

(2017 SP) Towards Evaluating the Robustness of Neural Networks

1.Summary

In this paper, the author proposed three different attack methods to generate adversarial examples towards neural network models under different distance metrics. These methods defeat the state-of-art defense strategies which are thought to be robust previously. The author also proved that a high-confidence adversarial example for one model can apply to another model too. Even in the situation that the latter is defended using the state-of-art method. The author suggest that the methods in this paper show be a benchmark in future defense attempts.

2.Challenge

The main challenge in finding adversarial examples is that the example generated should be distinguished with the origin example as little as possible. Furthermore, there are already some defensive methods that let all of the previous attack algorithms invalid. So new attack methods should be proved to work effectively even with these defensive methods.

3.Main Idea

3.1.Targeted attack

Let $C(x)$ be the correct label of an input example x . Given an valid example x we can find a similar sample x' with a target t and $t \neq C(x)$. What an attacker should do is to find an valid example that is as close as possible to the origin input under some distance metrics. This can be demonstrated as the formulation below.

$$\begin{aligned} \min \quad & D(x, x + \delta) \\ \text{s.t.} \quad & C(x + \delta) = t \\ & x + \delta \in [0, 1] \end{aligned}$$

D is the distance metric. The main distance metrics used in finding adversarial examples are L_0 , L_2 and L_∞ norm.

- L_0 distance: the number of indexes i such that $x'_i \neq x_i$.
- L_2 distance: the standard Euclidean distance between x and x' .
- L_∞ distance: the maximum change among all indexes:

$$L_\infty = \max(|x_1 - x'_1|, \dots, |x_n - x'_n|)$$

3.2.Objective function

Because the objective function is hard to train, the author convert the function to a different but equivalent form:

$$\min \quad D(x, x + \delta) + c \cdot f(x + \delta)$$

$$s. t. \quad x + \delta \in [0, 1]$$

The function f is defined to be ≤ 0 if and only if $C(x) = t$.

The assumption here is that we can find a proper constant c to make the optimal solution to the latter matches the optimal solution to the former.

There are many possible choices for f . With examining on the real dataset, the author choose the follow function:

$$f(x + \delta) = (\max_{i \neq t} (Z(x + \delta)_i) - Z(x + \delta)_t)^+$$

3.3.Box constraints

To ensure the modification yields a valid image, there is a constraint on δ : we must have

$0 \leq x_i + \delta_i \leq 1$. This is called "box constraint" in the field of optimization.

The normal gradient descent algorithm can meet the requirement. Previously used solution is to modify the gradient descent. But these modifications work badly in the author's method.

To train the model with unmodified gradient descent, the author choose to change the variable used to train the model. Specifically, instead of optimizing the variable δ , the author defined a new variable w and set:

$$\delta = \frac{1}{2}(\tanh(w) + 1) - x$$





































































































Because \tanh varies from -1 to 1, $x + \delta$ is promised to be within the valid range. Then the standard gradient descent can be used to find the adversarial examples.

3.4. L_2 attack

Putting these ideas together, the L_2 attack can be expressed as:

$$\min \quad \left\| \frac{1}{2}(\tanh(w) + 1) - x \right\|_2^2 + c \cdot f\left(\frac{1}{2}(\tanh(w) + 1)\right)$$

Applying gradient descent to this objective often results in a local minimum. To solve this problem, the author randomly choose a set of starting points which are close to the origin image. The follow image shows the result of the L_2 attack for each source/target pair. We can see that almost all attacks are visually indistinguishable from the origin digit.

		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										





































































































3.5. L_0 attack

Unlike the L_2 attack, the L_0 norm is not differentiable and therefore is ill-suited for gradient descent.

The resolution adopted is to maintain a set of fixed pixels. The set is initially empty. At each step, perform L_2 attack on restricted pixels that are not in the fixed set and then compute the gradient of f with respect to δ . Let's denote it as g . Then, try to select a pixel with a minimum $g_i \cdot \delta_i$ and add it into the fixed set. This means to eliminate the pixel that contribute least to the attack. After a few steps, we can finally find a minimum subset of pixels required to generate an adversarial example.

With a warm-start method, the L_0 attack can be perform as fast as the L_2 attack. The following

image show the result of L_0 result for each of source/target pair.

		Target Classification (L_0)									
		0	1	2	3	4	5	6	7	8	9
Source Classification	0										
	1										
	2										
	3										
	4										
	5										
	6										
	7										
	8										
	9										

3.6. L_∞ attack

As the L_0 attack above, the L_2 attack's objective function is indifferentiable too. Using a gradient descent directly may not get a good effect because δ will oscillate between two different points. The reason for this phenomenon is the L_∞ metric only penalizes the targets δ_i , leaving other δ_j unmodified even though they are big too.

