Machine Learning Security Against Adversarial Attacks

Neha Prasad (nehap), Tenzin Ukyab (tsu), Erica Liu (ericaliu)

May 15 2019

Contents

1 Introduction

After hearing Madry's talk during lecture, we were inspired to delve deeper into the security side of image classification machine learning. Currently, state of the art machine learning models are very brittle, which means that a slight transformation of input that might be undetectable by humans can drastically change the resulting classification by the model. This can be manipulated to force certain outputs in order to take advantage of the model. As a result, machine learning algorithms need to adapt in order to respond to such attacks. In this paper, we aim to survey a set of existing techniques to defend against adversarial input, then compare and evaluate these methods.

Initial research as to why such attacks are possible by Goodfellow et al., found that neural networks are not good at local generalization [1]. For a small enough $\varepsilon > 0$, local generalization means that an input x + r, where $||r|| < \varepsilon$, will get classified the same as x with high probability. Other key insights are that adversarial examples are cross-model and cross-dataset transferable and the Lipschitz bound of deep neural nets show instability in the function.

The underlying explanation for all this phenomena concluded by Goodfellow et al. however, is that adversarial examples are a natural consequence of the linearity of the models [2]. For an adversarial example x* = x + r where $||r|| < \varepsilon$, in a linear model: $w^T x^* = w^T x + w^T r$ if $r = sign(w)$ and the inputs are in a sufficiently high dimension such that $w^T r > w^T x$. This is surprising because neural nets themselves have high non-linearity. By the universal approximator theorem, a neural net with at least one hidden layer can represent any function [3]. The problem is not with deep neural nets in general but it arises when training these neural nets. In order for them to be easily optimized, models use linear activation functions like ReLU. Highly non-linear models like Radial Basis Function (RBF) networks are more resistant. Adversarial examples are transferable because most neural nets create linear models over different subsets of the same training set. Adversarial examples generalize in the same way non-adversarial examples generalize similarly across many neural nets.

The reason why all of this is worrisome is because in such attacks on DNNs, vulnerabilities can be leveraged for specific practical applications. For example, Eykholt et al. engineered graffiti-looking black and white tape placed on stop signs with the purpose of confusing self-driving cars [4]. Sharif et al. has created patterned glasses that can be worn to fool state-of-the-art face recognition into classifying the wearer as a different person, as seen in Figure 1 [5]. Many other examples such as these exist and many more can be generated with the current state of deep neural network models.

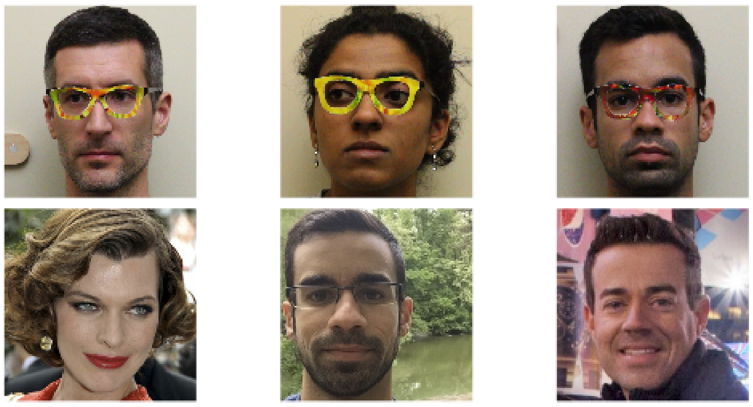

Figure 1: Examples of glasses worn to fool facial recognition into thinking that the faces in the first row of images were the same as the faces in the second row of images. [5]

In general there are multiple different possible threat models. One case is where the adversary has the capability to poison the data or place adversarial examples in the data while training. Data poisoning can come in the form of incorrectly labelling data purposefully [6] or adding an undetectable backdoor into the model using undetectable perturbations to the image [7]. Other attacks include model stealing where the adversary reverse engineers the model [8] or black box attacks where the adversary constructs adversarial examples just from queries.

Research into defense against adversarial attacks falls into two main categories that mirror general security mechanisms: detection and prevention. Detection mechanisms try to detect the adversarial examples before they are fed into the neural net. This mechanism focuses on defending against data poisoning. There are multiple categories of prevention mechanisms. One mechanism tries to make the models themselves robust to the adversarial attacks. This mechanism tries to defend against black box attacks by making the model itself resistant to adversarial perturbations. Another mechanism tries to transform the input before training in order to erase the effects of the adversary on the input. The last mechanism is gradient hiding, which is a white box attack defense mechanism. This paper will survey detection methods, input transformation methods, gradient hiding methods, and defense algorithms. It will compare the methods, examine the current state of the adversarial example field, and try to predict where the field is going.

