Exploratory Analysis of Input Perturbation Attacks and their Privacy Tradeoffs

Jack Bennett Longwell
Department of Electrical and Computer Engineering
Toronto Metropolitan University
Toronto, Canada
jlongwell@torontomu.ca

***Abstract*—In an era where society is rapidly progressing towards a thoroughly digitized landscape, the ubiquity of machine learning tools has become a defining characteristic of our technological advancement. This proliferation, while beneficial, brings to the forefront an issue of incredible importance: the privacy and security of our data. In a world where data is the new currency, its privacy is more critical than ever. With machine learning integrated into the fabric of everyday life, there is an unprecedented potential for data misuse, raising risks to privacy and security. This necessitates a revitalization of classical security measures, particularly in the realm of machine learning. Our paper delves into this challenge by scrutinizing two prominent methods designed to enhance the adversarial robustness of machine learning algorithms: Projected Gradient Descent (PGD) and Fast Gradient Sign Method (FGSM). We explore the benefits of these defensive measures, as well as their privacy trade-offs, asserting that while they contribute to robustness against adversarial attacks, they also raise pivotal questions about the balance between security enhancement and data privacy preservation. These analyses are crucial in guiding the development of machine learning tools that uphold the long-standing principles of data privacy.**

***Index Terms*—differential privacy, neural networks, machine learning, security, projected gradient descent, fast gradient sign method**

I. Introduction

The integration of machine learning tools into everyday life represents a significant paradigm shift in our society. Machine learning-based products have become prominent in every sector, and these models have the potential to transform each inch of our world, including how business is done, how patients are seen, and how our government is run. These tools, however powerful, all come with various privacy and security risks. As the world moves towards an increasingly digitized world, the security and privacy of our data is more important than ever. For these reasons, researchers have increasingly looked to differential privacy to help mitigate these problems.

Differential privacy concerns the systematic assurance of privacy and security of a dataset. By using rigorous mathematical definitions and first principles, researchers have developed increasingly sophisticated techniques with the goal of stopping adversarial methods. Using these advanced techniques, we can provide mathematical privacy guarantees against a variety of attack types.

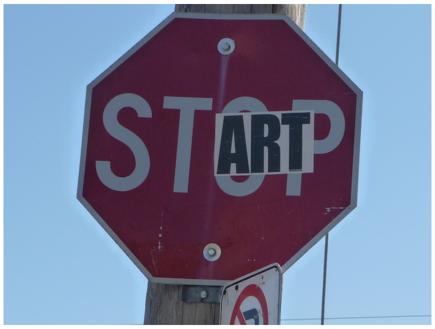

Fig. 1. Perturbed stop sign

Of the many kinds of adversarial attacks, one type is called an input perturbation attack. To give an illustrative example, say that we have a model that is specifically designed to detect stop signs. Now, suppose that the model is presented with a picture of a stop sign, but with a large sticker over the top (Fig. 1). Humans are able to discern that the sticker is simply some graffiti and is irrelevant. A machine learning model, however, may have a lot more trouble. This could become a severe issue if a model like this were to be deployed to a self-driving car. Hence, these models must be trained to be as robust as possible against a variety of adversarial attacks that have the potential to compromise algorithm function.

One of the few common traits that all machine learning models have in common is that they all must be trained on a dataset. Since many models are designed in fields such as healthcare, these datasets used for training are often supposed to be fully private, as they involve confidential patient data. We know that releasing the dataset to the public is off-limits, but sometimes even this is not enough to ensure privacy. Machine learning models that are deployed publicly via any method are still necessarily revealing information by providing an output. Membership adversarial attacks can use this information to recreate training datasets and breach privacy.

Each of these styles of attacks, input perturbation attacks and membership attacks, have been extensively researched. Novel methodologies are constantly being proposed to protect against these attacks, only for new methods of these attacks to be created to breach those proctections. Two techniques of attacking models via input perturbation attacks are called the Fast Gradient Sign Method (FGSM) [1] and Projected Gradient Descent (PGD) [2]. To make our model robust against these attacks, we simply need to train our models on perturbed data. After training our models, we will perform simple membership attacks against each of them. We explore the similarities, differences, and tradeoffs among the different techniques, with the purpose of investigating how defending against one attack can weaken an algorithm against another.

