Adversarial Machine Learning
(AI Safety and Security Textbook)

August 1, 2017

Contents

0 Introduction

With all of the resources and interest recently concentrated into AI, we might expect intelligent agents to emerge in a variety of forms, each with its own set of unique capabilities. The possibilities are endless, but we've already seen examples where we have difficulty comprehending the decisions an AI makes.

In the past decade, machine learning and more specifically neural networks have powered the explosion of AI advancements. Researchers who leverage neural networks have made unprecedented progress towards solving open problems in areas such as image recognition, audio processing, language modeling, and many others. State-of-the-art algorithms often outperform humans in some of these areas.

Despite these advancements, perturbations to the input data that are imperceivable to humans can fool neural networks. In fact if this perturbation is carefully crafted, the neural network will be highly confident in its incorrect prediction! To understand what these sort of perturbations can look like, look at figure 1.

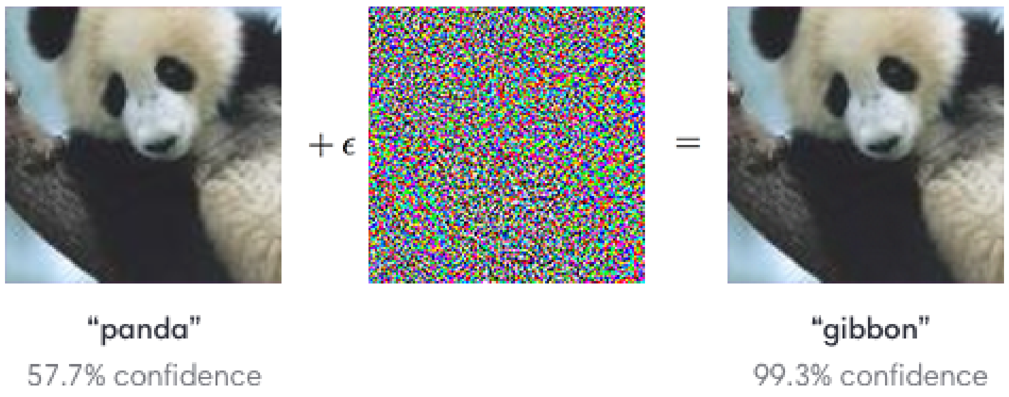


Figure 1: An adversary takes an image of a panda (on the left) applies a perturbation (which just looks like noise) and confuses the classifier to mislabel the image as a gibbon, without changing the image in a perceivable amount. [1]

These sorts of images and data that are similar to it are known as *adversarial examples* and the subfield of machine learning that studies failure cases such as these is known as *adversarial machine learning.*

1 Self-Driving Cars and Adversarial Machine Learning: A Case Study

Self-driving cars have received an enormous amount of attention in the past few years. Companies like Uber, Google, GM, and Apple are racing to be the first to develop robust and safe self-driving cars for commercial use.

Recent developments in machine learning have made building a self-driving feasible. But what exactly does a good driver need to pay attention to? Other cars? The weather? Obstacles? Accounting for hundreds of dynamic variables like these is what makes developing self-driving cars so challenging. However difficult this problem is, the social benefit of a solution is immense. In 2016, there were over 40,000 traffic fatalities in the US alone [2], a majority of which are caused by lapses of human judgment and limitations of human perception.

Self-driving cars will eventually outperform human drivers, given the sheer quantity of data that a computer can process. By leveraging advanced tools like LIDAR and installing cameras around the vehicle, a self-driving car can maintain an almost constant 360 degree understanding of its environment. To deal with this large volume of data, self-driving car companies turn towards neural networks. Neural networks can effectively recognize obstacles and road signs in real time. However, as we saw at the beginning of this chapter, these algorithms are also susceptible to attack. An adversary could develop inputs that easily fool a car's neural network into making an overconfident, erroneous, and potentially fatal decision.

What if, in a world full of self-driving cars, an attacker put a seemingly transparent sticker onto a stop sign that fools a self-driving car vision system into thinking the stop sign is in fact a green light? Although the sticker isn't noticeable to humans, our lack of understanding of how exactly neural networks recognize objects can lead to extremely hazardous situations for the car's passengers and other cars on the road.

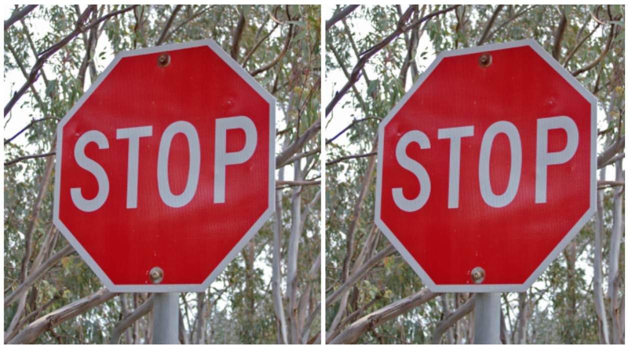


Figure 2: Left: A normal image of a stop sign. Right: a stop sign after an attacker applies a perturbation

2 Black-Box Function Model

To understand why adversarial examples exist, it's important that to have an understanding behind how machine learning models work. At the most general level, we can view a machine learning model as a general black box: it takes in some input, e.g. an image or stock ticker information, processes the data according to some internal logic or rule set, and produces a desired output. The input is typically referred to as an *example* and is denoted by X, while the output is referred to as the *label*, denoted by Y.