2 Generating Adversarial Examples

This section will highlight some methods of generating adversarial examples. These attacks can be categorized into targeted or untargeted and by choice of distance measurement. Targeted methods generate adversarial examples that are classified with a chosen particular class, whereas untargeted methods generate adversarial examples that are classified with any other class that is not the true one. The distance metric is usually the L_p-norm distance metric thresholded by a ϵ.

- L_∞-norm is the maximum change in every dimension, which means it is the maximum change to each of the pixels in an image.
- L_2-norm is the Euclidean distance between the clean and perturbed image.
- L_0-norm is the number of pixels altered in the image instead of the amount all the pixels were perturbed.

2.1 Fast Gradient Sign Method

The premier method of creating adversarial examples is through the fast gradient sign method (FSGM) introduced by Goodfellow et al. [2]. It is an untargeted L_∞-norm attack. It finds the smallest possible sign gradient perturbation to the input image that will result in a different classification. This is done on an image x and correct label y with a loss function $J(\cdot, \cdot)$.

$$x' = x + \epsilon \cdot sign(\nabla_x J(g(x), y))$$

This can be efficiently calculated with backpropagation.

2.2 Jacobian Saliency Map Approach

Jacobian Saliency Map Approach (JSMA) was proposed by Papernot et al. as an L_0-norm targeted attack [9]. It tried to limit the number of pixels changed by iteratively perturbing pixels that have high adversarial saliency scores until an adversarial example with the desired target class is reached. The adversarial saliency map is the Jacobian matrix of the DNN model at the input image x. It is meant to capture how much changing that pixel will increase the output score of the target class.

2.3 Iterative FGSM or Projected Gradient Descent

Kurakin et al. introduce an L_∞-norm untargeted attack [10]. It is essentially an iterative version of FGSM. They iteratively use FGSM with a finer distortion, followed by an ϵ-ball clipping.

$$x'_N = Clip_{x,\epsilon}(x'_{N-1} + \alpha sign(\nabla_x J(x'_{N-1}, y)))$$

$$x'_0 = x$$

Here x'_N is the adversarial image at the N'th iteration and x is the clean image. Madry et al. equates this with projected gradient descent (PGD) [11].

2.4 Carlini and Wagner

Carlini and Wagner introduce L_2-norm, L_∞-norm, and L_0-norm targeted attacks [12]. The L_2 attack uses a logits based objective function and converts the target variable to the argtanh space to make it easier to optimize. It also uses binary search to find good coefficients to optimize between prediction and distance terms. All this allows for finding smaller perturbations than before, making it a stronger attack. The L_∞ attack reduces a threshold for limiting perturbations iteratively until no solution can be found. This makes the L_∞-norm problem easier to solve. Their L_0 repeats their L_2 attack to find the least important features and freeze them until the attack fails.

3 Detection Methods

There are many ways of doing adversarial example detection. This section will highlight three categories of detection methods: sample statistics, training a detector, and prediction inconsistency. This section will describe and provide examples of each. Then it will describe general failures in trying to detect adversarial examples.

3.1 Sample Statistics

In general, this approach tries to quantify the distributional difference between clean images and adversarial images. Then, it tries to find these differences in other images to see if they are adversarial. Grosse et al., for example, uses the statistical test of maximum mean discrepancy [13]. Maximum mean discrepancy tries to figure out if the images are drawn from the same underlying distribution. Feiman et al. uses kernel density estimation which measures a difference between an unknown input and a set of clean inputs using their representations in the last hidden layer of the DNN [14]. Both these approaches are computationally expensive and require lots of data. Also, the undetectable nature of adversarial examples makes it logically infeasible to try to find statistical differences between images in general.

3.2 Training a Detector

The general idea here is to modify the neural net to detect adversarial examples in addition to clean examples. The inherent problem with this approach is that it requires a lot of adversarial examples to train on and might overfit to a particular subset of adversarial examples and not generalize well. Grosse et al. adds a new "adversarial" class to the last layer of DNN [13]. Metzen et al. branches the DNN at a middle layer, making one branch an adversarial detector [15].

3.3 Prediction Inconsistency

The main idea of prediction inconsistency is to measure the difference between the predictions of multiple different models. One example of this is feature squeezing as seen in Figure 2. It takes the input image and "squeezes" out unnecessary input features in various ways. Then the squeezed images and the clean image are given to the model to classify. If the model returns a prediction for the squeezed images that are far from the clean image then the image is probably adversarial because the squeezed images are meant to only have features relating to the true classification, not the adversarial ones. In this case, it is not given to the model as training input. There are many ways that the input images can be squeezed. Wu et al. use depth reduction and a smoothing spatial filter [16] and Liang et al. use scalar quantization by another research group [17].

