Adversarial Examples in Machine Learning

Rey Reza Wiyatno

Agenda

Intro

Adversarial Attacks

Real World Adversaries

Adversarial Defenses and Circumvention Strategy

Takeaways

Agenda

Intro

Adversarial Attacks

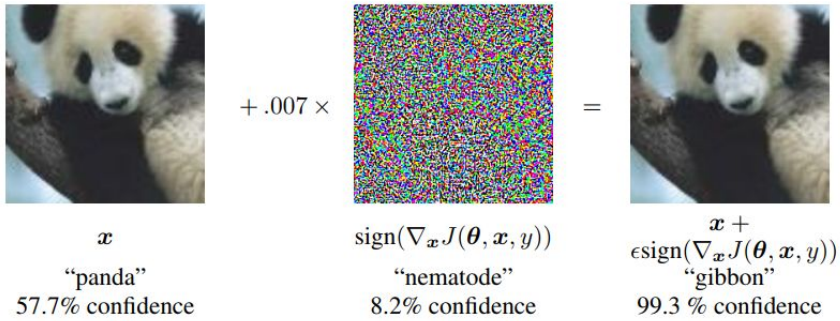
Real World Adversaries

Adversarial Defenses and Circumvention Strategy

Takeaways

Adversarial Examples

Inputs specifically designed to cause a model to make mistakes in its prediction, although they look like valid inputs to a human.



Adversarial example (right) misclassified as "Gibbon" with high confidence (Goodfellow et al., 2015).

Explaining and Harnessing Adversarial Examples (Goodfellow et al., 2015)

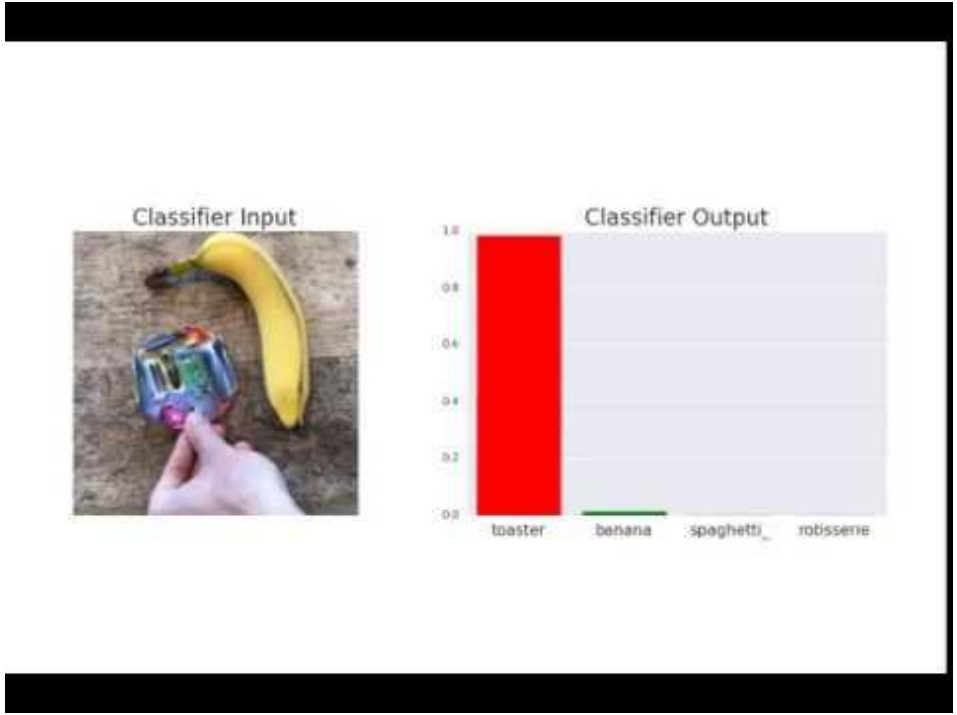
Standard Definition in Classification Task

Generate adversarial example ***x'*** that looks like an original example ***x***, but is mispredicted by a model. Formally:

Given a model ***f*** and an input ***x***, find ***x'*** where $\|x - x'\|_p < \epsilon$, such that $f(x) \neq f(x')$

