Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 3, 2021, Pages. 8405 - 8412
Received 16 February 2021; Accepted 08 March 2021.

One Pixel Attack for Fooling Neural Networks

Shubham Sinha[1], S.S. Saranya[2]

[1]Final Year Student, Computer Science Engineering, SRM Institute of Science and Technology, Kattankulathur Campus, Tamil Nadu, India. E-mail: sr7006@srmist.edu.in

[2]Assistant Professor, Department of Computer Science and Engineering, School of Computing, SRM Institute of Science and Technology, Kattankulathur Campus, Tamil Nadu, India. E-mail: saranyas6@srmist.edu.in

ABSTRACT

Recent research says that by adding relatively small amount of perturbation in the input vector of DNN (Deep Neural Network), the output can easily be altered. In this project we will be performing the attack by modifying only one pixel of the input vector. To do that we will be proposing a novel method, that will help us to generate one-pixel adversarial perturbation based on something called DE (Differential Evolution). This will be a black box attack (having less target information) and it can fool more types of neural networks because of features of DE. The results for this test shows that few of the original images present in CIFAR-10 testing dataset and few from the ImageNet testing images can be attacked to minimum of one target class just by changing one pixel. The same vulnerability is present in the original dataset of CIFAR-10. Thus, this attack explores a different take on adversarial ML, showing that current Deep Neural Networks are susceptible to such low dimension attacks.

KEYWORDS

DNN, DE, Pixel, Vector, Dataset, Vulnerability, Attack.

Introduction

Recent studies have shown that DNNs are not that much secure and are vulnerable to various types of attacks; however it is still not confirmed what causes these vulnerabilities, various researchers are doing their research in order to find out the reason for this Studies have shown that adding perturbation on original images can make Deep Neural Network misclassify the image. Since then, many algorithms are being generated to form this so called “adversarial image”. So the general idea behind the adversarial image is to add a little amount of perturbation in the natural image. This perturbation may or may not be visible to the human eye but it won't change the image recognition ability of human but it can change the classifier's output. The perturbation should not be large otherwise it would be visible to human eye as well.

This paper will be perturbing only one pixel and this will be a black box attack, the only thing that is needed is the probability labels of the classifier for the natural image. This attack will be having followingadvantages:

1. Effectiveness - On Kaggle CIFAR-10 dataset, non-targeted attacks on three different neural networks gave 68.71%, 71.66%, and 63.53% success rates and on the original CIFAR-10 dataset we got 22.60%, 35.20%, and 31.40% success rates.
2. Semi-Black Box Attack - Here there’s no need for any information about the internal of the DNN, here the only thing which is needed is the probability labels for the target class.
3. Flexibility - This attack will work on all types of DNN models.

There will be a locality analysis as well. In locality analysis, attack will be executed near to the successful attack pixel, same pixel perturbation will be used but with a different pixel coordinates. The success rate of pixel near to successful pixel is actually effective and almost same among different DNNs. This actually shows that vulnerable areas are the receptive fields, not the pixels or neurons. This confirms a fascinating property that is being shared among Deep Neural Networks which does not depends on the DNN model or success rate of the attack.

Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 3, 2021, Pages. 8405 - 8412
Received 16 February 2021; Accepted 08 March 2021.

AllConv

Car Car Deer Horse Ship
Airplane (82.4%)Airplane (82.0%)Dog (70.7%)Dog (80.0%)Car (90.7%)

NiN

Bird Deer Dog Horse Ship
Frog (88.8%)Dog (86.4%) Cat (75.5%) Frog (99.9%) Airplane (62.7%)

VGG

Bird Cat Cat Deer Ship
Frog (86.5%) Bird (66.2 %)Dog (78.2%)Airplane (99.9%) Airplane (88.2%)

Figure 1. One Pixel Attack success on 3 types of DNN on CIFAR-10 Dataset

Bassinet Cup Hamster Kettle
Paper Towel (16.2%) Soup Bowl (16.7%) Nipple (42.3%) Joystick (37.3%)

Figure 2. One Pixel Attack success on ImageNet Dataset

Related Works

A. Methodology

So the images that cause classifier to misclassify the image are called adversarial samples. So we have a function that takes image as an input and gives and output that is the probability label, so let $f(x) \epsilon R^k$ be the output of the classifier where $x \epsilon R^{m*n}$ is the input.

Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 3, 2021, Pages. 8405 - 8412
Received 16 February 2021; Accepted 08 March 2021.

Here R represents real numbers, m and n are dimensions of the input image, and k is the number of classes.

So now we need to define an adversarial sample$x^{'}$, which will be as follows:

$$x^{'} = x + \varepsilon_x$$
$$\{ x^{'} \epsilon R^n \mid argmax\big(f(x^{'})\big) \neq argmax(f(x)) \}$$

in which $\varepsilon_x \ \epsilon R^{m*n}$ is small amount of perturbation added to the input image and argmax is an operation that gives the maximum value from a target function.

Now we need to search for adversarial samples, to do this we've to use those perturbation vectors that will increment the given soft-label $f(x)_t$ in which the 't' represents the target class' index.

So we can say that the whole scenario can be written as the following optimization problem:

$$maximize_{\varepsilon_x} f(x + \varepsilon_x)_t$$
$$subject\ to \|\varepsilon_x \ \| \leq L$$

When talking about untargeted attacks, we need to minimize the soft label for the outputted class $f(x)_t$.
So out equation would be:

$$maximize_{\varepsilon_x} -f(x + \varepsilon_x)_t$$
$$subject\ to \|\varepsilon_x \ \| \leq L$$

The value of ε should be small, it should not be large enough that it changes the class of the input image, this would negate the adversarial sample creation and it will not be a misclassification.

So we've $f(x)$ as our target, which is basically an image classifier which will take n dimensional input vector "X", where$X = (x_1, x_2, \dots, x_n)$ is the original image being classified to class t correctly so probability of X belonging to class t will be $f_t(x)$ and we've another vector $e(x) = (e_1, e_2, \dots., e_n)$ which is the perturbation according to the X.

The target class be called adv and maximum modification allowed will be L and L is always measured as length of the vector$e(x)$. So when it comes to targeted attacks, the main goal is to get an optimal solution for this equation:

$$maximize_{e(x)^*} f_{adv}(x + e(x))$$
$$subject\ to \| e(x) \ \| \leq d$$

Here the "d" is a tiny number, usually d = 1 in our case of one pixel attack.

Till now we can say that we need to find 2 values:

- Which pixel we need to perturb
- Value of the modification

B. Differential Evolution

Differential Evolution also known as DE, is an optimization algorithm which is used to solve complex optimization problems. Differential Evolution belongs to the class of Evolutionary Algorithms.

To optimize a problem solution using DE, we first need an initial solution set (usually it is some random values) then in each iteration another set of candidate solution will be generated, they are called children. After that children will be compared to their parents, the child will survive if it's fitness value is better than it's parent. Initially 400 child solutions are reproduced then in next iteration again 400 child will be produced and so on. Normally we've 100 as the limit but this could be changed.

Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 3, 2021, Pages. 8405 - 8412
Received 16 February 2021; Accepted 08 March 2021.

Storn and Price introduced this algorithm in 1990s. Many books have been published on theoretical as well as practical aspects of using DE in domains like parallel computing, multi-objective optimization, constrained optimization etc.

So many variants of DE are being developed in an effort to improve optimization performance. There exist different schemes for performing crossover and mutation of agents in the basic algorithm mentioned above.

