Adversarial Attacks in Machine Learning

Sai Sasank Y
Computer Science & Engineering, PES University
Emails: sai.sasank.yadati@gmail.com

Sandeep Mysore
Computer Science & Engineering, PES University
Emails: sandeepmysore1711@gmail.com

***Abstract*—Machine Learning has paved its way into many aspects of our digital lives. Many applications we interact with have some form of Machine Learning to empower them. This ranges from online shopping, social networks to voice assistants, healthcare and finance. While it is irrefutable to say Machine Learning has boosted the capabilities of software systems, it is also worthwhile discussing the vulnerabilities and security concerns that arise in systems that leverage this technology. One such concern pertains to Adversarial Attacks. It involves fooling the models by carefully constructing examples that cause the models to make a wrong prediction. In this paper, we discuss adversarial attacks and its examples. We try and convince that the problem is worthy of our concern. We also compare and contrast various methods attempted to fight them.**

***Index Terms*—machine learning, adversarial examples, adversarial attacks, deep neural networks**

I. Adversarial Attacks

In a typical adversarial attack, an adversary constructs an example what we refer to as an adversarial example with an aim to trick the model to predict a wrong output. These attacks may have different intentions behind them can still be carried out on various models using some standard methods. We will look at different types of attacks classified based on different parameters.

It is to be noted that adversarial attacks are not vulnerabilities of any particular model, but the whole class of models that use gradient based learning. This includes simple models like logistic regression as well as complex models such as deep reinforcement learning models.

A. Black-box attacks and White-box attacks

These attacks are classified into black-box and white-box attacks [1] based on adversary's knowledge of the model in use. The attacks where the adversary has only access to the output (label or confidence scores) of the model and nothing more are referred to as black-box attacks. However, most adversarial attacks are usually white-box attacks where the adversary has the knowledge of training data, hyper-parameters and architecture of the model.

B. Targeted attacks and Non-Targeted attacks

Adversarial attacks can be targeted or non targeted [1]. Targeted attacks involve the attacker aiming for a particular output using the adversarial example. Non targeted attacks do not aim for any particular output as long as the output is wrong.

C. False Positive attacks and False Negative attacks

There are two types of attacks depending upon the targeted output [1]. Type 1 Error or a False Positive attack is where the attacker samples a negative example and tricks the model into predicting a positive label. And similarly, we have Type 2 Error or False Negative attack where the attacker targets a negative label.

D. One time attacks and Iterative attacks

Adversarial attacks are also classified based on number of runs needed to come up with a suitable adversarial example [1]. One time attacks are attacks where the adversary runs through the model once to come up with an acceptable adversarial example. Iterative attacks take multiple runs to modify and optimize the adversarial examples.

II. Adversarial Examples

In this section, we will take a look at some adversarial examples and see how the attacker could lead the model to wrongly classify the input.

A. Panda and Gibbon

Let us consider an example, [2] where we try to feed adversarial example to GoogLeNet that is trained on ImageNet. The example is an image of a Panda (see Figure 1) that's initially classified as Panda with about 57.7% confidence. The attacker adds a small calculated perturbation to this image and the output of the model predicts a Gibbon with about 99.3% confidence.

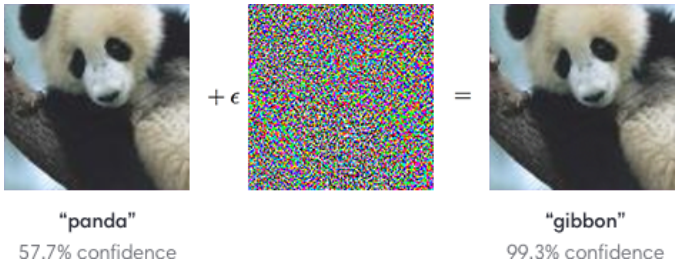

Fig. 1. An example applied to GoogLeNet that is trained on ImageNet. A small amount of noise is added to an image of a Panda to make the model wrongly classify it as a Gibbon.

This adversarial example is constructed using a fast gradient sign method. See [2] for the details of how one can calculate the perturbation needed to generate the adversarial example. This method involves calculating the gradient which can be done using backpropagation.