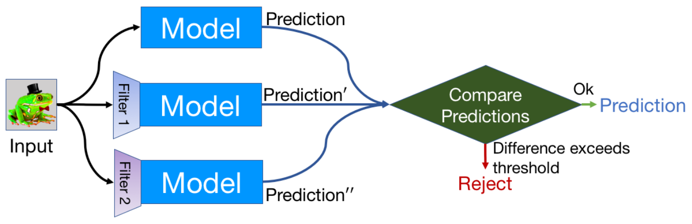

Figure 2: Visualization of Feature Squeezing [16]

4 Input Transformation

The following are some input transformation methods. They transform the input before giving it to the model to train on which leaves the model protected against adversarial examples.

4.1 Principal Component Analysis

Bhagoji et al. uses principal component analysis to reduce the dimensionality of the input image, so that only the top K principal components are left for the DNN to train on. In order for the attacker to create an adversarial example, they have to apply much bigger perturbations. PCA can also be used for detection because they place a high weight on larger principal components [18] by using it in analyzing hidden layers of the DNN [19].

4.2 Normalization

There are many possible approaches to normalizing a image. Feiman et al. suggests adding randomization to the network [14]. Li et al. suggests using an averaging filter to blur the image [18].

5 Failures of Detection and Input Transformation

The above detection and input transformation methods, however, were all shown to be ineffective by Carlini and Wagner [20]. Through rigorous testing and measurements they find that most of the methods were vulnerable to white box tests and weak to strong attacks. They suggest that methods developed in the future take into account the white box security of the methods because they are completely ignored right now. All of the 10 papers Carlini and Wagner cover only evaluate their methods using FGSM and JSMA attacks. They should have evaluated their models with the strongest attacks known, the CW and PGD attacks. Methods that are evaluated with the CW and PGD attacks, such as feature squeezing [16], however, sometimes do not evaluate with a strong enough adversary. Sharma et al. show that increasing the κ constraint coefficient and L_∞ constraint ϵ coefficient for CW and PGD attacks, respectively, lead to bypassing feature squeezing [21].

Carlini et al. conclude that security evaluation for detection methods need to be improved to a higher standard. They also conclude that detection of adversarial examples is an inherently hard process because adversarial examples are constructed to be hard to detect [20]. From this the general conclusion is that research should steer toward defense methods instead of detection methods. The next three defense algorithms are covered in depth to see if they fare better than these detection and input detection methods.

6 Obfuscated Gradients

A popular yet flawed method for defending against adversarial examples is obfuscated gradients. Gradients and gradient descent are often used to construct adversarial examples. The idea behind obfuscated gradients is that if you can design the gradient to be meaningless, hard to perform, or non-existent, the adversary can no longer use it to generate adversarial examples.

In their paper on obfuscated gradients [22], Athalye, Carlini, and Wagner identified three main methods of obfuscating gradients:

- Shattered gradients: obfuscate the gradient by making it incorrect or non-existent. Examples include designing the gradient to be non-differentiable or designing the gradient so that following it does not lead to local maxima.

- Stochastic gradients: obfuscate the gradient by introducing randomness (for example, randomizing input before feeding it into the network or randomizing the network itself), rendering the gradient meaningless.

- Vanishing/exploding gradients: obfuscate the gradient by iterating through the network multiple times, feeding the results of one iteration as input into the next, producing what is essentially a very deep neural network in which the gradient vanishes.

Athalye, Carlini, and Wagner found ways to circumvent these three methods as well. On networks using shattered gradients, they performed a forward pass through the network as is but used differentiable approximations of non-differentiable layers in the backward pass to generate adversarial examples. On networks using stochastic gradients, they computed the gradient over the expected transformation to the input. On vanishing/exploding gradients, they used reparameterization to change functions so that differentiating on them would not cause the gradient to vanish or explode.

Obfuscated gradients are not a secure defense against adversarial examples because there are ways to estimate an obfuscated gradient, as Athalye, Carlini, and Wagner demonstrated. Despite this shortcoming, 7 of 9 proposed defense methods at ICLR 2018 relied on obfuscated gradients. Of those 7, Athalye, Carlini, and Wagner completely circumvented 6 and partially circumvented 1. The two that didn't rely on obfuscated gradients are two of the most robust defenses proposed yet. These two defenses were Cascade Adversarial Training (Section 7), proposed by Na et al., and Robust Adversarial Training (Section 8), proposed by Madry et al.