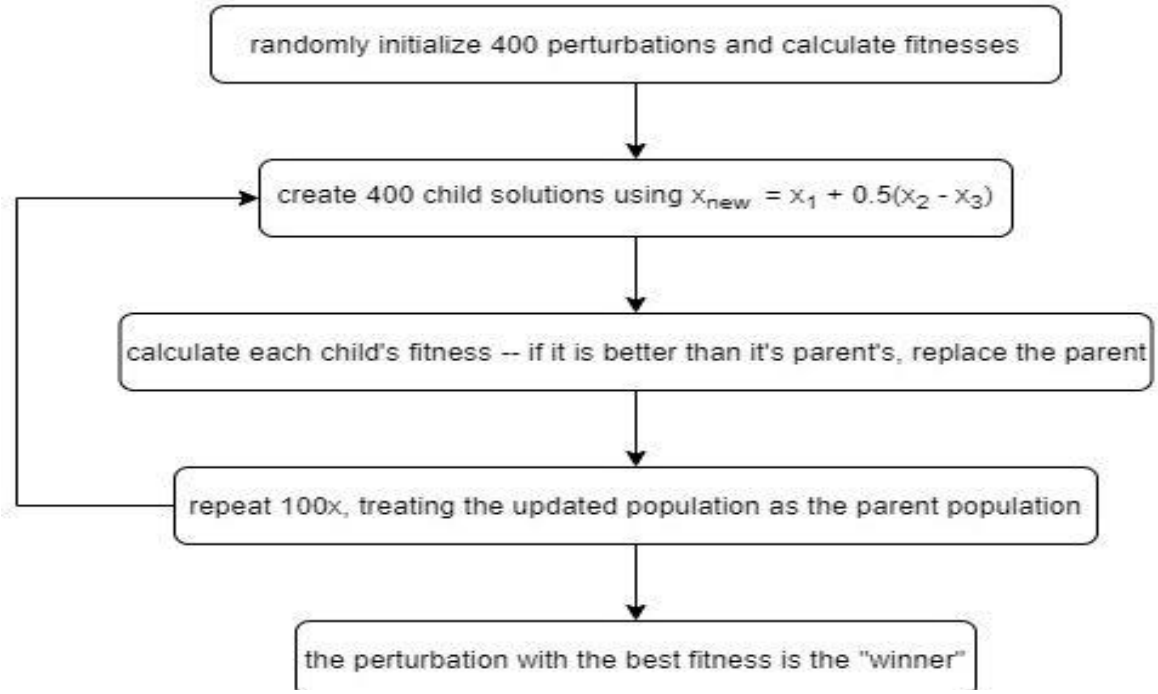

Figure 3. A flowchart describing DE Fitness is defined as the confidence in the correct label; x_1, x_2, and x_3 are randomly selected members of the parent population.

We'll encode our perturbation into an array (numpy array), this will be our candidate solution which has to be optimized by Differential Evolution. The perturbation is actually a tuple having 5 values, namely x-y coordinates of the pixel and RGB value of the perturbation. Each perturbation will be modifying one pixel only. The candidate solution is generated by the formula:

$$x_i(g+1) = x_{r_1}(g) + F\left(x_{r_2}(g) - x_{r_3}(g)\right)$$
$$where\ r1 \neq r2 \neq r3$$

Where x_i belongs to the set of candidate solutions, r_1, r_2, and r_3 are arbitrary numbers, F has a value of 0.5 this is called scale parameter and g is the current index of generation. As mentioned earlier, the stop criterion is 100 iterations but when the probability label for the target class is greater than or equal to 90% the premature stop criterion will occur, and when talking about non-targeted attacks it will stop when original class probability is lower than 5%.

C. Models

We'll be training 3 types of models viz. All Convolution Neural Network, Network in Network (NiN), and VGG16 as our target networks on CIFAR-10 Dataset.
The structures are defined in the following tables:

Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 3, 2021, Pages. 8405 - 8412
Received 16 February 2021; Accepted 08 March 2021.

Table 1. AllConv

"conv2d layer(kernel=3, stride = 1, depth=96)
conv2d layer(kernel=3, stride = 1, depth=96)
conv2d layer(kernel=3, stride = 2, depth=96)
conv2d layer(kernel=3, stride = 1, depth=192)
conv2d layer(kernel=3, stride = 1, depth=192)
dropout(0.3)
conv2d layer(kernel=3, stride = 2, depth=192)
conv2d layer(kernel=3, stride = 2, depth=192)
conv2d layer(kernel=1, stride = 1, depth=192)
conv2d layer(kernel=1, stride = 1, depth=10)
average pooling layer(kernel=6, stride=1)
flatten layer
softmax classifier"

Table 2. NiN

"conv2d layer(kernel=5, stride = 1, depth=192)
conv2d layer(kernel=1, stride = 1, depth=160)
conv2d layer(kernel=1, stride = 1, depth=96)
max pooling layer(kernel=3, stride=2)
dropout(0.5)
conv2d layer(kernel=5, stride = 1, depth=192)
conv2d layer(kernel=5, stride = 1, depth=192)
conv2d layer(kernel=5, stride = 1, depth=192)
average pooling layer(kernel=3, stride=2)
dropout(0.5)
conv2d layer(kernel=3, stride = 1, depth=192)
conv2d layer(kernel=1, stride = 1, depth=192)
conv2d layer(kernel=1, stride = 1, depth=10)
average pooling layer(kernel=8, stride=1)
flatten layer
softmax classifier"

