hand, if a single fixed random vector was added to the output of the neural network after classifying an input, this would not be a good candidate for a secret. By subtracting the observed output of the model with the expected output, the secret can be easily determined.

2.3 Restrict Attacks to the Defense's Threat Model

Attack work should always evaluate defenses under the threat model the defense states. For example, if a defense paper explicitly states "we intend to be robust to L_2 attacks of norm no greater than 1.5", an attack paper must restrict its demonstration of vulnerabilities in the defense to the generation of adversarial examples with L_2 norm less than 1.5. Showing something different, e.g., adversarial examples with L_∞ norm less than 0.1, is important and useful research[5] (because it teaches the research community something that was not

[5] See for example the work of Sharma & Chen (2017); Song et al. (2018) who explicitly step outside of the threat model of the original defenses to evaluate their robustness.

previously known, namely, that this system may have limited utility in practice), but is not a *break* of the defense: the defense never claimed to be robust to this type of attack.

2.4 Skepticism of Results

When performing scientific research one must be skeptical of all results. As Feynman concisely put it, "the first principle [of research] is that you must not fool yourself—and you are the easiest person to fool." This is never more true than when considering security evaluations. After spending significant effort to try and develop a defense that is robust against attacks, it is easy to assume that the defense is indeed robust, especially when baseline attacks fail to break the defense. However, at this time the authors need to completely switch their frame of mind and try as hard as possible to show their proposed defense is ineffective.[6]

Adversarial robustness is a negative goal – for a defense to be truly effective, one needs to show that *no attack* can bypass it. It is only that by failing to show the defense is ineffective to adaptive attacks (see below) that we can believe it will withstand future attack by a motivated adversary (or, depending on the motivation of the research, that the claimed lower bound is in fact an actual lower bound).

2.5 Adaptive Adversaries

After a specific threat model has been defined, the remainder of the evaluation focuses on *adaptive adversaries*[7] which are adapted to the specific details of the defense and attempt to invalidate the robustness claims that are made.

This evaluation is the most important section of any paper that develops a defense. After the defense has been defined, ask: *what attack could possibly defeat this defense?* All attacks that might work must be shown to be ineffective. An evaluation that does not attempt to do this is fundamentally flawed.

Just applying existing adversarial attacks with default hyperparameters is not sufficient, even if these attacks are state-of-the-art: all existing attacks and hyperparameters have been adapted to and tested only against *existing* defenses, and there is a good chance these attacks will work sub-optimally or even fail against a new defense. A typical example is gradient masking (Tramèr et al., 2017), in which defenses manipulate the model's gradients and thus prevent gradient-based attacks from succeeding. However, an adversary aware of the defense may recover these gradients through a black-box input-label queries, as shown by Papernot et al. (2017), or through a different loss function, as demonstrated by Athalye et al. (2018). In other words, gradient masking may make optimization-based attacks fail but that does not mean that the space of adversarial perturbations decreased.

Defending against non-adaptive attacks is necessary but not sufficient. It is our firm belief that **an evaluation against non-adaptive attacks is of very limited utility**.

Along the same lines, there is no justification to study a "zero-knowledge" (Biggio et al., 2013) threat model where the attacker is not aware of the defense. "Defending" against such an adversary is an absolute bare-minimum that in no way suggests a defense will be effective to further attacks. Carlini & Wagner (2017a) considered this scenario only to demonstrate that some defenses were completely ineffective even against this very weak threat model. The authors of that work now regret not making this explicit and discourage future work from citing this paper in support of the zero-knowledge threat model.

[6] One of the reasons it is so easy to accidentally fool oneself in security is that mistakes are very difficult to catch. Very often attacks only fail because of a (correctable) error in how they are being applied. It has to be the objective of the defense researcher to ensure that, when attacks fail, it is because the defense is correct, and not because of an error in applying the attacks.

[7] We use the word "adaptive adversary" (and "adaptive attack") to refer to the general notion in security of an adversary (or attack, respectively) that *adapts* to what the defender has done (Herley & van Oorschot, 2017; Carlini & Wagner, 2017a).

It is crucial to actively attempt to defeat the specific defense being proposed. On the most fundamental level this should include a range of sufficiently different attacks with carefully tuned hyperparameters. But the analysis should go deeper than that: ask why the defense might prevent existing attacks from working optimally and how to customize existing attacks or how to design completely new adversarial attacks to perform as well as possible. That is, applying the same mindset that a future adversary would apply is the only way to show that a defense might be able to withstand the test of time.

These arguments apply independent of the specific motivation of the robustness evaluation: security, worst-case bounds or human-machine gap all need a sense of the maximum vulnerability of a given defense. In all scenarios we should assume the existence of an "infinitely thorough" adversary who will spend whatever time is necessary to develop the optimal attack.

2.6 Reproducible Research: Code & Pre-trained Models

Even the most carefully-performed robustness evaluations can have subtle but fundamental flaws. We strongly believe that releasing full source code and pre-trained models is one of the most useful methods for ensuring the eventual correctness of an evaluation. Releasing source code makes it much more likely that others will be able to perform their own analysis of the defense.[8] Furthermore, completely specifying all defense details in a paper can be difficult, especially in the typical 8-page limit of many conference papers. The source code for a defense can be seen as the definitive reference for the algorithm.

It is equally important to release pre-trained models, especially when the resources that would be required to train a model would be prohibitive to some researchers with limited compute resources. The code and model that is released should be the model that was used to perform the evaluation in the paper to the extent permitted by underlying frameworks for accelerating numerical computations performed in machine learning. Releasing a *different* model than was used in the paper makes it significantly less useful, as any comparisons against the paper may not be identical.

Finally, it is helpful if the released code contains a simple one-line script which will run the full defense end-to-end on the given input. Note that this is often different than what the defense developers want, who often care most about performing the evaluation as efficiently as possible. In contrast, when getting started with evaluating a defense (or to confirm any results), it is often most useful to have a simple and correct method for running the full defense over an input.

There are several frameworks such as CleverHans (Papernot et al., 2018) or Foolbox (Rauber et al., 2017) as well as websites[9,10,11] which have been developed to assist in this process.

3 Specific Recommendations: Evaluation Checklist

While the above overview is general-purpose advice we believe will stand the test of time, it can be difficult to extract specific, actionable items from it. To help researchers *today* perform more thorough evaluations, we now develop a checklist that lists common evaluation pitfalls when evaluating adversarial robustness. Items in this list are sorted (roughly) into three categories.

