Towards Evaluating the Robustness of Neural Networks

S&P 17

Nicholas Carlini & David Wagner
University of California, Berkeley

Web Security & Privacy Lab

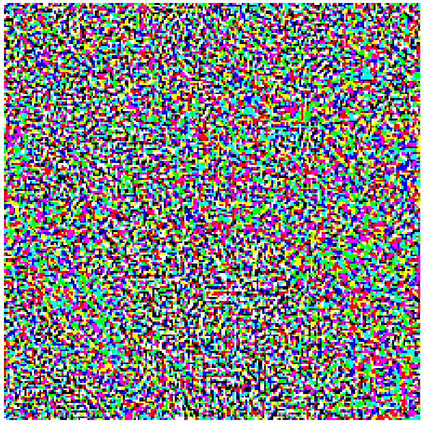
KAIST

What this paper is about?

- Problem
  - **How should we evaluate if a defense to adversarial examples if effective?**
- Contribution
  - Introduce **new attacks** with the three different distance metrics. The new attacks are significantly more effective than previous
  - Propose a way of generating a high-confidence adversarial example that break a defensive distillation

Adversarial Example*

Classified as panda



Small adversarial noise

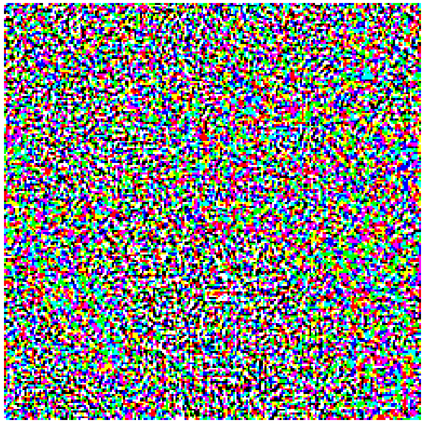
Classified as gibbon

Web Security & Privacy Lab

KAIST

Why should we care?

+



= ?

Small adversarial noise

It could make a big accident for autonomous vehicles

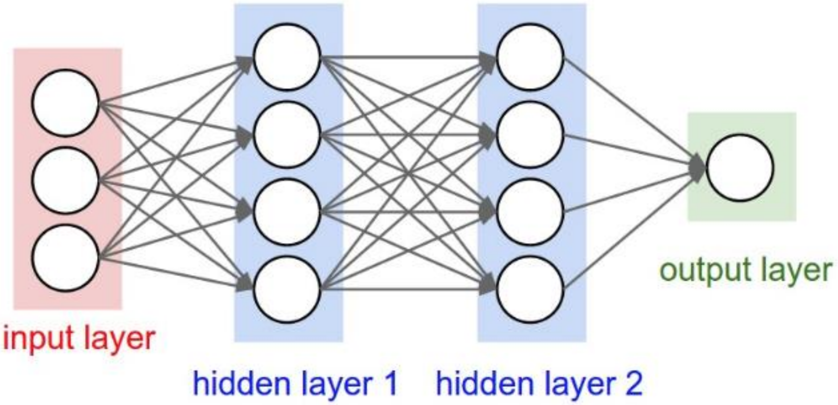
Evaluating Defenses

1) Construct proofs for lower bound on robustness
    1) Very difficult to do precisely in practice, but one can use approximations
2) **Demonstrate attack for upper bound on robustness**
    1) This paper introduces a strong attack and suggest to use it as a benchmark

Threat Model

- Adversary has access to model parameters
- Goal: construct adversarial examples

Original input



warplane

Attack: find a new input (similar to original input) but classified as another class t (untargeted or targeted)

Attacker knows the classifier

bird

Distance Metrics

- The key question in adversarial examples
  - How much distortion we must add to cause a misclassification?
  - Different depending on the domain, The space of images → ***Lp norm***

- Notation
  - L_p $distance = \|x - x'\|_p$, $\|v\|_p = \left(\sum_{i=1}^{n} |v_i|^p\right)^{\frac{1}{p}}$
- Three widely-used distance metrics
  - L_0 $distance$ (ex. $|x_1 - x_1'| + \ldots + |x_n - x_n'|$)
  - L_2 $distance$ (ex. $\sqrt{(x_1 - x_1')^2 + \ldots + (x_n - x_n')^2}$
  - L_∞ $distance$ (ex. $\max(|x_1 - x_1'|, \ldots, |x_n - x_n'|)$

Neural Network

- Neural networks F : **$\boldsymbol{m}$-class classifier**
- The output of the network : **softmax** function
  - The feature of the output vector y : $(0 \leq y_i \leq 1 \; and \; y_1 + \cdots y_m = 1)$
  - y_i is treated as the probability that input x has class i