7 Cascade Adversarial Training

Injecting adversarial examples during training is known to improve the robustness of one-step attacks, which are namely attacks that generate an adversarial perturbation by performing a single step computation. The opposite are iterative methods that perform the same computation multiple times to achieve a single perturbation. To tackle the more complex challenge of unknown iterative attacks, Taesik Na and his team at Georgia Tech propose two novel methods: cascade adversarial training and low-level similarity learning. [23]

7.1 Cascade Adversarial Training

Cascade adversarial training uses an already defended network for generating "iterative" adversarial images, and then incorporates those images along with one-step adversarial images (from the network being trained) into the training set. This process can be seen in Figure 3.

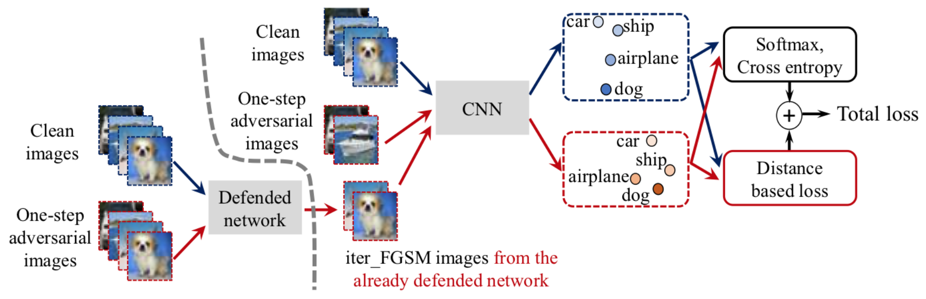

Figure 3: Diagram of cascade adversarial training [23].

The reasoning for the approach is based on the observation that it is efficient to attack an undefended network with iterative adversarial examples crafted from another undefended network and it is efficient to attack a defended network with iterative adversarial examples crafted from another defended network. There exists a high correlation in the strength of the attack between examples generated from similar networks. Therefore, in order to build a robust network, it is most optimal to construct adversarial images generated from networks trained with the same strategy. For cascade adversarial training, a defended network is used to generate the "iterative" adversarial images.

7.2 Low-Level Similarity Learning

The goal of Low-Level (pixel-level) Similarity learning is to add additional regularization to encourage a network to be insensitive to adversarial perturbation. In particular, this method involves using adversarial images along with their corresponding clean images in the training set. In addition, the distance between the embeddings of the clean and adversarial images is penalized. The distance-based loss encourages two similar images (clean and adversarial) to produce the same outputs, not necessarily the true labels.

The success of low-level similarity learning can be attributed to the fact that small differences in input should not drastically change the high level feature representation (embedding). Moreover, by penalizing the differences between the adversarial and clean images during regularization, convolution filters gradually learn how to ignore such pixel-level perturbation.

The one issue that low-level similarity learning poses is that during white-box attacks, low-level similarity learning serves as a good regularizer for adversarial images, but results in reduced accuracy for classifying clean images. Distance-based loss pulls adversarial embeddings toward their corresponding clean embeddings. During this process, clean embeddings from other classes might also be moved toward the decision boundary which results in decreased accuracy for the clean images.

7.3 Results

Based on white-box and black-box attack analysis, combining Cascade Adversarial Training with Low-Level Similarity learning improves robustness against iterative adversarial attacks, but at the expense of decreased accuracy for one-step attacks compared to the baseline defended network. Further research is necessary to improve the robustness against iterative attacks without sacrificing the accuracy for one-step attacks.

8 Robust Adversarial Training

In their paper, Madry et. al. seek to address the problem of adversarial attacks through training more robust classifiers [11]. Most classifiers are trained to do well against a standard loss function based on empirical risk minimization (ERM). For inputs x and their corresponding labels y, the goal is to find parameters θ such that you minimize the loss $L(x, y, \theta)$.

Training based on ERM, however, does not produce models that are robust against attacks using adversarially crafted examples. To train models that are robust against adversarial examples, Madry et al. instead suggest training models to do well against a robust loss function dependent on the set of perturbations an adversary can perform on the input data. To do so, Madry et al. specify a threat model and a security guarantee:

- Threat model: for each data point x, the adversary can apply a perturbation d from the perturbation set S such that the "true" label of x has not been altered, but the model changes and outputs an incorrect label for x. Madry et. al. specify legal perturbations using l_∞-ball bounds.

- Guarantee: the model will be robust against adversarial examples generated as specified in the threat model.

They then alter the standard loss function to account for robustness against adversarially crafted examples. The new robust loss function is:

$$\min_\theta \rho(\theta) \text{ where } \rho(\theta) = E_{(x,y)\sim D}[\max_{\delta \in S} L(\theta, x + \delta, y)]$$

where x, y, and θ are what they were above, S is the set of possible perturbations, D is the data distribution, and δ is the perturbation applied on the data point x. We now judge a model by how low its loss is for inputs that have had perturbations that produce maximum lost applied to them. Madry et al. state that solving this saddle problem and obtaining a small loss from it is key to training a network robust against adversarial attacks.