The items contained below are **neither necessary nor sufficient** for performing a complete adversarial example evaluation, and are intended to list common evaluation flaws. There likely exist completely ineffective defenses which satisfy all of the below recommendations;

[8] In their analysis of the ICLR 2018 defenses (Athalye et al., 2018), the authors spent five times longer re-implementing the defenses than performing the security evaluation of the re-implementations.

[9] `https://robust-ml.org`

[10] `https://robust.vision/benchmark/leaderboard/`

[11] `https://foolbox.readthedocs.io/en/latest/modules/zoo.html`

conversely, some of the strongest defenses known today do *not* check off all the boxes below (e.g. Madry et al. (2017)).

We encourage readers to be extremely careful and **not directly follow this list** to perform an evaluation or decide if an evaluation that has been performed is sufficient. Rather, this list contains common flaws that are worth checking for to identify potential evaluation flaws. Blindly following the checklist without careful thought will likely be counterproductive: each item in the list must be taken into consideration within the context of the specific defense being evaluated. Each item on the list below is present because we are aware of several defense evaluations which were broken and following that specific recommendation would have revealed the flaw. We hope this list will be taken as a collection of recommendations that may or may not apply to a particular defense, but have been useful in the past.

This checklist is a living document that lists the most common evaluation flaws as of May 14, 2019. We expect the evaluation flaws that are common today will *not* be the most common flaws in the future. We intend to keep this checklist up-to-date with the latest recommendations for evaluating defenses by periodically updating its contents. Readers should check the following URL for the most recent revision of the checklist: `https://github.com/evaluating-adversarial-robustness/adv-eval-paper`.

3.1 Common Severe Flaws

There are several common severe evaluation flaws which have the potential to completely invalidate any robustness claims. Any evaluation which contains errors on any of the following items is likely to have fundamental and irredeemable flaws. Evaluations which intentionally deviate from the advice here may wish to justify the decision to do so.

- §3 **Do not mindlessly follow this list**; make sure to still think about the evaluation.
- §2.2 **State a precise threat model** that the defense is supposed to be effective under.
  - The threat model assumes the attacker knows how the defense works.
  - The threat model states attacker's goals, knowledge and capabilities.
  - For security-justified defenses, the threat model realistically models some adversary.
  - For worst-case randomized defenses, the threat model captures the perturbation space.
  - Think carefully and justify any ℓ_p bounds placed on the adversary.
- §2.5 Perform **adaptive attacks** to give an upper bound of robustness.
  - The attacks are given access to the full defense, end-to-end.
  - The loss function is changed as appropriate to cause misclassification.
  - §4.3 **Focus on the strongest attacks** for the threat model and defense considered.
- §2.6 Release **pre-trained models and source code**.
  - Include a clear installation guide, including all dependencies.
  - There is a one-line script which will classify an input example with the defense.
- §4.2 Report **clean model accuracy** when not under attack.
  - For defenses that abstain or reject inputs, generate a ROC curve.
- §5.2 Perform **basic sanity tests** on attack success rates.
  - Verify iterative attacks perform better than single-step attacks.
  - Verify that iterative-attacks use sufficient iterations to converge.
  - Verify that attacks use sufficient random restarts to avoid sub-optimal local minima.
  - Verify increasing the perturbation budget strictly increases attack success rate.
  - With "high" distortion, model accuracy should reach levels of random guessing.
- §5.3 Generate an **attack success rate vs. perturbation budget** curve.
  - Verify the x-axis extends so that attacks eventually reach 100% success.
  - For unbounded attacks, report distortion and not success rate.
- §5.4 Verify **adaptive attacks** perform better than any other.

  - Compare success rate on a per-example basis, rather than averaged across the dataset.
  - Evaluate against some combination of black-box, transfer, and random-noise attacks.
- §5.7 Describe the **attacks applied**, including all hyperparameters.

3.2 Common Pitfalls

There are other common pitfalls that may prevent the detection of ineffective defenses. This list contains some potential pitfalls which do not apply to large categories of defenses. However, if applicable, the items below are still important to carefully check they have been applied correctly.

- §4.3 Apply a **diverse set of attacks** (especially when training on one attack approach).
  - Do not blindly apply multiple (nearly-identical) attack approaches.
- §4.4 Try at least one **gradient-free attack** and one **hard-label attack**.
  - Try Chen et al. (2017b); Uesato et al. (2018); Ilyas et al. (2018a); Brendel et al. (2017).
  - Check that the gradient-free attacks succeed less often than gradient-based attacks.
  - Carefully investigate attack hyperparameters that affect success rate.
- §4.5 Perform a **transferability attack** using a similar substitute model.
  - Select a substitute model as similar to the defended model as possible.
  - Generate adversarial examples that are initially assigned high confidence.
  - Check that the transfer attack succeeds less often than white-box attacks.
- §4.6 For randomized defenses, properly **ensemble over randomness**.
  - Verify that attacks succeed if randomness is assigned to one fixed value.
  - State any assumptions about adversary knowledge of randomness in the threat model.
- §4.7 For non-differentiable components, **apply differentiable techniques**.
  - Discuss why non-differentiable components were necessary.
  - Verify attacks succeed on undefended model with those non-differentiable components.
  - Consider applying BPDA (Athalye et al., 2018) if applicable.
- §4.8 Verify that the **attacks have converged** under the selected hyperparameters.
  - Verify that doubling the number of iterations does not increase attack success rate nor significantly change the adversarial loss.
  - Plot attack effectiveness versus the number of iterations.
  - Run attacks with multiple random starting points and retain the best one.
  - Explore different choices of the step size or other attack hyperparameters.
- §4.9 Carefully **investigate attack hyperparameters** and report those selected.
  - Start search for adversarial examples at a random offset. Try multiple random starting points for each input.
  - As for the number of attack iterations, verify that increasing the number of random restarts does not affect the attack's success rate or the adversarial loss.
  - Investigate if attack results are sensitive to any other hyperparameters.
- §5.1 **Compare against prior work** and explain important differences.
  - When contradicting prior work, clearly explain why differences occur.
  - Attempt attacks that are similar to those that defeated previous similar defenses.
  - When comparing against prior work, ensure it has not been broken.
- §4.10 Test **broader threat models** when proposing general defenses. For images:
  - Apply rotations and translations (Engstrom et al., 2017).
  - Apply common corruptions and perturbations (Hendrycks & Dietterich, 2019).
  - Add Gaussian noise of increasingly large standard deviation (Ford et al., 2019).