Existing Attack Algorithms

- Existing attacks for generating adversarial examples
  - L-BFGS
  - Fast Gradient Sign
  - JSMA
  - Deepfool

L-BFGS*

- Given an image x,
  - Find a different image x' (adversarial example) under L_2 *distance*
- They model the problem as a constrained minimization problem
  - Loss function = cross-entropy
  - Perform line search to find the constant c > 0
    - Yields an adversarial example of minimum distance

$$\begin{array}{rl} \text{minimize} & \|x - x'\|_2^2 \\ \text{such that} & C(x') = l \\ & x' \in [0,1]^n \end{array} \longrightarrow \begin{array}{rl} \text{minimize} & c \cdot \|x - x'\|_2^2 + \text{loss}_{F,l}(x') \\ \text{such that} & x' \in [0,1]^n \end{array}$$

Web Security & Privacy Lab

KAIST

Fast Gradient Sign

- Generation of adversarial examples under L_∞ ***distance***
  - Designed primarily to be fast instead of producing close adversarial example
- $x' = x - \epsilon \cdot sign(\nabla loss_{F,t}(x))$
  - ϵ is chosen to be sufficiently small so as to be undetectable, $t = target\ label$
  - Determine in which direction the pixel's intensity based on gradient of loss function
  - Faster rather than optimal

JSMA (Jacobian-based Saliency Map Attack)

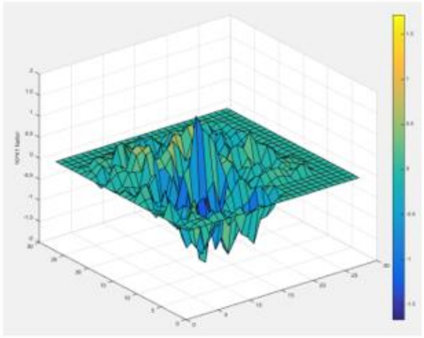
- Generation of adversarial examples under $\boldsymbol{L0}$ $\boldsymbol{distance}$
- Greedy algorithm
  - picks pixels to modify one at a time, increasing target classification on each iteration
  - use gradient $\nabla Z\ x\ l$ to compute a saliency map
    - Modeling of the Impact each pixel has on the resulting classification
    - Large value: increase likelihood of the model labeling the image as the target class l
  - Given the saliency map, JSMA picks the highest value on each iteration

Web Security & Privacy Lab

KAIST

JSMA (Jacobian-based Saliency Map Attack)

Jacobian-based Saliency Map Attack (JSMA)

Compute $\nabla F(x)$ →  → Modify x

Saliency Map

- Stop it
  - When a set threshold of pixels are modified which makes the attack detectable or JSMA succeeds in changing the classification

Deepfool

- Untargeted attack technique under $\boldsymbol{L2\ distance}$
- "It is efficient and produces closer adversarial examples than the L-BFGS"
- Consider that the target neural networks are totally linear
  - The hyperplane separates each class from another → ***False***
  - Take a step towards that solution until a true adversarial example is found

How to train models

- Train two networks for the MNIST and CIFAR-10 classification
- Use one pre-trained network for the ImageNet classification
- Model and training approach are identical for the defensive distillation of previous papers

Layer Type	MNIST Model	CIFAR Model
Convolution + ReLU	3×3×32	3×3×64
Convolution + ReLU	3×3×32	3×3×64
Max Pooling	2×2	2×2
Convolution + ReLU	3×3×64	3×3×128
Convolution + ReLU	3×3×64	3×3×128
Max Pooling	2×2	2×2
Fully Connected + ReLU	200	256
Fully Connected + ReLU	200	256
Softmax	10	10

Model architecture

Parameter	MNIST Model	CIFAR Model
Learning Rate	0.1	0.01 (decay 0.5)
Momentum	0.9	0.9 (decay 0.5)
Delay Rate	-	10 epochs
Dropout	0.5	0.5
Batch Size	128	128
Epochs	50	50

Model parameter

Paper Approach

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

distance between x and x+δ

x+δ is classified as target class *t*

each element of x+δ in [0,1] (for a valid image)

C() is highly non-linear

Web Security & Privacy Lab

KAIST

Approach (Objective function)

Try to find a good f()

$\boldsymbol{f}$ ***such that*** $\boldsymbol{C}(\boldsymbol{x}+\boldsymbol{\delta}) = \mathbf{t}$ if and only if $\boldsymbol{f}(\boldsymbol{x}+\boldsymbol{\delta}) \leq \mathbf{0}$