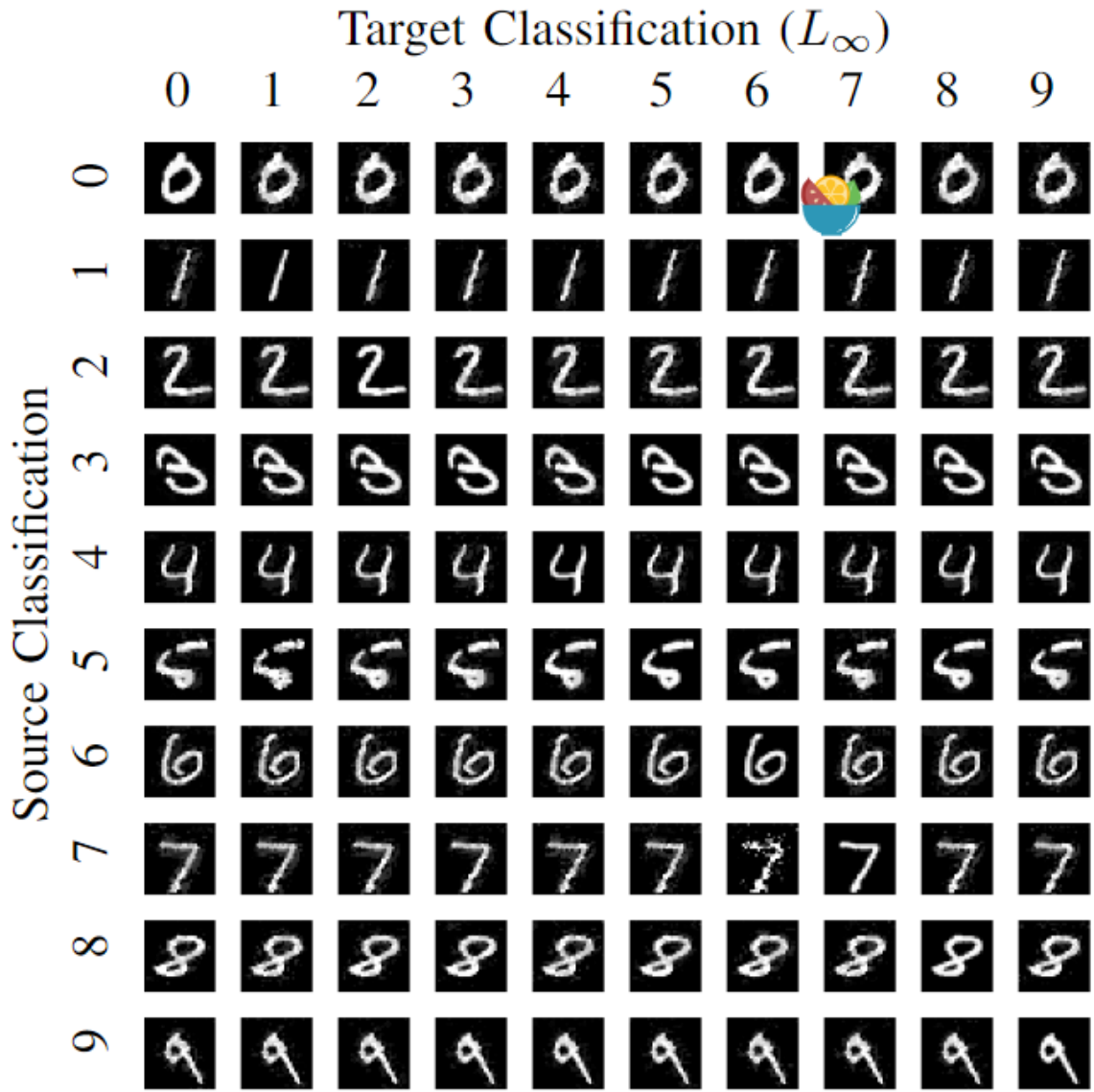
To solve this problem, the author first modify the objective function

$$\min c \cdot f(x + \delta) + \sum_i [(\delta_i - \tau)^+]$$

Using this function, we can penalize all pixels whose intensity is larger than a threshold.

The value of τ is set to 1 initially. Everytime when all δ_i is less than τ , it is reduced by a factor of 0.9.

The attack result is shown below



3.7.Attack evaluation

The author evaluate the three attack algorithms on both original models and models protected with distillation. The datasets used in evaluation are MNIST, CIFAR and ImageNet.

	Best Case				Average Case				Worst Case			
	MNIST		CIFAR		MNIST		CIFAR		MNIST		CIFAR	
	mean	prob	mean	prob	mean	prob	mean	prob	mean	prob	mean	prob
Our L_0	8.5	100%	5.9	100%	16	100%	13	100%	33	100%	24	100%
JSMA-Z	20	100%	20	100%	56	100%	58	100%	180	98%	150	100%
JSMA-F	17	100%	25	100%	45	100%	110	100%	100	100%	240	100%
Our L_2	1.36	100%	0.17	100%	1.76	100%	0.33	100%	2.60	100%	0.51	100%
Deepfool	2.11	100%	0.85	100%	-	-	-	-	-	-	-	-
Our L_∞	0.13	100%	0.0092	100%	0.16	100%	0.013	100%	0.23	100%	0.019	100%
Fast Gradient Sign	0.22	100%	0.015	99%	0.26	42%	0.029	51%	-	0%	0.34	1%
Iterative Gradient Sign	0.14	100%	0.0078	100%	0.19	100%	0.014	100%	0.26	100%	0.023	100%

