DEFENSE: ADVERSARIAL TRAINING

32

- Modified loss function:

- Augment the adversarial examples into training dataset.

$$\tilde{J}(\theta, x, y) = \alpha J(\theta, x, y) + (1 - \alpha)J(\theta, x + \epsilon sign(\nabla_x J(\theta, x, y)))$$

Algorithm 1 Adversarial training of network N.
Size of the training minibatch is m. Number of adversarial images in the minibatch is k.

```
Randomly initialize network N
repeat
    Read minibatch B = {X^1, ..., X^m} from training set
    Generate k adversarial examples {X^1_adv, ..., X^k_adv} from corresponding
     clean examples {X^1, ..., X^k} using current state of the network N
    Make new minibatch B' = {X^1_adv, ..., X^k_adv, X^{k+1}, ..., X^m}
    Do one training step of network N using minibatch B'
until training converged
```

Kurakin, A., Goodfellow, I., & Bengio, S. (2016). Adversarial machine learning at scale

DEFENSE: ADVERSARIAL TRAINING

- Madry et al. proposed Adversarial Training as saddle point optimisation problem.

$$\min_{\theta} \rho(\theta) \text{ where } \rho(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\max_{\delta\in\mathcal{S}} L(\theta, x+\delta, y)\right]$$

- They proved using Danskin's Theorem that Adversarial Training is solving this saddle point problem.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. (2017). Towards deep learning models resistant to adversarial attacks

DEFENSE: ADVERSARIAL TRAINING

- Modified loss function:

$$\tilde{J}(\theta, x, y) = \alpha J(\theta, x, y) + (1-\alpha) J(\theta, x + \epsilon sign(\nabla_x J(\theta, x, y)))$$

- Augment the adversarial examples into training dataset.

Algorithm 1 Adversarial training of network N.
Size of the training minibatch is m. Number of adversarial images in the minibatch is k.

1: Randomly initialize network N
2: **repeat**
3: Read minibatch $B = \{X^1, \ldots, X^m\}$ from training set
4: Generate k adversarial examples $\{X^1_{adv}, \ldots, X^k_{adv}\}$ from corresponding clean examples $\{X^1, \ldots, X^k\}$ using current state of the network N
5: Make new minibatch $B' = \{X^1_{adv}, \ldots, X^k_{adv}, X^{k+1}, \ldots, X^m\}$
6: Do one training step of network N using minibatch B'
7: **until** training converged

Kurakin, A., Goodfellow, I., & Bengio, S. (2016). Adversarial machine learning at scale