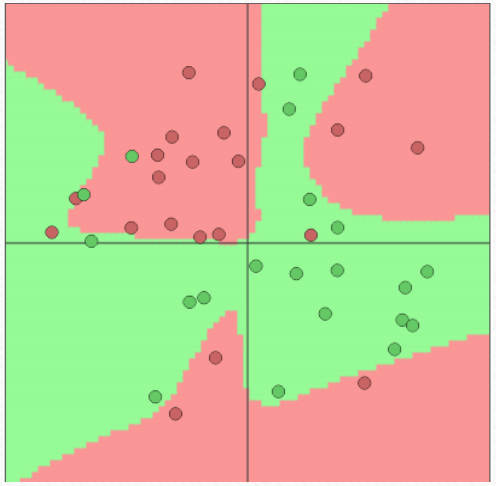
Towards Evaluating the Robustness of Neural Networks

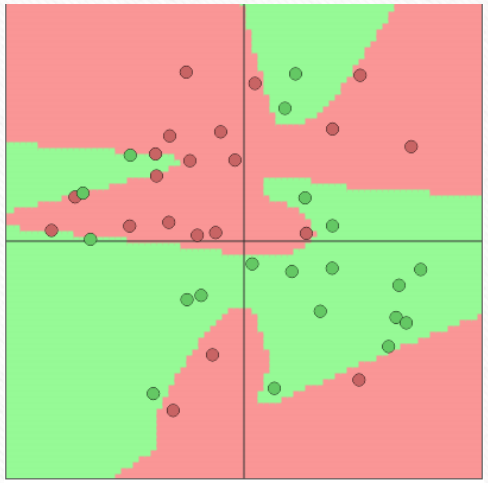
Nicholas Carlini and David Wagner

Slides by Ori Cohen and Shaked Cohen

Recap – What are Neural Networks

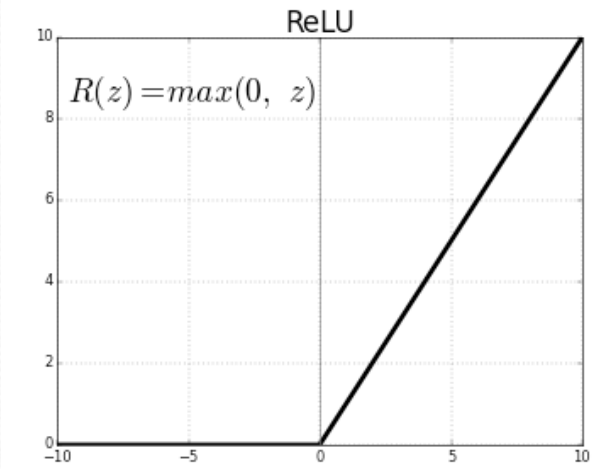
- Neural Network is a **function** $F(x) = y$ with trainable parameters that learns a given mapping

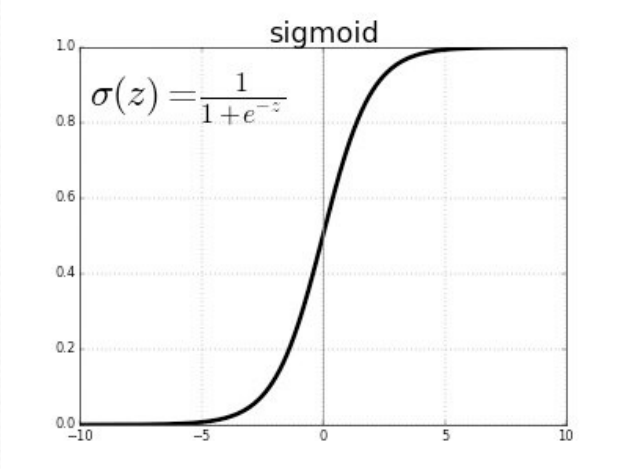




Recap – What are Neural Networks

- We add some non linearity in the form of activation function to be able to represent some more complex functions





Output of (Classification) Neural Network

- Softmax function, ensures that the output vector y satisfies $0 \leq y_i \leq 1$ and $y_1 + \cdots + y_m = 1$

$$\text{Softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

- The output vector y is thus treated as a probability distribution
- y_i is the probability that input x has class i