3.3 Special-Case Pitfalls

The following items apply to a smaller fraction of evaluations. Items presented here are included because while they may diagnose flaws in some defense evaluations, they are not necessary for many others. In other cases, the tests presented here help provide additional evidence that the evaluation was performed correctly.

- §4.1 Investigate if it is possible to use **provable approaches**.
  - Examine if the model is amenable to provable robustness lower-bounds.
- §4.11 **Attack with random noise** of the correct norm.
  - For each example, try 10,000+ different choices of random noise.
  - Check that the random attacks succeed less-often than white-box attacks.
- §4.12 Use both **targeted and untargeted attacks** during evaluation.
  - State explicitly which attack type is being used.
- §4.13 **Perform ablation studies** with combinations of defense components removed.
  - Attack a similar-but-undefended model and verify attacks succeed.
  - If combining multiple defense techniques, argue why they combine usefully.
- §4.14 **Validate any new attacks** by attacking other defenses.
  - Attack other defenses known to be broken and verify the attack succeeds.
  - Construct synthetic intentionally-broken models and verify the attack succeeds.
  - Release source code for any new attacks implemented.
- §5.5 Investigate applying the defense to **domains other than images**.
  - State explicitly if the defense applies only to images (or another domain).
- §5.6 Report **per-example attack success rate**: $\underset{x\in\mathcal{X}}{\text{mean}}\,\min_{a\in\mathcal{A}} f(a(x))$, not $\min_{a\in\mathcal{A}}\,\underset{x\in\mathcal{X}}{\text{mean}}\, f(a(x))$.

4 Evaluation Recommendations

We now expand on the above checklist and provide the rationale for each item.

4.1 Investigate Provable Approaches

With the exception of this subsection, all other advice in this paper focuses on performing heuristic robustness evaluations. Provable robustness approaches are preferable to only heuristic ones. Current provable approaches often can only be applied when the neural network is explicitly designed with the objective of making these specific provable techniques applicable (Kolter & Wong, 2017; Raghunathan et al., 2018; Weng et al., 2018). While this approach of designing-for-provability has seen excellent progress—the best approaches today can certify some (small) robustness even on ImageNet classifiers (Lecuyer et al., 2018)—often the best heuristic defenses offer orders of magnitude better (estimated) robustness.

Proving a lower bound of defense robustness guarantees that the robustness will never fall below that level (if the proof is correct). We believe an important direction of future research is developing approaches that can generally prove arbitrary neural networks correct. While work in this space does exist (Katz et al., 2017; Tjeng et al., 2019; Xiao et al., 2019; Gowal et al., 2018), it is often computationally intractable to verify even modestly sized neural networks.

One key limitation of provable techniques is that the proofs they offer are generally only of the form "for some *specific* set of examples $\mathcal{X}$, no adversarial example with distortion less than ε exists". While this is definitely a useful statement, it gives no proof about any *other* example $x' \notin \mathcal{X}$; and because this is the property that we actually care about, provable techniques are still not provably correct in the same way that provably correct cryptographic algorithms are provably correct.

4.2 Report Clean Model Accuracy

A defense that significantly degrades the model's accuracy on the original task (the *clean* or *natural* data) may not be useful in many situations. If the probability of an actual attack is very low and the cost of an error on adversarial inputs is not high, then it may be unacceptable to incur *any* decrease in clean accuracy. Often there can be a difference in the impact of an error on a random input and an error on an adversarially chosen input. To what extent this is the case depends on the domain the system is being used in.

For the class of defenses that refuse to classify inputs by abstaining when detecting that inputs are adversarial, or otherwise refuse to classify some inputs, it is important to evaluate how this impacts accuracy on the clean data. Further, in some settings it may be acceptable to refuse to classify inputs that have significant amount of noise. In others, while it may be acceptable to refuse to classify adversarial examples, simple noisy inputs must still be classified correctly. It can be helpful to generate a Receiver Operating Characteristic (ROC) curve to show how the choice of threshold for rejecting inputs causes the clean accuracy to decrease.

4.3 Focus on the Strongest Attacks Possible

Use optimization-based attacks. Of the many different attack algorithms, optimization-based attacks are by far the most powerful. After all, they extract a significant amount of information from the model by utilizing the gradients of some loss function and not just the predicted output. In a white-box setting, there are many different attacks that have been created, and picking almost any of them will be useful. However, it is important to *not* just choose an attack and apply it out-of-the-box without modification. Rather, these attacks should serve as a starting point to which defense-specific knowledge can be applied.

We have found the following three attacks useful starting points for constructing adversarial examples under different distance metrics:

- For ℓ_1 distortion, start with Chen et al. (2017a).
- For ℓ_2 distortion, start with Carlini & Wagner (2017c).
- For ℓ_∞ distortion, start with Madry et al. (2017).

Note, however, that these attacks were designed to be effective on standard neural networks; any defense which modifies the architecture, training process, or any other aspect of the machine learning algorithm is likely to affect their performance. In order to ensure that they perform reliably on a particular defense, a certain amount of critical thinking and hyper-parameter optimization is necessary. Also, remember that there is no universally *strongest* attack: each attack makes specific assumptions about the model and so an attack that is best on one model might perform much worse on another (Schott et al., 2019). It is worth considering many different attacks.

Do not use Fast Gradient Sign (exclusively). The Fast Gradient Sign (FGS) attack is a simple approach to generating adversarial examples that was proposed to demonstrate the linearity of neural networks (Goodfellow et al., 2014). It was never intended to be a strong attack for evaluating the robustness of a neural network, and should not be used as one.

Even if it were intended to be a strong attack worth defending against, there are many defenses which achieve near-perfect success at defending against this attack. For example, a weak version of adversarial training can defend against this attack by causing gradient masking (Tramèr et al., 2017), where locally the gradient around a given image may point in a direction that is not useful for generating an adversarial example.

While evaluating against FGS can be a component of an attack evaluation, it should never be used as the only attack in an evaluation.

At the same time, even relatively simple and efficient attacks (DeepFool and JSMA) (Moosavi-Dezfooli et al., 2016; Papernot et al., 2016b) can still be useful for evaluating adversarial robustness if applied correctly (Jetley et al., 2018; Carlini & Wagner, 2016). However doing so often requires more care and is more error-prone; applying gradient-based attacks for many iterations is often simpler despite the attacks themselves being slower.

Do not *only* use attacks during testing that were used during training. One pitfall that can arise with adversarial training is that the defense can overfit against one particular attack used during training (e.g. by masking the gradients) (Tramèr et al., 2017). Using the same attack during both training and testing is dangerous and is likely to overestimate the robustness of the defense. This is still true even if the attack is perceived as the strongest attack for the given threat model: this is probably not true any more given that the defense has been specifically tuned against it.

