Practical Adversarial Attacks Generation to Machine Learning Models
Explore insight of pixel Attacks with DE algorithm

C. Tao
School of Information Technology and Electrical Engineering
The University of Queensland, Qld., 4072, Australia

Abstract

With the development of machine learning, convolutional neural network (CNN) based methods have achieved great progress recently. However, this also leads to machine learning models being exposed to malicious attacks, especially adversarial attacks have become one of the most popular evasion attacks. In this paper, I explore the insight of the one-pixel attack, a black-box attack with a highly limited situation that only modifies one pixel of the original image by using a differential evolution algorithm. Compared to other popular white-box attacks, the original One-pixel black-box attack costs high computation and requires much time to achieve the goal. In this case, I try to improve the original pixel-attack method by changing the optimization algorithm and making this attack more adaptive when facing real-world issues.

1 Introduction

As the development of machine learning, especially deep learning can model more complex algorithm functions by using large datasets, research and probing on adversarial attacks has been a popular and important area. Preventing malicious attacks can be crucial for machine learning models to approach robustness.

The adversarial attacks are generated by adding some noises or perturbations to benign images, which makes the raw image become an adversarial example. According to Goodfellow et al. [1], these adversarial examples can make the DNN become vulnerable, which causes damages to DNN misclassify. Some adversarial examples are easy to be perceptible to human eyes due to the pixel variation. However, some examples are difficult to detect by machine learning models and humans. In this case, adding only one tiny amount of well-designed perturbation for creating adversarial images has been proposed by Su et al. [3], namely the one-pixel attack. Their works have several advantages: It is effective and comparable to other white-box adversarial attacks such as FGSM. Also, it is a semi-black-box attack that only needs the class labels rather than the whole network structures and gradients. Moreover, it is flexible since it can attack various types of machine learning models even the gradient calculation is difficult [3]. In this situation, adversarial examples generation can be treated as an optimization problem. Specifically, we can assume that a pretrained image classifier f, such as vgg16 network, can receive inputs x = (x1, …, xn) as the benign image and classify the true label class as t. The probability of x for class t could be ft(x). Then, the adversarial perturbation, which relates to x, can be the vector e(x) = (e1, …, en). And the limitation of the maximum modification values notes as L.

However, according to Su et al. [3], the limitation of the maximum modification value will be 1 since the attack aims to change only one pixel to the noise from the original image. Thus, the core task for the one-pixel attack is to find the most suitable pixel that needs to be modified. And the question will be changed like below:

$$\begin{aligned} \underset{e(\mathbf{x})^*}{\text{maximize}} \quad & f_{adv}(\mathbf{x} + e(\mathbf{x})) \\ \text{subject to} \quad & \|e(\mathbf{x})\|_0 \leq d, \end{aligned}$$

Fig 1. Optimization problem for pixel attack

which d = 1 for one-pixel attack.

2 Method

My goal is to improve the original One-pixel attack performance and also try to find a way to make it time-sufficient. Firstly, I'll introduce the initial algorithm of the One-pixel attack, then explain my proposed improvements on them.

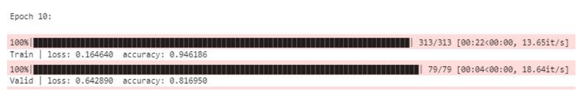
The original one-pixel attack mainly adopts Differential evolution (DE), which is a population-based algorithm that aims to solve complex multi-modal optimization problems [4]. Since DE belongs to evolutionary algorithms, DE can keep the higher quality value during each iteration until it finds the 'best' solution to a problem. According to Su et al. [3], there are three advantages to using differential evolution: firstly, the solution of this algorithm is based on the higher probability, which means there is no need to compute every exact pixel value for one image, especially only one pixel needs to be perturbated. This saves much time and much space. The other feature is that DE requires less information from the target model than other optimization algorithms, such as gradient descent. This not only makes the one-pixel attack to be a black-box type but also makes it a more efficient attack tactic. Lastly, DE is considered as a simple and flexible method. It can be adopted to attack various models' classifiers rather than focus on one specific classifier. However, DE still needs to gain the probability labels of the model, which make it become a semi black-box style.

To improve the original algorithm, I will try the self-adaptive differential evolution algorithm, which is a variation of DE. According to Qin et al. [2], the original DE highly depends on the parameter settings to achieve good performance. Thus, self-adaptive differential evolution can automatically adapt the parameter settings during the evolution process, which may improve the performance of the one-pixel attack in different situations.

3 Experiments

The dataset I use for this experiment is CIFAR-10. And the network model structure is the pretrained vgg16 from PyTorch library so far. And I trained the vgg16 network with some parameter changes to fit CIFAR-10 dataset use. Because the time is limited, I set the epoch as 10 rather than the previous value 100. And I still got a fair result of the accuracy is around 82% on the test set (see Fig 2).