Table 3.VGG16

"conv2d layer(kernel=3, stride = 1, depth=64)
conv2d layer(kernel=3, stride = 1, depth=64)
max pooling layer(kernel=2, stride=2)
conv2d layer(kernel=3, stride = 1, depth=128)
conv2d layer(kernel=3, stride = 1, depth=128)
max pooling layer(kernel=2, stride=2)
conv2d layer(kernel=3, stride = 1, depth=256)
conv2d layer(kernel=3, stride = 1, depth=256)
conv2d layer(kernel=3, stride = 1, depth=256)
max pooling layer(kernel=2, stride=2)
conv2d layer(kernel=3, stride = 1, depth=512)
conv2d layer(kernel=3, stride = 1, depth=512)
conv2d layer(kernel=3, stride = 1, depth=512)
max pooling layer(kernel=2, stride=2)
conv2d layer(kernel=3, stride = 1, depth=512)
conv2d layer(kernel=3, stride = 1, depth=512)
conv2d layer(kernel=3, stride = 1, depth=512)
max pooling layer(kernel=2, stride=2)
flatten layer
fully connected(size=2048)
fully connected(size=2048)
softmax classifier"

Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 3, 2021, Pages. 8405 - 8412
Received 16 February 2021; Accepted 08 March 2021.

The settings of the networks are very close to the original models but they have slight modifications so that we can get the maximum classification accuracy. Both of the targeted and non-targeted attacks will work on these models.

For both of the attacks, 500 natural images will be chosen from the CIFAR-10 Dataset randomly.

Here we will be using Kaggle CIFAR-10 dataset but not the original one. The dataset contains around 3,00,000 images which actually helps us to visually inspect the followings: duplication, blurring, clipping, rotation, adding few arbitrary bad pixels and so on. This data set is more practical since it simulates most common scenarios that image can contain random noise.

For each natural image present in the data set, total of nine targeted attacks were executed which were trying to disturb the image to other nine classes. Till here only targeted attack was launched and the efficacy of the non targeted attack is completely dependenton the targeted attack results evaluation.
If any of the original image can be perturbed to at least one of the target class out of the total classes present, the non-targeted attack on this image will succeed.

For ImageNet data set, we will be performing one non targeted attack using the previously used parameters of DE. We will be launching the attack on ImageNet data set by making use of a fitness function whose aim is to decrement the probability label of the actual class.

Results

Table 4. Success Rate of the Attack

	LeNet	**ResNet**
1 Pixel Attack (Original)	58%	21%
Random Pixel Attack	4.8%	3%
Nearby Pixel Attack	33%	31.3%

Table 4 shows the success rate of attack on both of the randomly chosen pixels, and nearby pixels. To obtain this result, first one pixel attack has to be executed. After that the same perturbation has to be used to change an arbitrary pixel in the true image and after that evaluate the success of the method. Now to obtain the results, one needs to execute both nearby and random pixel attack once per image for each successful attack in samples of the data set.

After calculating the mean for all activation maps for the preceding successful and un-successful attacks, one can tell if there exist any obvious difference between successful and failed attacks. The graph is shown in the figure x. One can see that even the average difference failed to distinguish between successful and un-successful attacks.

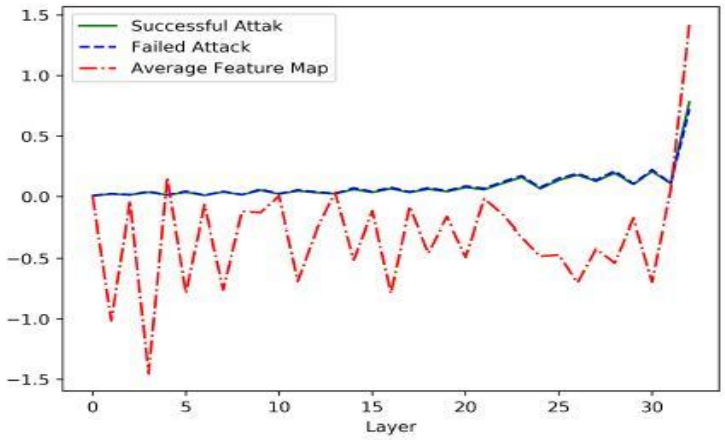

Figure 4. Average difference for all layers

Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 3, 2021, Pages. 8405 - 8412
Received 16 February 2021; Accepted 08 March 2021.