Applying many nearly-identical attacks is not useful. When applying a diverse set of attacks, it is critical that the attacks are actually diverse. For example, the Basic Iterative Method (BIM) (also called i-FGSM, iterated fast gradient sign) (Kurakin et al., 2016) is nearly identical to Projected Gradient Descent (Madry et al., 2017) modulo the initial random step. Applying both of these attacks is less useful than applying one of these attacks and another (different) attack approach.

4.4 Apply Gradient-Free Attacks

To ensure that the model is not causing various forms of gradient masking, it is worth attempting gradient-free attacks. There are several such proposals. The following three require access to model confidence values, and therefore some forms of gradient masking (e.g., by thresholding the model outputs) will prevent these attacks from working effectively. Nevertheless, these attacks are effective under many situations where standard gradient-based attacks fail:

- ZOO (Chen et al., 2017b) (and also (Liu et al., 2017; Bhagoji et al., 2018; Tu et al., 2018)) numerically estimates gradients and then performs gradient descent, making it powerful but potentially ineffective when the loss surface is difficult to optimize over.
- SPSA (Uesato et al., 2018) was proposed specifically to evaluate adversarial example defenses, and has broken many. It can operate even when the loss surface is difficult to optimize over, but has many hyperparameters that can be difficult to tune.
- NES (Ilyas et al., 2018a) is effective at generating adversarial examples with a limited number of queries, and so generates adversarial examples of higher distortion. One variant of NES can handle scenarios in which only the confidence values of the top-k classes or only the label corresponding to the most confident output (see below) are observed. This family of approaches can be further strengthened by including and transferring over the existing priors (Ilyas et al., 2018b).

Hard label (or "decision-based") attacks differ from gradient-free confidence-based attacks in that they only require access to the `arg max` output of the model (i.e., the label corresponding to the most confident output). This makes these attacks much slower, as they make many more queries, but defenses can do less to accidentally prevent these attacks.

- The Boundary Attack (Brendel et al., 2017) is general-purpose hard-label attack and performs a descent along the decision boundary using a rejection sampling approach. In terms of the minimum adversarial distance the attack often rivals the best white-box attacks but at the cost of many more queries.

The most important reason to apply gradient-free attacks is as a test for gradient masking (Tramèr et al., 2017; Athalye et al., 2018; Uesato et al., 2018). Gradient-free attacks should almost always do worse than gradient-based attacks (see §5.2). Not only should this be true when averaged across the entire dataset, but also on a per-example basis (see §5.6) there should be very few instances where gradient-based attacks fail but gradient-free attacks succeed. Thus, white-box attacks performing worse than gradient-free attacks is strong evidence for gradient masking.

Just as gradient masking can fool gradient-based attack algorithms, gradient-free attack algorithms can be similarly fooled by different styles of defenses. Today, the most common class of models that causes existing gradient-free attacks to fail is *randomized models*. This may change in the future as gradient-free attacks become stronger, but the state-of-the-art gradient-free attacks currently do not do well in this setting. (Future work on this challenging problem would be worthwhile.)

4.5 Perform a Transferability Analysis

Adversarial examples often transfer between models (Papernot et al., 2016a). That is, an adversarial example generated against one model with a specific architecture will often also be adversarial on another model, even if the latter is trained on a different training set with a different architecture.

By utilizing this phenomenon, *transfer attacks* can be performed by considering an alternative, substitute model and generating adversarial examples for that are strongly classified as a wrong class by that model. These examples can then be used to attack the target model. Such transfer attacks are particularly successful at circumventing defenses that rely on gradient masking. Since the adversarial example is generated on an independent model, there is no need to directly optimize over the (often poorly behaved) landscape of the defended model. An adversarial example defense evaluation should thus attempt a transferability analysis to validate that the proposed defense breaks transferability. If it does not, then it is likely to not actually be a strong defense, but just appears like one.

What should be the source model for a transferability analysis? A good starting point is a model as similar to the defended model as possible, trained on the same training data. If the defended model adds some new layers to a baseline model, it would be good to try the baseline. Using the undefended baseline allows optimization-based attacks to reliably construct high-confidence adversarial examples that are likely to transfer to the defended model. We also recommend trying a strong adversarially trained model (Madry et al., 2017), which has been shown to be a strong source model for adversarial examples. It is also more effective to generate adversarial examples that fool an ensemble of source models, and then use those to transfer to a target model (Liu et al., 2016).

4.6 Properly Ensemble over Randomness

For defenses that randomize aspects of neural network inference, it is important to properly generate adversarial examples by ensembling over the randomness of the defense. The randomness introduced might make it difficult to apply standard attacks since the output of the classifier as well the loss gradients used for optimization-based attack now become stochastic. It is often necessary to repeat each step of a standard attack multiple times to obtain reliable estimates. Thus, by considering multiple different choices of randomness, one can generate adversarial examples that are still for a new choice of randomness.

Defenses should be careful that relying on an exponentially large randomness spaces may not actually make attacks exponentially more difficult to attack. It is often the case that one can construct adversarial examples that are resistant to such randomness by simply constructing examples that are consistently adversarial over a moderate number of randomness choices.

Verify that attacks succeed if randomness is fixed. If randomness is believed to be an important reason why the defense is effective, it can be useful to sample one value of randomness and use only that one random choice. If the attacks fail even when the randomness is disabled in this way, it is likely that the attack is not working correctly and should be fixed. Once the attack succeeds with randomness completely disabled, it can be helpful to slowly re-enable the randomness and verify that attack success rate begins to decrease.

4.7 Approximate Non-Differentiable Layers

Some defenses include non-differentiable layers as pre-processing layers, or later modify internal layers to make them non-differentiable (e.g., by performing quantization or adding extra randomness). Doing this makes the defense much harder to evaluate completely: gradient masking becomes much more likely when some layers are non-differentiable, or have difficult-to-compute gradients.

In many cases, it can happen that one accidentally introduces non-differentiable layers (Athalye et al., 2018) (e.g., by introducing components that, while differentiable, are not usefully-differentiable) or layers that cause gradients to vanish to zero (e.g., by saturating activation functions (Brendel & Bethge, 2017; Carlini & Wagner, 2016)).

In these cases it can be useful to create a differentiable implementation of the layer (if possible) to obtain gradient information. In other case, applying BPDA (Athalye et al., 2018) can help with non-differentiable layers. For example, if a defense implements a pre-processing layer that denoises the input and has the property that $\text{denoise}(x) \approx x$, it can be effective to approximate its gradient as the identity function on the backward pass, but compute the function exactly on the forward pass.