There are two types of problems that models try to solve: classification, which identifies categories such as objects in images or harmless vs malignant tumors, and regression, which makes predictions for real-valued labels like the price of a house or the price of a stock ten days from now. To ensure that our model identifies the right objects or makes the right predictions, we have to show it many examples of what we want it to learn, such as a thousand images of cats and their breeds or a million different houses and their prices. This process is known as *training* and the data used is known as *training data*. We use a *loss function* to compare the outputs and the expected label as well as quantitatively measure how much the model has improved. Whenever we use the model to make a prediction, we can say that the model performs *inference*.

This view of the machine learning model is useful, because it lets us consider the nature of adversarial machine learning from a more abstract point of view. An attacker could modify the input X in some small way, unnoticeable to the human eye, to make this model output an incorrect Y. This effectively allows the adversary to "hack" the model's prediction system without humans noticing anything wrong, until damage has been done.

We can generally think about machine learning models as belonging to two categories: white box, and black box. White box models are interpretable: we can understand them and their components intuitively. Black box models, while effective, are not interpretable: we can understand their inputs and outputs, but we cannot explain intuitively how the model works internally.

3 Types of Adversarial Threats

As we begin talking about adversarial threats to machine learning models, we must be able to compare various forms of attack. For simplicity's sake, we will focus primarily on classification, or category identification problems, although threats for regression problems are similar. Attacks are defined both by the particular model targeted and by the attacker's knowledge about the model. We can then build a taxonomy of attacks based on these two variables.

Papernot et. al. [3] defines four attacks that an adversary can use on a machine learning model, ordered by increasing difficulty:

confidence reduction threats: this attack reduces a model's confidence in its prediction. Similar to how frosted glass obscures people's vision, an adversary may want to add noise in a system in order to make the model less accurate. These tend to be the easiest attacks for an adversary to carry out because they only require a small perturbation (such as adding noise to the image of the panda in Figure 1) to be successful. By rendering a model's classification more ambiguous, we can introduce indecisiveness.

source misclassification threats: a misclassification attack is a more extreme version of a confidence reduction attack. It shifts a prediction further, causing the classifier to incorrectly label the example with some incorrect class.

targeted misclassification threats: these attacks aim to fool the model into a particular

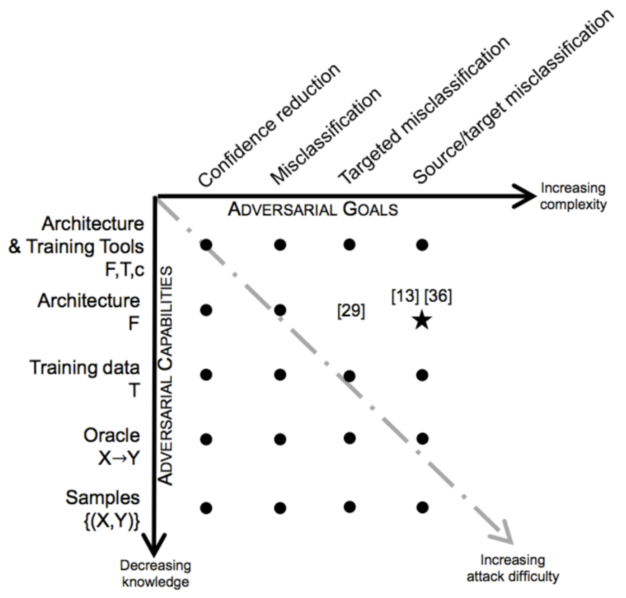


Figure 3: Note that the horizontal axis denotes the attacker's objective, increasing in complexity as we move to the right. The vertical axis denotes the amount of information the attacker must know in order to execute the attack, decreasing in value as we move downwards. Note that the more complex the attack, the more difficult it would be for an adversary to perform it. Likewise, the more an attacker knows about the model, the easier it is to attack it. [3]

classification (target class), by generating examples that look like noise, but which can cause a model to be highly confident in a particular prediction. To carry out this attacks, an adversary typically needs to have data from the target class.

source/target misclassification threats: the most specific attacks, aim to force the model into misidentifying an example from class A as belonging to a particular class B. These are the hardest to achieve because there is very little wiggle room for both the direction and the amount of manipulation. Source/target misclassification attacks are also the most dangerous attacks.

The information available to the attacker establishes the difficulty of the attack. On one end of the spectrum, imagine an adversary who knows how the model was designed and what data it was trained on. Clearly, this adversary is better prepared to attack the model than one who only has access to a model's predictions. Papernot defines a spectrum of levels of information that could be available to the user as follows:

Training data and architecture: the adversary knows everything about the model and

