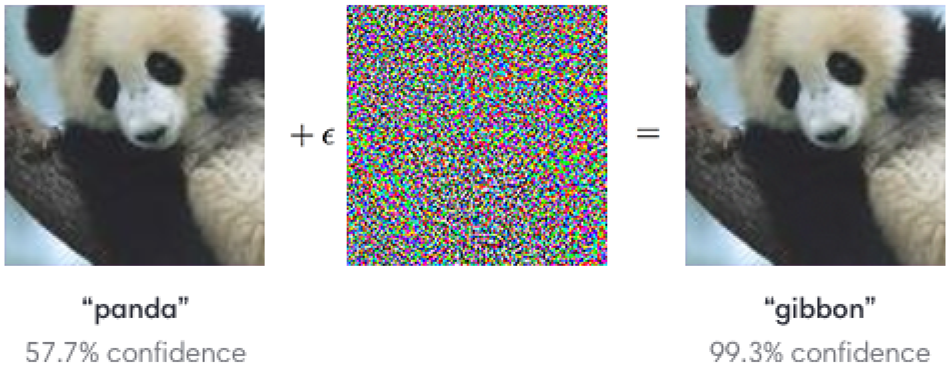
Attacking Machine Learning with Adversarial Examples

Adversarial examples are inputs to machine learning models that an attacker has intentionally designed to cause the model to make a mistake; they're like optical illusions for machines. In this post we'll show how adversarial examples work across different mediums, and will discuss why securing systems against them can be difficult.

February 24, 2017
7 minute read

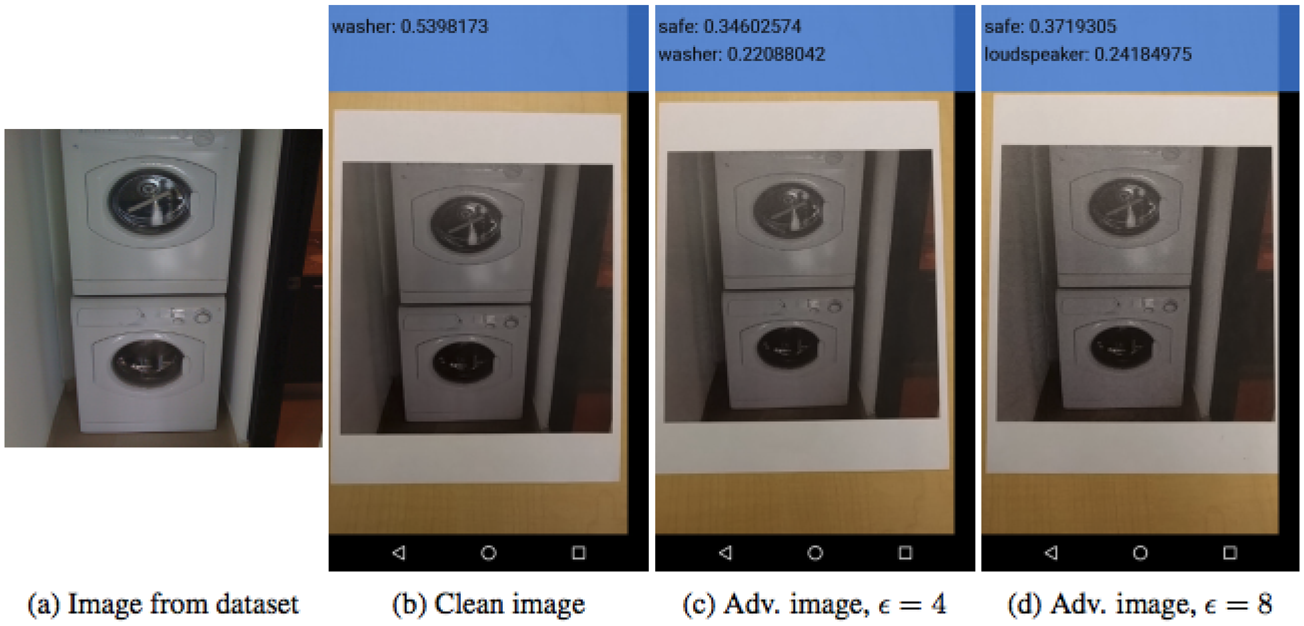
At OpenAI, we think adversarial examples are a good aspect of security to work on because they represent a concrete problem in AI safety that can be addressed in the short term, and because fixing them is difficult enough that it requires a serious research effort. (Though we'll need to explore many aspects of machine learning security to achieve our goal of building safe, widely distributed AI.)