4.8 Verify Attack Convergence

On at least a few inputs, confirm that the number of iterations of gradient descent being performed is sufficient by plotting the attack success and adversarial loss versus the number of iterations. These plots should eventually plateau; if not, the attack has not converged and more iterations can be used until it does. Plotting the adversarial loss is often more useful than plotting the attack success rate, as the former has a finer granularity and may reveal that the attack is still making progress even though the success rate seems to have plateaued. The number of iterations necessary is generally inversely proportional to step size and proportional to distortion allowed. Dataset complexity is often related.

For example, on CIFAR-10 or ImageNet with a maximum ℓ_∞ distortion of 8/255, white-box optimization attacks generally converge in under 100 or 1000 iterations with a step size of 1. However, black-box attacks often take orders of magnitude more queries, with attacks requiring over 100,000 queries not being abnormal. For a different dataset, or a different distortion metric, attacks will require a different number of iterations. While it is possible to perform an attack correctly with very few iterations of gradient descent, it requires much more thought and care (Engstrom et al., 2018). For example, in some cases even a million iterations of white-box gradient descent have been found necessary (Qian & Wegman, 2018).

There are few reasonable threat models under which an attacker can compute 100 iterations of gradient descent, but not 1000. Similarly, as stated in Athalye et al. (2018), increasing the time it takes an adversary to find an adversarial example from one second to ten seconds does typically not constitute an increase in robustness. Of course, if a future defense manages to provide strong (and preferably provable) lower bounds on the computational cost of an attack, this might still be interesting if these lower bounds are prohibitively large (e.g., the attack requires at least 2^{30} gradient evaluations). In general, because the threat model must not constrain the approach an attacker takes, it is worth evaluating against a strong attack with many iterations of gradient descent, as well as many different random restarts (to avoid converging to sub-optimal local minima (Madry et al., 2017; Mosbach et al., 2018)).

Ensure doubling attack iterations does not increase attack success rate or adversarial loss. Because choosing any fixed bounded number of iterations can be difficult to generalize, one simple and useful method for determining if "enough" iterations have been performed is to try doubling the number of iterations and check if this improves on the adversarial examples generated. This single test is not the only method to select the number of iterations, but once some value has been selected, doubling the number of iterations can be a useful test that the number selected is probably sufficient.

4.9 Carefully Investigate Attack Hyperparmeters

Start search from random offsets. When performing iterative attacks, it is useful to begin the search for adversarial examples at random offsets away from the initial, source input. This can reduce the distortion required to generate adversarial examples by 10% (Carlini & Wagner, 2017c), and help avoid causing gradient masking (Tramèr et al., 2017; Madry et al., 2017).

Moreover, even if the attack uses sufficient iterations to converge, repeating the attack with multiple randomly chosen starting points decreases the likelihood of ending up in a sub-optimal local minima (Madry et al., 2017; Mosbach et al., 2018). Doing this every time is not necessary, but doing it at least once to verify it does not significantly change results is worthwhile. Prior work has found that using at least 10 random starting points is effective.

As for verifying attack convergence, a good test would be to check that doubling the number of random restarts (e.g., from 10 to 20) does not affect the attack's success rate and adversarial loss.

Carefully select attack hyperparameters. Many attacks support a wide range of hyperparameters. Different settings of these parameters can make order-of-magnitude differences in attack success rates. Fortunately, most hyperparameters have the property that moving them in one direction strictly increases attack power (e.g., more iterations of gradient descent is typically better). When in doubt, chose a stronger set of hyperparameters.

All hyperparameters and the way in which they were determined should be reported. This allows others to reproduce the results and helps to understand how much the attacks have been adapted to the defense.

4.10 Test General Robustness for General-Purpose Defenses

Some defenses are designed not to be robust against any one specific type of attack, but to generally *be robust*. This is in contrast to other defenses, for example adversarial training (Goodfellow et al., 2014; Madry et al., 2017), which explicitly set out to be robust against one specific threat model (e.g., l_∞ adversarial robustness).

When a defense is arguing that it generally improves robustness, it can help to also verify this fact with easy-to-apply attacks. Work in this space exists mainly on images; below we give three good places to start, although they are by no means the only that are worth applying.

- Transform the image with a random rotation and translation (Engstrom et al., 2017). This attack can be performed brute-force and is thus not susceptible to gradient masking.
- Apply common corruptions and perturbations (Hendrycks & Dietterich, 2019) that mimic changes that may actually occur in practice.
- Add Gaussian noise with increasingly large standard deviation (Ford et al., 2019). Adversarially robust models tend to be more resistant to random noise compared to their standard counterparts (Fawzi et al., 2016).

In all cases, if these tests are used as a method for evaluating general-purpose robustness it is important to *not train on them directly*: doing so would counter their intended purpose.

4.11 Try Brute-Force (Random) Search Attacks

A very simple sanity check to ensure that the attacks have not been fooled by the defense is trying random search to generate adversarial examples within the threat model. If brute-force random sampling identifies adversarial examples that other methods haven't found, this indicates that other attacks can be improved.

We recommend starting by sampling random points at a large distance from the original input. Every time an adversarial example is found, limit the search to adversarial examples of strictly smaller distortion. We recommend verifying a hundred instances or so with a few hundred thousand random samples.

4.12 Targeted and Untargeted Attacks

In theory, an untargeted attack is strictly easier than a targeted attack. However, in practice, there can be cases where targeting any of the $N - 1$ classes will give superior results to performing one untargeted attack.

At the implementation level, many untargeted attacks work by *reducing* the confidence in the correct prediction, while targeted attacks work by *increasing* the confidence in some other prediction. Because these two formulations are not directly inverses of each other, trying both can be helpful.

4.13 Attack Similar-but-Undefended Models

Defenses typically are implemented by making a series of changes to a base model. Some changes are introduced in order to increase the robustness, but typically other changes are also introduced to counteract some unintended consequence of adding the defense components. In these cases, it can be useful to remove all of the defense components from the defended model and attack a model with only the added non-defense components remaining. If the model still appears robust with these components not intended to provide security, it is likely the attack is being artificially broken by those non-defense components.

Similarly, if the defense has some tunable constants where changing (e.g., by increasing) the constant is believed to make generating adversarial examples harder, it is important to show that when the constant is not correctly set (e.g., by decreasing it) the model is vulnerable to attack.