Since stochastic gradient descent (SGD) is a common and effective way to train a network, gradients of the robust loss function will need to be calculated. Madry et al. compute the gradient of the outer minimization problem by computing the gradient at the maximizer of the inner maximization problem (this is shown to be a valid method by an approximation of Danskin's Theorem and successful experiments run by Madry et al.).

Because finding a maximizing perturbation corresponds to finding an attack on the network, common algorithms for constructing adversarial examples can be used as inner maximization algorithms. One such algorithm is FGSM (see Section 2.1):

$$x + \epsilon * sgn(\nabla_x L(\theta, x, y))).$$

This can be viewed as a single-step maximization. FGSM^k, which Madry et al. equate essentially with PGD (see Section 2.3), can be used to do a multi-step maximization.

Madry et al. found that PGD is especially helpful for robust training because the local maxima it finds tend to be close in loss value for both normally and adversarially trained networks. This clustering of loss values suggests that PGD will generally find the local maxima with the greatest loss values and that models trained to be robust against PGD adversaries will consequently be robust against all first-order adversaries, adversaries that use only first-order information (e.g. gradients). If the adversary uses only first-order information, it probably won't find a significantly better local maxima than PGD would; all local maxima found by PGD tend to be close in loss value, implying that there probably aren't points with higher loss value. This is validated by Madry et al.'s experiments.

Madry et al. solve the saddle problem by using PGD to solve the inner maximization problem and using SGD to solve the outer minimization problem. The general process Madry et al. use to find the best parameters for an adversarially robust model is [24]:

1. Sample a data point x, y.
2. Compute the maximizer x^* of the robust loss $\rho_{x,y}(\theta)$.
3. Compute the gradient $g = \nabla_\theta L(f_\theta(x^*), y)$.
4. Update θ with the gradient g.
5. Repeat steps 1-4 until convergence.

where f is the classifier and θ are the parameters of the model. This process, using PGD for (2) and SGD for (3), performed well on the MNIST and CIFAR10 datasets. Madry et al. tested their defense against a variety of attacks, including the strongest attacks known, the CW and PGD attacks (see Section 5). On the MNIST dataset, the networks Madry et al. trained were very robust, maintaining 89.3% accuracy against the most successful attack. They were less robust on the CIFAR10 dataset, with 45.8% accuracy on the most successful attack, but still achieved generally high accuracy across the different attacks run. Madry et al.'s experiment descriptions and results are shown in Figure 4.

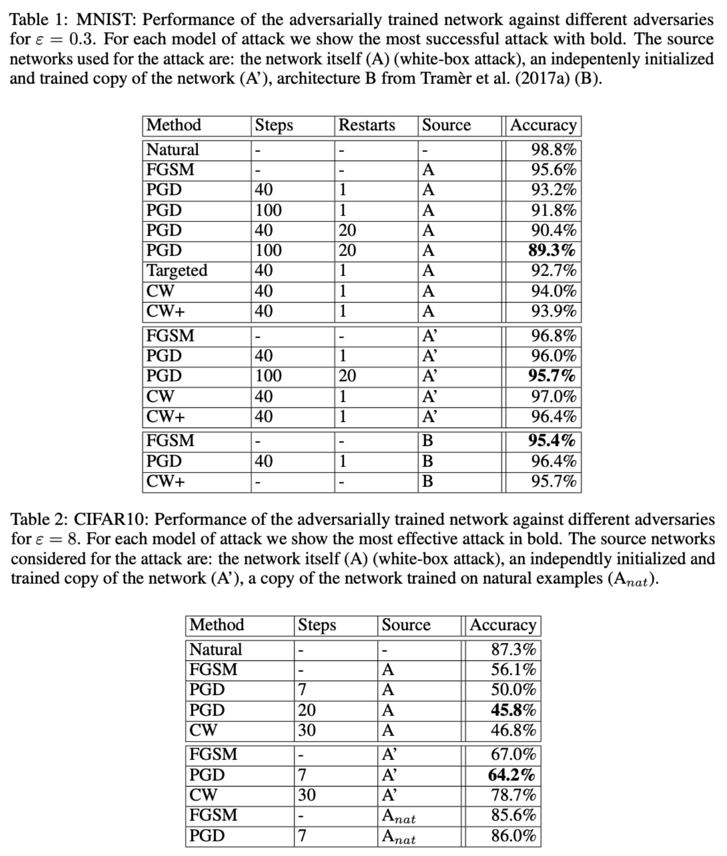

Table 1: MNIST: Performance of the adversarially trained network against different adversaries for $\varepsilon = 0.3$. For each model of attack we show the most successful attack with bold. The source networks used for the attack are: the network itself (A) (white-box attack), an indepentenly initialized and trained copy of the network (A'), architecture B from Tramèr et al. (2017a) (B).