	Best Case				Average Case				Worst Case			
	MNIST		CIFAR		MNIST		CIFAR		MNIST		CIFAR	
	mean	prob	mean	prob	mean	prob	mean	prob	mean	prob	mean	prob
Our L_0	10	100%	7.4	100%	19	100%	15	100%	36	100%	29	100%
Our L_2	1.7	100%	0.36	100%	2.2	100%	0.60	100%	2.9	100%	0.92	100%
Our L_∞	0.14	100%	0.002	100%	0.18	100%	0.023	100%	0.25	100%	0.038	100%

TABLE VI

COMPARISON OF OUR ATTACKS WHEN APPLIED TO DEFENSIVELY DISTILLED NETWORKS. COMPARE TO TABLE IV FOR UNDISTILLED NETWORKS.

The results have shown that:

1. On unsecured models, all of the three attacks get a success rate of 100% and the mean distance is fewer when comparing to previous method.
2. On models protected by defensive distillation, all previous proposed methods fail to find a adversarial example. However the success rate of algorithms in this paper generate adversarial examples without any defeat.

3.8.Transferability

Transferability refers to the phenomenon that an adversarial example for a model will often be an adversarial example for a different model, even the two models are different and trained with different datasets.

So defense methods must prove they can cut off this transferability. The author trains a standard model, and use the adversarial examples for the standard model to attack other two models. One of them is another standard model and the other is a model protected by defensive distillation. The author also modify the loss function f :

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

the constant k is used to depict the confidence of the generated adversarial examples. The bigger the constant is, the stronger that we believe that the examples generated will results in an misclassification.

The result of this experiment shows that for the three methods proposed, a transferability is always feasible on both of the two models. The only difference is that the distilled model need a larger constant k .

4.Strength

1. The three algorithms proposed in this paper obtain a huge advance compared to the previous method. They successfully prove that the state-of-art defensive strategy still has very big limitations.
2. The author demonstrate many concepts at a high level, giving the reader an intuitive understand on the mechanism on the attack and defense methods and why they success/fail.

5.Weakness

1. In this paper, the author constraints his focus only on the computer vision, which is just a part of the deep learning. Other fields such as NLP use models which are very different. What will happen when using this method in these different models is remained for further validation.
2. The speed of these attacks is typically slower that previous methods, which may constraints their application in some real-time scenarios.

(2018 VTC-Fall)Detecting and Mitigating Spoofing Attack against an Automotive Radar

1.Solved problem

- Show the vulnerabilities of PyCRA in the context of automotive radar systems. The method will turn off the rader periodically which can decrease the availability of rader system on autonomous vehicles.

- Propose STCR, a new defense method that uses the spatio-temporal challenge-response scheme.
- Using Matlab to evaluate and compare the accuracy and robustness of PyCRA and STCR.

2.Main idea

The PyCRA uses a method that periodically turn off the radar. The assumption behind it is that the attacker can't discover this behavior and will continuously emit spoofing signals. As a result, if the receiver detects the signal received is higher than a threshold between the turn-off period, this signal is deemed as the attack signal.

However, this method may cause some potential risks in the autonomous driving scenario because the vehicle should detect the obstacles in real time.

The modification the author made is to use a spatio-temporal scheme that transmits radar signal toward randomly selected directions. If the signal received is from a different direction, it's considered to be the attack signal.

3.Highlights worth learning

- The insight to find potential problems in state-of-art attack methods
- Use change in space to substitute the change in time domain.

(2018 CVPR)Robust Physical-World Attacks on Deep Learning Visual Classification

1.Solved problem

- The author introduced Robust Physical Perturbation(RP_2) to generate physical perturbations for physical-world objects to cause the misclassification of neural network models.
- The author proposed an evaluation methodology to study the effectiveness of physical perturbations in both lab and field scenarios.

2.Main Idea

There are two major categories of perturbation attack methods. One is digital adversarial attack which is performed by changing the intensities of pixels, the other is physical adversarial attack which tries to change the real world objects.

The digit-only methods don't work well in real-world scenarios because of the complex environment conditions. So in this paper the author proposed to perform an attack by adding stickers to a real object. The author chose to add black stickers to the stop signs alongside the road.

To achieve this goal, the author first generated adversarial examples with a procedure similar to the digit-only attack but with some consideration on some physical world challenges. Then, a mask is applied to guarantee that all of the spoofed points are located inside the target sign. The author evaluated their method in both lab and field scenarios with different angles and distances to the stop sign and all of these tests show an excellent performance.

3.Highlights worth learning

- The mask used to constraint the generated spoofed points inside the object makes it possible to perturb the real object indeed.

