Towards Evaluating the Robustness of Neural Networks


Nicholas Carlini  David Wagner
University of California, Berkeley
Adversarial Example

\[ x \]

“panda”
57.7% confidence

\[ + .007 \times \text{sign}(\nabla_x J(\theta, x, y)) \]

“nematode”
8.2% confidence

\[ = \]

\[ x + \epsilon \text{sign}(\nabla_x J(\theta, x, y)) \]

“gibbon”
99.3% confidence
Contents

• Background and Formalization
  • Neural Network
  • Generating Adversarial Examples
  • Distance Metrics: L-2, L-infinity, L-0

• Existing Attacks

• Carlini’s Attacks

• Experiments
  • Dataset & target models: MNIST, CIFAR-10, ImageNet
  • More effective than previous methods
Formalization

• Neural network as a function \( g : X \rightarrow Y \).

\[
g(x) = f_L(f_{L-1}(\ldots((f_1(x))))
\]

(Typically softmax() as last layer.)

• The goal of an adversary in evasion attack
  
  • Given \( x \in X \) and \( g(\cdot) \), find an \( x' \in X \) such that:

  Untargeted: \( g(x) \neq g(x') \land \Delta(x,x') \leq \varepsilon \)

  Or targeted: \( g(x) = l \land \Delta(x,x') \leq \varepsilon \)
Distance Metric

- $L^p$-norm $\|x\|_p = (|x_1|^p + |x_2|^p + \cdots + |x_n|^p)^{\frac{1}{p}}$

  - $L^2$-norm $\|x\|_2 = (x_1^2 + x_2^2 + \cdots + x_n^2)^{\frac{1}{2}}$

  - $L^\infty$-norm $\|x\|_\infty = \max \{|x_1|, |x_2|, \ldots, |x_n|\}$

- $L^0$-“norm”

  $|x_1|^0 + |x_2|^0 + \cdots + |x_n|^0$ (0$^0 = 0$)

- ...

Which one is the best for vision task?
Existing Attacks

- $L^2$ Adversary
  - L-BFGS

- $L^\infty$ Adversary
  - Fast Gradient Sign Method
  - Iterative Gradient Sign Method

- $L^0$ Adversary
  - Jacobian-based Saliency Map Approach
$L^2$ Adversary: L-BFGS

- Straightforward form, difficult to solve directly.
  \[
  \begin{aligned}
  \text{minimize} & \quad \|x - x'\|_2^2 \\
  \text{such that} & \quad C(x') = l \\
  & \quad x' \in [0, 1]^n
  \end{aligned}
  \]

- Revised form
  \[
  \begin{aligned}
  \text{minimize} & \quad c \cdot \|x - x'\|_2^2 + \text{loss}_{F,l}(x') \\
  \text{such that} & \quad x' \in [0, 1]^n
  \end{aligned}
  \]
  - Softmax-cross-entropy loss
  - Using L-BFGS-B as solver, which supports box constraints.
  - Try many values of $c$ to get the minimum $L^2$
$L^\infty$ Adversary: FGSM & iterative

• Fast Gradient Sign Method

$$x' = x - \epsilon \cdot \text{sign}(\nabla \text{loss}_{F,t}(x))$$

• Iterative Gradient Sign Method

$$x'_0 = x$$

$$x'_i = \text{clip}_\epsilon(x'_{i-1} - \alpha \cdot \text{sign}(\nabla \text{loss}_{F,t}(x'_{i-1})))$$
$L^0$ Adversary: JSMA

• Jacobian-based Saliency Map Approach (JSMA)

• Basic idea: find the most influential pixels and change to maximum or minimum

• Iterative algorithm:
  1. If misclassified, terminate.
  2. Calculate the saliency map (Jacobian matrix).
  3. Pick a pair of pixels that will (1) enlarge the score of target label and (2) lower the score on other labels.
  4. Modify the pixel pair to maximum (or minimum) values. Goto 1.
Carlini’s Attacks

• $L^2$ Adversary

• $L^0$ Adversary

• $L^\infty$ Adversary
Carlini’s $L^2$ Adversary

• Using logits-based objective instead of softmax-cross-entropy loss.

\[
C(x + \delta) = t \text{ if and only if } f(x + \delta) \leq 0
\]

\[
f_6(x') = \left( \max_{i \neq t} (Z(x')_i - Z(x')_t) \right)^+
\]

• Handle box constraint by changing variables.

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

• Since $-1 \leq \tanh(w_i) \leq 1$, we have $0 \leq x_i + \delta_i \leq 1$

• More options on optimizers: Adam.
Carlini’s $L^2$ Adversary

• Final form:

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

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

• How to choose $c$?
  • Too large, always gets $f(x^*) \leq 0$, but the L2 distance might be large.
  • Too small, may not get $f(x^*) \leq 0$, attack fails.
  • Binary search!

• Another trick: Multiple starting-point gradient descent.
Carlini’s $L^0$ Adversary

• Find out unimportant pixels and fix the values, iteratively.

• Iterative algorithm:
  1. Run L2 adversary on $x'$ and restore the fixed pixels, terminate if attack fails.
  2. Compute $g = \nabla f(x + \delta)$
  3. Select pixel $i = \arg \min_i g_i \cdot \delta_i$ and fix it. Goto 1.

• How to select $c$ for L2?
  • Search from a very low value until L2 is successful. Double $c$ till threshold.

• Warm-start trick.

• Compared with JSMA
  • Grows Vs. Shrinks an allowed set; Less like salt and pepper perturbations.
Carlini’s $L^\infty$ Adversary

• Naïve form

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

• Only penalize the (single) largest entry, easy to get oscillating.

• Revised form

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

• Reduce tau (x0.9) iteratively if all entries smaller than tau.
• Choose c: the same as L0
• Warm-start iteration.
Experimental Results

• Produce adversarial examples with smaller $L^p$
  • Logits-based objective function instead of loss
  • Handle box constraint by using tanh()
  • Tricks: warm-start search, multi starting points.
  • …

• Effective on Defensive distillation.
  • Bypassing softmax().

• Significantly slower (not suited for adversarial training)
  • $L^0$: 2x – 10x slower than optimized JSMA
  • $L^2$ and $L^\infty$: 10x – 100x slower.
Conclusion

• Improved $L^2$, $L^\infty$ and $L^0$ attack methods.
• Proved defensive distillation is not a good defense.
• Towards evaluation of robustness.