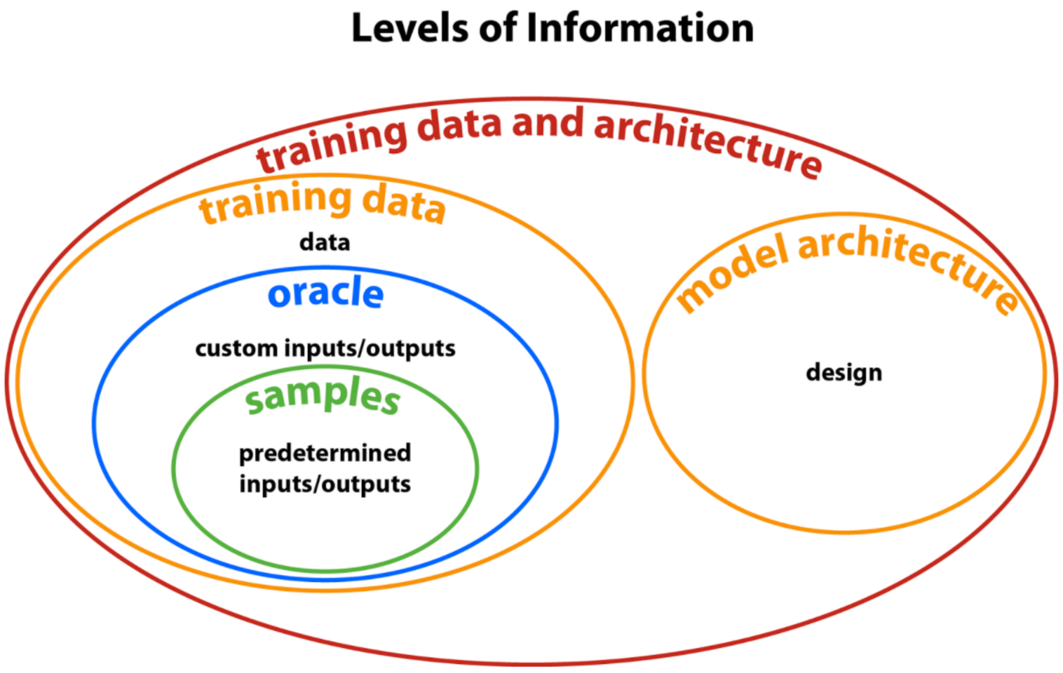


Figure 4: A visualization of the 5 levels of information. Color indicates severity of the information.

training data, from the exact design of the architecture and the model's learned parameters, to the data used to train the model.

Model architecture: The adversary knows something about the model, but nothing about the data used to train the model.

Training data: The adversary knows something about the data used to train the model, but nothing about the model.

Oracle - The network is effectively just a black box to the attacker. As such, the attacker can present the model with any input, and observe the corresponding output.

Samples - The attacker has only a list of pre-determined inputs and their corresponding outputs. Unlike the Oracle, the attacker cannot present the model with new inputs.

Engineers can improve a model's defense against adversarial example attacks by hiding information about the design of the model and the data it was trained on. That way, attacks, especially complex ones, become less likely to be successful.

With this in mind, we encourage you to reference back to figure 3, as it will serve as the backbone of our adversarial attack taxonomy. If you understand this figure and the relationships within it, you will be better equipped to understand the tradeoff between model exposure and vulnerability, which stands at the heart of adversarial machine learning.

Many attacks in adversarial machine learning fall into these five categories. Of course, some attacks are more effective than others. We will cover models from across these categories, but we will focus primarily on those which are both more immediately threatening and impactful across various situations.

4 High-Level Problem Overview

In this section we show how an attacker would generate adversarial examples (see section 0 for more details). Suppose an agent runs a machine learning model for some prespecified purpose. This agent could be a company, government agency, a hobbyist, etc. An attacker with the desire to fool this model will first have to acquire some information about it. The amount of information will dictate what kind of attacks an adversary could perform, as discussed in more detail in section 3. Leveraging this information, the attacker can build a pseudo-model that learns to perform the same task as the original model. The adversary can then find weaknesses in the pseudo-model by running one of the attacks described in section 6. Although the original model and the pseudo-model differ, in [4] Papernot shows that an adversary's attack against P may also fool M with high probability. Interestingly, in many cases, the pseudo-model's architecture doesn't need to exactly replicate the model's architecture in order to attack it. In [5] Papernot et. al show that adversarial attacks can transfer between machine learning models - often even between models that operate completely differently from each other, such as decision trees and neural networks. This implies that an attacker can get away with roughly approximating the desired architecture in order to successfully create transferable adversarial attacks.

In short, adversaries builds replicas of the models they would like to attack, find flaws in these replica models, and then take advantage of these flaws in the original model.

Let's ground this discussion in a real-world scenario: imagine we operate a fictional self-driving car company called DriveYourself.AI. While developing our self-driving car, we collected thousands of hours of video data and spent millions of investor dollars developing the machine learning-based vision system. Like any other self-driving car startup, we use this exact same neural network – our best performing model – in each car that comes off the manufacturing line.

We happen to sell our car to a malicious individual who wants to attack this system (for example, the malicious individual could seek to crash a car in order to harm the passengers). The attacker proceeds to collect information about the neural network. They uses this information to build a pseudo-model, a process we will discuss in more detail in section 5. By leveraging the methods discussed in section 6, the attacker can then easily generate attacks against their pseudo-model, which will also fool the true vision system. However, this attack is not just limited to the car that the attacker has access to. Considering we use the same neural network as our vision system across the board, this attack would be able to affect all of our cars.

5 Building a Pseudo Model

As mentioned in the previous section, to successfully attack a model, an adversary must have enough information about the model to create a rough copy of the target. We refer to this as a pseudo-model. With a good pseudo-model, an adversary can generate adversarial examples that will be likely to fool the original target as well, even if the adversary's copy isn't entirely identical to the original model.

The simplest form of attack falls in the second largest category of Figure 4, when an adversary has direct access to the design of our model, the training data, and the parameters . For our self-driving car company, this would mean our adversary found this information on breaking into the vision system. With this information, creating a pseudo-model is straightforward - just assemble the model.

Let's say DriveYourself.AI acknowledged this kind of attack could happen, so we've encrypted the network details and hidden the training data, such that our system acts like a black box. The model is still operational: it will stop the car when it sees a stop sign, it will yield at a yield sign, etc. However, the adversary no longer has access to the model structure or training data, and thus copying the model exactly or using the training data for the pseudo-model are impossible.