II. Methodology

A. Input Perturbation Attacks

1) FGSM is a technique that uses the gradient of the loss function to find ways to maximize the error. It is a simple one-step process that uses an input image as data to construct a perturbed adversarial example that is likely to fool a model. Specifically, we construct adversarial examples with the following formula:

$$x_{adv} = x_{inp} + \epsilon * sign(\nabla_x J(\theta, x, y)) \tag{1}$$

For all our following work, we use $\epsilon = 0.3$.

2) PGD is very similar in that it also constructs adversarial images to try and fool a model. The main difference is that PGD is an iterative process instead of a one-step process. For k iterations, we will take our input X, and add some small random perturbation to get x_0. For each iteration k, we get:

$$x^{'}_{k+1} = x_k + \alpha * sign(\nabla_x J(\theta, x, y)) \tag{2}$$

Where:

$$\alpha = \epsilon / Steps \tag{3}$$

We also perform a projection to each update to ensure that the perturbation remains within some specified norm ball. For the following work, we use $\epsilon = 0.3$ and $\alpha = 0.3/30 = 0.01$.

B. Membership Attacks

For the sake of the brevity of this project, we will be testing simple naive membership attacks. We assume the existence of some discrete classification model. We assume that we can query this model as much as we would like. Once queried, all we are given access to is the softmaxed logit of the category that the model classifies our input as. This output can be thought of as a probability of classification, or a measure of confidence. We have a set of data samples and we would like to decide whether each sample was used in the training of the model or not. Our classification model is defined as:

$$f(\theta, x_i) = \tilde{y_i} \tag{4}$$

Where a function f with parameters θ and x_i outputs softmaxed logit of the output class $\tilde{y_i}$. We sum and average the outputs as:

$$\mu = \frac{1}{n} * \sum_{i=1}^{n} \tilde{y_i} \tag{5}$$

We use this μ as a threshold to determine whether a sample was part of the training data or not. We define the classification of training set or not as:

$$g(f(\theta, x_i)) = \begin{cases} 1 & \text{if } \tilde{y} \geq \mu \\ 0 & \text{else} \end{cases} \tag{6}$$

The limitations of simple attacks such as this are commonly understood, its use is for exploratory purposes more than overall accuracy.

C. Training

To train models to be adversarially robust against attacks that use FGSM and PGD, all we have to do is include perturbed images in our training. That will allow our model to generalize to both normal images and perturbed ones. We will be training 21 Convolutional Neural Network (CNN) classification models on the MNIST dataset [3]. The dataset is one of the most common for evaluating computer vision models. It consists of 60000 training images along with 10000 testing images. The images consist of hand-drawn Arabic digits from zero to ten. These 21 models will be split between 3 different techniques: normal, FGSM and PGD. We use the packages from CleverHans (v4.0.0) [4] for computation. Each of the seven models in the three classes will be trained using increasing sizes of the dataset, starting at 160 and increasing by factors of two until it grows to 10240, which will be set to 10000 instead due to testing set size limitations. Each model will be trained for 20 epochs with a learning rate of 1e-3. For the full architecture details, please see the accompanying code. We record accuracy on the held-out test set of size equal to the training set. The accuracy of normal images, FGSM adversarial images and PGD adversarial images are recorded.

Once a model has been trained, it will be subjected to the membership attack as defined above. The accuracy of our method in discerning the train samples from the test samples is recorded for each of the models. We also compare distributions between true training samples and true testing samples and calculate the differences in loss.

