An empirical investigation of sketching as a defense against adversarial attacks

Rufeng Ma, Alex Gittens
Rensselaer Polytechnic Institute

Introduction:

An adversarial attack is a technique to find a perturbation, imperceptible to humans, that changes the prediction of a machine learning model. See Figure 1 for an example [1].

Adversarial attacks can be generated in various ways: using the L-BFGS algorithm, the fast gradient sign method, PGD, momentum iterative attack, and distributional adversarial attack. Adversarial attacks can be targeted, or untargeted: targeted attacks attempt to change the label of an input to a specific label, while untargeted attacks simply attempt to change the label.

A state-of-the-art defense, PGD adversarial training, generates adversarial examples during training to train the model to be robust [2]. Another class of proactive defenses uses randomness to simulate the effect of adversarial attacks during training, to learn a model that is robust to small perturbations of its inputs.

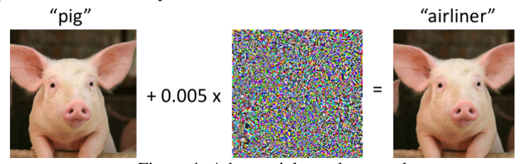

Figure 1, Adversarial attack example

Sketching is a randomized dimensionality reduction technique that is used to decrease the computational cost of algorithms that work with high-dimensional data [3]. We consider the use of sketching as a novel regularization method for deep learning by adding carefully crafted noise using perturbation layers.

Experimental setup:

We use the MNIST dataset for our training and testing, and use a fully connected feedforward network with three hidden layers. During training we insert sketching layers after the first two hidden layers. During testing we remove these perturbation layers. See Figure 3.

We parameterize the perturbation layers with parameters c1 and c2. These control the accuracy of the sketching procedure: high values mean very accurate sketching (little noise insertion). We tune c1 and c2 from none (meaning no perturbation) to 50000. We investigate the use of only one perturbation layer as well as the use of two.

We trained models in the standard manner on clean data, and also using PGD adversarial training. In training, we generate a new random sketching matrix for each perturbation layer in each minibatch. At test time, we evaluate both the clean accuracy on the test set, and the adversarial accuracy using the PGD attack algorithm to determine the accuracy of the models under adversarial attack.

An algorithm and the Network:

A basic idea for inserting robustness is to add noise to the output of layer L − 1 before sending it to layer L.

$$a_l = \omega^l(O_{l-1} + \varepsilon) + b_l$$
$$O_l = \sigma(a_l)$$

We use sketching as a method of noise injection. In general, sketching algorithms take a x and produce an output sketch vector that is close to x [4]. Here we replace the $\omega^l O_l$ with $\omega^l SS^T O_l$,where S is a gaussian random matrix.

$$a_l = \omega^l SS^T O_{l-1} + b_l$$

$$S \epsilon \mathbb{R}^{n_{l-1} \times C},$$

$$S_{ij} \sim i.i.d\ N\left(0, \frac{1}{\sqrt{c}}\right)$$

Here c is one dimension of the matrix S. The other dimension of S is n_{l-1}, the dimension of the last output of L-1. The matrix SS^T is a random approximation of the identity matrix, and as c approaches infinity, the perturbation effect goes way. See Figure 2.

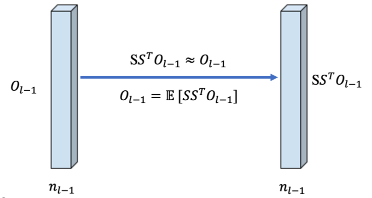
O_{l-1}
$SS^T O_{l-1} \approx O_{l-1}$
$O_{l-1} = \mathbb{E}[SS^T O_{l-1}]$
$SS^T O_{l-1}$
n_{l-1}
n_{l-1}

Figure 2, the operation computed by the random perturbation layer

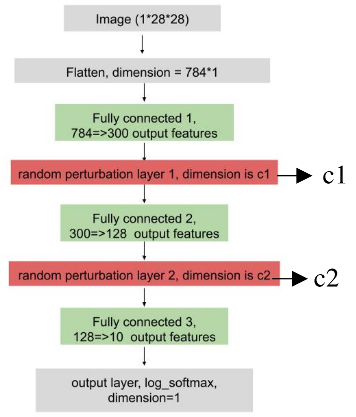
Image (1*28*28)
Flatten, dimension = 784*1
Fully connected 1, 784=>300 output features
random perturbation layer 1, dimension is c1
c1
Fully connected 2, 300=>128 output features
random perturbation layer 2, dimension is c2
c2
Fully connected 3, 128=>10 output features
output layer, log_softmax, dimension=1

Figure 3, the architecture of the network

In our experiments, we use up to two random perturbation layers.