4.14 Validate any new attack algorithms introduced

Often times it can be necessary to introduce a new attack algorithm tailored to some specific defense. When doing so, it is important to carefully evaluate the effectiveness of this new attack algorithm. It is unfortunately easy to design an attack that will never be effective, regardless of the model it is attacking.

Therefore, when designing a new attack, it is important to validate that it is indeed effective. This can be done by selecting alternate models that are either known to be insecure or are intentionally designed to be insecure, and verify that the new attack algorithm can effectively break these defenses.

5 Analysis Recommendations

After having performed an evaluation (consisting of at least some of the above steps), there are several simple checks that will help identify potential flaws in the attacks that should be corrected.

5.1 Compare Against Prior Work

Given the extremely large quantity of defense work in the space of adversarial examples, it is highly unlikely that any idea is completely new and unrelated to any prior defense. Although it can be time-consuming, it is important to review prior work and look for approaches that are similar to the new defense being proposed.

This is especially important in security because any attacks which were effective on prior similar defenses are likely to still be effective. It is therefore even more important to review not only the prior defense work, but also the prior attack work to ensure that all known attack approaches have been considered. An unfortunately large number of defenses have been defeated by applying existing attacks unmodified.

Compare against true results. When comparing against prior work, it is important to report the accuracy of prior defenses under the strongest attacks on these models.If a defense claimed that its accuracy was 99% but followup work reduced its accuracy to 80%, future work should report the accuracy at 80%, and **not** 99%. The original result is wrong.

There are many examples of defenses building on prior work which has since been shown to be completely broken; performing a literature review can help avoid situations of this nature. Websites such as `RobustML`[12], are explicitly designed to help track defense progress.

5.2 Perform Basic Sanity Tests

Verify iterative attacks perform better than single-step attacks. Iterative attacks are strictly more powerful than single-step attacks, and so their results should be strictly superior. If an iterative attack performs worse than a single-step attack, this often indicates that the iterative attack is not correct.

If this is the case, one useful diagnostic test is to plot attack success rate versus number of attack iterations averaged over many attempts, to try and identify if there is a pattern. Another diagnostic is to plot the model loss versus the number of attack iterations for a single input. The model loss should be (mostly) decreasing at each iteration. If this is not the case, a smaller step size may be helpful.

Another issue could be that the iterative attack is not converging properly. A useful test is to verify that increasing the number of iterations or the number of random restarts have only marginal effect on the attack success rate and model loss.

Verify increasing the perturbation budget strictly increases attack success rate. Attacks that allow more distortion are strictly stronger than attacks that allow less distortion. If the attack success rate ever decreases as the amount of distortion allowed increases, the attack is likely flawed.

With "high" distortion, model accuracy should reach levels of random guessing. Exactly what "high" means will depend on the dataset. On some datasets (like MNIST) even with noise bounded by an ℓ_∞ norm of 0.3, humans can often determine what the digit was. However, on CIFAR-10, noise with an ℓ_∞ norm of 0.2 makes most images completely unrecognizable.

Regardless of the dataset, there are some accuracy-vs-distortion numbers that are theoretically impossible. For example, it is not possible to do better than random guessing with a ℓ_∞ distortion of 0.5: any image can be converted into a solid gray picture. Or, for MNIST, the median L_2 distortion between a digit and the most similar other image with a different label is less than 9, so claiming higher than this is not feasible.

5.3 Generate An Accuracy versus Perturbation Curve

One of the most useful diagnostic curves to generate is an accuracy versus perturbation budget curve, along with an attack success rate versus perturbation budget curve. These curves can help perform many of the sanity tests discussed in §5.2.

For some attack which produces minimally-distorted adversarial examples (Carlini & Wagner, 2017c) (as opposed to maximizing loss under some norm constraint (Madry et al., 2017)) generating these curves is computationally efficient (adversarial examples can be sorted by distance and then generating these curve can be accomplished by sweeping a threshold constant). For attacks that maximize loss given a fixed budget (Madry et al., 2017), generating this curve naively requires calling the attack once for each value of the perturbation budget, but this can be made more efficient by performing binary search on the budget on a per-example basis.

[12] `https://www.robust-ml.org/`

Perform an unbounded attack. With unbounded distortion, any attack should eventually reach 100% success, even if only by switching the input to actually be an instance of the other class. If unbounded attacks do not succeed, this indicates that the attack is being applied incorrectly. The curve generated should have an x-axis with sufficient perturbation so that it reaches 100% attack success (0% model accuracy).

For unbounded attacks, measure distortion, not success rate. The correct metric for evaluating unbounded attacks is the distortion required to generate an adversarial example, not the success rate (which should always be 100%). The most useful plot is to show success rate (or model accuracy) vs. distortion, and the most useful single number is either the mean or median distance to adversarial examples, when using unbounded attacks. To make measuring success rate meaningful, another option is to artificially bound the attack and report any adversarial example of distance greater than some threshold a failure (as long as this threshold is stated).

5.4 Verify white-box attacks perform better than black-box attacks

Because white-box attacks are a strict super-set of black-box attacks, they should perform strictly better. In particular, this implies that gradient-based attacks should, in principle, outperform gradient-free attacks. Gradient-free attacks doing better often indicates that the defense is somehow masking gradient information and that the gradient-based attack could be improved to succeed more often.

When this happens, look at instances for which adversarial examples can be found with black-box but not white-box attacks. Check if these instances are at all related, and investigate why white-box attacks are not finding these adversarial examples.

5.5 Investigate Domains Other Than Images

Adversarial examples are not just a problem for image classification, they are also a problem on sentiment analysis, translation, generative models, reinforcement learning, audio classification, and segmentation analysis (among others). If a defense is limited to one domain, it should state this fact explicitly. If not, then it would be useful to briefly consider at least one non-image domain to investigate if the technique could apply to other domains as well. Audio has properties most similar to images (high dimensional mostly-continuous input space) and is therefore an easy point of comparison. Language processing is much different (due to the inherently discrete input space) and therefore defenses are often harder to apply to this alternate domain.

5.6 Report the Per-Example Attack Success Rate

When evaluating attacks, it is important to report attack success rate on a per-example basis instead of averaged over the entire attack. That is, report $\operatorname*{mean}_{x \in \mathcal{X}} \min_{a \in \mathcal{A}} f(a(x))$, not $\min_{a \in \mathcal{A}} \operatorname*{mean}_{x \in \mathcal{X}} f(a(x))$.

This per-example reporting is strictly more useful than a per-attack reporting, and is therefore preferable. This is true despite the fact that in practice the results for the *worst* attack is often very close to the true per-example worst-case bounds.