Conclusions

This paper showed that how a pixel change can cause an influence that spreads throughout the layers of the DNNs. The next module will use Activation Maps to properly examine the behavior of the perturbation within the DNN and it will tell us about the conflicting salience hypothesis.

This paper highlighted the following things:

- **Vulnerability of Receptive Fields:** Success rate on pixels near to the successfully attacked pixel are equally likely vulnerable. This actually shows that it's not neurons or pixels which are vulnerable but the receptive fields.
- **Similarity between Successful and Failed Attacks:** Both failed and successful perturbation surprisingly shows similar behavior and this behavior is nowhere related to the decrement in confidence with the label of the class. There exists some un-successful attacks which failed to change the confidence but they may have influenced all of the layers of the neural network and they also behave in similar fashion to successful attacks.

References

[1] Image Credits: ImageNet, and CIFAR-10.

[2] Su, J., Vargas, D.V., & Sakurai, K. (2019). One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, *23*(5), 828-841.

[3] Vargas, D.V., &Su, J. (2019). Understanding the one-pixel attack: Propagation maps and locality analysis. *arXiv preprint arXiv:1902.02947*.

[4] Khan, U., Woods, W., &Teuscher, C. (2019). Exploring and Expanding the One-Pixel Attack.

[5] Athalye, A., Engstrom, L., Ilyas, A., & Kwok, K. (2018). Synthesizing robust adversarial examples. *In International conference on machine learning*, *PMLR*, 284-293.

[6] Athalye, A., Carlini, N., & Wagner, D. (2018). Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *In International Conference on Machine Learning*, *PMLR*, 274-283.

[7] Barreno, M., Nelson, B., Sears, R., Joseph, A.D., &Tygar, J.D. (2006). Can machine learning be secure?. *In Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, 16-25.

[8] Symposium on Information, computer and communications security, 16–25, 2006.

[9] Barreno, M., Nelson, B., Joseph, A.D., &Tygar, J.D. (2010). The security of machine learning. *Machine Learning*, *81*(2), 121-148.

[10] Brown, T.B., Mané, D., Roy, A., Abadi, M., & Gilmer, J. (2017). Adversarial patch. *arXiv preprint arXiv:1712.09665*.

[11] Buckman, J., Roy, A., Raffel, C., & Goodfellow, I. (2018). Thermometer encoding: One hot way to resist adversarial examples. In *International Conference on Learning Representations*.

[12] Carlini, N., & Wagner, D. (2017). Adversarial examples are not easily detected: Bypassing ten detection methods. *In Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, 3-14.

[13] Carlini, N., & Wagner, D. (2017). Magnet and efficient defenses against adversarial attacks are not robust to adversarial examples. *arXiv preprint arXiv:1711.08478*.

[14] Carlini, N., & Wagner, D. (2017). Towards evaluating the robustness of neural networks. *In IEEE symposium on security and privacy (sp)*, 39-57.

Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 3, 2021, Pages. 8405 - 8412
Received 16 February 2021; Accepted 08 March 2021.

[15] Dang, H., Huang, Y., & Chang, E.C. (2017). Evading classifiers by morphing in the dark. *In Proceedings of the ACM SIGSAC conference on computer and communications security*, 119-133.

[16] Dhillon, G.S., Azizzadenesheli, K., Lipton, Z.C., Bernstein, J., Kossaifi, J., Khanna, A., &Anandkumar, A. (2018). Stochastic activation pruning for robust adversarial defense. *arXiv preprint arXiv:1803.01442*

[17] Goodfellow, I.J., Shlens, J., &Szegedy, C. (2014a). Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*

[18] Goodfellow, I.J., Shlens, J., &Szegedy, C. (2014b). Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*

[19] Grosse, K., Manoharan, P., Papernot, N., Backes, M., & McDaniel, P. (2017). On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280*

[20] Guo, C., Rana, M., Cisse, M., & Van Der Maaten, L. (2017). Countering adversarial images using input transformations. *In ICLR.arXiv preprint arXiv:1711.00117*