If an adversary has expertise about the problem at hand, they can make sure their pseudo-model structure has roughly the same level of complexity as the target model, improving the chances attacks will transfer to the target. This is especially relevant for a car vision system - the set of architectures that are useful for computer vision have many overlapping structural details that make finding a transferable attack fairly easy.

Our engineers seemed to have covered all possible attack vectors, but could an adversary use the model's predictive ability to their advantage? In [4], Papernot et. al demonstrate how an adversary could use a model to construct a dataset by using it to classify or regress hundreds of inputs, which could then be used to train a pseudo-model. The attacker uses the model as an "oracle" to gain a glimpse into how it works. While comparable to obtaining the training set directly, the oracle method allows for greater freedom, as the adversary can present the oracle with any input they desire.

Now consider a final model, where the adversary only has access to a set of samples that have been output by a model - the lowest amount of information available according to the taxonomy of 4. Here, the only piece of information the adversary has access to are inputs and outputs predefined by the model architects. Notice that this level of security is quite rare (for example, a self-driving car could never reach this level of security, since it needs to process new images for safe autonomous driving). While the target model is more secure than before, if the set of input and output pairs that are available to the adversary is sufficiently large, it can be used effectively to train a pseudo-model.

6 Attacks Against a Model

Once an adversary has built a pseudo-model, they must remember that our final goal is to generalize what we know about pseudo-model so we can transfer these weaknesses back onto the original model and use them to our advantage. Below we will describe methods for finding weaknesses in the pseudo-model.

Fast Gradient Sign Method

For the more mathematically inclined reader, this equation formalizes the attack: $x_{adv} = x + \epsilon\nabla_x J(\theta, x, y)$

An adversary determines the direction of greatest input perturbation by calculating the Jacobian of the loss function J with respect to the model parameters θ, the input x and the true label y ($\nabla_x J(\theta, x, y)$). The attacker normalizes the amount of perturbation by applying the sign function, then adds this normalized vector scaled by ϵ to the input example x, creating the adversarial example x_{adv}. It should be noted that this technique only works on models that are differentiable, such as logistic regression and neural networks. [1]

The value of ϵ is chosen to be small enough such that the perturbation doesn't alter the input by any human visible amount, but the perturbed input is still able to fool the model.

An adversary could extend this attack into a source/target misclassification.

$$x_{adv} = x - \epsilon\nabla_x J(\theta, x, y_{target})$$

Instead of ascending the gradient of the loss function, raising the loss value, and thus reducing confidence in the model, the adversary descends the gradient with respect to a target y_{target} in place of the true label y.

For both versions, if a single perturbation is not sufficient to cause misclassification, an adversary could repeat the process, replacing x in the gradient calculation with the updated x_{adv} after each iteration [6].

While other attacks exist, they are chiefly variations on FGSM. For example, the Jacobian-based Saliency Map Attack (JSMA) updates one feature at a time, rather than ascending a gradient in feature space. JSMA provides finer-grain control by updating features one by one, and can thus generate adversarial examples with less perturbation. However, its added precision comes at an increased computational cost [3].

7 Defenses Against Adversarial Attacks

Training Phase Defenses

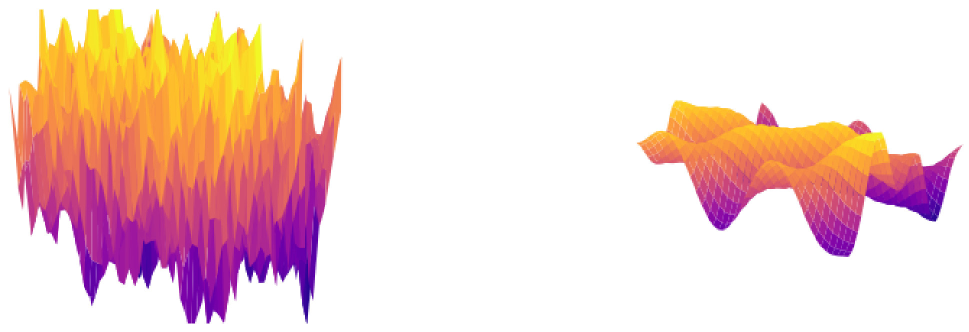
The most straight-forward defense to adversarial attacks is to build a stronger model during training - either by altering the training dataset or the training regimen.

Adversarial Training

One approach to reduce the effectiveness of adversarial attacks is to augment the training dataset with adversarial examples. By learning to classify particularly challenging examples, models can develop a better understanding of the true data distribution. *Adversarial training* protects models against known adversarial example generation attacks by training on the adversarial examples themselves. Defenders craft adversarial examples, add them to the training dataset, and then finetune the model with the newly augmented dataset.

We can better understand this defense by introducing error surfaces. Error surfaces describe the relationship between the domain of possible inputs to a model and the value of a model's cost function. A steep gradient on the error surface indicates that a small perturbation in the input will drastically increase the value of the model's cost function. A model trained without exposure to adversarial examples might have many points like these; its error surface would be jagged, resembling knives sticking out of the ground. In this case, an adversary could easily find adversarial examples using an attack such as FGSM.

Adversarial training smooths out the imperfections in the error surface. The adversary then has a more difficult time finding a perturbation that is both small enough to remain



(a) Error surface before adversarial training (b) Error surface after adversarial training

Figure 5: Model error surface visualization before and after adversarial training. The xy-axes are two features of the input and the z axis is the error surface, also known as the loss surface. The problem is binary classification - a model with high error is considered part of class A and a model with low error is considered part of class B. Left: Error surface of a trained target model. The surface is very jagged, with large gradients at many points, facilitating gradient-based attacks. Right: Error surface of a target model trained adversarially. The surface is much smoother, reducing the effect of small perturbations on model performance.