Results:

- **What is the impact of sketching on the clean accuracy?**

Figure 4 shows the accuracy of a network with two perturbation layers as c1=c2 ranges from none to 50000. These results show that the test and train accuracy increase as c1 and c2 increase from 2 to 5000. Also, the best accuracy with sketching occurs at c1=c2=5000. Both of these trends reflect the fact that inserting more noise hurts the clean accuracy. Importantly, though we see that between c=10 and c=200 the accuracy soared from 30% to over 80%. So, sketching hurts clean accuracy in general, but sufficiently mild sketching only has a mild impact on the clean accuracy.

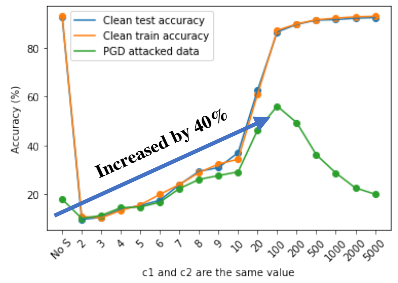

Figure 4, Clean accuracy and PGD accuracy

- **How much does sketching improve robustness?**

In Figure 4, sketching helps improve the model robustness in a specific range. We found c1=c2=100 is a sweet spot that maximizes the robustness of the model. This much sketching improvse the adversarial accuracy by 40% while only mildly impacting the clean accuracy.

- **Effects of sketching using standard training**

Figure 5 shows the adversarial accuracy as c1 and c2 vary from none to 5000 using standard training. Note that the four corners correspond to essentially no sketching. The trend is that mild sketching on the first layer (c1 not too large or small) and more aggressive sketching on the second layer (c2 > c1) gives better adversarial accuracy than standard training. We find a sweet spot when c1=100 and c2=200.

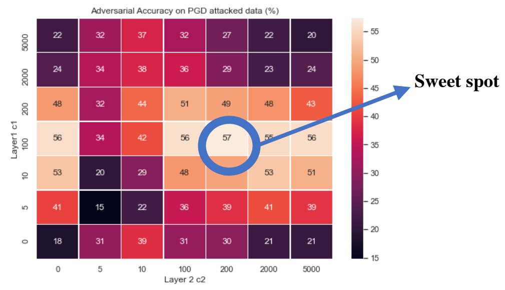

Figure 5, Heatmap of adversarial accuracy, model trained on clean MNIST dataset, 0 means no sketching.

Results:

- **Effect of sketching using adversarial training**

Figure 6 shows that sketching does not help improve adversarial accuracy when adversarial training is used. Figure 7 confirms that more adversarial accuracy is achieved when c2 > c1.

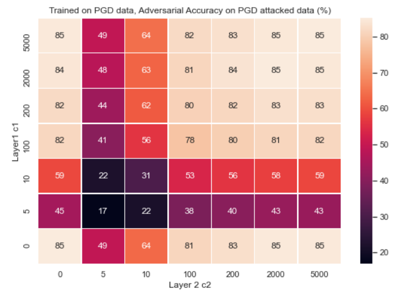

Figure 6, Heatmap of adversarial accuracy, model trained on PGD attacked MNIST dataset, 0 means no sketching.

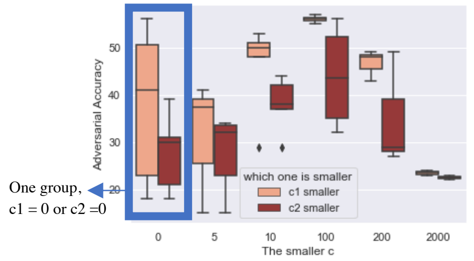

Figure 7, Boxplot showing models wit less aggressive sketching on the first layer are more accurate. 0 means no sketching is used.

Conclusion:

Training with an appropriate amount of sketching significantly improves adversarial accuracy if using standard training. To see this benefit, we find that we need to be careful in sketching the lower layers (more accurately, less noisily) compared to sketching in the higher layers (less accurately, more noisily). Sketching doesn't help when using PGD adversarial training. This is evidence that sketching helps explore the adversarial attack space, but not as well as explicitly optimizing using PGD does. Future work is to make sketching or other forms of randomized explorations of the adversarial space work as well as or synergize with PGD.

References:

[1] Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199, 2013.
[2] Adversarial Attacks and Defenses in Deep Learning. https://doi.org/10.1016/j.eng.2019.12.012
[3] Short and Deep: Sketching and Neural Networks, https://openreview.net/pdf?id=r1br_2Kge
[4] Woodruff, David P. "Sketching as a Tool for Numerical Linear Algebra." Foundations and Trends® in Theoretical Computer Science 10.1–2 (2014): 1-157.