$s = correct\ classification$ $e^{+} = \max(e, 0)$

$softplus\ (x) = \log(1 + \exp(x))$

$lossF,s\ (x) = cross\ entropy\ loss\ for\ x$

$$f_1(x') = -\text{loss}_{F,t}(x') + 1$$

$$f_2(x') = (\max_{i \neq t}(F(x')_i) - F(x')_t)^+$$

$$f_3(x') = \text{softplus}(\max_{i \neq t}(F(x')_i) - F(x')_t) - \log(2)$$

$$f_4(x') = (0.5 - F(x')_t)^+$$

$$f_5(x') = -\log(2F(x')_t - 2)$$

$$f_6(x') = (\max_{i \neq t}(Z(x')_i) - Z(x')_t)^+$$

f_6 is the best one!

$$f_7(x') = \text{softplus}(\max_{i \neq t}(Z(x')_i) - Z(x')_t) - \log(2)$$

Web Security & Privacy Lab

KAIST

Approach (Objective function)

Initial formulation

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

difficult to solve

$C(x+\delta) = t$ *if and only if* $f(x+\delta) \leq 0$

Replace non-linear constraint with objective function

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

Approach (Box Constraint)

- To ensure the modification yields a valid image, δ $constraint$
  - $0 \leq xi + \delta i \leq 1$ $for\ all\ i$ <= Box constraint
- Three different methods of approaching this problem
  - Projected gradient descent
    - One step of standard gradient descent, clips all the coordinates to be within the box
  - Clipped gradient descent
    - Not clip xi on each iteration, it incorporates the clipping into the objective function
    - $f\ (x + \delta) \rightarrow f\ (\min(\max(x + \delta), 0), 1)$

Web Security & Privacy Lab

KAIST

Approach (Box Constraint)

- Change of variables
  - Use a new variable w, apply a change-of-variables and optimize over w

$$\delta_i = \frac{1}{2}(\tanh(w_i) + 1) - x_i$$

$$-1 \leq \tanh(w_i) \leq 1$$

Approach (Objective function)

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

$$\Big\downarrow \quad \text{Change of variables}$$

$$\begin{aligned} \text{minimize} \quad & \mathcal{D}(x, x+\delta) + c \cdot f(x+\delta) \\ \text{such that} \quad & \frac{1}{2}(\tanh(w_i)+1) \end{aligned}$$

How to choose target class

- Average case
  - Select the target class **uniformly at random** among the labels that are not the correct label
- Best case
  - Perform the attack against all incorrect classes, and report the target class that was **least difficult** to attack
- Worst case
  - Perform the attack against all incorrect classes, and report the target class that was **most difficult** to attack

Evaluation of Approach

- Evaluate the quality of adversarial examples (Random 1,000 instances)
  - each objective function and method to enforce the box constraint
- Worst performing objective function → **cross-entropy loss**

	Best Case						Average Case						Worst Case					
	Change of Variable		Clipped Descent		Projected Descent		Change of Variable		Clipped Descent		Projected Descent		Change of Variable		Clipped Descent		Projected Descent	
	mean	prob	mean	prob	mean	prob	mean	prob	mean	prob	mean	prob	mean	prob	mean	prob	mean	prob
f_1	2.46	100%	2.93	100%	2.31	100%	4.35	100%	5.21	100%	4.11	100%	7.76	100%	9.48	100%	7.37	100%
f_2	4.55	80%	3.97	83%	3.49	83%	3.22	44%	8.99	63%	15.06	74%	2.93	18%	10.22	40%	18.90	53%
f_3	4.54	77%	4.07	81%	3.76	82%	3.47	44%	9.55	63%	15.84	74%	3.09	17%	11.91	41%	24.01	59%
f_4	5.01	86%	6.52	100%	7.53	100%	4.03	55%	7.49	71%	7.60	71%	3.55	24%	4.25	35%	4.10	35%
f_5	1.97	100%	2.20	100%	1.94	100%	3.58	100%	4.20	100%	3.47	100%	6.42	100%	7.86	100%	6.12	100%
f_6	1.94	100%	2.18	100%	1.95	100%	3.47	100%	4.11	100%	3.41	100%	6.03	100%	7.50	100%	5.89	100%
f_7	1.96	100%	2.21	100%	1.94	100%	3.53	100%	4.14	100%	3.43	100%	6.20	100%	7.57	100%	5.94	100%

Web Security & Privacy Lab

KAIST

Approach (Objective function)

$$\text{minimize} \quad \mathcal{D}(x, x+\delta) + c \cdot f(x+\delta)$$
$$\text{such that} \quad \frac{1}{2}(\tanh(w_i) + 1)$$