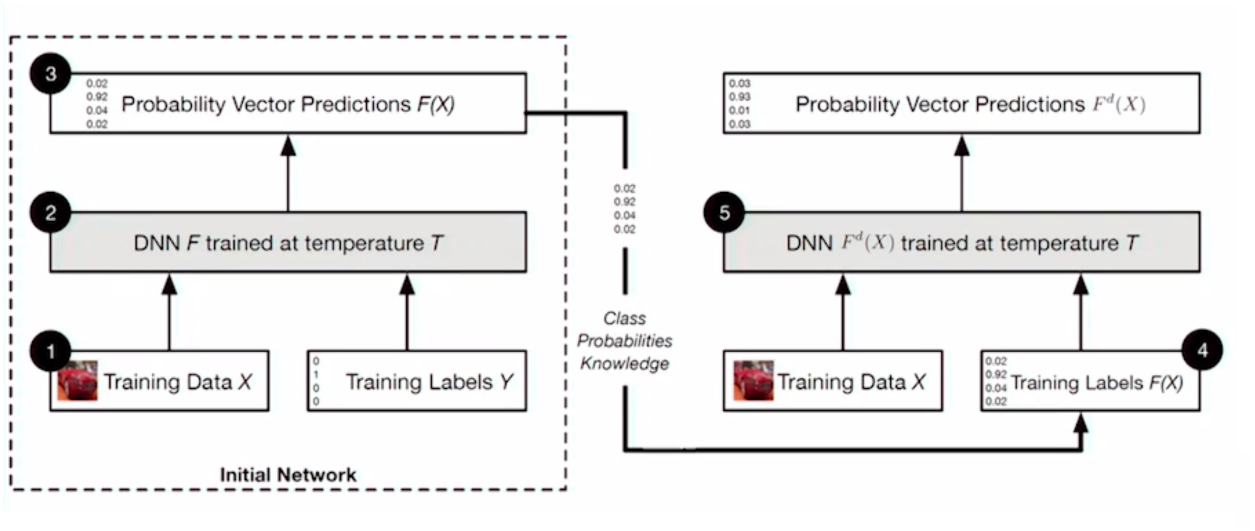
imperceivable by humans, yet significant enough to properly attack the classifier. A simple example with two feature dimensions is displayed in 5.

You can imagine that if a defender used all possible adversarial examples when training the model, the model would be entirely robust. However, the space of all possible examples is very large, so defenders can only hope that this brute force defense samples from the space sufficiently well. Using a very fast technique like FGSM still ensures that the defender can efficiently generate a sizable dataset and use it to smooth out much of the model's error surface. While adversarial training prevents exploitation of extreme gradients, adversaries can still find adversarial examples by leveraging subtle gradients.

Defensive Distillation

Another defense method proposed in [7] is called defensive distillation. Hinton et al [8] as a method to transfer knowledge from complicated neural networks (or ensembles of neural networks) to simple neural networks. However, distillation can also be used as an adversarial defense.

Figure 6: Defensive distillation uses two models. The first network produces very smooth probability outputs, which has good defensive properties. The second network uses those probabilities to generate very discrete probabilities, which are used for confident class predictions.


3
0.02
0.92
0.04
0.02
Probability Vector Predictions F(X)
2
DNN F trained at temperature T
1
Training Data X
0
1
0
0
Training Labels Y
Initial Network
0.02
0.92
0.04
0.02
Class Probabilities Knowledge
0.03
0.93
0.01
0.03
Probability Vector Predictions $F^d(X)$
5
DNN $F^d(X)$ trained at temperature T
Training Data X
0.02
0.92
0.04
0.02
Training Labels F(X)
4

During defensive distillation, the defender divides training into two parts: training a standard model that outputs smooth probabilities and training a second model that uses these smooth probabilities to predict labels.

Normally, a model goes directly from inputs to very discrete probabilities that are used for confident predictions. For example, instead of saying the output is 54% likely to be a dog and 46% likely to be a cat(highly uniform probability distribution), the model would be predisposed to say the output is 98% likely to be a dog and 2% likely to be a cat(highly nonuniform probabilities). A nonuniform probability output allows for confident predictions, but a uniform probability output is more useful for defensive purposes as it is less biased towards choosing one class over another.

In essence, the goal of defensive distillation is to incorporate the advantages of both the defensive properties of uniform probability outputs and the confidence benefits of nonuniform probability outputs. For a formal description of how defensive distillation uses this two-stage process, we refer readers to [7].

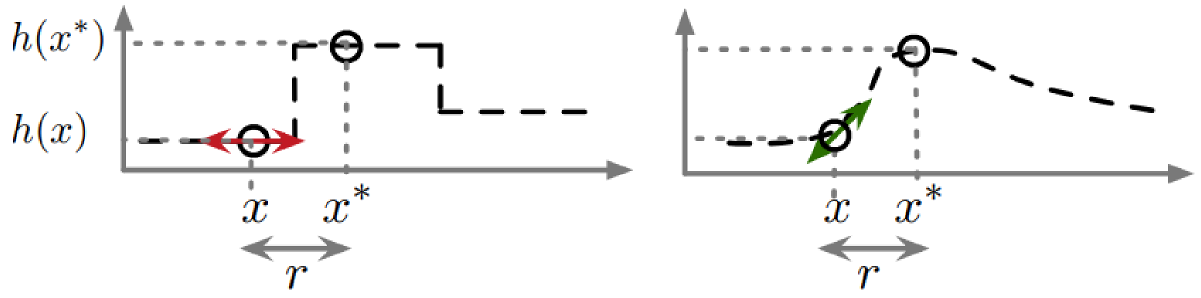


Figure 7: Diagram of gradient masking from [9] In (a) the target model has a continuous surface with a non-zero gradient. An adversary can leverage this for an attack by perturbing x up the gradient to adversarial $x*$.(b) The same model but with gradient masking. An adversary could no longer use the gradient to improve adversarial example $x*$.