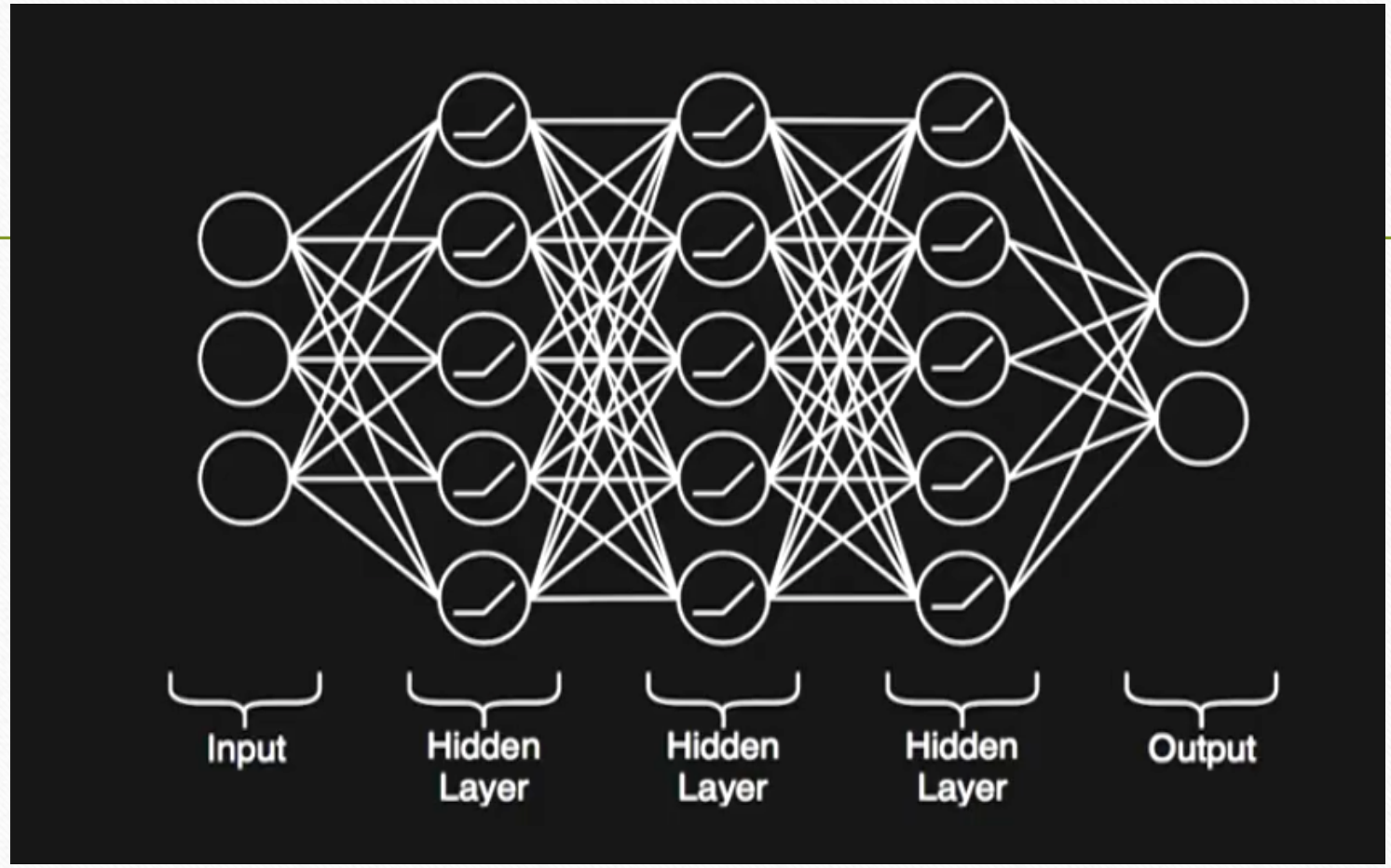
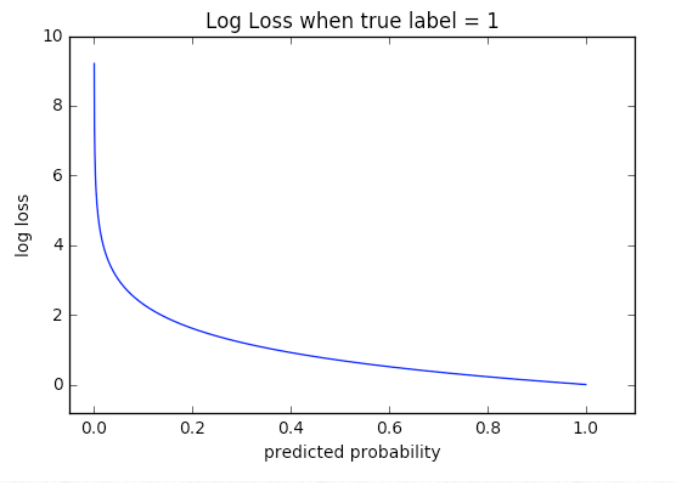