$\downarrow$

Calculate distance via L0, L2 or L

$$\text{minimize} \quad \|x - x'\|_p$$
$$\text{such that} \quad f(x+\delta) \le 0$$
$$\frac{1}{2}(\tanh(w_i) + 1)$$

Web Security & Privacy Lab

KAIST

Discretization

- In a valid image, pixel intensity must be a (discrete) integer
- This work models pixel intensity as a real number in the range [0,1]
  - With continuous optimization problem
  - Round to the nearest integer $255(x_i + \delta_i)$
- Possible to degrade the quality of the adversarial example
  - If needed, perform greedy search on the lattice defined by the discrete solution

L_2 attack

- Formulation

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

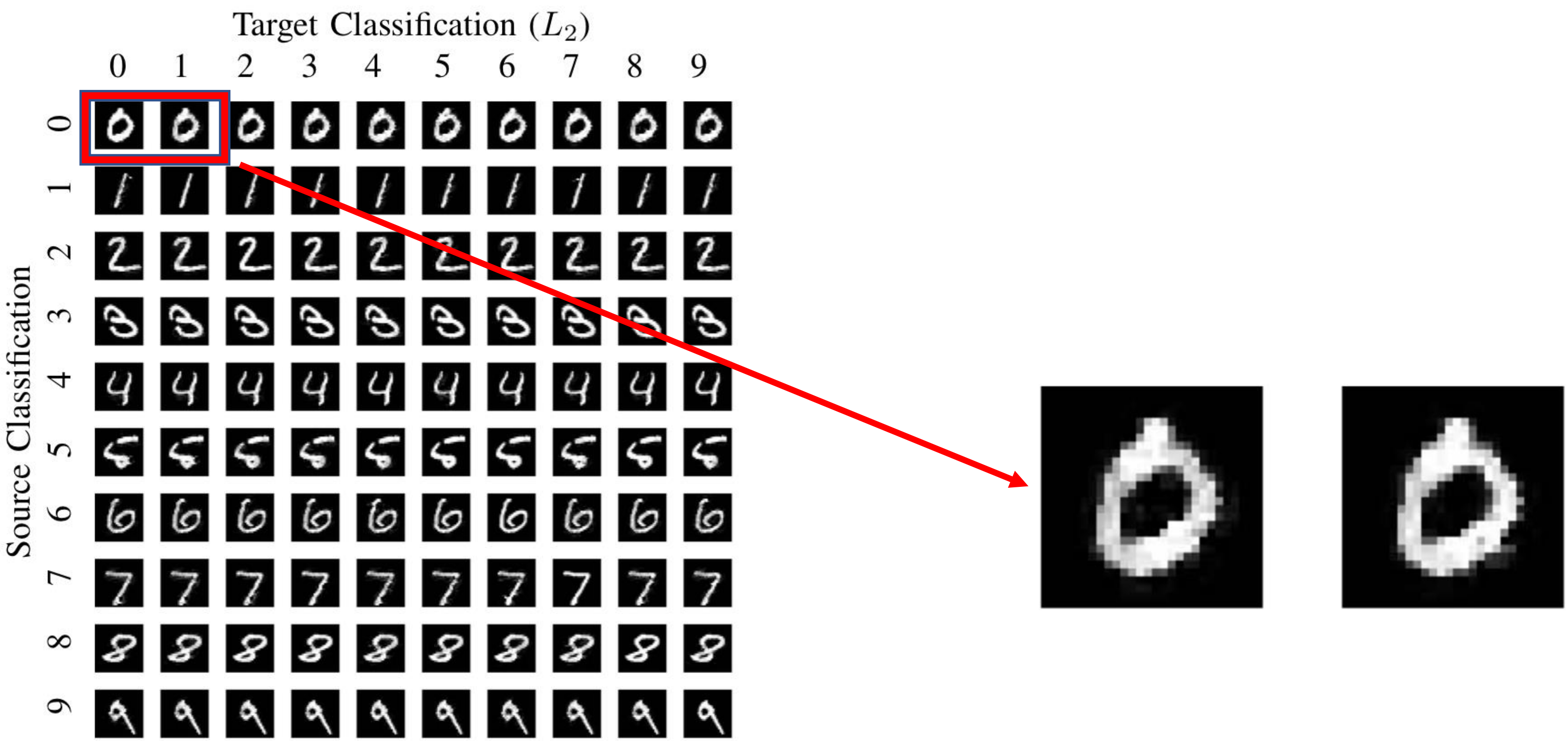
Change of variables $\Big\downarrow$ $\delta_i + x_i = \frac{1}{2}(\tanh w_i + 1)$