III. Results and Analysis

A. Input Perturbation Attacks

The results of our classifiers are reported (Fig. 2, top left). Our normal classifier performs very strongly on normal data. Its accuracy quickly climbs to the mid-nineties within the first couple of increases in dataset size. The accuracy of our normal models on perturbed data, however, is very poor. It only once achieves an accuracy greater than ten percent, an astonishing 85% decrease in performance. This displays the strength of our input perturbation methods, we can render an otherwise incredibly strong classifier to functionally useless levels of accuracy. We can also compare this to the performance of our models trained on perturbed data. Using FGSM training (Fig. 2, top right), we are easily able to get our model to be just as robust towards adversarial

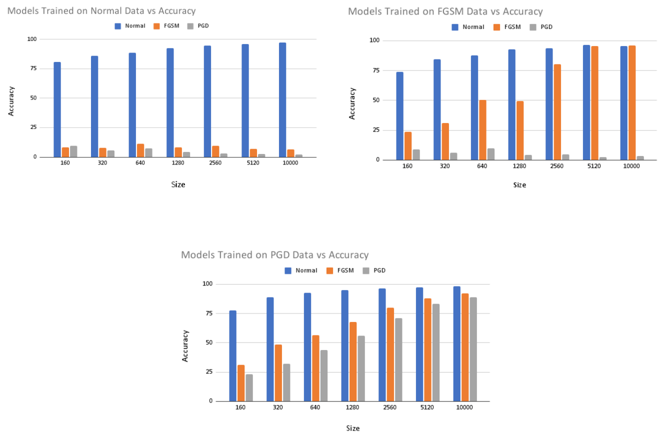

Fig. 2. Accuracy of each model

images perturbed with FGSM as it is with normal images. We report relatively negligible levels of accuracy drop even in normal image classification, easily in the high nineties. This method does however seem to only be moderately effective in smaller sample sizes(160-1280). We can see that for the first few iterations, we only achieve accuracy levels around 25-50% against FGSM images. Finally, we report the accuracy of our models trained with images perturbed with PGD (Fig. 2, bottom). Our PGD models present lots of promise. They retain very high accuracy on normal data, at some iterations outperforming both other techniques. Using PGD, we also see strong increases in robustness. Not only do we slowly increase our accuracy across iterations to around 88% against PGD data, but we also increase robustness against FGSM at the same time. This is to be expected, as PGD is essentially just a more complicated version of FGSM. Still, it is reassuring to see that we can effectively kill two birds with one stone. We do not achieve the exact same level of robustness against FGSM using PGD as we do when we use FGSM, but the differences are minimal. PGD is clearly the more effective technique for overall robustness if you could only choose one.

Next we observe the plots of loss gaps (Fig. 3). The loss gap is defined here as the difference between the loss of the samples in the training set and those that are part of the testing set. We expect that models in general will have a lower loss when using it on training data, as those were the samples that have already been seen, and hence, the parameters of the model should already be trained to minimize the loss. Those in the testing should in theory have more loss or error. We see that our models that are trained on only normal data have considerably higher loss gaps between their training and testing sets than those trained on FGSM or PGD. I believe that this is counter to the general consensus that models trained to be robust often have larger loss gaps or that they somehow are more overfitted and do not generalize as well to non-training data. In our case at least, this is not what we observe.