B. Washer and Safe

Another interesting example of a black-box attack [3] (the attack is carried out without access to the model) on a mobile application. A clean image (see Figure 2) is taken from the dataset (a) and is used to construct adversarial images with different amounts of adversarial perturbation ε . Then both the clean and adversarial examples are printed on the paper. The TensorFlow Camera Demo application was used to classify these printed images. The clean image (b) is correctly classified as a "washer" even after printing, while the adversarial images (c) and (d) are wrongly classified as a "safe".

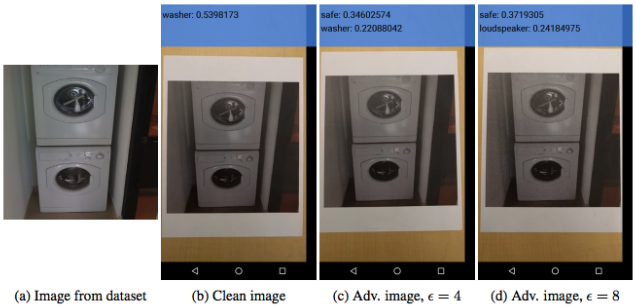

Fig. 2. *Adversarial examples printed out on paper and photographed with a smartphone and make the classifier to mislabel a "washer" as a "safe". The clean image (b) is however correctly classified even after printing.*

C. Stop and Yield

Let us look at one more example [4], where we look at images (see Figure 3) of sign boards that are potentially consumed by self driving vehicles or traffic management systems.

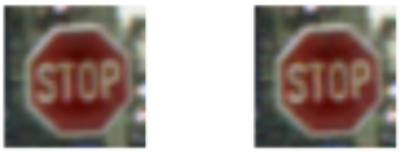

Fig. 3. Image of a sign board that says "STOP". Although indifferentiable to naked eye, the image on the right has some perturbation added to it.

To the eyes of a human, these sign images look absolutely the same. Our vision system identifies each as a stop sign. But in fact, the image on the left is an original image of the stop sign. However, the image on the right is produced by carefully adding a precisely calculated perturbation which makes a Deep Neural Network classify it as a yield sign. Any adversary can use this adversarial example to cause an autonomous vehicle that isn't resistant to such examples, to exhibit undesired, perhaps even dangerous behavior.

III. Is the problem worth studying?

It is important to consider the implications given the possibilities of carrying out adversarial attacks. If an attacker can craft adversarial examples, essentially it means that the attacker can fool your model at will. Consider the potential risks of such possibilities in systems like face recognition, object detection systems in self driving vehicles and almost any other ML systems in production. This could cause the users, owners and stakeholders of the systems a lot of damage in terms of money, time, business and reputation. Hence, we firmly believe adversarial attacks are definitely worthy of studying.

IV. Methods to generate Adversarial Examples

We will discuss three methods [3] to generate adversarial examples. We will also compare and contrast these methods below.

A. Fast method

This is perhaps the simplest and very effective method to generate adversarial examples. We first linearize the cost function and then find the solution for the perturbation by imposing the constraint of maximizing the cost subject to L_∞. ε is a hyperparameter that influences the magnitude of the perturbation. This method is called fast because it can be done in one pass of backpropagation.

$$perturbation,\ p\ =\ \varepsilon.sign(\nabla_X\ J(X,\ y_{true}))$$

It is interesting to note that this is a non targeted attack.

B. Basic Iterative method

This method is an extension of fast method. The idea is to apply the above equation, multiple times, with small step size and clipping of the image is done every iteration to keep the pixel values within the epsilon neighborhood.

The number of iterations is determined heuristically. This choice is a trade-off between optimizing the max-norm and the cost of computation to generate the perturbation and therefore the adversarial example.

$$\boldsymbol{X}_0^{adv} = \boldsymbol{X},\quad \boldsymbol{X}_{N+1}^{adv} = Clip_{X,\epsilon}\left\{\boldsymbol{X}_N^{adv} + \alpha\,\mathrm{sign}(\nabla_X J(\boldsymbol{X}_N^{adv}, y_{true}))\right\}$$

This is also a non targeted attack, because we are only increasing the cost of the correct class.

C. Iterative Least Likely Class method

The above methods we have discussed are non targeted and simply increase the cost of the correct class in an attempt to generate the perturbation. They do not specify which of the other classes the model should select. This could lead to not so interesting misclassifications in datasets with large number of classes like ImageNet.