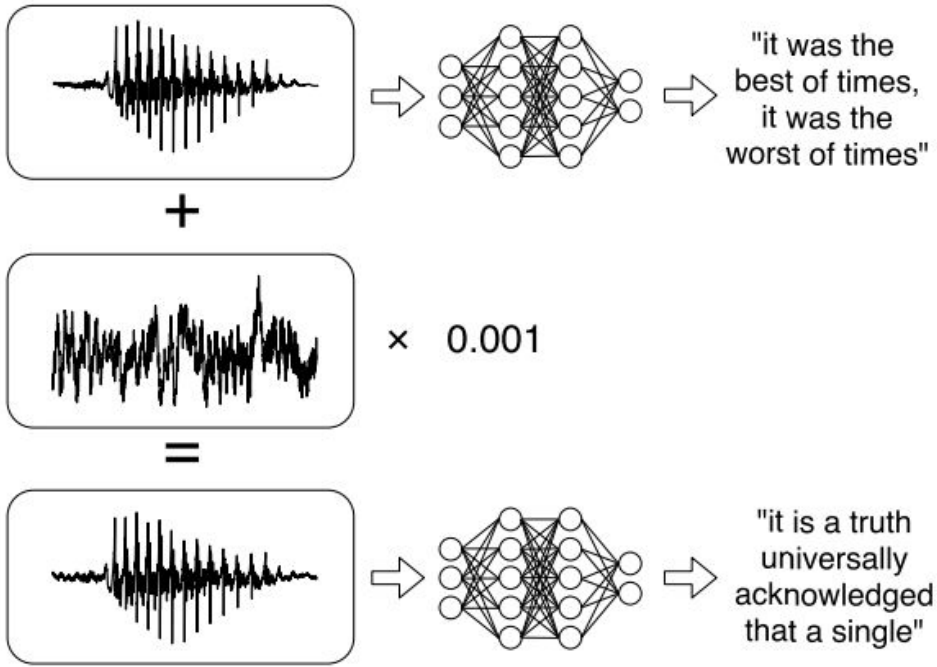
Note: other perceptual similarity metrics other than the L_p norm can also be used.

Adversarial Examples: Adversarial Patch



Adversarial Patch (Brown et al., 2017)

Adversarial Examples: Adversarial Audio



Demo: https://nicholas.carlini.com/code/audio_adversarial_examples/

Audio Adversarial Example: Targeted Attacks on Speech-to-Text (Carlini & Wagner, 2018)

Common Terms

Adversarial attacks: methods to generate adversarial examples.

Adversarial defenses: methods to defend against adversarial examples.

Adversarial robustness: property to resist misclassification of adversarial examples.

Adversarial detection: methods to detect adversarial examples.

Transferability: adversarial examples generated to fool a specific model can also be used to fool other models.

Common Terms

Adversarial perturbation: difference between original example and its adversarial counterpart

Whitebox attack: when attacker have full access to the victim model

Blackbox attack: when attacker only have access to victim's output

Targeted attack: when attacker wants an adversary to be mispredicted in a specific way

Non-targeted attack: when attacker does not care if an example is mispredicted in a specific way

Agenda

Intro

Adversarial Attacks

Real World Adversaries

Adversarial Defenses and Circumvention Strategy

Takeaways

This categorization is only based on methods covered in this presentation and does not encompasses every single attack or defense method that exist as of today.

Attack & Defense Methods

Attack strategies:

Whitebox:

- Direct gradient step(s): FGSM, BIM, R+FGSM, DAG
- Gradient-based greedy algorithm: JSMA
- Iterative optimization: L-BFGS, C&W, UAP, Adversarial Eyeglasses, RP_2, EOT
- Parameterized optimization: ATN

Blackbox:

- Decision boundary approximation: SBA
- Evolutionary algorithm: One Pixel Attack
- Finite difference method: ZOO

Defense strategies:

Data augmentation:

- Adversarial training
- Ensemble adversarial training
- PGD adversarial training

Gradient regularization:

- DCN
- Defensive distillation

Input manipulation:

- PixelDefend

Detection methods:

- Adversary detector networks
- Feature squeezing

Fast Gradient Sign Method (FGSM)

- Single-step **gradient ascent**

$$x' = x + \epsilon \cdot \text{sign}(\nabla_x \mathcal{L}(x, y))$$

- Is above equation for targeted/non-targeted?
- If targeted, how to make it untargeted (and vice-versa)?
- Why use “sign” operator?