Method	Steps	Restarts	Source	Accuracy
Natural	-	-	-	98.8%
FGSM	-	-	A	95.6%
PGD	40	1	A	93.2%
PGD	100	1	A	91.8%
PGD	40	20	A	90.4%
PGD	100	20	A	**89.3%**
Targeted	40	1	A	92.7%
CW	40	1	A	94.0%
CW+	40	1	A	93.9%
FGSM	-	-	A'	96.8%
PGD	40	1	A'	96.0%
PGD	100	20	A'	**95.7%**
CW	40	1	A'	97.0%
CW+	40	1	A'	96.4%
FGSM	-	-	B	**95.4%**
PGD	40	1	B	96.4%
CW+	-	-	B	95.7%

Table 2: CIFAR10: Performance of the adversarially trained network against different adversaries for $\varepsilon = 8$. For each model of attack we show the most effective attack in bold. The source networks considered for the attack are: the network itself (A) (white-box attack), an independtly initialized and trained copy of the network (A'), a copy of the network trained on natural examples (A_{nat}).

Method	Steps	Source	Accuracy
Natural	-	-	87.3%
FGSM	-	A	56.1%
PGD	7	A	50.0%
PGD	20	A	**45.8%**
CW	30	A	46.8%
FGSM	-	A'	67.0%
PGD	7	A'	**64.2%**
CW	30	A'	78.7%
FGSM	-	A_{nat}	85.6%
PGD	7	A_{nat}	86.0%

Figure 4: Description and results of Madry et al.'s experiments [11].

One drawback to this more robust way of training classifiers Madry et al. proposed is that it is significantly more computationally expensive than the standard way of training classifiers. Training to satisfy the standard ERM loss function requires just one forward and one backward pass through the network, while training to satisfy the robust saddle function requires k additional forward and backward passes where k is the step size of the PGD adversary.

What sets Madry et al. apart from other attempts to protect against adversarial attacks? Madry et al. are more general in their approach. One shortcoming of many attempts to protect against adversarial attacks noted by both Madry et al. and Carlini and Wagner is that many attempts are too specific in the kinds of attacks they are defending against. Madry et al. note that because of this, these attempts do not offer a good idea of the kinds of guarantees they offer. Madry et al. on the other hand provide a more specific security guarantee against a broad class of attacks. Their approach to protect against adversarial attacks focuses on prevention against a defined class of attacks.

9 Comparison of Madry and Na

9.1 Differences

The differences between the two models are summarized in the following table. The main areas where the two differ are in their main idea, process, attacks addressed, and drawbacks.

	Madry	**Na**
Main Idea	Focus on a threat model and concrete security guarantee	Effective to attack and defend a model with iterative images drawn from networks trained with same strategy
Process	Solve saddle problem to find best parameters for model	Train model, generate and add adversarial examples to training set, train second model
Attacks Addressed	All First-order Attacks	Just Iterative Attacks
Drawbacks	Can be computationally expensive (using $FGSM^k$ results in k more forward and backward passes through the network)	Performance against iterative attacks at the expense of performance against one-step; relies on already-defended networks

9.2 Similarities

Despite the differences in their methods, Madry's and Na's defenses share many qualities that set them apart from the rest of the field.

Both defense methods are general. They are designed for robustness against general classes of adversarial attacks rather than for robustness against specific kinds of adversarial attacks. Focusing on robustness against specific attacks often leads to an unclear idea of how a model will behave against attacks that aren't addressed by its defense and makes fulfilling specific security guarantees difficult. Madry and Na avoid these pitfalls by designing defenses that aim to produce generally robust models.

Both defenses also seek to prevent the success of adversarial examples, rather than detecting adversarial examples and discarding them before they are fed into the model. Detection of adversarial examples is inherently difficult because adversarial examples are designed to be hard to detect, and detection methods are easily bypassed, as observed by Carlini and Wagner [20]. Instead of attempting to detect adversarial examples, Madry's and Na's defenses seek to train their models to be robust to them, preventing adversarial examples from causing significant loss rather than detecting and removing them.

Both Madry and Na also modified the loss functions against which they trained their models. Though they modified the loss functions in different ways, with Madry converting it into a saddle problem to evaluate loss on maximally perturbed inputs and Na accounting for distance-based loss between the clean and adversarial embeddings, both did so to account for adversarially-produced distance between an input's "true" label and the label outputted by the model. Madry and Na both understood the nature of their adversary and adapted their training processes to account for it, resulting in more robust models.

