FREE VS FAST ADVERSARIAL TRAINING

Giuseppe Capaldi
University of Rome "La Sapienza"
capaldi.1699498@studenti.uniroma1.it

Gianluca Capozzi
University of Rome "La Sapienza"
capozzi.1693255@studenti.uniroma1.it

September 24, 2020

ABSTRACT

In this work we propose a comparison between two methods for adversarial training: Free and Fast.
The code is available at `https://github.com/not-a-genius/neuralNetworkExam`.

1 Introduction

Adversarial training is a method for training robust deep neural networks, training on adversarial examples as a defense against adversarial attacks. This is typically considered to be more expensive than standard training due to the necessity of constructing adversarial examples via a first-order method like projected gradient descent (PGD). The papers presented show that it is possible to train empirically robust models using methods no more costly than standard training in practice. These methods show results comparable to train a model against PGD but at lower cost in terms of computational time. The goal of adversarial training is to learn a model which is not only accurate on the data (accuracy on train and test data) but also accurate on adversarially perturbed versions of the data (i.e. accuracy on images perturbed using the PGD attack). This leads to particular attention at the results we obtained after adversarial training experiments, aware of the presence of an acceptable compromise in accuracy, given a large increase in robustness due to the tradeoff between robustness and generalization [1, 2, 3].

1.1 Related Works

Our work is based on “Adversarial training for free!” paper [4], that presented an algorithm able to eliminate the overhead cost of generating adversarial examples by recycling the gradient information computed when updating the model parameters. While researching related work in the field we discovered a more recent paper, called "Fast is better than free: Revisiting adversarial training" [5] presenting an already known method called FGSM, previously considered ineffective due to what the paper calls “catastrophic overfitting”. This failure condition is preventable with the use of random initialization points. Moreover the same paper reported a further acceleration in training even for free algorithm thanks to standard techniques for efficient training, including cyclic learning rate and mixed-precision arithmetic. This caught our interest and we decided to try to replicate the reported results and compare them with our implementation, in order to confirm or deny these thesis.

1.2 Dataset and architecture

We decided to choose CIFAR-10 as the dataset on which our experiments have been conducted to adapt to the available hardware (our GPU is an Nvidia RTX 2070 super). The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.
Both WRN32 and ResNet50 have been tried as models to train on but their lack of accuracy even at high numbers of epochs suggested us to switch to shallower models, so we used PreActResNet18 which is based on ResNet18.

2 Adversarial Machine Learning

Adversarial machine learning is a machine learning technique that attempts to fool models by supplying deceptive input (called also adversarial example). Adversarial examples exploit the way artificial intelligence algorithms work to disrupt the behavior of artificial intelligence algorithms. In the past few years, adversarial machine learning has become an active area of research as the role of AI continues to grow in many of the applications we use. There's growing concern that vulnerabilities in machine learning systems can be exploited for malicious purposes. The main idea to dealt with adversarial examples is represented by adversarial training; it can be traced back to [6], in which models were hardened by producing adversarial examples and injecting them into training data [4].
This technique can be summarized as follows: given a network f_θ parametrized by θ, a dataset (x_i, y_i), a loss function l and a threat model Δ, the learning problem consists in the following optimization problem:

$$min_\theta \textstyle\sum_i l(f_\theta(x_i + \delta), y_i)$$

A typical choice for the adversarial model is to take $\Delta = \{\delta : ||\delta||_\infty \leq \epsilon\}$ for some $\epsilon > 0$.

3 PGD attack

The PGD attack is a white-box attack, which means that the attacker has access to the model's weights, giving to the attacker much more power than a black-box attack (in which the model's weights are not known) as he can specifically craft the attack to fool the chosen model without having to rely on transfer attacks, which often result in human-visible perturbations. The key to understanding the PGD attack is to frame finding an adversarial example as a contrained optimisation problem. PGD attempts to find the perturbation that maximises the loss of a model on a particular input while keeping the size of the perturbation smaller than a specified amount referred to as epsilon. This constraint is usually expressed as the l^2 or l^∞ norm of the perturbation and it is added so the content of the adversarial example is the same as the unperturbed sample. The PGD algorithm can be summarised with the 4 steps below although the attacker is free to apply any optimisation improvements such as momentum, Adam, multiple restarts etc...

1. Start from a random perturbation in the l_p ball around a sample;
2. Take a gradient step in the direction of the greatest loss;
3. Project perturbation back into the l_p ball if necessary;
4. Repeat steps 2-3 until convergence.