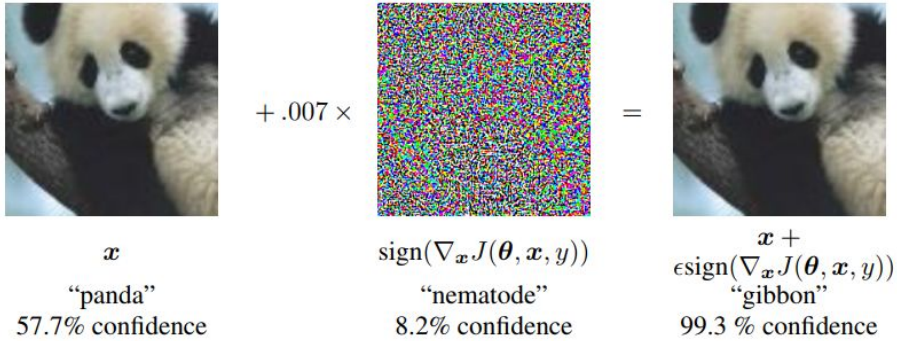
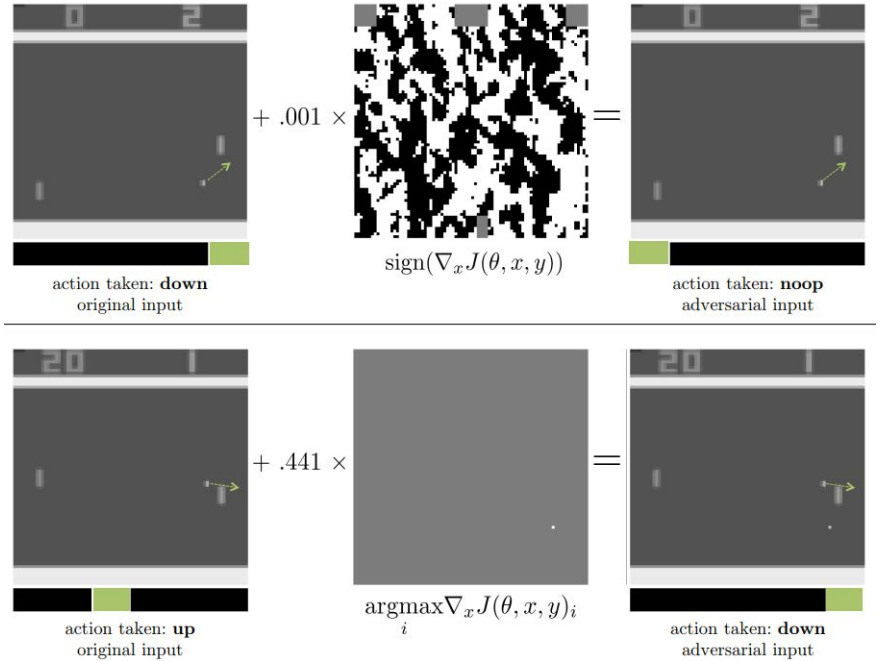


Illustration of FGSM (Goodfellow et al., 2015).

Note: ***ℒ(x,y)*** and ***J(x,y)*** denote the training loss function, ***x*** is an input, ***y*** is the true label of ***x***, ***ϵ*** is a small constant where ***ϵ*** > 0

Explaining and Harnessing Adversarial Examples (Goodfellow et al., 2015)

Fast Gradient Sign Method (FGSM)



Applying FGSM to fool reinforcement learning models (Huang et al., 2017).

Adversarial Attacks on Neural Network Policies (Huang et al., 2017)

Basic Iterative Method (BIM)

- Iterative variant of FGSM

$$\boldsymbol{X}_0^{adv} = \boldsymbol{X}, \quad \boldsymbol{X}_{N+1}^{adv} = Clip_{X,\epsilon}\left\{\boldsymbol{X}_N^{adv} + \alpha\,\mathrm{sign}\left(\nabla_X J(\boldsymbol{X}_N^{adv}, y_{true})\right)\right\}$$

- Targeted attack variant of BIM called Iterative Least-Likely Class Method (ILLCM)

$$\boldsymbol{X}_0^{adv} = \boldsymbol{X}, \quad \boldsymbol{X}_{N+1}^{adv} = Clip_{X,\epsilon}\left\{\boldsymbol{X}_N^{adv} - \alpha\,\mathrm{sign}\left(\nabla_X J(\boldsymbol{X}_N^{adv}, y_{LL})\right)\right\}$$

$$y_{LL} = \arg\min_y \{p(y|\boldsymbol{X})\}$$

$$Clip_{X,\epsilon}\{\boldsymbol{X}'\}(x,y,z) = \min\left\{255, \boldsymbol{X}(x,y,z)+\epsilon, \max\{0, \boldsymbol{X}(x,y,z)-\epsilon, \boldsymbol{X}'(x,y,z)\}\right\}$$

- What is the perceptual similarity metric used here?