To get an idea of what adversarial examples look like, consider this demonstration from *Explaining and Harnessing Adversarial Examples*: starting with an image of a panda, the attacker adds a small perturbation that has been calculated to make the image be recognized as a gibbon with high confidence.



An adversarial input, overlaid on a typical image, can cause a classifier to miscategorize a panda as a gibbon.

The approach is quite robust; recent research has shown adversarial examples can be printed out on standard paper then photographed with a standard smartphone, and still fool systems.



Adversarial examples can be printed out on normal paper and photographed with a standard resolution smartphone and still cause a classifier to, in this case, label a "washer" as a "safe".

Adversarial examples have the potential to be dangerous. For example, attackers could target autonomous vehicles by using stickers or paint to create an adversarial stop sign that the vehicle would interpret as a 'yield' or other sign, as discussed in *Practical Black-Box Attacks against Deep Learning Systems using Adversarial Examples*.

Reinforcement learning agents can also be manipulated by adversarial examples, according to new research from UC Berkeley, OpenAI, and Pennsylvania State University, *Adversarial Attacks on Neural Network Policies*, and research from the University of Nevada at Reno, *Vulnerability of Deep Reinforcement Learning to Policy Induction Attacks*. The research shows that widely-used RL algorithms, such as DQN, TRPO, and A3C, are vulnerable to adversarial inputs. These can lead to degraded performance even in the presence of pertubations too subtle to be percieved by a human, causing an agent to move a pong paddle down when it should go up, or interfering with its ability to spot enemies in Seaquest.

Adversarial Attacks: Seaquest, A3C, L2-Norm

If you want to experiment with breaking your own models, you can use cleverhans, an open source library developed jointly by Ian Goodfellow and Nicolas Papernot to test your AI's vulnerabilities to adversarial examples.

Adversarial examples give us some traction on AI safety

When we think about the study of AI safety, we usually think about some of the most difficult problems in that field — how can we ensure that sophisticated reinforcement learning agents that are significantly more intelligent than human beings behave in ways that their designers intended?

Adversarial examples show us that even simple modern algorithms, for both supervised and reinforcement learning, can already behave in surprising ways that we do not intend.

Attempted defenses against adversarial examples

Traditional techniques for making machine learning models more robust, such as weight decay and dropout, generally do not provide a practical defense against adversarial examples. So far, only two methods have provided a significant defense.

Adversarial training: This is a brute force solution where we simply generate a lot of adversarial examples and explicitly train the model not to be fooled by each of them. An open-source implementation of adversarial training is available in the cleverhans library and its use illustrated in the following tutorial.

Defensive distillation: This is a strategy where we train the model to output probabilities of different classes, rather than hard decisions about which class to output. The probabilities are supplied by an earlier model, trained on the same task using hard class labels. This creates a model whose surface is smoothed in the directions an adversary will typically try to exploit, making it difficult for them to discover adversarial input tweaks that lead to incorrect categorization. (Distillation was originally introduced in *Distilling the Knowledge in a Neural Network* as a technique for model compression, where a small model is trained to imitate a large one, in order to obtain computational savings.)

Yet even these specialized algorithms can easily be broken by giving more computational firepower to the attacker.

A failed defense: “gradient masking”

To give an example of how a simple defense can fail, let’s consider why a technique called “gradient masking” does not work.

“Gradient masking” is a term introduced in *Practical Black-Box Attacks against Deep Learning Systems using Adversarial Examples*. to describe an entire category of failed defense methods that work by trying to deny the attacker access to a useful gradient.

Most adversarial example construction techniques use the gradient of the model to make an attack. In other words, they look at a picture of an airplane, they test which direction in picture space makes the probability of the “cat” class increase, and then they give a little push (in other words, they perturb the input) in that direction. The new, modified image is mis-recognized as a cat.

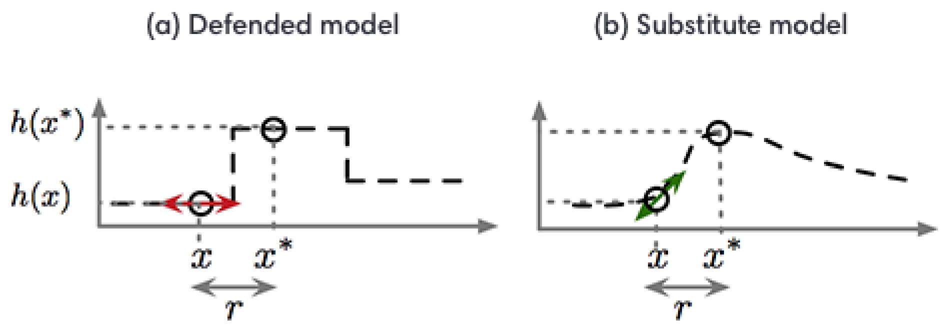
But what if there were no gradient — what if an infinitesimal modification to the image caused no change in the output of the model? This seems to provide some defense because the attacker does not know which way to “push” the image.