Gradient Blocking Defenses

A promising defense is to directly hide the feedback signals that adversaries rely on to construct adversarial attacks. As explained in Section 2 of this chapter, attackers find ways to trick models by perturbing the input a little and observing how that changes the model's confidence in its classification of the input. This perturbation is typically performed in the direction of the gradient. As such, by hiding information about the gradient from the attacker, we effectively prevent these attacks from happening. This process is known as *gradient masking.*

A simple form of gradient masking involves hiding the probability distribution of a classification model, and only returning the most probable output. A neural network tasked to differentiate between dogs and cats might return 70% dog and 30% cat for a certain image. However, the attacker will only be able to see the highest probably label: ̈dog. ̈ This ̈maskingb̈ars the attacker from discovering the direction of optimal input perturbation, and thus makes it more difficult to use gradient based attacks.

This seems great! Why do we need other defenses if this simple method is available? Despite rendering attacks difficult to carry out, gradient masking doesn't eliminate adversarial samples – it simply hides the model's inherent weaknesses. Concealing the gradient may make it difficult for the adversary to create adversarial samples, but if discovered, they will still fool the network. Even when treating the model as a black box, an adversary could still find these adversarial samples.

It turns out the promising defenses we've mentioned above owe their efficacy to unintentional gradient masking. Adversarial training and defensive distillation both result in smoother loss surfaces near training examples, especially where an attacker might expect to receive gradient signals from the target model.

Unfortunately, this dependence implies that adversarial training and defensive distillation can be attacked using the same methods that successfully penetrate gradient masking defenses. discussed in Carlini et al. [10] include using properly chosen loss functions, calculating gradients from the pre-softmax layer, and transferring adversarial samples from a pseudo-model. While these are all white box attacks, Carlini suggests that these attacks are robust enough to work in black box settings by leveraging adversarial sample transferability.

Adversarial Sample Detection Defenses

Adversarial sample detection defenses rely on actively distinguishing between normal and adversarial samples during *inference* (the stage during which a trained model is actively classifying new inputs). Adversarial sample detection defenses can be thought of as spam filters, tuned to detect adversarial examples. Either the model itself, or a separate model altogether, can seek to identify and filter out harmful inputs before they are processed.

Detectors

To recognize harmful inputs, we can use a *detector* that identifies harmful inputs. While there have been numerous proposals for detector schemes, we focus on Xu et al.'s *feature squeezing* [11].

To separate adversarial samples from normal samples, feature squeezing leverages model robustness. A model is robust if it is able to properly classify an example even after slight transformation. For example, an image recognition model should be able to recognize an image even after rotation or scaling. The detector uses this information to compare a model's prediction before and after a transformation of the input. The idea is that if an adversarially crafted example is misclassified as a cat, it will successfully classify the image as a cat even after the adversarial example is transformed. In the *feature squeezing* strategy, the transformations reduce the color bit depth of each pixel and smooth the image using a spatial filter. Mathematically, reducing the color bit depth of pixels in an image means reducing the granularity of the bits that represent the intensity of each color channel (usually RGB for images). While 8 bits are usually used to store intensities between [0,1] for each color channel, feature squeezing suggests reducing the bit depth by up to 1 bit per channel. Intuitively, this transformation results in an image that may accentuate slight perturbations to color channels on the input. The other feature squeezing transformation is a spatial smoothing filter, colloquially known as blur. For each pixel in the image, the filter uses a window to average out intensities of neighboring pixels. This transformation seeks to "undo" sparse perturbations on individual pixels.

The motivation behind those two feature squeezing transformations is to "undo" any adversarial pixel-level perturbations imperceptible to humans.

However, as shown in He et al. [12], if an attacker knows what kind of detector is being used, the attacker can often times include tricking the detector as part of the adversarial sample generation process! This type of attack that actively circumvents defensive detectors is called an active attack. Active attackers are able to generate robust adversarial samples that consistently fool target networks even after many transformations (even printouts of these adversarial image samples have been known to fool target models).

Reformers

Recent work [13] by Meng et al. introduces a reformer as an adversarial defense. A reformer takes in an example, finds the closest matching example in the training dataset, then adjusts the input by pushing the it closer to this other example. A special neural network called an autoencoder has been particularly successful at performing this function. An autoencoder attempts to learn the mapping from an input to an input, basically replacing the label with the example again.. A naively architected autoencoder would learn the identity function. By creating a bottleneck in the architecture, the model learns the optimal way to compress the information into a simplified representation. The simplified representation then makes it easier for the reformer to "fix" an adversarial example as there are less features and thus less possible failure points. If you're interested in learning more about autoencoders, please read [14].

The reformer autoencoder is trained with non-adversarial examples in the hopes of learning a surface that represents the key aspects of normal samples. Then, when the reformer is presented with an adversarial example x, the reformer tries to find a normal example y on the learned surface that is close to x. The reformer tries to "undo" the adversarial perturbations the attacker uses to transform the original input. Then, the reformed normal example y will be classified by the target model as the true label for x.