Image taken from Nicholas Carlini talk at IEEE

Loss Function

- Measure how accurate the network is
- Example: **Cross-Entropy**



Evaluating Robustness?

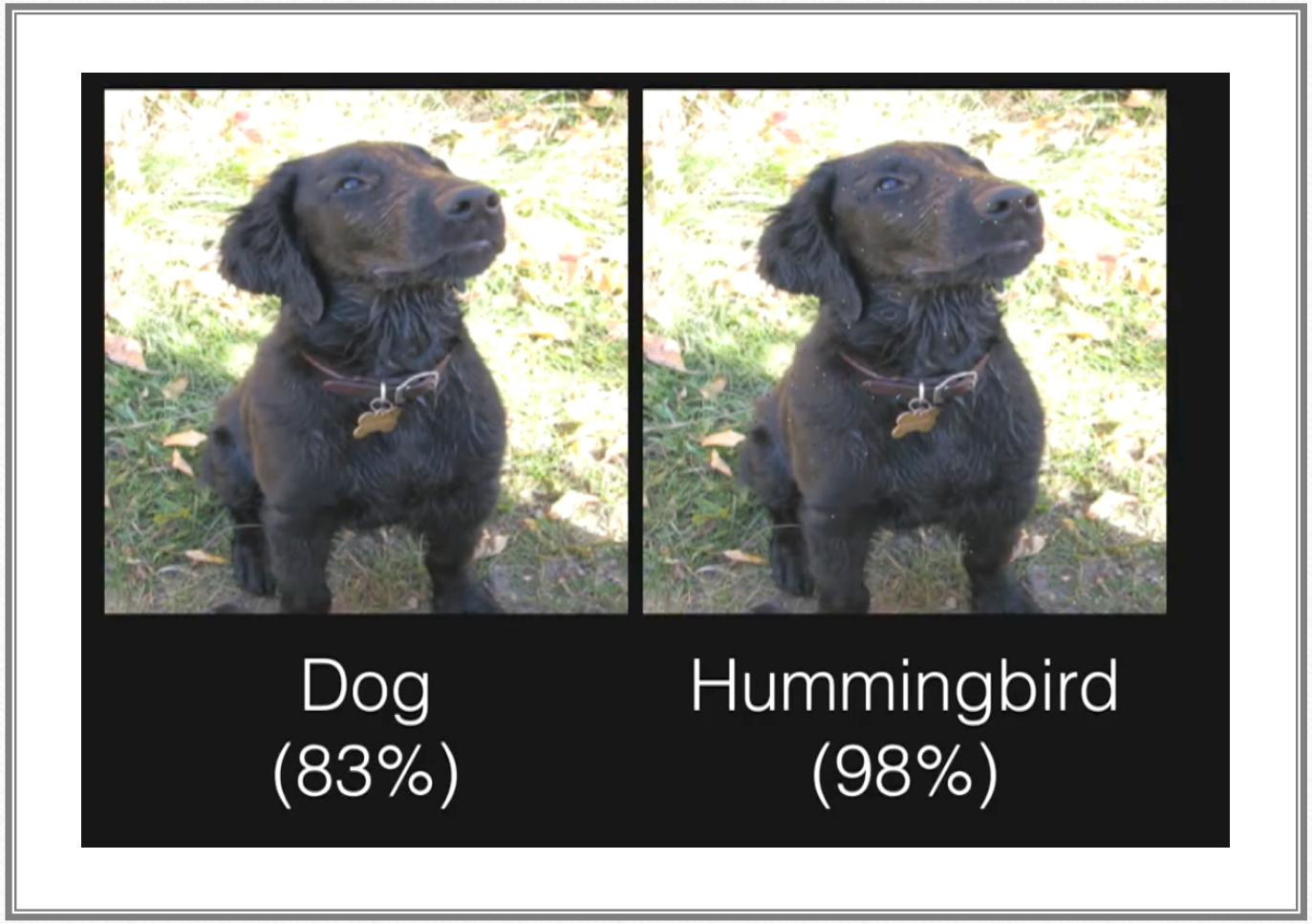
- **Construct a proof of robustness**
- "For any two inputs that are closer than delta, their outputs differ by no more than epsilon…"
- Main limitation – only realistic on networks with hundreds of neurons
- Does not scale to millions of neurons models used in practice

