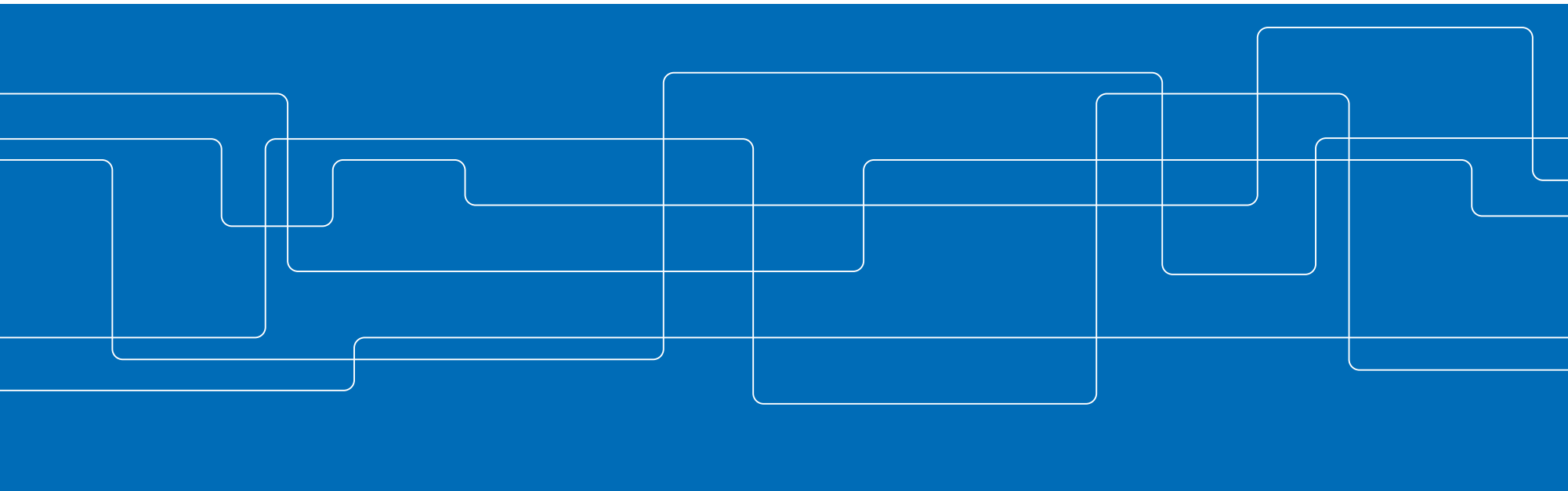
KTH ROYAL INSTITUTE
OF TECHNOLOGY

Towards Evaluating the Robustness of Neural Networks

Authors : Nicholas Carlini and David Wagner

Presenter: Ezgi Korkmaz



Neural Networks and Notations

A neural network is a function $F(x) = y$,
takes input $x \in \mathbb{R}^m$,
produces an output $y \in \mathbb{R}^m$,
where model F depends on model parameters θ.

Focus of the paper m class classifier.

$$0 \leq y_i \leq 1 \qquad y_1 + ... + y_m = 1 \tag{1}$$

The output vector treated as probability distribution,

$$F(x) = softmax(Z(x)) = y \tag{2}$$

where $Z(x) = z$ the output of all layers.

Neural Networks and Notation

Neural network will consists of layers,

$$F = softmax \circ F_n \circ F_{n-1} \circ ... \circ F_1 \tag{3}$$

$$F_i(x) = \sigma(\theta_i \cdot x) + \hat{\theta}_i \tag{4}$$

- σ a nonlinear activation function,
- θ_i a matrix of model weights,
- $\hat{\theta}_i$ a vector of model biases,

σ could be tanh, sigmoid, ReLU or ELU
Paper focused on ReLU.
Image classification is primary evaluation domain.
$h \times x$ pixel-grey-scale image two dimensional vector
$x \in \mathbb{R}^{hw}$ where x_i scaled to be in the range $[0, 1]$
A color RGB 3 dimensional vector $x \in \mathbb{R}^{3hw}$

Neural Networks and Notation

A neural network is a function $F(x) = y$,
takes input $x \in \mathbb{R}^m$,
produces an output $y \in \mathbb{R}^m$,
where model F depends on model parameters θ.

- The output vector y treated as probability distribution
- The element y_i the probability the input x has class i
- Classifier assigns the label $C(x) = argmax_i F(x)_i$
- $C^*(x)$ is the correct label

L-BFGS

Given an image x, find an image x' labeled differently.
Model as constrained minimization problem.

$$\begin{aligned} \text{minimize} \quad & ||x - x'||_2^2 \\ \text{subject to} \quad & C(x') = l \\ & x' \in [0, 1]^n \end{aligned}$$

Can be very difficult to solve,

$$\begin{aligned} \text{minimize} \quad & c \cdot ||x - x'||_2^2 + loss_{F,l}(x') \\ \text{subject to} \quad & x' \in [0, 1]^n \end{aligned}$$

L-BFGS

$$\begin{aligned} \text{minimize} \quad & c \cdot ||x - x'||_2^2 + loss_{F,l}(x') \\ \text{subject to} \quad & x' \in [0, 1]^n \end{aligned}$$

where $loss_{F,l}$ is a function mapping an image to a positive real number.
The common loss function is cross entropy.
Perform line search to find constant $c > 0$
Solve this optimization problem for multiple values of c

FGSM & IFGSM

It is optimized for L_∞ distance metric,

$$x' = x - \epsilon \cdot sign(\nabla loss_{F,t}(x)) \tag{5}$$

Designed to be fast instead of optimal.
IFGSM instead of taking single step of size ϵ in the direction of gradient sign multiple smaller steps α taken and clip by ϵ.

$$x'_i = x_{i-1} - clip_\epsilon(\alpha \cdot sign(\nabla loss_{F,t}(x'_{i-1})) \tag{6}$$

Superior results compared to FGSM.