However, reformers are still vulnerable to active attackers, just as detectors are. With the knowledge that a defender uses a reformer, the attacker could attempt to target the reformer directly, generating adversarial examples that both fool the reformer and the target model.

8 Problem Domains

We have introduced adversarial machine learning attacks, explained how they are carried out, and how to defend against them. While academic research of adversarial attacks has focused on attacking image classification models, we foresee rapid expansion of adversarial methods in all domains where machine learning models can be useful. Below, we sample several domains that we believe are especially vulnerable to adversarial attacks in the future.

- **Face Recognition**: Consider a face recognition system. An adversary could impersonate anybody by launching a targeted misclassification threat. The attacker would simply need to generate an adversarial image that confuses the model into believing with high confidence that the image is someone with access to the information the system secures. Facial recognition systems that rely solely on machine learning based vision systems are not robust to adversarial attacks and most attack plans described in section 8 could penetrate such a system. Consider the case in which a facial recognition system is used to authenticate a login to an ATM or a bank. Successful adversarial attacks could have significant ramifications, allowing for instant access to any account. On the other hand, an adversary could leverage only source misclassification to keep account holders from accessing their money by requesting a ransom.

- **Anti-virus**: Consider an anti-virus system. An attacker could leverage a source/ target misclassification attack. An adversary could build a virus, where its featurization makes the virus look like an approved program to the anti-virus system [15]. A hacker could create a Trojan with a particular placement of comments, or another method of code obfuscation, that masks the virus' nefarious nature. While this scenario is on the radar of modern security research, attacks evolve very frequently and anti-virus developers must know how to defend their machine learning models against adversarial attack.

- **Google**: Google's search algorithm is a prime candidate for adversarial targeting. One adversarial attack that could be leveraged against this type of system is a source/target misclassification threat, wherein an adversary could change the ranking and reputation of any particular web page by intelligently perturbing its contents and history. Consider the case in which a website owner seeks to boost his website's reputation, in order to increase page views, and thus ad revenue. Such an individual could learn the optimal "perturbations" to apply to their website, such that the website itself becomes an adversarial example, and thus appears frequently in top search results. This is already being utilized by organizations in the form of data-driven advertising and marketing.

  Additionally, Google's crucial role in society as a knowledge engine makes adversarial attacks against it all the more alarming due to the immediate reach of such an attack. With adversarial attacks, dictators may be able to covertly censor traffic by subtly performing source misclassification, making unapproved content inaccessible by search. Radical news outlets could redirect legitimate search queries for facts to biased or fabricated political pieces. In a world already suffering from man-made censorship of information, the possibility of using AI to optimally "guide" the dogma and understanding of the masses is a very frightening prospect.

- **Drones**: Finally, consider you own an autonomous drone delivery company. Your team has created an amazing set of self-driving drones that zoom around San Francisco delivering food to a bustling city. A combination of different sensory inputs (e.g. computer vision, LIDAR, gps, etc.) guides the autonomous robots. An adversary would only have to steal one drone to have access to the model itself, (either directly or as an oracle) and, as discussed in earlier sections, it's not difficult to then build an effective pseudo-model. Next, the adversary could launch a slew of targeted attacks on your fleet of drones as they zip around the city. This attack could cost thousands in damages, engineering man hours to refactor the model, but more importantly, it could jeopardize the physical safety of city dwellers.

  The other examples posed their own unique challenges and use cases; however, this attack differs in that it deals with instantiations in the real world. Adversaries in other examples had to deal with remote, inaccessible software hidden behind layers of cloud infrastructure, whereas the adversary in this scenario simply had to walk out into the street. As AI becomes more powerful, and we start to see real-world instantiations in the form of automata and embedded AI systems, we will also simultaneously start to see adversarial AI become a more prevalent.

9 Trends, Insights, and Recent Developments

Some skeptics of adversarial examples claim that attacks on machine learning models are often unfeasible or impractical. However, security should not rely entirely on the assumption that the attacker would not devote enough effort to break the system.

Barring any attack on the model itself, an adversary could focus on attacking the data used to train a model – a process known as poisoning. In a poisoning attack, the adversary exerts control over the learning process by inserting carefully crafted samples into the training dataset. These samples are designed to skew the data distribution and degrade the performance of the model. Poisoning attacks are effective against models that are frequently re-trained with real-world data. Given the data gathered to train one model will be shared with all of its counterparts, it is relatively straightforward to poison hundreds, if not thousands, of products such as cars and facial recognition security cameras by infecting the data collected by a single product.

As an example, malicious email campaigns are constantly evolving to circumvent spam filters. Anti-spam systems needs to collect data from recent spam campaigns to stay up-to-date. Manually labeling this data is infeasible, so the system relies on automated training data collection. For example, it might store a list of email addresses known to regularly send spam and automatically mark emails sent from this blacklisted address as "spam" in the training data. The adversary in control of these addresses could send out perfectly legitimate emails, and the system would learn to identify them as "spam-like". If the automatic labeling system mislabels a significant number of these emails, the trained spam detection model will learn the wrong class distributions, and begin labeling legitimate emails as spam.

One potential defense against poisoning attacks is to thoroughly sanitize the training data. Practically, this amounts to detecting and removing suspicious points and outliers, and only using data from trusted data repositories. Although large volumes of data are often necessary to produce good machine learning models, untrustworthy or unverified data repositories act as new attack vectors in the learning process. Another sanitation method is to use a "reject on negative impact" analysis on new data. In this setup, the system compares model performance before and after training on the suspect samples. If the new data significantly decreases test accuracy on legitimate examples, the system discards the new samples.