This reporting also avoids another common pitfall in which two defenses A and B are compared on different attacks X and Y leading to statements such as "A is more robust than B against attack X while B is more robust than A on attack Y". Such results are not useful, even more so if one attack is strong and the other is weak. Instead, by comparing defenses on a per-example based (i.e. the optima per example over all attacks), one defense can be selected as being stronger. (Note this still requires the defenses are evaluated *under the same threat model*. It is not meaningful to compare a ℓ_∞-robust defense against a ℓ_0-robust defense.)

5.7 Report Attacks Applied

After having performed a complete evaluation, as discussed above, defense evaluations should report all attacks and include the relevant hyperparameters. It is especially important to report the number of iterations for optimization-based attacks.

Note that it is not necessary to report the results from every single attack in all details, especially if an attack is highly similar to another and yields similar results. In this case the space can rather be used to describe how the attacks were applied. A good guideline is as follows: do not show a massive table with many similar attacks that have not been adapted to the defense. Instead, rather show a shorter table with a more diverse set of attacks that have each been carefully adapted and tuned.

6 Conclusion

Evaluating adversarial example defenses requires extreme caution and skepticism of all results obtained. Researchers must be very careful to not deceive themselves unintentionally when performing evaluations.

This paper lays out the motivation for how to perform defense evaluations and why we believe this is the case. We develop a collection of recommendations that we have identified as common flaws in adversarial example defense evaluations. We hope that this checklist will be useful both for researchers developing novel defense approaches as well as for readers and reviewers to understand if an evaluation is thorough and follows currently accepted best practices.

We do not intend for this paper to be the definitive answer to the question "*what experiments should an evaluation contain?*". Such a comprehensive list would be impossible to develop. Rather, we hope that the recommendations we offer here will help inspire researchers with ideas for how to perform their own evaluations. [13]

In total, we firmly believe that developing robust machine learning models is of great significance, and hope that this document will in some way help the broader community reach this important goal.

Acknowledgements

We would like to thank Catherine Olsson and Úlfar Erlingsson for feedback on an early draft of this paper, and David Wagner for helpful discussions around content.

References

Anish Athalye and Nicholas Carlini. On the robustness of the cvpr 2018 white-box adversarial example defenses. *arXiv preprint arXiv:1804.03286*, 2018.

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018.

Marco Barreno, Blaine Nelson, Russell Sears, Anthony D Joseph, and J Doug Tygar. Can machine learning be secure? In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pp. 16–25. ACM, 2006.

[13] We believe that even if all defenses followed all of the tests recommended in this paper, it would still be valuable for researchers to perform re-evaluations of proposed defenses (for example, by following the advice given in this paper and investigating new, adaptive attacks). For every ten defense papers on arXiv, there is just one paper which sets out to re-evaluate previously proposed defenses. Given the difficulty of performing evaluations, it is always useful for future work to perform additional experiments to validate or refute the claims made in existing papers.

Marco Barreno, Blaine Nelson, Anthony D Joseph, and JD Tygar. The security of machine learning. *Machine Learning*, 81(2):121–148, 2010.

Arjun Nitin Bhagoji, Warren He, Bo Li, and Dawn Song. Practical black-box attacks on deep neural networks using efficient query mechanisms. In *European Conference on Computer Vision*, pp. 158–174. Springer, 2018.

Battista Biggio, Giorgio Fumera, and Fabio Roli. Multiple classifier systems for robust classifier design in adversarial environments. *International Journal of Machine Learning and Cybernetics*, 1(1-4):27–41, 2010.

Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, pp. 387–402. Springer, 2013.

Wieland Brendel and Matthias Bethge. Comment on "biologically inspired protection of deep networks from adversarial attacks". *arXiv preprint arXiv:1704.01547*, 2017.

Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. *arXiv preprint arXiv:1712.04248*, 2017.

Nicholas Carlini. Is ami (attacks meet interpretability) robust to adversarial examples? *arXiv preprint arXiv:1902.02322*, 2019.

Nicholas Carlini and David Wagner. Defensive distillation is not robust to adversarial examples. *arXiv preprint arXiv:1607.04311*, 2016.

Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 3–14. ACM, 2017a.

Nicholas Carlini and David Wagner. Magnet and "efficient defenses against adversarial attacks" are not robust to adversarial examples. *arXiv preprint arXiv:1711.08478*, 2017b.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 39–57. IEEE, 2017c.

Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David Wagner, and Wenchao Zhou. Hidden voice commands. In *USENIX Security Symposium*, pp. 513–530, 2016.

Pin-Yu Chen, Yash Sharma, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh. Ead: elastic-net attacks to deep neural networks via adversarial examples. *arXiv preprint arXiv:1709.04114*, 2017a.

Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 15–26, 2017b.

Cory Cornelius. The efficacy of shield under different threat models. *arXiv preprint arXiv:1902.00541*, 2019.

George E Dahl, Jack W Stokes, Li Deng, and Dong Yu. Large-scale malware classification using random projections and neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pp. 3422–3426. IEEE, 2013.

Nilesh Dalvi, Pedro Domingos, Sumit Sanghai, Deepak Verma, et al. Adversarial classification. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 99–108. ACM, 2004.

Gamaleldin Elsayed, Shreya Shankar, Brian Cheung, Nicolas Papernot, Alexey Kurakin, Ian Goodfellow, and Jascha Sohl-Dickstein. Adversarial examples that fool both computer vision and time-limited humans. In *Advances in Neural Information Processing Systems*, pp. 3914–3924, 2018.

Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. A rotation and a translation suffice: Fooling CNNs with simple transformations. *arXiv preprint arXiv:1712.02779*, 2017.

Logan Engstrom, Andrew Ilyas, and Anish Athalye. Evaluating and understanding the robustness of adversarial logit pairing. *arXiv preprint arXiv:1807.10272*, 2018.

Alhussein Fawzi, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Robustness of classifiers: from adversarial to random noise. In *Advances in Neural Information Processing Systems*, pp. 1632–1640, 2016.

Nic Ford, Justin Gilmer, Nicolas Carlini, and Dogus Cubuk. Adversarial examples are a natural consequence of test error in noise. *arXiv preprint arXiv:1901.10513*, 2019.

Justin Gilmer, Ryan P Adams, Ian Goodfellow, David Andersen, and George E Dahl. Motivating the rules of the game for adversarial example research. *arXiv preprint arXiv:1807.06732*, 2018.

