Adversarial Attacks and Defenses in Images, Graphs and Text: A Review

- 2020, 108 citations

I hope I can turn these notes into the *Preliminaries* thesis section

Vulnerable networks

- CNN
- FC DNN
- RNN
- GCN (graph convolutional networks) - used in fraud detection
  - only necessary to change couple of edges

Counter-measures

- Gradient masking
- Robust optimization
- Adversary detection

Deep neural-nets reason differntly -> understanding adversarial attacks should help understand this difference

Definitions and notations

Threat model

Adversary's goal

- Poisoning attack - change the behavior of DNN by modifying/inserting few train examples
  - public honeypot - collection of training data for malware detectors
- evasion attack - craft fake examples classifier cannot recognize
  - targeted
  - untargeted

Adversary's knowledge

- White-box attack - widely studied, easily analyzed mathematically
- Black-box attack - practical
- Semi-white (gray) box attack - train generative model in white-box setting, then use in black-box scenario (TREMBA)

Victim models

- conventional machine learning models - SVM, Naive-Bayes
- DNN - not well understood how they work, studying security necessary
  - FC
  - CNN - sparse version of FC
  - GCN
  - RNN

Security evaluation

Robustness

- **Minimal pertubation** - The smallest pertubation to fool the network
  - $\delta_{min} = \arg\min_{\delta} \|\delta\| \text{ s.t. } F(x+\delta) \neq y$
- **Robustness** - The norm of minimal pertubation on particular example
  - $r(x, F) = \|\delta_{min}\|$

- **Global robustness** - Expectation of robustness over the whole dataset
  - $\rho(F) = \mathop{\mathbb{E}}_{x \sim \mathcal{D}} r(x, F)$

Adversarial risk (loss)

- **Most-adversarial example** - Given classifier F, datapoint x and ϵ ball, x_{adv} is the adversarial example with the largest loss
  - x_{adv} is the point, where the classifier is the most likely to be fooled
  - $x_{adv} = \arg\max_{x'} \mathcal{L}(x\ F)$ s.t. $\|x' - x\| \leq \epsilon$
- **Adversarial loss** - Loss value of the most-adversarial example
  - $\mathcal{L}_{adv}(x) = \mathcal{L}(x_{adv}) = \max_{\|x'-x\|<\epsilon} \mathcal{L}(\theta, x', y)$
- **Global adversarial loss (adversarial risk)** - The expectation of adversarial loss over the data distribution $\mathcal{D}$
  - $\mathcal{R}_{adv}(F) = \mathop{\mathbb{E}}_{x \sim \mathcal{D}} \mathcal{L}_{adv}(x) = \mathop{\mathbb{E}}_{x \sim \mathcal{D}} \max_{\|x'-x\|<\epsilon} \mathcal{L}(\theta, x', y)$

Adversarial risk vs. risk

- The concept of *Adversarial risk* is similar to the definition of classifier risk (empirical risk)
  - $\mathcal{R}(F) = \mathop{\mathbb{E}}_{x \sim \mathcal{D}} \mathcal{L}(\theta, x, y)$
  - Global adversarial risk (loss) is in a sense empirical risk but on the most adversarial examples, low empirical risk doesn’t have to mean low adversarial risk

Generating adversarial examples

Studying adversarial examples in the image domain essential, because: - perceptual similarity between fake and original is intuitive (unlike in other domains - graphs, audio) - imaga data have simple structure

Studied datasets

- MNIST
- CIFAR10
- ImageNet

All attacks summarization table

White-box attacks

Given classifier C (model F), victim sample (x, y), synthesize fake image x', that is perceputally similar to original x, but fools the classifier C - find x' satisfying $\|x' - x\| \leq \epsilon$, such that $C(x') = t \neq y$ - $\|\cdot\|$ usually l_p norm

Biggio’s attack

- ECML 2013
- adversarial examples on MNIST targeting SVMs and 3-layer FC DNNs
- inspired studies on safety of deep learning

Szegedy’s limited-memory BFGS (L-BFGS)

- Dec 2013
- L-BFGS is an optimization algorithm, that leverages estimates of second order partial derivates information
- first to attack image classifiers
- find minimally distorted adversarial example x' by solving:
  - $\min \|x - x'\|_2^2$, s.t. $C(x') = t$ and $x' \in [0, 1]^m$
- loss function:
  - $\min c\|x - x'\|_2^2 + \mathcal{L}(\theta, x', t)$, s.t. $x' \in [0, 1]^m$
- by increasing constant c throughout the optimization, while keeping the adversarial example outside of the correct decision boundary, we can approximately find adversarial example with minimal pertubation

Fast gradient sign method (FGSM)

- Dec 2014, Goodfellow et al.
- **non-target**
  - $x' = x + \epsilon \text{sgn}(\nabla_x \mathcal{L}(\theta, x, y))$
  - maximises loss of correct classification
  - note that $x' = x - \epsilon \text{sgn}(\nabla_x \mathcal{L}(\theta, x, y_{least-likely}))$ is also valid, it's would be the first step of the iterative Least-likely class method
- **target**
  - $x' = x - \epsilon \text{sgn}(\nabla_x \mathcal{L}(\theta, x, t))$
  - minimize loss of target class
- For targeted attack setting, this can be framed as one-step 1-bit-precision gradient descent to solve:
  - $\min \mathcal{L}(\theta, x', t)$ s.t. $\|x' - x\|_\infty \leq \epsilon$ and $x' \in [0,1]^m$
  - resulting x' is vertex (extreme point) of ϵ hypercube around x
- only one backprop, very fast
- used for producing train samples for adversarial training