We can easily imagine some very trivial ways to get rid of the gradient. For example, most image classification models can be run in two modes: one mode where they output just the identity of the most likely class, and one mode where they output probabilities. If the model’s output is “99.9% airplane, 0.1% cat”, then a little tiny change to the input gives a little tiny change to the output, and the gradient tells us which changes will increase the probability of the "cat" class. If we run the model in a mode where the output is just "airplane", then a little tiny change to the input will not change the output at all, and the gradient does not tell us anything.

Let's run a thought experiment to see how well we could defend our model against adversarial examples by running it in "most likely class" mode instead of "probability mode." The attacker no longer knows where to go to find inputs that will be classified as cats, so we might have some defense. Unfortunately, every image that was classified as a cat before is still classified as a cat now. If the attacker can guess which points are adversarial examples, those points will still be misclassified. We haven't made the model more robust; we have just given the attacker fewer clues to figure out where the holes in the models defense are.

Even more unfortunately, it turns out that the attacker has a very good strategy for guessing where the holes in the defense are. The attacker can train their own model, a smooth model that has a gradient, make adversarial examples for their model, and then deploy those adversarial examples against our non-smooth model. Very often, our model will misclassify these examples too. In the end, our thought experiment reveals that hiding the gradient didn't get us anywhere.

The defense strategies that perform gradient masking typically result in a model that is very smooth in specific directions and neighborhoods of training points, which makes it harder for the adversary to find gradients indicating good candidate directions to perturb the input in a damaging way for the model. However, the adversary can train a *substitute* model: a copy that imitates the defended model by observing the labels that the defended model assigns to inputs chosen carefully by the adversary.



A procedure for performing such a model extraction attack was introduced in the black-box attacks paper. The adversary can then use the substitute model's gradients to find adversarial examples that are misclassified by the defended model as well. In the figure above, reproduced from the discussion of gradient masking found in *Towards the Science of*

Security and Privacy in Machine Learning, we illustrate this attack strategy with a one-dimensional ML problem. The gradient masking phenomenon would be exacerbated for higher dimensionality problems, but harder to depict.

We find that both adversarial training and defensive distillation accidentally perform a kind of gradient masking. Neither algorithm was explicitly designed to perform gradient masking, but gradient masking is apparently a defense that machine learning algorithms can invent relatively easily when they are trained to defend themselves and not given specific instructions about how to do so. If we transfer adversarial examples from one model to a second model that was trained with either adversarial training or defensive distillation, the attack often succeeds, even when a direct attack on the second model would fail. This suggests that both training techniques do more to flatten out the model and remove the gradient than to make sure it classifies more points correctly.

Why is it hard to defend against adversarial examples?

Adversarial examples are hard to defend against because it is difficult to construct a theoretical model of the adversarial example crafting process. Adversarial examples are solutions to an optimization problem that is non-linear and non-convex for many ML models, including neural networks. Because we don't have good theoretical tools for describing the solutions to these complicated optimization problems, it is very hard to make any kind of theoretical argument that a defense will rule out a set of adversarial examples.

Adversarial examples are also hard to defend against because they require machine learning models to produce good outputs *for every possible input*. Most of the time, machine learning models work very well but only work on a very small amount of all the many possible inputs they might encounter.

Every strategy we have tested so far fails because it is not *adaptive*: it may block one kind of attack, but it leaves another vulnerability open to an attacker who knows about the defense being used. Designing a defense that can protect against a powerful, adaptive attacker is an important research area.

Conclusion

Adversarial examples show that many modern machine learning algorithms can be broken in surprising ways. These failures of machine learning demonstrate that even simple algorithms can behave very differently from what their designers intend. We encourage machine learning researchers to get involved and design methods for preventing adversarial examples, in order to close this gap between what designers intend and how algorithms behave. If you're interested in working on adversarial examples, consider joining OpenAI.

For more information

To learn more about machine learning security, follow Ian and Nicolas's machine learning security blog cleverhans.io.

Authors
Ian Goodfellow, Nicolas Papernot, Sandy Huang, Rocky Duan, Pieter Abbeel & Jack Clark

Filed Under
Research

OpenAI

Company
API
Projects
Blog
About
Jobs

Latest
Research
Announcements
Events
Milestones

Resources
Newsroom
Brand Assets
Papers
Charter