Adversarial Examples in the Physical World (Kurakin et al., 2017)
Adversarial Machine Learning at Scale (Kurakin et al., 2017)

Random FGSM (R+FGSM)

- FGSM variant with a random starting point

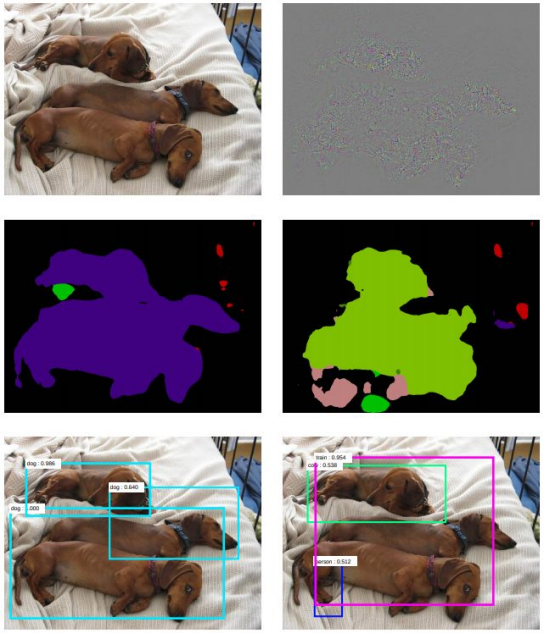
$$x^{\text{adv}} = x' + (\varepsilon - \alpha) \cdot \texttt{sign}\left(\nabla_{x'} J(x', y_{\text{true}})\right), \quad \text{where} \quad x' = x + \alpha \cdot \texttt{sign}(\mathcal{N}(\mathbf{0}^d, \mathbf{I}^d))$$

- Designed to circumvent a defense method called adversarial training (Goodfellow et al., 2015) in its naive implementation
- We will come back to the motivation later in defense section

Ensemble Adversarial Training: Attacks and Defenses (Tramèr et al., 2018)
Explaining and Harnessing Adversarial Examples (Goodfellow et al., 2015)

Dense Adversary Generation (DAG)

- Methods to attack semantic segmentation and object detection models
- High-level explanation: think the attack scenario as if we have model with multiple outputs, and try to misclassify the outputs as much as possible



DAG attack examples
(Xie et al., 2017)

Adversarial Examples for Semantic Segmentation and Object Detection (Xie et al., 2017)

This categorization is only based on methods covered in this presentation and does not encompasses every single attack or defense method that exist as of today.

Attack & Defense Methods

Attack strategies:

Whitebox:

- Direct gradient step(s): FGSM, BIM, R+FGSM, DAG
- Gradient-based greedy algorithm: JSMA
- Iterative optimization: L-BFGS, C&W, UAP, Adversarial Eyeglasses, RP_2, EOT
- Parameterized optimization: ATN

Blackbox:

- Decision boundary approximation: SBA
- Evolutionary algorithm: One Pixel Attack
- Finite difference method: ZOO

Defense strategies:

Data augmentation:

- Adversarial training
- Ensemble adversarial training
- PGD adversarial training

Gradient regularization:

- DCN
- Defensive distillation

Input manipulation:

- PixelDefend

Detection methods:

- Adversary detector networks
- Feature squeezing

Jacobian-based Saliency Map Attack (JSMA)

- Modify pixel values based on saliency map ***S(x,t)*** greedily

$$S(\mathbf{X}, t)[i] = \begin{cases} 0 \text{ if } \frac{\partial \mathbf{F}_t(\mathbf{X})}{\partial \mathbf{X}_i} < 0 \text{ or } \sum_{j \neq t} \frac{\partial \mathbf{F}_j(\mathbf{X})}{\partial \mathbf{X}_i} > 0 \\ \left(\frac{\partial \mathbf{F}_t(\mathbf{X})}{\partial \mathbf{X}_i}\right) \left|\sum_{j \neq t} \frac{\partial \mathbf{F}_j(\mathbf{X})}{\partial \mathbf{X}_i}\right| \text{ otherwise} \end{cases}$$

- **Saturates** 2 pixels (chosen based on some heuristics) per iteration
- **Removes** the saturated pixels from the set of pixels that are allowed to be modified at the end of each iteration
- Perceptual metric: L_0