In our case, adversarial training consists simply of putting the PGD attack inside the training loop, applying a kind of "data augmentation"; in this case, instead of performing random transformations as a preprocessing step, we create specific perturbations that best fool the model and indeed adversarially trained models do exhibit less overfitting when trained on small datasets.
The PGD attack is based on the simplest version of what is called Fast Gradient Sign Method, used to approximate the inner maximization of Delta. This could be seen as a relatively inaccurate approximation of the inner maximization for l_∞ perturbations, and has the following closed form (as shown in [7]):

$$\delta^\star = \epsilon \cdot sign(\nabla_x l(f(x), y)).$$

A better approximation of the inner maximization is to take multiple, smaller FGSM steps of size alpha. When the iteration leaves the threat model, it is projected back to the set Δ. Multiple restarts within the threat model Δ typically improve the approximation of the inner maximization even further. The combination of all these techniques is the PGD attack. The problem of such technique is that the number of gradient computations here is proportional to O(MK) in a single epoch, where M is the size of the dataset and K is the number of steps taken by the PGD adversary. This is K times greater than standard training so adversarial training is typically K times slower than standard training. The pseudo-code of the PGD attack is reported in Algorithm 1.

Algorithm 1 PGD adversarial training for T epochs, given some radius ϵ, adversarial step size α and K PGD steps and a a dataset of size M for a network f_θ

```
for $t = 1...T$ do
    for $i = 1...M$ do
        // Perform PGD adversarial attack
        $\delta = 0$ // or randomly initialized
        for $j = 1...K$ do
            $\delta = \delta + \alpha \cdot sign(\nabla_\delta l(f_\delta(x_i + \delta), y_i))$
            $\delta = max(min(\delta, \epsilon), -\epsilon)$
        end for
        $\theta = \theta - \nabla_\theta l(f_\theta(x_i + \delta), y_i)$ // Update model weights with some optimizer, e.g. SGD
    end for
end for
```

The execution of K-PGD in [8] took four days on a Titan X (with model WideResNet and dataset CIFAR-10). This led us to consider the reported results obtained using this adversarial training, without running them directly.

4 Free Adversarial training for free!

Algorithm 2 "Free" Adversarial Training (Free-m)

```
Require: Training samples X, perturbation bound $\epsilon$, learning rate $\tau$, hop steps $m$
Initialize $\theta$
$\delta \leftarrow 0$
for $epoch = 1...N_{ep}/m$ do
    for $minibatch\ B \in X$ do
        for $i = 1...m$ do
            // Update $\theta$ with stochastic gradient descent
            $g_\theta \leftarrow \mathbb{E}_{(x,y)\in B}\ \nabla_\theta[l(x + \delta, y, \theta)]$
            $g_{adv} \leftarrow \nabla_x l(x + \delta, y, \theta)$
            $\theta \leftarrow \theta - \tau g_\theta$
            // Use gradients calculated for the minimization step to update $\delta$
            $\delta \leftarrow +\epsilon \cdot sign(g_{adv})$
            $clip(\delta, -\epsilon, \epsilon)$
        end for
    end for
end for
```

5 Fast is better than free

Algorithm 3 FGSM adversarial training for T epochs

```
Require: Dataset of size M, radius $\epsilon$, K PGS steps, step size $\alpha$
for $t = 1...T$ do
    for $i = 1...M$ do
        // Perform FGSM adversarial attack
        $\delta = Uniform(-\epsilon, \epsilon)$
        $\delta = \delta + \alpha \cdot sign(\nabla_\delta l(f_\theta(x_i + \delta), y_i))$
        $\delta = max(min(\delta, \epsilon), -\epsilon)$
        // Update model weights with some optimizer, e.g. SGD
        $\theta = \theta - \nabla_\theta l(f_\theta(x_i + \delta), y_i)$
    end for
end for
```

6 Conclusions

References

[1] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. ICLR, 1050:11, 2018

[2] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P Xing, Laurent El Ghaoui, and Michael I Jordan. Theoretically principled trade-off between robustness and accuracy. ICML, 2019a.

[3] Ali Shafahi, W Ronny Huang, Christoph Studer, Soheil Feizi, and Tom Goldstein. Are adversarial examples inevitable? ICLR, 2019a.

[4] Ali Shafahi, Mahyar Najibi, Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! arXiv preprint arXiv:1904.12843, 2019b.

[5] Eric Wong and Leslie Rice and J. Zico Kolter. Fast is better than free: revisiting adversarial learning, arxiv:2001.03994, 2020.

[6] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. ICLR, 2015.

[7] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572, 2014.

[8] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. ICLR, 2017.