To be able to produce interesting misclassifications we can use the iterative least likely class method. This method can be used to generate an adversarial example for a desired target class. For a particular target class, we choose the least likely

class according to the prediction of the trained network on image X.

$$y_{LL} = \arg\min_{y} \{p(y|\boldsymbol{X})\}.$$

The least likely class is usually very dissimilar from the original class. So, this method of attack can be used to generate more interesting mistakes. Putting this together, we have the following:

$$\boldsymbol{X}_0^{adv} = \boldsymbol{X}, \quad \boldsymbol{X}_{N+1}^{adv} = Clip_{X,\epsilon}\left\{\boldsymbol{X}_N^{adv} - \alpha \operatorname{sign}\left(\nabla_X J(\boldsymbol{X}_N^{adv}, y_{LL})\right)\right\}$$

The hyperparameters here are the number of iterations and the coefficient of perturbation.

D. Comparison of Methods

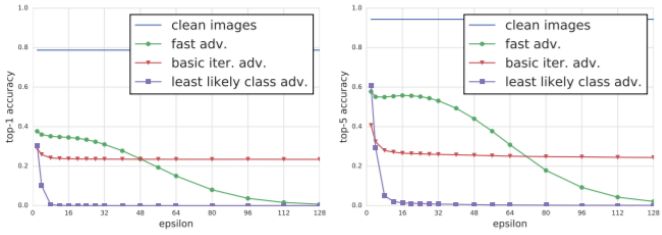

Fig. 4. Graph on left describes top-1 accuracy and graph on right describes top-5 accuracy.

The above attacks are carried out on Inception v3 and the accuracy was computed on all 50,000 images of validation set of ImageNet. For every combination of method and epsilon, the accuracy was computed on all the images. Accuracy on clean images is used as a baseline.

V. Attempted Defenses against Adversarial Attacks

A. Gradient Masking

This technique or rather set of techniques was first described in [4]. These techniques suggest constructing a model that does not permit the use of gradients that give away information. Models such as kNN classifier fall into this category. Such models make constructing adversarial examples more difficult, but are still vulnerable to adversarial examples that fool the gradient based models. It also discusses [4] the issue that if the defender tries to avoid attacks by not disclosing the gradient sensitivities of the model, these directions can still be discovered other ways, meaning the same attack can still succeed. It is showed that the black-box attack based on transfer from a substitute model overcomes gradient masking defenses. We do not know of any fully effective defense mechanism yet, and we only discuss the ones that have given us the best empirical successes below.

B. Adversarial Training

Using adversarial examples during training is shown to increase the robustness of significantly descriptive models, such as DNNs [2]. To evaluate this defense strategy using a correct implementation, they [4] train locally, using a codebase that helps generate adversarial examples at each step. For every batch, adversarial examples are generated and trained on. It is observed that for ε = 0.15, the defense can fail against the black-box attack with adversarial examples crafted on the substitute and misclassified by the model at rates up to 71.25%. However, for ε = 0.3, the black-box attack is not effective anymore.

C. Defensive Distillation

The distillation method was originally discussed in [5] as a method for compression where a small model is trained to imitate a large model in order to save computational costs. This is a strategy where we train the model to output probabilities of different classes instead of hard decisions about the output class [6] [7]. 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.

Acknowledgment

We would like to thank our guide Prof. V. R. Badri Prasad for helping us in conducting this study. We would also like to thank the Dept. of Computer Science & Engineering of PES University for this wonderful opportunity.

References

[1] X. Yuan, P. He, Q. Zhu, and X. Li*, "Adversarial Examples- Attacks and Defenses for Deep Learning."

[2] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and Harnessing Adversarial Examples."

[3] A. Kurakin, I. J. Goodfellow, and S. Bengio, "ADVERSARIAL EXAMPLES IN THE PHYSICAL WORLD."

[4] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical Black-Box Attacks against Machine Learning."

[5] G. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network."

[6] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks."

[7] I. Goodfellow, N. Papernot, S. Huang, Y. Duan, P. Abbeel, and J. Clark, "Attacking Machine Learning with Adversarial Examples."