Madry's and Na's defenses are also both well-evaluated. In 2017, Carlini and Wagner observed that most proposed defenses were not evaluated strongly enough, and as a result were not as robust as their papers claimed they were [20]. Carlini and Wagner suggested that researchers test their defenses against strong attacks, including white box attacks, and test their defenses on multiple datasets (for example, both CIFAR10 and MNIST). Madry and Na both test their defenses against multiple attacks, including strong attacks formulated by Carlini and Wagner, and both test their defenses on CIFAR10 and MNIST.

Madry and Na set good examples of how defenses against adversarial examples should be formulated and evaluated, but their defenses also share some shortcomings with the rest of the field that future research is starting to address. Because Madry's work is more generally straightforward and applicable, it has been having more influence on the field and many recent papers are focusing more on evaluating his work than on evaluating Na's Cascade Adversarial Training method.

10 Future of the Field

Looking into the future for the field of adversarially robust machine learning, there are two things to consider. One is the future possible research directions and the other is the expectations and standards that those research directions will be evaluated on. Madry and Na's work are great research directions to future improve and pursue. There are even more adversarially robust algorithms in development that provide other research directions. However, without stricter evaluation criteria and standards of evaluation, research conclusions might be misguided and slow down development. This section will address problems in the field and other promising algorithms that have been proposed in ICLR 2019. Then, it will address the question of evaluation standards.

10.1 Future of the Madry Algorithm

The adversarial training procedure proposed by Madry is one of the most effective and popular methods to defend against adversarial examples. As a result, there are a lot of papers that shed light on the weaknesses of the Madry algorithm. Sharma et al. show us how it is very susceptible to L_1 amd L_2-based attacks [26]. This is due to the unrealistic constraint put on the adversary. Madry's algorithm only considers a ε=0.3 scaled L_∞ attack.

Another example is the ICLR 2019 paper by Zhang et al. [25]. It demonstrates the practicality and hardness of adversarial training by showing that the effectiveness of adversarial training has a strong correlation with the distance between a test point and the "manifold" of training data in the network. Test examples that are relatively far away from this "manifold" are more likely to be vulnerable to adversarial attacks.

Consequentially, an adversarial training based defense is susceptible to a new class of attacks, the "blind-spot attack", where the input images reside in "blind-spots" of the empirical distribution of training data. For MNIST, for example, these "blind-spots" can be easily found by simply scaling and shifting image pixel values. The "blind-spot attacks" have the following two properties.

1. They are still drawn from the ground-truth data distribution and classified correctly by the model.
2. Adversarial training cannot provide good robustness properties on these images, and we can easily find their adversarial examples with small distortions using a simple gradient based attack.

Although the blind-spot attack is beyond the threat model considered in adversarial training, the argument is that adversarial training is unlikely to scale well to datasets that lie in a high dimensional manifold, as the limited training data only guarantees robustness near these training examples.

These results suggest that MNIST is far from being solved in terms of adversarial robustness. Schott et al.'s paper from ICLR 2019 presents a novel robust classification model that performs analysis by synthesis using class-conditional data distributions [27]. The goal of this method is to learn a causal model of the inputs, and by classifying a new sample according to the class under which it has the highest likelihood. This approach is referred to as the Bayesian classifier.

10.2 Robustness Evaluation Criteria

Based on the pitfalls presented above about detection methods, input transformation, gradient hiding, and Madry's algorithms, it is obvious an evaluation standard for the robustness of algorithms is necessary. Without it, researchers mistakenly announce the robustness of the algorithm and impede progress in the field. Here are some suggestions for what researchers should consider while evaluating their methods:

1. Consider an adversary with complete white box access to the model.
2. Do not assume that information can be hidden about the system because it can be estimated.
3. Consider the strongest adversary possible. This means all possible threshold parameters of attacks and stronger attack methods.

This is of course not an exhaustive list. Carlini and many other leading researchers in the field have compiled a more detailed and thorough list of recommendations in hopes of creating a standard [28]. It lists out all the possible attacks to consider and a methodology for evaluation. It is a living document that people are invited to improve and work on. They do not only mention evaluation standards but also mention best practices in order to facilitate the advancement of the field. Some examples include publishing code used and standards on reporting accuracy to help with the reproducibility of the research.

