Adversarial Examples with MNIST

Evan Lavender
edl43@drexel.edu

Abstract

Deep neural networks have been widely implemented in a variety of tasks. Despite this, they can be easily fooled by small intentional perturbations to their input. These perturbations can be calculated directly using knowledge of the model, or indirectly using estimates. The *Fast Gradient Sign Method* (FGSM) and *Natural Evolution Strategy* (NES) are shown to generate adversarial examples for different models. The same techniques are used to train new models. Including adversarial examples in the training process is shown to decrease the success rate of further adversarial attacks.

1 Introduction

Deep learning has made significant progress in a wide domain of tasks and is being applied in many safety-critical environments [Yuan et al., 2017]. Despite this success, there are many concerns in the field of safety and security. [Szegedy et al., 2013] found that they could cause a deep neural network to misclassify an image by applying a certain hardly perceptible perturbation that maximizes the network's prediction error. They decided to call these misclassified images *Adversarial Examples*.

Networks can also misclassify an image containing physical perturbations on real objects. With a perturbation in the form of only black and white stickers, [Eykholt et al., 2017] are able to attack a real stop sign, causing targeted misclassification. [Szegedy et al., 2013] suggest that this property is instrinsic to current deep neural networks.

Adversarial attacks can be categorized as either white-box or black-box. The difference is determined by how much information the attacker is allowed to know of the network. In a white-box attack, full knowledge of the model is allowed. This includes architecture, inputs, outputs, weights, and gradients. In a black-box attack, only knowledge of the inputs and outputs is allowed.

The goal of the attack can be categorized as either a misclassification or source/target misclassification. A misclassification attack wants the output classification to be incorrect, regardless of what it actually is. A source/target misclassification attack wants the network to predict a desired target classification.

2 Related Work

To generate adversarial examples, [Szegedy et al., 2013] used a *L-BFGS* method. This attack used an expensive linear search method which was time-consuming and impractical [Yuan et al., 2017].

[Goodfellow et al., 2014] proposed the *Fast Gradient Sign Method* (FGSM). This method is "fast" because it does not require an iterative procedure to compute examples [Kurakin et al., 2016]. The sign of the gradient of the cost function with respect to the input x is scaled by a hyperparameter ϵ and added to x :

$$x^{adv} = x + \epsilon sign(\nabla_x J(\hat{y}, y))$$

[Goodfellow et al., 2014]. The ϵ parameter controls *how much* the image is allowed to change. As the ϵ value is increased, the perturbations become more perceptible.

These are examples of white-box attack methods. They require specific knowledge of the network in order to calculate the exact gradients needed to perturb the input. [Papernot et al., 2016] introduce a novel black-box method in which they train a local model to substitute for the target network. The local model is used to create adversarial examples that are misclassified by the target network [Papernot et al., 2016]. In [Szegedy et al., 2013], they found that adversarial examples genereted by one network are still statistically hard for another network, even if it was trained with different hyperparameters or on a different set of examples entirely.

Zeroth Order Optimization (ZOO) is a different kind of black-box method by [Chen et al., 2017]. Instead of leveraging transferability from substitute models, ZOO-based attacks directly estimate the gradients of the targeted network.

3 Methodology

3.1 Iterative Fast Gradient Sign Method

The white-box FGSM can be extended into the *Basic Iterative Method* [Kurakin et al., 2016]. The "fast" method is applied multiple times with a small step size α, with the result of each intermediate step being clipped to ensure they are in an ϵ-neighborhood of the original image:

$$x_0^{adv} = x, x_{N+1}^{adv} = Clip_\epsilon\{x_N^{adv} + \alpha sign(\nabla_x J(\hat{y}, y))\}$$

The algorithm in 1 is used for general misclassification. If the goal is a source/target misclassification, then the algorithm can be modified in two ways. First, x^{adv} is to be returned if $\hat{y} == y$. Second, the scaled gradient is subtracted from the *noise*. This will cause a *decrease* in the loss function for target y.

3.2 Natural Evolution Strategy

Evolution Strategies (ES) are a class of heuristic search procedures inspired by natural evolution: At every iteration, a population of examples are perturbed and their objective function value is evaluated [Salimans et al., 2017]. Samples are recombined to form the

Algorithm 1: Basic Iterative Method (BIM or I-FGSM)

Input: iterations N, epsilon ϵ, step size α, model M, input x, class y
$noise = \text{zeros}(x.\text{shape})$
for $i = 1, 2, 3, ...N$ **do**
  $x^{adv} = x + noise$
  $\hat{y} = M(x^{adv})$
  return x^{adv} if $\hat{y} \neq y$
  $noise = Clip_{\epsilon}\{noise + \alpha sign(\nabla_x J(\hat{y}, y))\}$
end

population for the next iteration, until the objective is fully optimized [Salimans et al., 2017].