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

Optimized with gradient descent

  - f is based on the best objective function found earlier
- Multiple starting-point gradient descent
  - Randomly sample points uniformly from the ball of radius r
- To avoid bad local minimum

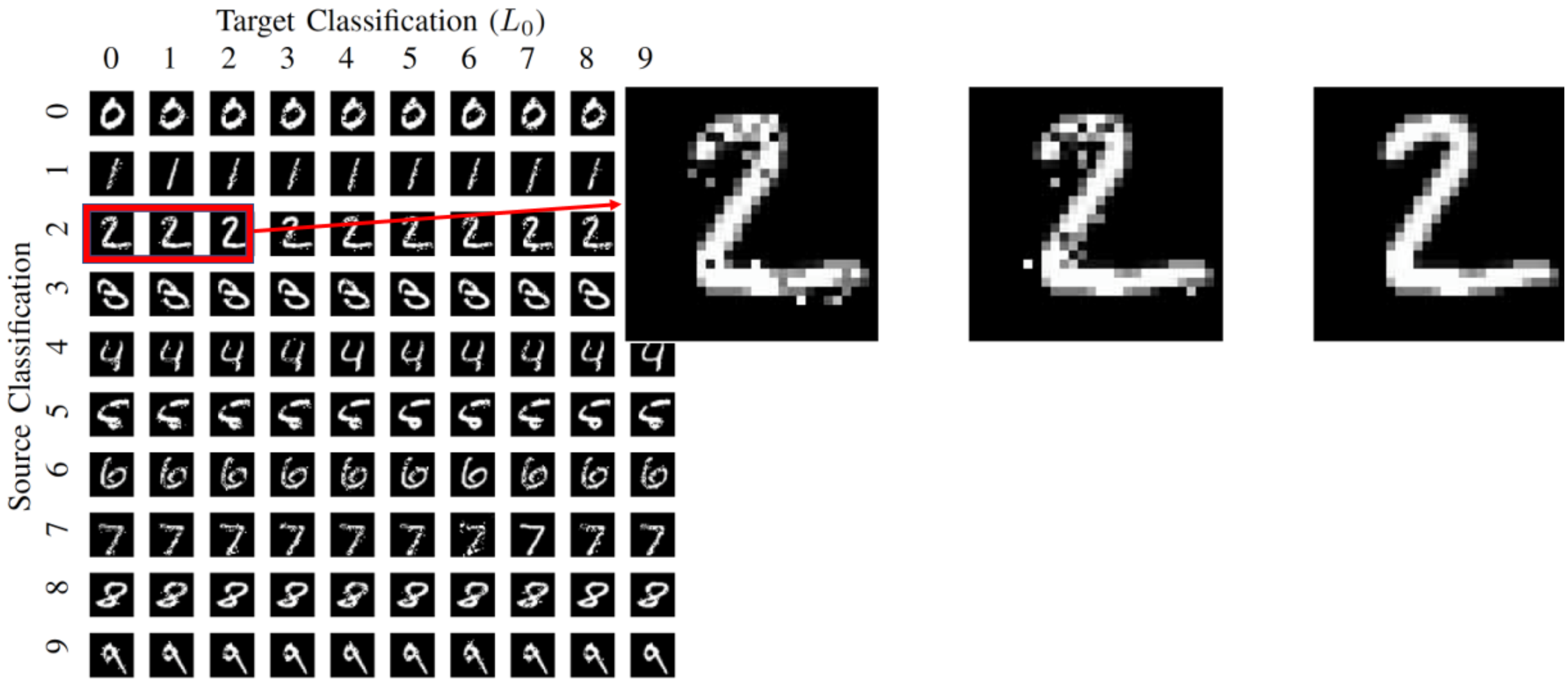
L_2 $attack$ example



L_0 *attack*

- Try to find some pixels don't have much effect on clas sifier output
- Use an iterative algorithm
  - for (each iteration):
    1. identify some pixels that don't have effect on its classification
    2. fix those pixels
    3. if (a minimal subset of pixels that can be modified to generate an adversarial example)
       - break
- In each iteration
  - Use $L_2 attack$ to identify which pixels are unimportant
- L0 distance is non-differentiable