Evaluating Robustness?

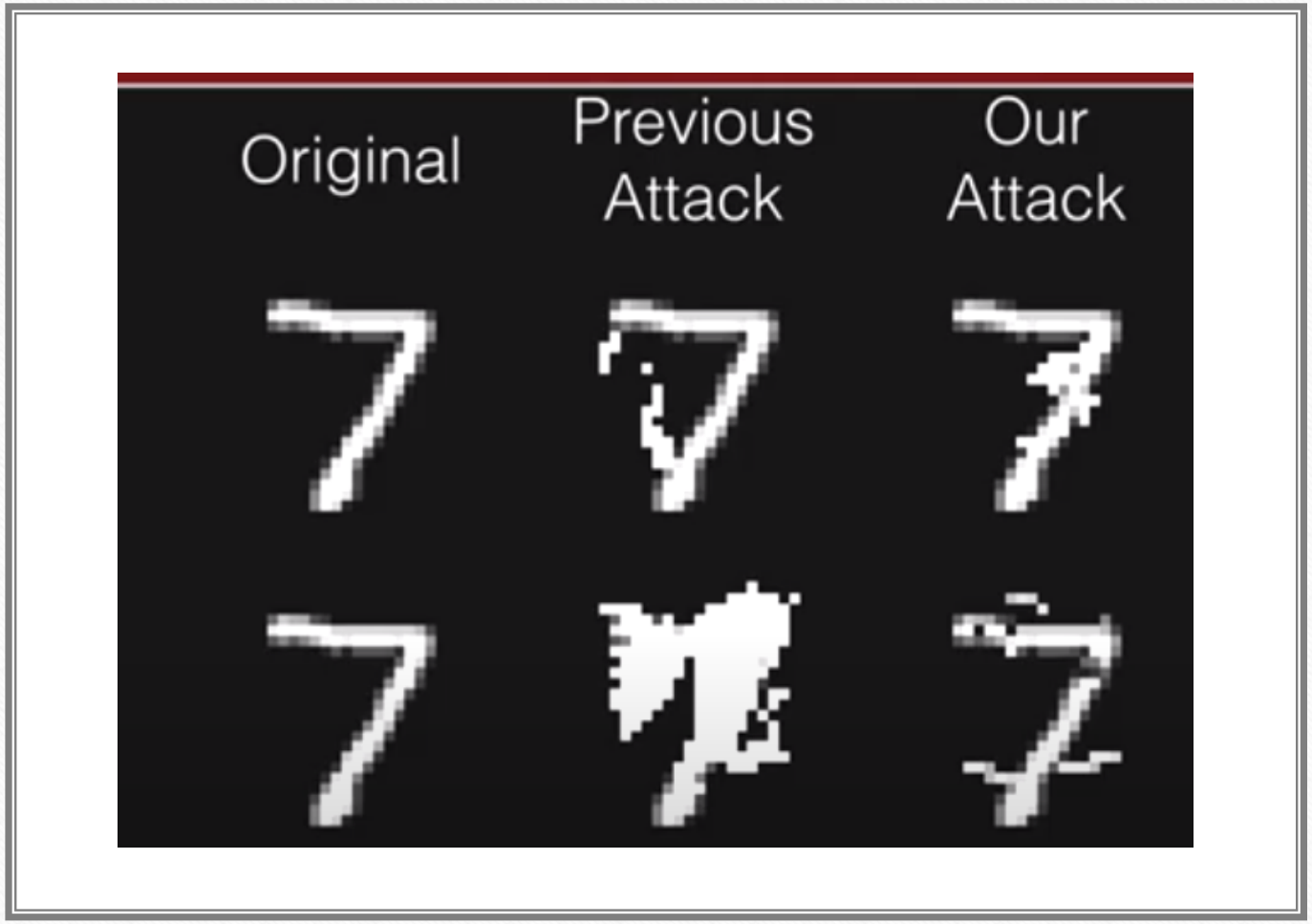
- **Demonstrate constructive attack**
- Easy, but comes with the price of generalization
- The fact that you immune to some kind of “attack” does not imply that you’re robust to other ”stronger” attack