Note: In the figure, ***F(x)*** and ***ŷ(·)*** are interchangeable and denote the model's prediction, ***x*** denotes a normal input, and ***t*** denotes the target misclassification class

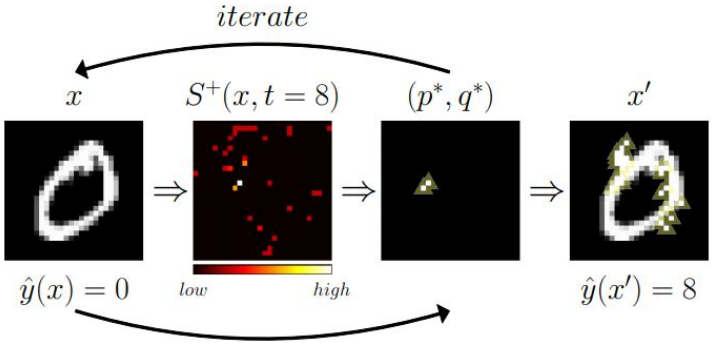
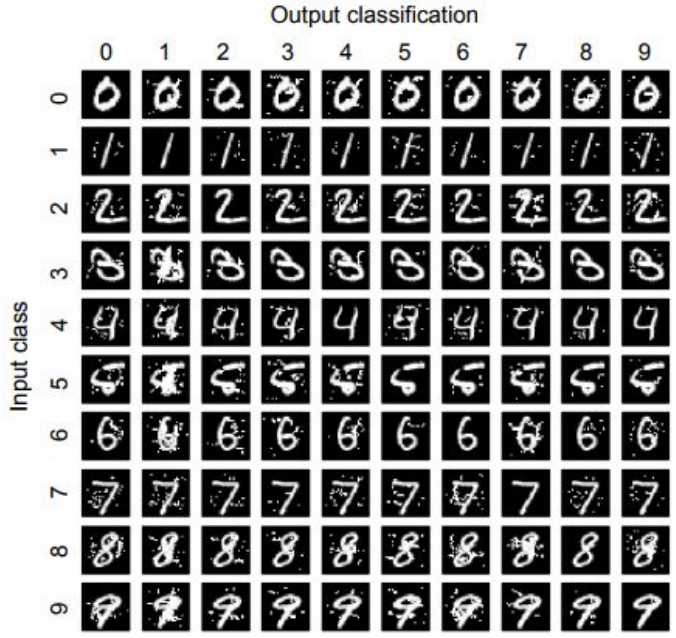


Illustration of JSMA (Wiyatno & Xu, 2018).

The Limitations of Deep Learning in Adversarial Settings (Papernot et al., 2015)
Maximal Jacobian-based Saliency Map Attack (Wiyatno & Xu, 2018)

Jacobian-based Saliency Map Attack (JSMA)



Adversaries generated by JSMA (Papernot et al., 2015).

The Limitations of Deep Learning in Adversarial Settings (Papernot et al., 2015)

This categorization is only based on methods covered in this presentation and does not encompasses every single attack or defense method that exist as of today.

Attack & Defense Methods

Attack strategies:

Whitebox:

- Direct gradient step(s): FGSM, BIM, R+FGSM, DAG
- Gradient-based greedy algorithm: JSMA
- Iterative optimization: L-BFGS, C&W, UAP, Adversarial Eyeglasses, RP_2, EOT
- Parameterized optimization: ATN

Blackbox:

- Decision boundary approximation: SBA
- Evolutionary algorithm: One Pixel Attack
- Finite difference method: ZOO

Defense strategies:

Data augmentation:

- Adversarial training
- Ensemble adversarial training
- PGD adversarial training

Gradient regularization:

- DCN
- Defensive distillation

Input manipulation:

- PixelDefend

Detection methods:

- Adversary detector networks
- Feature squeezing

L-BFGS Attack

- Model adversarial example generation as optimization problem
- Use L-BFGS as optimizer
- Given a victim model ***f***, find ***r*** that minimizes:

$$c|r| + \text{loss}_f(x + r, l)$$

$$\text{subject to } x + r \in [0, 1]^m$$

Adversarial examples generated using this method (Szegedy et al., 2014).