This concept can be easily applied to create adversarial examples as a black-box attack method. The objective function being optimized is the loss function of the model, with the fitness of a sample being the value of the function. Because an entire population of random samples is evaluated, this method is akin to estimating the gradient of the loss function using finite differences.

Algorithm 2: Natural Evolution Strategy

Input: iterations N, population size n, epsilon ϵ, step size α, sigma σ, model M, input x, class y
$\mu = 0$
for $i = 1, 2, 3, ...N$ **do**
  $noise \sim \mathcal{N}(0, I)$
  $population = Clip_{\epsilon}\{\mu + \sigma * noise\}$
  $fitness = \text{zeros}(n)$
  for $j = 0, 1, 2, ...n$ **do**
    $x^{adv} = x + population_j$
    $\hat{y} = M(x^{adv})$
    return x^{adv} if $\hat{y} \neq y$
    $fitness_j = J(\hat{y}, y)$
  end
  $\mu = \mu + \alpha \frac{1}{n\sigma} * dot(noise.T, fitness)$
end

3.3 Adversarial Defense

A simple technique for building robustness against adversarial attacks is to use adversarial examples in the training data set. During training, each sample is perturbed with a probability p.

4 Evaluation

The MNIST database [LeCun et al., 2010] is used to evaluate both white-box (I-FGSM) and black-box (NES) attack methods. This consists of 70000 grayscale handwritten digits of size $28x28$. 100 randomly selected testing samples are used for adversarial example generation. Each attack is allowed 500 iterations, with epsilon values of $\epsilon = 0.1, 0.2, 0.3$, for both general misclassification and source/target misclassification. I-FGSM uses a step size $\alpha = 0.001$, and NES uses a step size of $\alpha = 0.05$ with sigma $\sigma = 0.25$. The median number of queries is taken using successful attacks only.

The network used is convolutional neural network based on [1]. Two new models are created through adversarial training and tested. These models are created by perturbing the training data with probability $p = 0.2$ for 10 iterations. The attack uses a randomly generated target. The I-FGSM and NES attacks uses a step size of $\alpha = 0.01, 0.2$ respectively. In the table 1 below, the value on the left represents a general misclassification attack, and the value on the right is for a source/target misclassification attack.

	Original		**I-FGSM Trained**		**NES Trained**	
Test Accuracy	.9805		.9810		.9826	
			ϵ **= 0.1**			
	I-FGSM	**NES**	**I-FGSM**	**NES**	**I-FGSM**	**NES**
Success Rate	0.12/0.04	0.08/0.03	0.1/0.04	0.07/0.03	0.08/0.04	0.08/0.04
Query Count	46/32	187/186	39/66	645/1321	40/43	668/867
			ϵ **= 0.2**			
Success Rate	0.86/0.23	0.43/0.08	0.61/0.12	0.38/0.03	0.71/0.15	0.40.06
Query Count	171/186	1291/1653	167/189	1226/2133	161/191	991/2002
			ϵ **= 0.3**			
Success Rate	1.0/0.95	0.9/0.6	1.0/0.84	0.85/0.42	1.0/0.89	0.84/0.5
Query Count	173/242	1123/1745	200/263	1251/1676	188/245	1115/1682

Table 1: Results

The I-FGSM outperforms the NES attack method both in terms of query count and success rate. This is to be expected, as the white-box attack method has full knowledge of the model. The adversarial training techniques resulted in increased testing accuracy and decreased adversarial success rate.