References

[1] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations* (ICLR), 2014. https://arxiv.org/pdf/1312.6199.pdf

[2] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. In *International Conference on Learning Representations* (ICLR), 2015. https://arxiv.org/pdf/1412.6572.pdf

[3] Kur Hornik, Maxwell Stinchcombe, and Halber White. Multilayer feedforward networks are universal approximators. In *Neural Networks* Vol 2, 1989. https://www.cs.cmu.edu/ epxing/Class/10715/reading/Kornick_et_al.pdf

[4] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, Dawn Song Robust Physical-World Attacks on Deep Learning Visual Classification. In *Computer Vision and Pattern Recognition* (CVPR 2018)

[5] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Micheal K. Reiter. Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition. In *ACM Conference on Computer and Communications Security* (ACM CCS), 2016. https://www.cs.cmu.edu/ sbhagava/papers/face-rec-ccs16.pdf

[6] Pang Wei Koh and Percy Liang. Understanding Black-box Predictions via Influence Functions. In *International Conference on Machine Learning*, 2017. https://arxiv.org/abs/1703.04730

[7] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain. In *IEEE*, 2017. https://arxiv.org/pdf/1708.06733.pdf

[8] Florian Tramer, Fan Zhang, Ari Juels, Micheal K. Reiter, and Thomas Ristenpart. Stealing Machine Learning Models via Prediction APIs. In *USENIX Security*, 2016 https://arxiv.org/pdf/1609.02943.pdf

[9] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The Limitations of Deep Learning in Adversarial Settings. In *IEEE European Symposium on Security and Privacy*, 2016. https://arxiv.org/abs/1511.07528

[10] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial Examples in the Physical World. In International Conference on Learning Representations (ICLR) Workshop, 2017. https://arxiv.org/abs/1607.02533

[11] Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2017. https://arxiv.org/pdf/1706.06083.pdf

[12] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy*, 2017. https://arxiv.org/abs/1608.04644

[13] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. On the (Statistical) Detection of Adversarial Examples. arXiv preprint 1702.06280, 2017.

[14] Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. Detecting Adversarial Samples from Artifacts. arXiv preprint 1703.00410, 2017.

[15] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On Detecting Adversarial Perturbations. In *ICLR*, 2017. https://arxiv.org/pdf/1702.04267.pdf

[16] Weilin Xu, and David Evans. Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks. In *Network and Distributed Systems Security Symposium* (NDSS), 2017. https://arxiv.org/pdf/1704.01155.pdf

[17] Bin Liang, Hongcheng Li, Miaoqiang Su, Xirong Li, and Wenchang Shi. Detecting Adversarial Image Examples in Deep Networks with Adaptive Noise Reduction. In *IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING*, 2017. https://arxiv.org/abs/1705.08378

[18] Xin Li and Fuxin Li. Adversarial Examples Detection in Deep Networks with Convolutional Filter Statistics. arXiv preprint arXiv:1612.07767, 2016. https://web.engr.oregonstate.edu/ lif/3060_merge.pdf

[19] Dan Hendrycks and Kevin Gimpel. Early Methods for Detecting Adversarial Images. In *International Conference on Learning Representations* (Workshop Track), 2017. https://arxiv.org/abs/1608.00530

[20] Nicholas Carlini and David Wagner. Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods. In *ACM Conference on Computer and Communications Security*, 2017. https://arxiv.org/pdf/1705.07263.pdf

[21] Yash Sharma and, Pin-Yu Chen. Bypassing Feature Squeezing by Increasing Adversary Strength. In *ICLR* (Workshop Track), 2018. https://openreview.net/pdf?id=SkV0ptkvf

[22] Anish Athalye, Nicholas Carlini, David Wagner. Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. In *ICML*, 2018. https://arxiv.org/abs/1802.00420

[23] Taesik Na, Jong Hwan Ko, and Saibal Mukhopadhyay. Cascade Adversarial Machine Learning Regularized with a Unified Embedding. In *ICLR*, 2018. https://openreview.net/pdf?id=HyRVBzap-

[24] Aleksander Madry, Ludwig Schmidt, Dimitris Tsipras. Training Robust Classifiers. https://gradientscience.org/robust_opt_pt1/

[25] Huan Zhang, Hongge Chen, Zhao Song, Duane Boning, Inderjit Dhillon, and Cho-Jui Hsieh. The Limitations of Adversarial Training and the Blind-Spot Attack. In *ICLR*, 2019. https://openreview.net/pdf?id=HylTBhA5tQ

[26] Yash Sharma, and Pin-Yu Chen. Attacking the Madry Defense Model with L1-Based Adversarial Examples. In *ICLR* Workshop Track, 2018. https://arxiv.org/pdf/1710.10733.pdf

[27] Lukas Schott, Jonas Rauber, Matthias Bethge, and Wieland Brendel. Towards the First Adversarially Robust Neural Network Model on MNIST. In *ICLR*, 2019. https://openreview.net/forum?id=S1EHOsC9tX

[28] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, Alexey Kurakin. On Evaluating Adversarial Robustness. arXiv preprint 1902.06705, 2019. https://arxiv.org/pdf/1902.06705.pdf