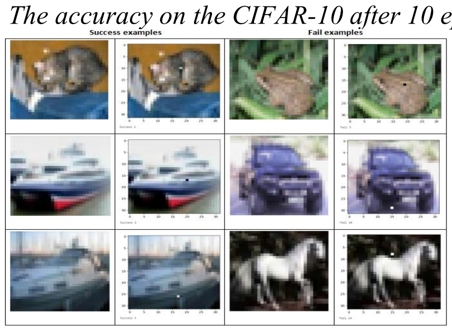
After this, I used the original one-pixel attack algorithm, which proposed by Sy et al. [3], to test my vgg16 model. Generally, the untargeted success rate of the original algorithm is around 50% through my process. Some works examples are provided (See Fig 3).



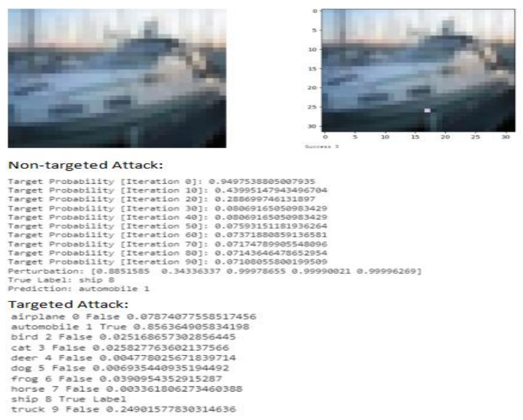
```
Epoch 10:

100%|████████████████████████████████████████████████| 313/313 [00:22<00:00, 13.65it/s]
Train | loss: 0.164640  accuracy: 0.946186
100%|████████████████████████████████████████████████| 79/79 [00:04<00:00, 18.64it/s]
Valid | loss: 0.642890  accuracy: 0.816950
```

Fig 2. The accuracy on the CIFAR-10 after 10 epochs.



(a) Successful attacks and fail attacks examples



```
Non-targeted Attack:
Target Probability [Iteration 0]: 0.9497538805007935
Target Probability [Iteration 10]: 0.43995147943496704
Target Probability [Iteration 20]: 0.288699746131897
Target Probability [Iteration 30]: 0.08069165050983429
Target Probability [Iteration 40]: 0.08069165050983429
Target Probability [Iteration 50]: 0.07593151181936264
Target Probability [Iteration 60]: 0.07371880859136581
Target Probability [Iteration 70]: 0.07174789905548096
Target Probability [Iteration 80]: 0.07143646478652954
Target Probability [Iteration 90]: 0.07108055800199509
Perturbation: [0.8851585  0.34336337 0.99978655 0.99990021 0.99996269]
True Label: ship 8
Prediction: automobile 1

Targeted Attack:
 airplane 0 False 0.07874077558517456
 automobile 1 True 0.856364905834198
 bird 2 False 0.025168657302856445
 cat 3 False 0.025827763602137566
 deer 4 False 0.004778025671839714
 dog 5 False 0.006935440935194492
 frog 6 False 0.0390954352915287
 horse 7 False 0.003361806273460388
 ship 8 True Label
 truck 9 False 0.24901577830314636
```

(b) untargeted attack & targeted attack examples

Fig 3. The approach of one-pixel attacks

4 General Discussion

Based on the result of the experiment, the original one-pixel attack achieves a decent result on CIFAR-10 with vgg16 network. However, it took over an hour to attack the whole test set on my desktop (GPU 1660ti, CPU Intel i5-10400F). Thus, either increasing the attack success rates or reducing the running time will be the core task for my proposed work. According to Zhou et al [4], their proposed algorithm adapted within one-pixel attack can achieve a wonderful output, which the model only has 13% to classify the image successfully. Therefore, changing the original algorithm may achieve significant improvements. In addition, change the number of perturbated pixels could also be a way to achieve my goal, for example, change one pixel to three or even more. And collect the output to evaluate this tactic.

5 References

[1] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and Harnessing Adversarial Examples." arXiv, Mar. 20, 2015. Accessed: Aug. 10, 2022. [Online]. Available: http://arxiv.org/abs/1412.6572
[2] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in *2005 IEEE Congress on Evolutionary Computation*, 2005, vol. 2, pp. 1785–1791 Vol. 2. doi: 10.1109/CEC.2005.1554904.
[3] J. Su, D. V. Vargas, and S. Kouichi, "One Pixel Attack for Fooling Deep Neural Networks," *IEEE Trans. Evol. Computat.*, vol. 23, no. 5, pp. 828–841, Oct. 2019, doi: 10.1109/TEVC.2019.2890858.
[4] T. Zhou, S. Agrawal, and P. Manocha, "Optimizing One-pixel Black-box Adversarial Attacks." arXiv, Apr. 30, 2022. Accessed: Oct. 07, 2022. [Online]. Available: http://arxiv.org/abs/2205.02116