L_0 *attack* example



L_∞ *attack*

- An iterative attack

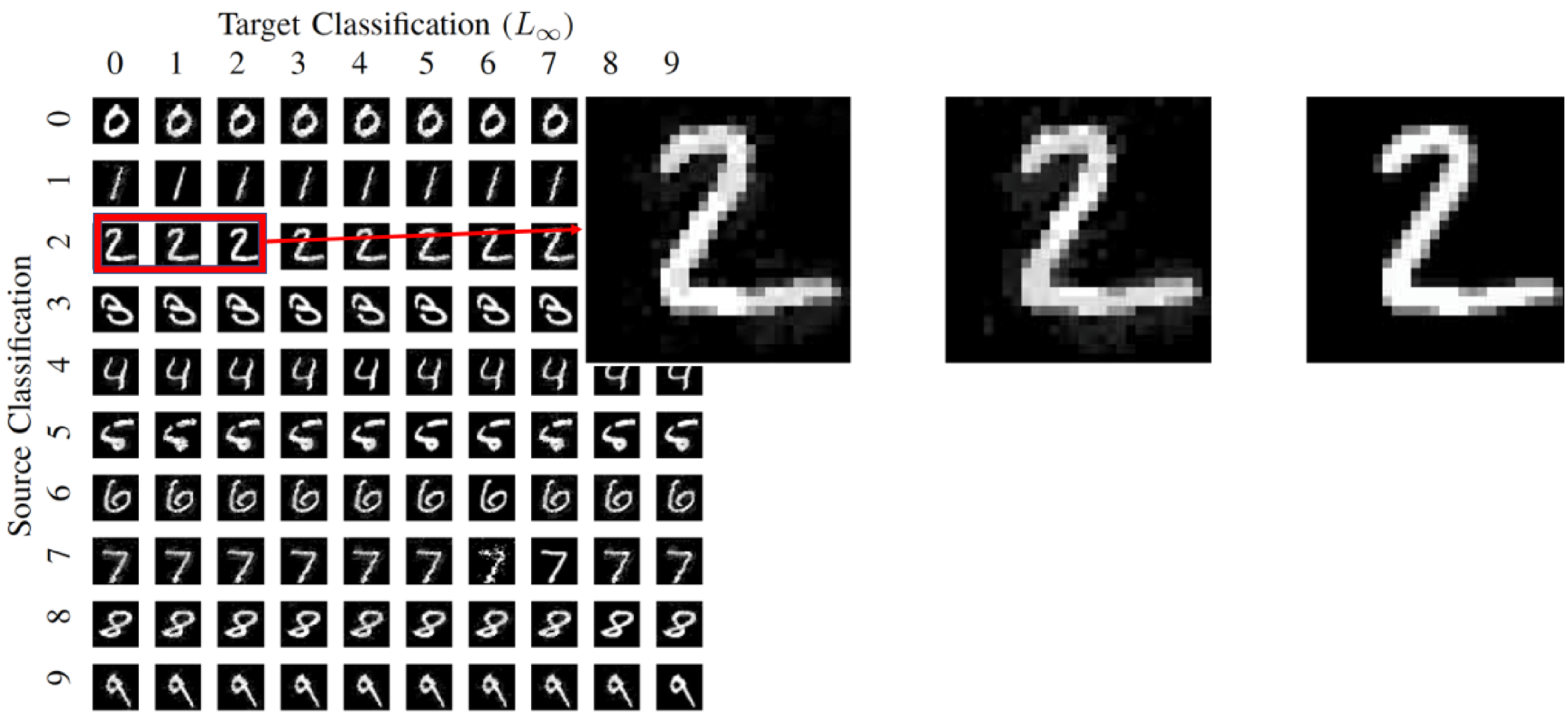
$$\text{minimize} \quad c \cdot f(x+\delta) + \|\delta\|_\infty$$