Carlini Wagner

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

where D is some distance metric.
This is hard to solve because $C(x+\delta) = t$ is nonlinear.
Define an objective function f such that $C(x+\delta) = t$
if and only if $f(x+\delta) \leq 0$

Carlini Wagner

Many possible choices for f :

- $f_1(x') = -loss_{F,t}(x') + 1$
- $f_2(x') = (max_{i \neq t} F(x')_i) - F(x')_t)^+$
- $f_3(x') = softplus(max_{i \neq t} F(x')_i) - F(x')_t) - log(2)$
- $f_4(x') = (0.5 - F(x')_t)$
- $f_5(x') = -log(2F(x')_t - 2)$
- $f_6(x') = -(max_{i \neq t}(Z(x')_i) - Z(x')_t)^+$

$(e)^+$ is $max(e, 0)$
$softplus(x) = log(1 + exp(x))$
$loss_{F,s}(x)$ cross entropy loss for x
s is the correct label.

Carlini-Wagner

Instead of formulating as,

$$\begin{aligned} \text{minimize} \quad & D(x, x+\delta) \\ \text{subject to} \quad & f(x+\delta) \leq t \\ & x+\delta \in [0,1]^n \end{aligned}$$

Formulate as,

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

where c is a suitable chosen constant

Box Constraints

To ensure the modification yields a valid image, constrain on δ
$0 \leq x_i + \delta_i \leq 1$

- Projected gradient descent
  - Perform one step gradient descent then clip within the box
- Clipped gradient descent
  - Does not clip x_i on each iteration
  - Replace $f(x + \delta)$ with $f(min(max(x + \delta, 0), 1))$
- Change of variables
  - Change of variables, optimize over w
  - $\delta_i = \frac{1}{2}(tanh(w_i) + 1) - x_i$
  - Since $-1 \leq tanh(w_i) \leq +1$ it will $0 \leq x_i + \delta_i \leq 1$

These steps will allow using the Adam optimizer

Discretization

- Pixel intensities modeled as continuous real number in the range [0,1]
- In a valid image pixel intensity is a discrete integer {0,1,...,225}
- In practice the integrality constraints are ignored, continuous optimization problem solved and rounded to the nearest integer.
- The intensity of ith pixel becomes $[255(x_i + \delta_i)]$
- Discretization is not important in FGSM cause the perturbation is bigger
- It is important in CW

L_2 Attack

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

where f defined as,

$$f(x') = max(max\{Z(x')_i : i \neq t\} - Z(x')_t - \kappa \qquad (7)$$

Misclassification occurs by adjusting κ
The parameter κ allows to control the desired confidence.

L_0 Attack

- L_0 metric is non-differentiable, ill-suited for standard gradient descent.
- Iterative algorithm
  - Modify the pixels in the allowed set δ be the solution of L_2
  - Compute the gradient of the objective function at the adversarial instance $g = \nabla f(x + \delta)$
  - Select the pixel $i = argmin_i g_i \cdot \delta_i$ fix i and remove i from the allowed set
  - g_i tells how much reduction in f we obtain per unit change to ith pixel
  - Multiply this by how much ith has changed
  - Repeat until L_2 fails to find an adversarial example.

L_∞ Attack

L_∞ metric is not fully differentiable, experimented on

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

The gradient descent stuck oscillating between two suboptimal solutions.

- Consider a case $\delta_i = 0.5$ and $\delta_j = 0.5 - \epsilon$
- L_∞ will only penalize δ_i not δ_j where $\frac{\partial}{\partial \delta_j}||\delta||_\infty$ be zero
- On the next iteration $\delta_i = 0.5 - \epsilon'$ and $\delta_j = 0.5 + \epsilon''$
- Will oscillate back and forth $\delta_i = \delta_j = 0.5$

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

After each iteration if $\delta_i < \tau$ for all i reduce τ by factor of 0.9

Evaluation

- L_0 and L_2 attack find adversarial examples with $2\times$ to $10\times$ lower distortion succeed with $\%100$ probability.
- L_∞ is comparable where on ImageNet to any desired label by only flipping the lowest bit of each pixel.

	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%

TABLE IV

COMPARISON OF THE THREE VARIANTS OF TARGETED ATTACK TO PREVIOUS WORK FOR OUR MNIST AND CIFAR MODELS. WHEN SUCCESS RATE IS NOT 100%, THE MEAN IS ONLY OVER SUCCESSES.

Distillation

- Reduce a large model to a smaller model
- First train the teacher model in the training set
- Then use the teacher to label each instance with soft labels
- For instance, the hard label for an image of hand-written 7 will be 7, soft label will be %80 will be 7 %20 will be 1
- Distillation can increase the accuracy on a test set.
- Defensive distillation is a uses distillation to increase the robustness of a neural network.
- Defensive distillation will include a temperature constant to softmax function

$$softmax(x, T)_i = \frac{e^{\frac{x_i}{T}}}{\sum_j e^{\frac{x_j}{T}}} \tag{8}$$

Defensive Distilled Network

Previous attacks fail to find adversarial examples.
CW has %100 success probability for each three distance metric
Distillation has no impact when CW is used.

	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.