Adversarial Examples

- Transform input x by a small amount and thereby change how x is classified
- Amount of change required can be so small as to be undetectable
- Security implications: self-driving cars, speech recognition etc.



Adversarial Examples



Adversarial Examples

Our Paper

- **Demonstrate constructive attacks**
- construct an upper bound on the robustness of neural networks.
- As a case study, use these attacks to demonstrate that defensive distillation* does not actually eliminate adversarial examples
- Illustrate the general need for better techniques to evaluate the robustness of neural networks

Distance Metrics (1/2)

- In each domain, the distance metric that we use is different
- Prior work suggests L_p norms in the space of images (our domain)
- L normalization over vector v: $\|v\|_p = \left(\sum_{i=1}^{n} |v_i|^p \right)^{\frac{1}{p}}$

Distance Metrics (2/2)

- L_0 - number of pixels that have been altered in an image
- L_2 - standard Euclidean distance: $\|x\|_2 = \sqrt{\sum_{i=1}^{n} |x_i|^2}$
- L_∞ - maximum change to any of the coordinates:

$$\|x - x'\|_\infty = \max(|x_1 - x'_1|, \ldots, |x_n - x'_n|).$$

White Box Assumption

- Adversary has complete access to a neural network
- Realistic assumption in light of prior work
- More on that later (transferability)

The Goal

- Finding an adversarial instance x' for an image x that:

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

- x is fixed, target class t, C classifier
- The goal is to find the smallest δ

Solution?

- Gradient Descent?
- Fails completely due to highly non-linear constraint
- Evaluating the NN in the constraint is difficult

Reformulation (1/3)

Distance: D(x, x') Loss (how far from t): g(x')

- minimize D(x, x') + g(x')
  s.t. x' is "valid"
- g(x') is a loss function on how far C(x') to target t
- if $C(x') = t$ then $g(x') \leq 0$
- if $C(x') \neq t$ then $g(x') > 0$

Reformulation (2/3)

- Example of valid g:
- $g(x') = 1 - C(x')_t$
- If $C(x')$ says $\Pr(t) = 1$ then $g(x') = (1 - C(x')_t) = 1 - 1 = 0$
- If $C(x')$ says $\Pr(t) = 0$ then $g(x') = (1 - C(x')_t) = 1 - 0 = 1$
- The more the classification is "far" from t the higher the loss

Reformulation (3/3)

- $g(x') = c \cdot f\ (x + \delta)$

Distance — Loss (how far from t)

- minimize $D(x, x + \delta) + c \cdot f\ (x + \delta)$
  such that $x + \delta \in [0, 1]$
- Best f empirically chosen

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

Box Constraints

- How to ensure the optimization produces valid image x'?
- Remember: $0 \leq x_i + \delta_i \leq 1$ for all
- Solution: Apply *Change of variables*

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

- Since $-1 \leq \tanh(w_i) \leq 1$, it follows that $0 \leq x_i + \delta_i \leq 1$, so the solution will automatically be valid

Discretization

- We model pixel intensities as a **continuous** real number in the range [0, 1]
- However, in a valid image, each pixel intensity must be a **discrete** integer in the range {0,…,255}
- In practice, solve the continuous optimization problem, and then round to the nearest integer

L2 Attack

- Given x, search for w that solves:

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

Loss (how far from t)