only penalizes the largest entry

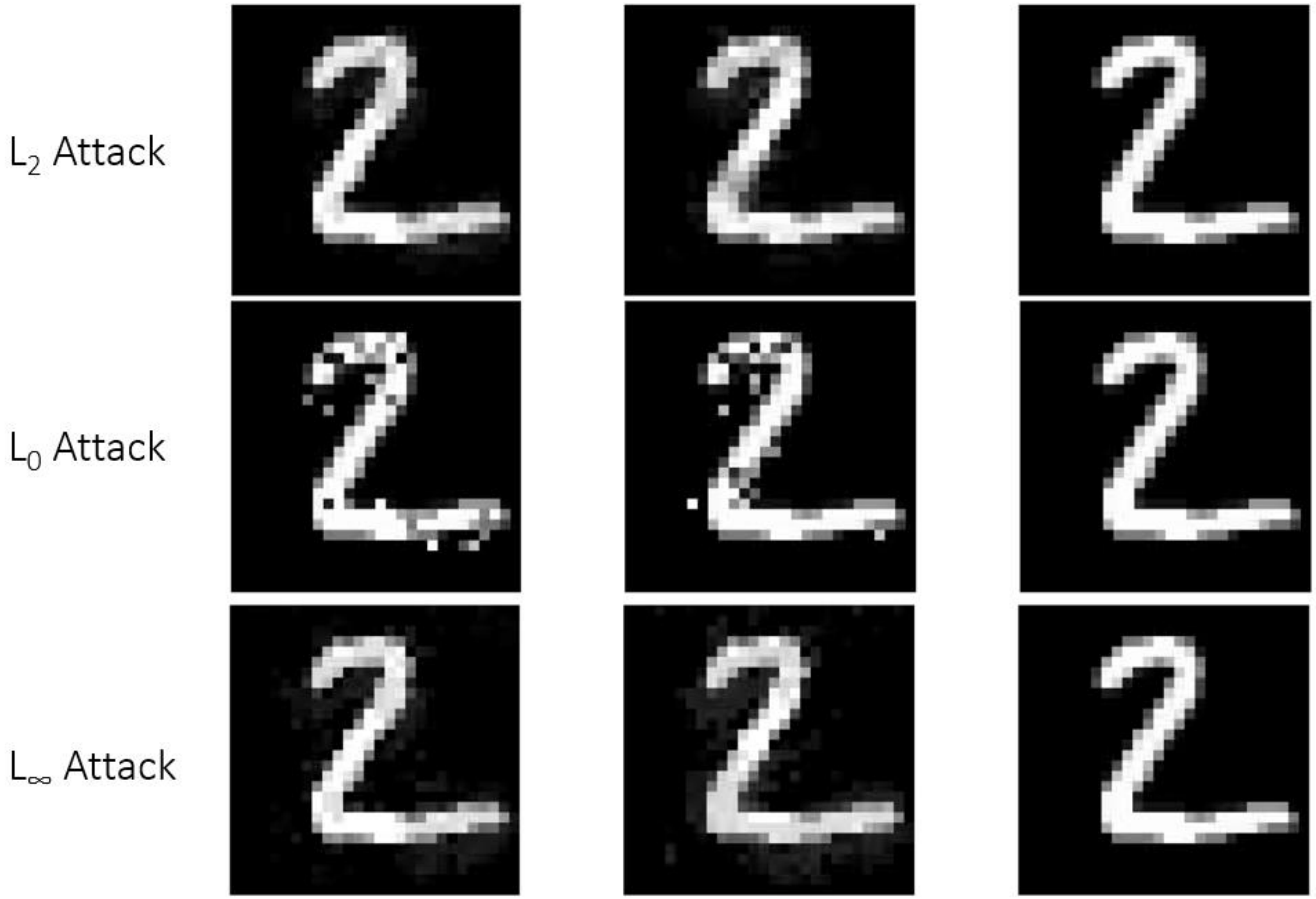
At each iteration, solve $$\text{minimize} \quad c \cdot f(x+\delta) + \cdot \sum_i \left[(\delta_i - \tau)^+\right]$$

τ := τ*0.9 if all δ_i < τ, else terminate the search

L_∞ *attack* example



Compare $L2$, L_0, and L_∞



Attack Evaluation

- Comparison
  - Targeted attacks (Deepfool, fast gradient sign, JSMA)
  - Best results previously reported in prior publications
- The attacks of this paper find **closer** adversarial examples and **never fail to find** adversarial examples

	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%

Attack Evaluation

- Evaluation on ImageNet
- JSMA is not applicable to ImageNet (299x299x3)
- By nature, it requires all combinations of pixel pairs(2^36)

	Untargeted		Average Case		Least Likely	
	mean	prob	mean	prob	mean	prob
Our L_0	48	100%	410	100%	5200	100%
JSMA-Z	-	0%	-	0%	-	0%
JSMA-F	-	0%	-	0%	-	0%
Our L_2	0.32	100%	0.96	100%	2.22	100%
Deepfool	0.91	100%	-	-	-	-
Our L_∞	0.004	100%	0.006	100%	0.01	100%
FGS	0.004	100%	0.064	2%	-	0%
IGS	0.004	100%	0.01	99%	0.03	98%

Defensive Distillation

- Defensive distillation can be applied to any feed-forward neural network and only requires a single re-training step, and is currently one of the only defenses giving strong security guarantees against adversarial examples.

$$\mathrm{softmax}(x, T)_i = \frac{e^{x_i/T}}{\sum_j e^{x_j/T}}$$

Where constant T > 0

Defensive Distillation

- Uses distillation, but with two significant change
  1) Teacher model size = distilled model size
  2) Defensive distillation uses a **large** distillation temperature
- Procedure of Defensive Distillation
  1) (Training phase) Train a network, the teacher network with temperature T
  2) (Training phase) Compute soft labels by apply the teacher network to each instance in the training set
  3) (Training phase) Train a distilled network on the soft labels
  4) (Test phase) run the distilled network with temperature 1