[1] https://github.com/pytorch/examples/tree/master/mnist

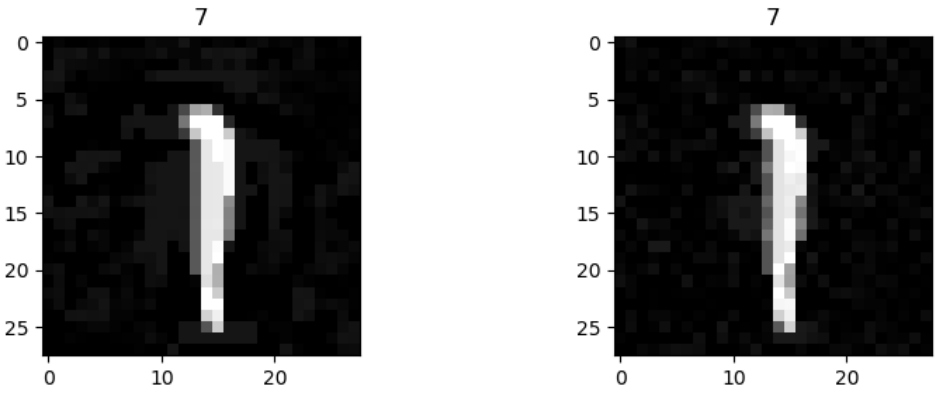

Figure 1: $1 \rightarrow 7$ with $\epsilon = 0.1$

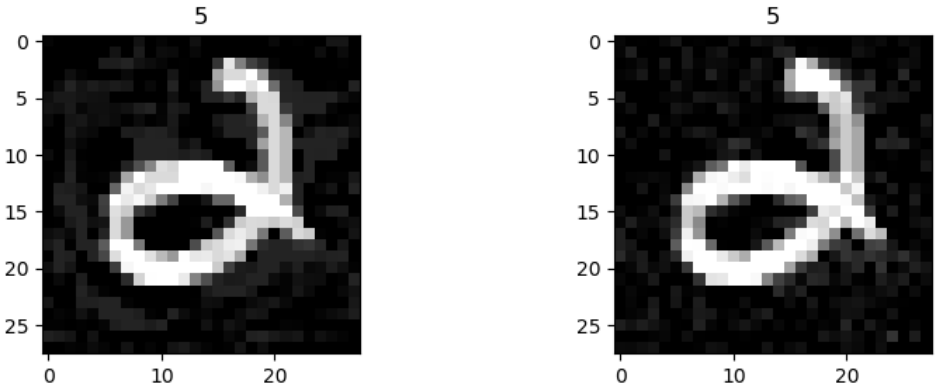

Figure 2: $2 \rightarrow 5$ with $\epsilon = 0.2$

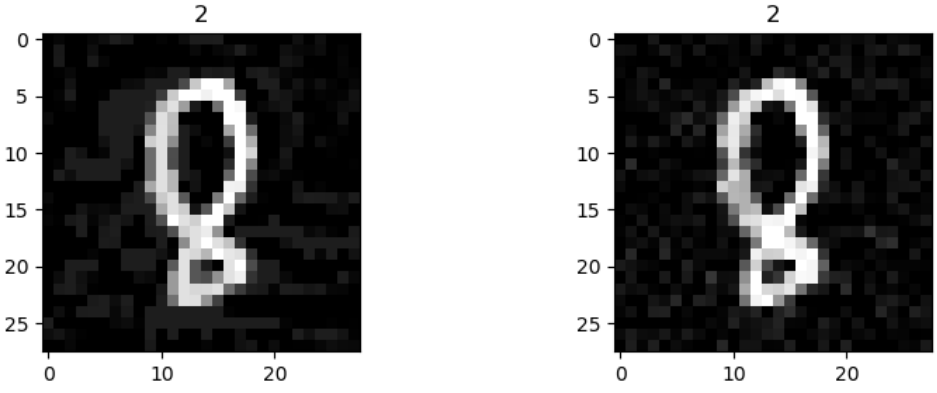

Figure 3: $8 \rightarrow 2$ with $\epsilon = 0.3$

5 Conclusions

The *Iterative Fast Gradient Sign Method* and *Natural Evolution Strategy* are shown to successfully generate adversarial examples for the MNIST data set. They are successful with the goal of general misclassification and source/target misclassification. Using these same methods to perform adversarial training, it was shown that the resulting models were less susceptible to attack. The test accuracy also increased as a result of the training.

6 Future Work

This work is just scratching the surface of the landscape of adversarial machine learning. Aside from the task of hyperparameter tuning, there are deep questions raised by the existence of adversarial examples. If a network can generalize well, how can it be confused by these examples, which are nearly indistinguishable from the regular examples [Szegedy et al., 2013]? Questions such as this open a huge area to explore for future research.

Future paths for this work specifically include:

- More data sets
- Experiment with using adversarial training to *strengthen* actual predictions instead of lowering them.
- See if examples created by the original network can fool the adversarially-trained networks.
- Identify patterns in misclassification, i.e. what is the most likely misclassification?

References

[Chen et al., 2017] Chen, P.-Y., Zhang, H., Sharma, Y., Yi, J., and Hsieh, C.-J. (2017). Zoo. *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security - AISec '17*.

[Eykholt et al., 2017] Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A., Kohno, T., and Song, D. (2017). Robust physical-world attacks on deep learning models.

[Goodfellow et al., 2014] Goodfellow, I. J., Shlens, J., and Szegedy, C. (2014). Explaining and harnessing adversarial examples.

[Kurakin et al., 2016] Kurakin, A., Goodfellow, I., and Bengio, S. (2016). Adversarial examples in the physical world.

[LeCun et al., 2010] LeCun, Y., Cortes, C., and Burges, C. (2010). Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2.

[Papernot et al., 2016] Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., and Swami, A. (2016). Practical black-box attacks against machine learning.

[Salimans et al., 2017] Salimans, T., Ho, J., Chen, X., Sidor, S., and Sutskever, I. (2017). Evolution strategies as a scalable alternative to reinforcement learning.

[Szegedy et al., 2013] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2013). Intriguing properties of neural networks.

[Yuan et al., 2017] Yuan, X., He, P., Zhu, Q., and Li, X. (2017). Adversarial examples: Attacks and defenses for deep learning.