As the field progresses, it seems as if adversaries shatter our defenses as soon as we construct them. However, there are several promising research directions. The first is model ensembling: combining information from multiple “base” models to make a single classification. Ensembling tends to highlight the strengths of base models when they perform well, and mask the weaknesses of base models when then perform poorly. One variant of this approach [16] seems to lessen the susceptibility of adversarially trained models to white box attacks. By consulting multiple sources before making a decision, this method smooths the error surface, rendering the model robust to adversarial attacks.

We may also see improved robustness to adversarial examples as large research groups (such as Google and Facebook) train models on gigantic datasets. By encompassing a greater range of input possibilities, these models train on large datasets may learn smoother error surfaces, and thus greater robustness to adversarial attacks.

On the other hand, we might also see advancements in adversarial robustness in low data scenarios. Most attacks and defenses seem to hinge either on on having certain levels of a priori information about the model and the problem domain, or adapting on the fly so as to mitigate the effects of their their lack of a priori knowledge about the system. Recently, new learning paradigms are emerging under the umbrella of meta learning [17], focusing on learning to learn. One-shot [18] meta learning systems are becoming increasingly popular as they promise fast training of powerful user-specific models without sacrificing generalization. We see this low-data learning algorithms as a promising new class of machine learning models to defend/attack.

10 Conclusion

Adversarial attacks and defenses constantly evolve alongside artificial intelligence at large. Currently, it seems that most attacks are very difficult to defend against. However, some defenses do show promise against scenarios where the attacker doesn’t have access to as much information about the model. While new and ingenious defensive strategies are developed often, adversarial attacks seem to permeate even the most promising lines of defense. When considering this arms race in adversarial learning, it seems as though the adversary currently has the upper hand. In order to be truly robust against any attack, a model would need to learn and retain all the possible knowledge/information about a learning problem, whether that involves something proactive (and impractical) such as training on all the possible training samples, an infinite space, or something catch and sanitize any and all adversarial attacks.

In spite of these limitations, advances in the field of meta-learning will allow researchers to theorize and implement more intelligent methods for defenders to prevent attacks and for adversaries to perpetuate attacks - leveling the playing field and moving it away from the data-driven paradigm. There is promise that this arms race will eventually reach an equilibrium, much like the equilibrium that the traditional computing security is at now.

One thing is for sure: before relying on and integrating any model into daily life, more research needs to be conducted to ensure the integrity of these models. Developing defenses will be critical as unsafe applications could come at the cost of human life.

Bibliography

[1] I. J. Goodfellow, J. Shlens, & C. Szegedy (2014). 'Explaining and harnessing adversarial examples.' *arXiv preprint arXiv:1412.6572* .

[2] D. of Statistics (2016). 'Nsc motor vehicle fatality estimates.' .

[3] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, & A. Swami (2016). 'The limitations of deep learning in adversarial settings.' In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pp. 372–387. IEEE.

[4] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, & A. Swami (2017). '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.

[5] N. Papernot, P. McDaniel, & I. Goodfellow (2016). 'Transferability in machine learning: from phenomena to black-box attacks using adversarial samples.' *arXiv preprint arXiv:1605.07277* .

[6] A. Kurakin, I. Goodfellow, & S. Bengio (2016). 'Adversarial machine learning at scale.' *arXiv preprint arXiv:1611.01236* .

[7] N. Papernot, P. McDaniel, X. Wu, S. Jha, & A. Swami (2015). 'Distillation as a defense to adversarial perturbations against deep neural networks.' In *Proceedings of the 37th IEEE Symposium on Security and Privacy.*

[8] G. Hinton, O. Vinyals, & J. Dean (2015). 'Distilling the knowledge in a neural network.' *arXiv preprint arXiv:1503.02531* .

[9] N. Papernot, P. McDaniel, A. Sinha, & M. Wellman (????). 'Towards the Science of Security and Privacy in Machine Learning.' *ArXiv e-prints* .

[10] N. Carlini & D. A. Wagner (2016). 'Defensive distillation is not robust to adversarial examples.' *CoRR* **abs/1607.04311**. URL `http://arxiv.org/abs/1607.04311`.

[11] W. Xu, D. Evans, & Y. Qi (2017). 'Feature squeezing: Detecting adversarial examples in deep neural networks.' *arXiv preprint arXiv:1704.01155* .

[12] W. He, J. Wei, X. Chen, N. Carlini, & D. Song (2017). 'Adversarial example defenses: Ensembles of weak defenses are not strong.' *arXiv preprint arXiv:1706.04701* .

[13] D. Meng & H. Chen (2017). 'Magnet: a two-pronged defense against adversarial examples.' *arXiv preprint arXiv:1705.09064* .

[14] P. Baldi (2012). 'Autoencoders, unsupervised learning, and deep architectures.' In *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, pp. 37–49.

[15] K. Grosse, N. Papernot, P. Manoharan, M. Backes, & P. McDaniel (2016). 'Adversarial perturbations against deep neural networks for malware classification.' *arXiv preprint arXiv:1606.04435* .

[16] F. Tramèr, A. Kurakin, N. Papernot, D. Boneh, & P. McDaniel (2017). 'Ensemble adversarial training: Attacks and defenses.' *arXiv preprint arXiv:1705.07204* .

[17] C. Finn, P. Abbeel, & S. Levine (2017). 'Model-agnostic meta-learning for fast adaptation of deep networks.' *arXiv preprint arXiv:1703.03400* .

[18] L. Fe-Fei et al. (2003). 'A bayesian approach to unsupervised one-shot learning of object categories.' In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pp. 1134–1141. IEEE.