

Evaluation of Robustness of Neural Nets

Aditya Narayan(15D070062), Pranav Sankhe(150070009)

EE769, IIT Bombay

aditya_narayan@iitb.ac.in

pranav_sankhe@iitb.ac.in

Abstract— We prove that the distillation security for neural networks is not secure for a certain attack algorithm(L2). Currently there is a trend towards the reliance on neural nets because of its promising learning capabilities. We want to take a step back and provide a word of caution to not rely on neural nets for security critical applications, even with existing state-of-the-art security measures

I. INTRODUCTION

Neural networks has become of significant importance because of its high effectiveness in various Machine Learning tasks. This includes things like speech processing, natural language processing etc

But there are ways to fool neural nets classification by slightly perturbing the image to be classified, in such a way that the tampering is visually undetectable.

This limits the domain of application of neural nets. For example, we can play a video with an hidden message to access neural net based voice control devices.

This need for security has allowed many researchers to look for security solutions. One of the recent, promising security termed *Defensive Distillation* had very good results initially by blocking the previously existing attacks against neural nets.

We used an attack algorithm to construct an upper bound on the robustness of this “security enhanced” neural net. As case study we use this attack to demonstrate that defensive distillation does not actually remove these adversarial examples.

We therefore demonstrate two very important needs when it comes to using neural nets in security critical areas

- There currently exists no secure way to do it

- Maintain the highest standard of safety while employing neural nets in these areas

Assumably, we might be better off to not use neural nets at all when it comes to security critical areas.

II. BACKGROUND

A. Defensive Distillation

To defensively distill a neural network, begin by first training a network with identical architecture on the training data in a standard manner. When we compute the softmax while training this network, replace it with a more-smooth version of the softmax (by dividing the logits by some constant T). At the end of training, generate the soft training labels by evaluating this network on each of the training instances and taking the output labels of the network.

The intuition here is that we’ll hopefully prevent overfitting. Some philosophies say that it’s the susceptibility of overfitting in neural nets that cause adversarial examples to hide behind blind spots. *Defensive Distillation* effectively tries to remove these blind spots.

B. Adversarial Examples

Given a valid input x and a target $t \neq C^*(x)$, it is often possible to find a similar input x' such that $C(x') = t$ yet x, x' are close according to some distance metric. An example x' with this property is known as an adversarial example.

C. Distance Metrics

The L_p distance is written $\|x - x'\|_p$, where the p-norm

$\| \cdot \|_p$ is defined as

$$\|v\|_p = \left(\sum_{i=1}^n |v_i|^p \right)^{\frac{1}{p}}$$

L2 distance measures the standard Euclidean (root-mean-square) distance between x and x' .

III. IMPLEMENTATION

A. Formulation of the problem

We formally define the problem of generating adversarial examples as follows:

$$\begin{aligned} & \text{minimize } D(x, x + \delta) \\ & \text{such that } C(x + \delta) = t \\ & \quad x + \delta \in [0, 1]^n \end{aligned}$$

where x is fixed, and the goal is to find δ that minimizes $D(x, x + \delta)$. That is, we want to find some small change δ that we can make to an image x that will change its classification, but so that the result is still a valid image. Here D is some distance metric.

The above formulation is difficult for existing algorithms to solve directly, as the constraint $C(x + \delta) = t$ is highly non-linear. Therefore, we express it in a different form that is better suited for optimization. We define an objective function f such that $C(x + \delta) = t$ if and only if $f(x + \delta) \leq 0$.

Now we use an alternative formulation as follows:

$$\begin{aligned} & \text{minimize } D(x, x + \delta) + c \cdot f(x + \delta) \\ & \text{such that } x + \delta \in [0, 1]^n, \delta \text{ is the perturbation} \end{aligned}$$

Here the choice of the value of c is such that the it is the minimal solution x^* of the problem. This is done empirically and turns out to be equivalent to the initial formulation.

Here f is a suitable objective function defined as

$$f(x') = \max(\max\{Z(x')_i, i \neq t\} - Z(x')_t, -\kappa)$$

where, $Z(\cdot)$ is the softmax function and κ is a hyperparameter

B. Data Preprocessing

The input was taken as pixels in the form of grayscale numbers. The following preprocesses were carried out prior to implementing the attack

- Box Constraints: Normalise $x + \delta$ between $[0, 1]^n$

- Change of Variables: Take δ_i as follows to satisfy the box constraints

$$\delta_i = \frac{1}{2}(\tanh(w_i) + 1) - x_i$$
- Discretization: We model pixel intensities as a (continuous) real number in the range $[0, 1]$. However, in a valid image, each pixel intensity must be a integer in the range $\{0, 1, \dots, 255\}$. This additional requirement is not captured in our formulation. In practice, we ignore the integrality constraints, solve the continuous optimization problem, and then round to the nearest integer: the intensity of the i th pixel becomes $\lfloor 255(x_i + \delta_i) \rfloor$. $\lfloor \cdot \rfloor$ is the greatest integer function.

C. L2-attack

Putting these ideas together, we obtain a method for finding adversarial examples that will have low distortion in the L2 metric. Given x , we choose a target class t (such that we have $t \neq C^*(x)$) and then search for w that solves





































































































$$\begin{aligned} & \text{minimize} \\ & \left\| \frac{1}{2}(\tanh(w) + 1) - x \right\|_2^2 + c \cdot f\left(\frac{1}{2}(\tanh(w) + 1)\right) \end{aligned}$$

We use the Adam's optimizer to solve this fast and accurately. An Adam's optimizer adds an acceleration term to the standard momentum based gradient descent algorithm we normally use.

In practice we notice that gradient descent algorithms tend to get stuck at local minimas. To overcome this problem we perform gradient descent starting from multiple starting points. This thus reduces the likelihood of the algorithm getting stuck at local minimas.

D. Results

We could break through the distillation security and get the neural net to misclassify the newly generated examples. The images in the next page show our results.

		Target Classification (L_2)									
		0	1	2	3	4	5	6	7	8	9
Source Classification	0										
	1										
	2										
	3										
	4										
	5										
	6										
	7										
	8										
	9										

IV. CONCLUSIONS

The existence of adversarial examples limits the areas in which deep learning can be applied. It is an open problem to construct defenses that are robust to adversarial examples. In an attempt to solve this problem, defensive distillation was proposed as a general-purpose procedure to increase the robustness of an arbitrary neural network.

REFERENCES

- [1] NICHOLAS CARLINI, DAVID WAGNER, Towards Evaluating the Robustness of Neural Nets
- [2] https://github.com/carlini/nn_robust_attacks.