Paper

- **Title:** Towards Evaluating the Robustness of Neural Networks
- **Authors:** Nicholas Carlini, David Wagner
- **arXiv link:** https://arxiv.org/abs/1608.04644

TL;DR

They introduce a new method to generate targeted adversarial examples for a neural network that are powerful enough to even render defensive distillation, the best defense method at the time. The attacks are tailored to three distance metrics - L_0, L_2 and L_∞

Targeted attacks

- Targeted attacks are those attacks that try to generate an adversarial example of a target false label, while untargeted attacks try to generate an adversarial example of any false label.
- Untargeted attacks are less powerful than targeted so they only consider targeted attacks.
- There are three methods to choose the 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

Method

- Formally finding adversarial examples can be written as,

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

  where x is the image, δ is to be determined, $C()$ is a classifier, t is the target label ($t \neq C(x)$) and $\mathcal{D}$ is a distance metric.
- The constraint $C(x+\delta) = t$ is non linear and makes the above problem difficult to solve.

- To solve that problem, introduce an objective function f such that $C(x+\delta) = t$ if and only if $f(x+\delta) \leq 0$. The following f worked best for here:

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

where $x' = x + \delta$ and $Z(x')$ is the output of classifier network just before applying softmax i.e. $Z(x')$ is the logits of x'.

- Now the optimization problem is:

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

where $c > 0$ is a hyperparameter.

- For L_p norm, the above is same as

$$\begin{aligned} &min\ ||\delta||_p + c \cdot f(x+\delta) \\ &\text{such that } x + \delta \in [0,1]^n \end{aligned}$$

- To ensure that $0 \leq x + \delta \leq 1$, the following three methods can be used
  - Projected gradient descent : Clip the coordinates to be within limits at each step. This works poorly for gradient descent approaches that have a complicated update step (for example, those with momentum).
  - Clipped gradient descent : Replace $f(x+\delta)$ with $f(min(max(x+\delta, 0), 1))$. But this can kill gradient updates when $x + \delta$ is out of range.
  - Change of variables : Introduce a variable w and optimize over w.

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

The above ensures $0 \leq x + \delta \leq 1$.

Attacks

1. L_2 attack : To find adversarial image with minimal L_2 distance from input image
   - $$\min_w\ ||\frac{1}{2}(tanh(w)+1) - x||_2^2 + c \cdot f(\frac{1}{2}(tanh(w)+1))$$
   where $f(x') = max(max\{Z(x')_i : i \neq t\} - Z(x')_t, -\kappa)$
   - Here, κ controls the confidence of misclassification.
2. L_0 attack : To find adversarial image with minimal L_0 distance from input image

- Due to non-differentiable L_0 norm, standard gradient descent does not work
- They use an iterative algorithm that, in each iteration, identifies some pixels that don't have much effect on the classifier output and then fixes those pixels, so their value will never be changed. This will get a minimal set of pixels whose values must be changed to change final label.
- Get δ using the L_2 method, compute $g = \nabla f(x + \delta)$, select pixel $i = arg\ min_i g_i \cdot \delta_i$ and add i to the set of pixels that should not change.
- Repeat above until L_2 fails to find an adversarial example.

3. L_∞ attack : To find adversarial image with minimal L_∞ distance from input image
   - As L_∞ is not fully differentiable, gradient descent does not perform well, it ends up affecting only the largest value making no change to other values.
   - So, they use an iterative attack that minimizes the L_∞ norm.
   - At each iteration

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

     where τ is initialized to 1, and at the end of each iteration if $\delta_i < \tau \forall i$, reduce τ by a factor of 0.9, else terminate.