Web Security & Privacy Lab

KAIST

Fragility of Existing Attacks

- Defensive distillation defeats existing attack algorithms and reduces their success probability from 95% to 0.5%
- L-BFGS, Deepfool, Fast Gradient Sign
  - The gradient of $F(\cdot)$ is zero almost always $\rightarrow$ prohibit the use of objective function
  - $L1 norm\ of\ Z(\cdot)$
    - (Undistilled network) mean value : 5.8, standard deviation : 6.4
    - (Distilled network) mean value : 482, standard deviation : 457
  - The value of $Z(\cdot)$ are 100 times larger $\rightarrow$ the output of F becomes ϵ in all components

Applying Our Attacks

- Re-implement defensive distillation on MNIST and CIFAR-10
  - The same model with a previous work
  - Temperature T=100, the value found to be most effective
- Success rate
  - **All** of the previous attacks **fail to find** adversarial examples
  - This work succeeds with **100% success probability** for three distance metrics

	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%

Effect of Temperature

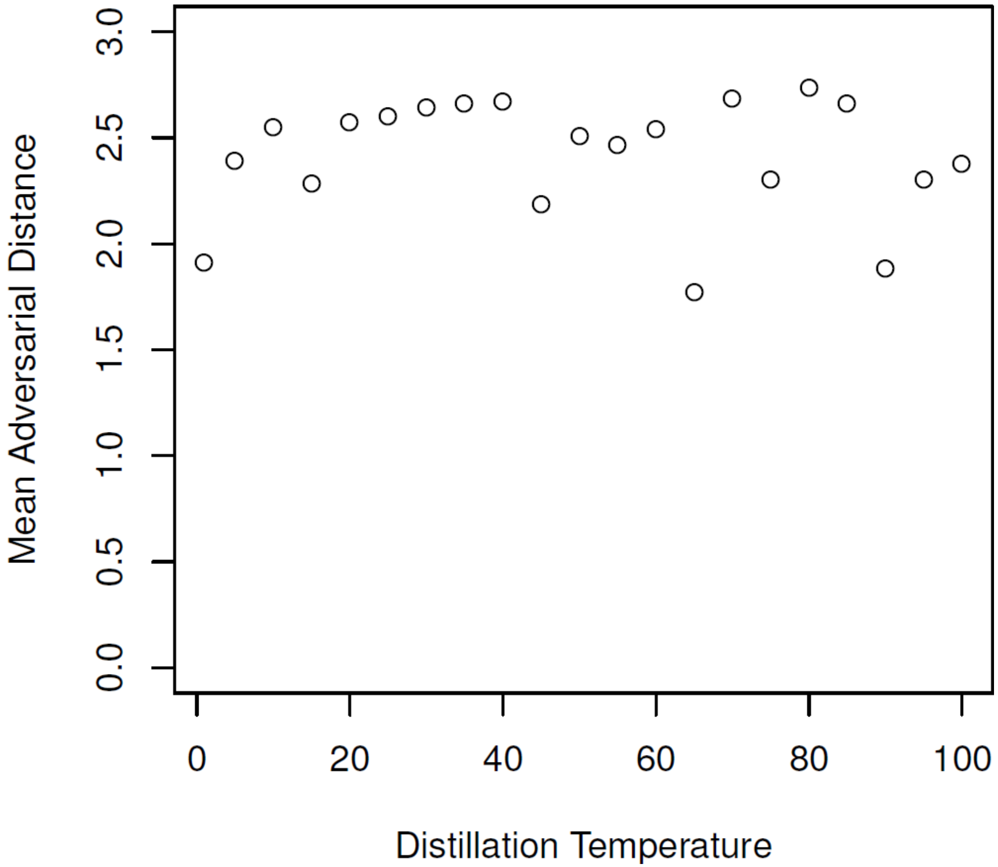
- Previous work*
  - Temperature ↑ means success rate (the creation of adversarial example) ↓
  - T=1 → 91%, T=5 → 24%, and T=100 →0.5%
- This work
  - **No effect** of temperature on the mean distance to adversarial examples
  - → **Increasing the distillation temperature** does not increase the **robustness of the neural network**

Web Security & Privacy Lab

KAIST

Effect of Temperature

Does a high distillation temperature increase the robustness of the network?
No



Conclusion

- How should we evaluate the effectiveness of defenses against adversarial attacks?
- Show robustness against powerful attacks
- This paper shows demonstrate upper bound on robustness
  - Not useful for neural network verification, useful for breaking neural network