Deep Fool

- Nov 2015
- hyperplane of decision boundary
  - $f(x) = F(x)_y - F(x)_t = 0$
- they linearize this hyperplane using Taylor expansion
  - $f'(x) \approx f(x_0) + \langle \nabla_x f(x_0), (x - x_0) \rangle = 0$
- they find orthogonal vector ω from x_0 to the hyperplane and move along its direction
- for instance LeNet can be fooled on over 90% test samples with $l_\infty \leq 0.1$

Jacobian-based saliency map attack (JSMA)

- Nov 2015, Papernot et al.
- They go a step backwards in the network, and instead of tracking loss $\mathcal{L}(x)$ they track gradients of all class outputs $\nabla F_j(x)$
- at each step single pixel is pertubed
- it is the one, that:
  - increases $F_t(x)$, must satisfy $\frac{\partial F_t(x)}{x_i} > 0$
  - decreases $\sum\limits_{j \neq t} F_j(x)$

Basic iterative method (BIM) / Projected gradient descent (PGD) attack

- iterative version of **FGSM**
  - $x_0 = x; \quad x^{t+1} = Clip_{x,\epsilon}(x^t + \alpha \text{sgn}(\nabla_x \mathcal{L}(\theta, x^t, y))).$
- $Clip_{x,\epsilon}(x')$ projects x' to the surface of ϵ-neighborhood ball $B_\epsilon(x)$ centered at x
- $B_\epsilon(x) : x' : \|x' - x\|_\infty \leq \epsilon.$
- step size α is set relatively small
- number of iterations set, such that the border can be reached (e.g., $iter = \frac{\epsilon}{\alpha} + 10$)
- **PGD** = BIM + random initialization
- BIM (PGD) searches for the *most-adversarial* example in the l_∞ ball $B_\epsilon(x)$, that is the example most likely to fool the target model

Carlini & Wagner's attack

- attack against "**defensive distilation**"
  - FGSM and L-BFGS not strong-enough against the distilation defense, gradients are orders of magnitude smaller
- They reframe the optimization problem as:
  - $\min \|x - x'\|_2^2 + c \cdot f(x', t)$, s.t. $x' \in [0,1]^m$
  - where $f(x', t) = (max_{i \neq t} Z(x')_i - Z(x')_t)^+$
    - maximizes classification for t and minimizes classification for all other classes
    - so called **"margin loss"**
    - there are many different ways to define valid loss function, but margin loss seems to work the best (probably)
  - only difference in formulation is that L-BFGS uses **cross-entropy** instead of **margin loss**
    - this formulation has a nice property, that when $C(x') = t$, then $f(x', t) = 0$, and the algorithm switches to optimizing only the **distance part of the objective**
    - efficient for finding the minimally distorted adversarial example

- quite very strong attack, useful for benchmarking

Ground truth attack

- attempt to rigorously test DNN robustness, to prove something mathematically
  - attempt to find **"provable strongest attack"**
- **ground truth adversarial example** - the closest adversarial example
- the attack is reframed as a satisfiability decision problem and solved by relevant solver
  - inefficient, doesn't scale to larger networks

Other l_p attacks

- previous attacks - l_2 or l_∞

One-pixel attack

- l_0 **norm** - number of pixels changed
- VGG16 on CIFAR10 can be attacked (63.5% of test samples) by changing only **one pixel**
- demonstrates the poor robustness of DNNs

EAD: Elastic-net Attack (on Deep NNs)

- combines l_1 and l_2 norm
- some strong models robust to l_∞ and l_2 are still vulnerable to l_1

Universal attack

- one universal pertubation, that misleads classifier on large number of test images
- find pertubation δ satisfying:
  1. $\|\delta\|_p < \epsilon$
  2. $\underset{x \sim D(x)}{\mathbb{P}}(C(x+\delta) \neq C(x)) \geq 1 - \sigma$
- the pertubation looks similar TREMBA decoder output
- successfully attack 85% ImageNet test samples under ResNet 152

Spatially transformed attack

- translation, rotation, distortion, while keeping the semantics intact, such that the pertubation invisible to a human

Unrestricted adversarial examples

- not restricted by a fixed l_p metric
- similarly to *Spatiaally transformed attack*, changes to the original image aren't made in the pixel space, but in some other latent space
  - l_p norm can change a lot, while the semantics can remain almost the same
- pretrained **AC-GAN** (auxiliary classifier generative adversarial network)
  - we find z_0 s.t. $\mathcal{G}(z_0) = x$
  - then we search for the adversarial example in the latent space neighbourhood of z_0
  - successful, if $C(\mathcal{G}(z_0 + \delta)) \neq C(\mathcal{G}(z_0))$

Physical world attacks

- stickers on the road surface, on the road signs
- surprisinngly, adversarial images crafted using FGSM (not so much by BIM, it overfits) are quite robust to natural transformations (different viewpoints, lighting, noise, distortion, etc.)

Eykholt's sticker attack on road signs

- use l_1 attack to roughly find salient places for stickers
- use l_2 attack to optimize the color of the stickers
- TODO: Why l_1 and then l_2? What property those attacks have, which is useful here?

Athalye's 3D adversarial object

- the famous real-life 3D printed **turtle-rifle**

- optimize the 3D structure and texture, such that the object is adversarial from any viewpoint, under any lighting, camera distance, rotation, background

Black-box attacks

Substitute model

- different DNNs share similar weaknesses, 2 CNNs trained on slightly different datasets will share a lot of adversarial examples
- we can take a substitute model with similar architecture, trained on similar dataset, use classic white-box approach to craft an adversarial example, and then use this adversarial image to attack the target model
    - if the target is also fooled by the same example as the substitute model, we say that this adversarial example is **transferable**
- the trick is to construct good “replica” synthetic training set
- we can also train a diverse ensemble of different models with different architectures and parameters
    - this further helps transferaability, examples are more general