Amir Globerson and Sam Roweis. Nightmare at test time: robust learning by feature deletion. In *Proceedings of the 23rd international conference on Machine learning*, pp. 353–360. ACM, 2006.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples (2014). *arXiv preprint arXiv:1412.6572*, 2014.

Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Relja Arandjelovic, Timothy Mann, and Pushmeet Kohli. On the effectiveness of interval bound propagation for training verifiably robust models. In *ArXiv preprint arXiv:1810.12715*, 2018.

Warren He, James Wei, Xinyun Chen, Nicholas Carlini, and Dawn Song. Adversarial example defenses: Ensembles of weak defenses are not strong. *arXiv preprint arXiv:1706.04701*, 2017.

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

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *International Conference on Learning Representations (ICLR)*, 2019.

Cormac Herley and Paul C van Oorschot. Sok: Science, security and the elusive goal of security as a scientific pursuit. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pp. 99–120. IEEE, 2017.

Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. *International Conference on Machine Learning (ICML)*, 2018a.

Andrew Ilyas, Logan Engstrom, and Aleksander Madry. Prior convictions: Black-box adversarial attacks with bandits and priors. In *International Conference on Learning Representations (ICLR)*, 2018b.

Saumya Jetley, Nicholas Lord, and Philip Torr. With friends like these, who needs adversaries? In *Advances in Neural Information Processing Systems*, pp. 10772–10782, 2018.

Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, pp. 97–117. Springer, 2017.

Auguste Kerckhoffs. La cryptographie militaire. *Journal des sciences militaires*, pp. 5–38, 1883.

Aleksander Kołcz and Choon Hui Teo. Feature weighting for improved classifier robustness. In *CEASâĂŹ09: sixth conference on email and anti-spam*, 2009.

J Zico Kolter and Eric Wong. Provable defenses against adversarial examples via the convex outer adversarial polytope. *arXiv preprint arXiv:1711.00851*, 1(2):3, 2017.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. In *Certified Robustness to Adversarial Examples with Differential Privacy*, pp. 0. IEEE, 2018.

Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. *arXiv preprint arXiv:1611.02770*, 2016.

Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. In *International Conference on Learning Representations (ICLR)*, 2017.

Daniel Lowd and Christopher Meek. Adversarial learning. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pp. 641–647. ACM, 2005.

Pei-Hsuan Lu, Pin-Yu Chen, Kang-Cheng Chen, and Chia-Mu Yu. On the limitation of magnet defense against l1-based adversarial examples. In *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pp. 200–214. IEEE, 2018a.

Pei-Hsuan Lu, Pin-Yu Chen, and Chia-Mu Yu. On the limitation of local intrinsic dimensionality for characterizing the subspaces of adversarial examples. *arXiv preprint arXiv:1803.09638*, 2018b.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2574–2582, 2016.

Marius Mosbach, Maksym Andriushchenko, Thomas Trost, Matthias Hein, and Dietrich Klakow. Logit pairing methods can fool gradient-based attacks. *arXiv preprint arXiv:1810.12042*, 2018.

Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016a.

Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pp. 372–387. IEEE, 2016b.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. 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.

Nicolas Papernot, Fartash Faghri, Nicholas Carlini, Ian Goodfellow, Reuben Feinman, Alexey Kurakin, Cihang Xie, Yash Sharma, Tom Brown, Aurko Roy, Alexander Matyasko, Vahid Behzadan, Karen Hambardzumyan, Zhishuai Zhang, Yi-Lin Juang, Zhi Li, Ryan Sheatsley, Abhibhav Garg, Jonathan Uesato, Willi Gierke, Yinpeng Dong, David Berthelot, Paul Hendricks, Jonas Rauber, and Rujun Long. Technical report on the cleverhans v2.1.0 adversarial examples library. *arXiv preprint arXiv:1610.00768*, 2018.

Haifeng Qian and Mark N Wegman. L2-nonexpansive neural networks. *arXiv preprint arXiv:1802.07896*, 2018.

Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. *arXiv preprint arXiv:1801.09344*, 2018.

Jonas Rauber, Wieland Brendel, and Matthias Bethge. Foolbox v0.8.0: A python toolbox to benchmark the robustness of machine learning models. *arXiv preprint arXiv:1707.04131*, 2017.

Jerome H Saltzer and Michael D Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9):1278–1308, 1975.

Lukas Schott, Jonas Rauber, Matthias Bethge, and Wieland Brendel. Towards the first adversarially robust neural network model on mnist. *International Conference for Learning Representations 2019*, 2019.

Yash Sharma and Pin-Yu Chen. Breaking the madry defense model with l_1-based adversarial examples. *arXiv preprint arXiv:1710.10733*, 2017.

Yash Sharma and Pin-Yu Chen. Bypassing feature squeezing by increasing adversary strength. *arXiv preprint arXiv:1803.09868*, 2018.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

Yang Song, Rui Shu, Nate Kushman, and Stefano Ermon. Generative adversarial examples. *arXiv preprint arXiv:1805.07894*, 2018.

Nedim Šrndic and Pavel Laskov. Detection of malicious pdf files based on hierarchical document structure. In *Proceedings of the 20th Annual Network & Distributed System Security Symposium*, pp. 1–16. Citeseer, 2013.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Vincent Tjeng, Kai Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. In *International Conference on Learning Representations (ICLR)*, 2019.

Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

Florian Tramèr, Pascal Dupré, Gili Rusak, Giancarlo Pellegrino, and Dan Boneh. Adversarial: Defeating perceptual ad-blocking. *arXiv preprint arXiv:1811.03194*, 2018.

Chun-Chen Tu, Paishun Ting, Pin-Yu Chen, Sijia Liu, Huan Zhang, Jinfeng Yi, Cho-Jui Hsieh, and Shin-Ming Cheng. Autozoom: Autoencoder-based zeroth order optimization method for attacking black-box neural networks. In *ArXiv preprint arXiv:1805.11770*, 2018.

Jonathan Uesato, Brendan O'Donoghue, Aaron van den Oord, and Pushmeet Kohli. Adversarial risk and the dangers of evaluating against weak attacks. *arXiv preprint arXiv:1802.05666*, 2018.

Tsui-Wei Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Duane Boning, Inderjit S Dhillon, and Luca Daniel. Towards fast computation of certified robustness for ReLU networks. In *International Conference on Machine Learning (ICML)*, 2018.

Kai Y. Xiao, Vincent Tjeng, Nur Muhammad Shafiullah, and Aleksander Madry. Training for faster adversarial robustness verification via inducing ReLU stability. In *International Conference on Learning Representations (ICLR)*, 2019.