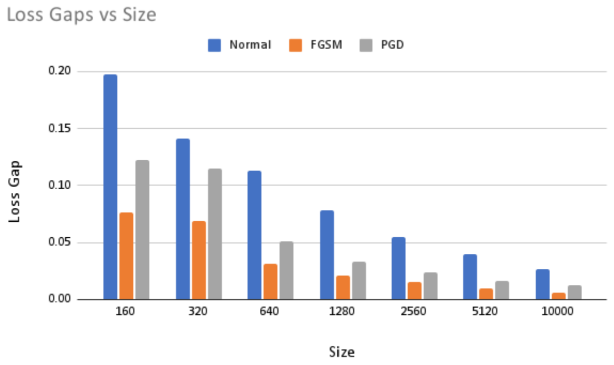

Fig. 3. Difference in Loss

B. Membership Attacks

We move on to membership attacks against our models. Firstly, we observe the distribution of chosen softmaxed logits between the training set and the test set (Fig. 4). We report the distribution with the largest loss gap(left), that being a normal model trained on 160 examples, and the model which has the smallest loss gap(right), the FGSM model with 10000 samples. We are attempting with our membership attacks to find some kind of underlying distribution or qualities that make the training set different from the testing set. From the graph on the left, we can see two distinct distributions. There is lots of overlap, but the chosen softmaxed logits for training data is generally much larger than those for the testing set samples. Using this, we are able to gain information about the training dataset that we otherwise would not be privy to. Contrast this instead with the model with the smallest loss gap. There is essentially 100% overlap between the training and testing data. This suggests that, at least by using only the softmaxed output logit, there is no difference that we can observe between the training and testing set. We would expect our membership attack to perform much poorer on the model on the right than the model on the left.

Finally, we report the results of our attack methodology as described in section II (Fig. 5). We can observe that, as our discussion above would suggest, the membership attack is strongest in the first few iterations of the dataset. This is due to the general overfitting of our model to the smaller

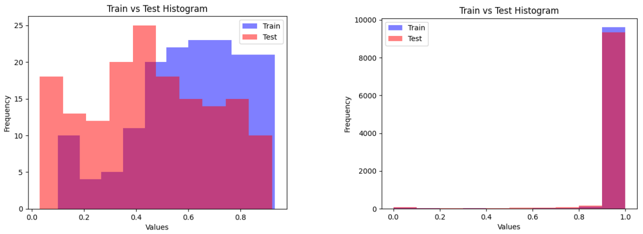

Fig. 4. Histogram and overlap of distributions

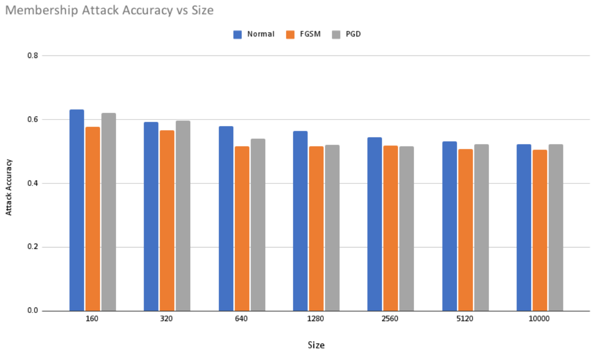

Fig. 5. Accuracy of our membership attacks

training set and their lack of generalizability to the wider distribution of unseen test data. Suprisingly, we observe that the highest accuracy is against our normal models, and the lowest average is against the FGSM models. This could be due to the particular dataset we use, or perhaps with the particular epsilon values we use. They all have a general downward trend, approaching the 0.5 mark of indifference. After using the largest training and testing set possible, 10000 samples each, we can see that the model that has the least privacy is the PGD method, although these differences are a matter of small basis points.

IV. Supplementary

Please find the tutorial I have made and all the code used to recreate these results in the ”Secure ML report full pipeline.ipynb” file attached with this report. MNIST dataset used can be accessed via TensorFlow’s datasets API. All of my results and metrics used to mark figures can be found at the exel sheet here

Explaining and Harnessing Adversarial Examples Ian J. Goodfellow, Jonathon Shlens, Christian Szegedy

References

[1] I. Goodfellow, J. Shlens, and C. Szegedy, (2014) “Explaining and Harnessing Adversarial Examples,” ArXiv. (Online article). https://arxiv.org/abs/1412.6572

[2] A. Kurakin, I. Goodfellow, and S. Bengio, (2016) “Adversarial examples in the physical world,” ArXiv. (Online article). https://arxiv.org/abs/1607.02533

[3] Y. LeCun, C. Cortes, and C. J. C. Burges, (2010) ”The MNIST database of handwritten digits,” (Online) http://yann.lecun.com/exdb/mnist/

[4] N. Papernot, F. Faghri, N. Carlini, et al. (2016) “Technical Report on the CleverHans v2.1.0 Adversarial Examples Library,” ArXiv. (Online article). https://arxiv.org/abs/1610.00768