- With f defined as

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

L0 Attack

- L0 is non-differentiable and therefore is ill-suited for standard gradient descent
- instead, we use an iterative algorithm. In short:
- Identifies least ”important” pixels using L2 adversary
- Fixes them so their value will not be changed at the next iterations
- Repeat until identifying **minimal*** subset of important pixels

L_∞ Attack (1/2)

- L_∞ is not fully differentiable, therefore gradient descent does not perform well for it
- Additionally, naively optimizing $\text{minimize} \quad c \cdot f(x+\delta) + \|\delta\|_\infty$ is problematic
- $\|\delta\|_\infty$ term only penalizes the largest entry in δ
- Oscillating between two suboptimal solutions

L_∞ Attack (2/2)

- We resolve this issue using an iterative attack
- Replace the L_∞ term in the objective function with a penalty for any terms that exceed τ (initially 1 and decays)

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

- This loss term penalizes all large values simultaneously

Attack Evaluation

- Evaluate on 1, 000 images in the test set that were initially classified correctly (MNIST, CIFAR-10, ImageNet)
- For each distance metric, across all three datasets, our attacks find closer adversarial examples than the previous state-of-the-art attacks
- Never fails to find an adversarial example
- performs better as the task complexity increases, as opposed to previous methods

Comparison to Previous Work (1/2)

	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%

Comparison to Previous Work ImageNet (2/2)

	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 (1/2)

- Modify Softmax to include *Temperature* constant T

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

- Note softmax(x,T) = softmax(x/T, 1)
- Increasing T results in ”softer” maximum (T->∞ is uniform distribution)
- Decreasing T results in “harder” maximum (T->0 is max)

Defensive Distillation (2/2)

Train a network *N0*, setting the temperature of the softmax to T during the training phase

Evaluate the train data with *N0*, using temperature T, to generate *soft labels*

Train distilled network *N1* on the *soft labels*, using temperature T

To classify new inputs, use *N1* with temperature 1 (regular softmax)

N0 and *N1* are of same shape and architecture

Back to Our Attack

- Our attack performs better than previous attacks on a standard neural network
- Previous attacks fail on a defensively distilled neural network
- Will our attack succeed on a defended network?
- Hell yeah

Defensive Distillation does not Impact Robustness

- In the original work, increasing the temperature was found to consistently reduce attack success rate
- Experiment is re-implemented with the improved attacks
- No effect of temperature over the mean distance to adversarial examples
- Correlation coefficient is $\rho = -0.05$

Transferability (1/2)

- Recent work has shown that an adversarial example for one model will often transfer to be an adversarial on a different model
- i.e. adversarial examples on **neural networks** transfer to **random forests**!
- How can we execute our attack on a blackbox model?

Transferability (1/2)

- Recent work has shown that an adversarial example for one model will often transfer to be an adversarial on a different model
- i.e. adversarial examples on **neural networks** transfer to **random forests**!
- How can we execute our attack on a blackbox model?
- Train your own whitebox model and attack it, and use the generated adversarial examples to attack the blackbox model

Transferability (2/2)

- Hence, if **model A** on classification task C* could be attacked using an adversarial examples on **another model B** on the same task C*…
- What's required from an effective defense?

Transferability (2/2)

- Hence, if **model A** on classification task C* could be attacked using an adversarial examples on **another model B** with the same task C*…
- What's required from an effective defense?
- Any defense that is able to provide robustness against adversarial examples must somehow **break the transferability property**
- Otherwise, we could run our attack on an easy-to-attack model, and then transfer those adversarial examples to the hard-to-attack model

Defensive Distillation does not Impact Robustness #2 (1/2)

- With transferability, we can demonstrate a second break of distillation by transferring attacks from a standard model to a defensively distilled model
- We accomplish this by finding high-confidence adversarial examples

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

- The larger κ, the stronger the classification of the adversarial example

Defensive Distillation does not Impact Robustness #2 (2/2)

- Hypothesis: The stronger the classification on the first model, the more likely it will transfer to the second model
- Experiment shows this hypothesis is empirically true
- Near-100% transferability at κ=20 in unprotected network, κ=40 for defensively distilled network

Conclusion

- It is **still** an open problem to construct defenses that are robust to adversarial examples
- Propose powerful attacks that show defensive distillation does not serve this purpose
- Our attacks more generally can be used to evaluate the efficacy of potential defenses

Full code of the attacks can be found here