Intriguing Properties of Neural Networks (Szegedy et al., 2014)

Carlini & Wagner (C&W) Attacks

- 3 variants based on similarity metrics: L_0, L_2, L_∞
- Similar to the L-BFGS attack, the problem is modeled as an optimization problem, but using **surrogate loss function**
- ***C&W*** L_2 attack tries to find w that minimizes:

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

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

- What is that objective function trying to achieve?

Note: $Z(x)$ is the logits vector of x, κ is a positive constant to adjust misclassification confidence, t denotes the index of the target misclassification class, e.g. $Z(x)_i$ denotes the i-th element of $Z(x)$.

Towards Evaluating the Robustness of Neural Networks (Carlini & Wagner, 2017)

Universal Adversarial Perturbation (UAP)

Universal perturbations:

Single perturbation that can be added to multiple inputs to make them adversarial

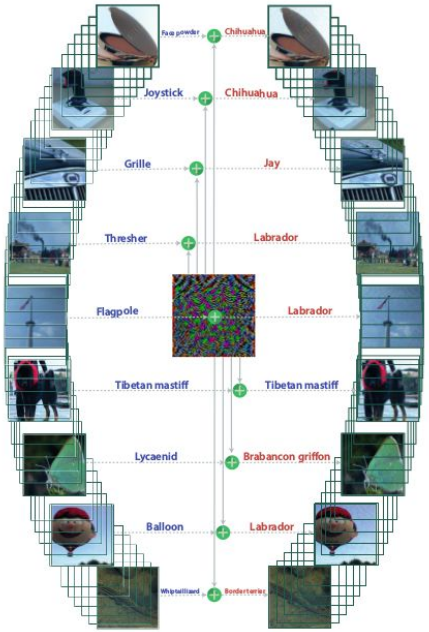


Illustration of UAP (Moosavi-Dezfooli et al., 2016).

Algorithm 1 Computation of universal perturbations.

input: Data points X, classifier $\hat{k}$, desired ℓ_p norm of the perturbation ξ, desired accuracy on perturbed samples δ.
output: Universal perturbation vector v.
Initialize $v \leftarrow 0$.
while $\text{Err}(X_v) \leq 1 - \delta$ **do**
  for each datapoint $x_i \in X$ **do**
    if $\hat{k}(x_i + v) = \hat{k}(x_i)$ **then**
      Compute the *minimal* perturbation that sends $x_i + v$ to the decision boundary:

$$\Delta v_i \leftarrow \arg\min_r \|r\|_2 \text{ s.t. } \hat{k}(x_i + v + r) \neq \hat{k}(x_i).$$

      Update the perturbation:

$$v \leftarrow \mathcal{P}_{p,\xi}(v + \Delta v_i).$$

    end if
  end for
end while

$$\mathcal{P}_{p,\xi}(v) = \arg\min_{v'} \|v - v'\|_2 \text{ subject to } \|v'\|_p \leq \xi$$

This categorization is only based on methods covered in this presentation and does not encompasses every single attack or defense method that exist as of today.

Attack & Defense Methods

Attack strategies:

Whitebox:

- Direct gradient step(s): FGSM, BIM, R+FGSM, DAG
- Gradient-based greedy algorithm: JSMA
- Iterative optimization: L-BFGS, C&W, UAP, Adversarial Eyeglasses, RP_2, EOT
- Parameterized optimization: ATN

Blackbox:

- Decision boundary approximation: SBA
- Evolutionary algorithm: One Pixel Attack
- Finite difference method: ZOO

Defense strategies:

Data augmentation:

- Adversarial training
- Ensemble adversarial training
- PGD adversarial training

Gradient regularization:

- DCN
- Defensive distillation

Input manipulation:

- PixelDefend

Detection methods:

- Adversary detector networks
- Feature squeezing

Adversarial Transformation Network (ATN)

- Train a neural network to generate adversaries
- 2 variants: Adversarial Autoencoder (AAE) and Perturbation ATN (P-ATN)
- AAE: Given a victim model ***f***, train a generator network G_t on dataset ***X*** to output adversarial examples ***X'*** such that ***f(X') = t***, where ***t*** is a target misclassification class
- Every G_t can only be used generate adversarial examples that are misclassified as ***t***

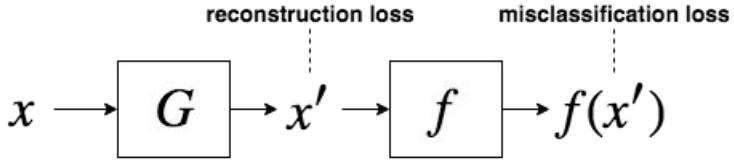
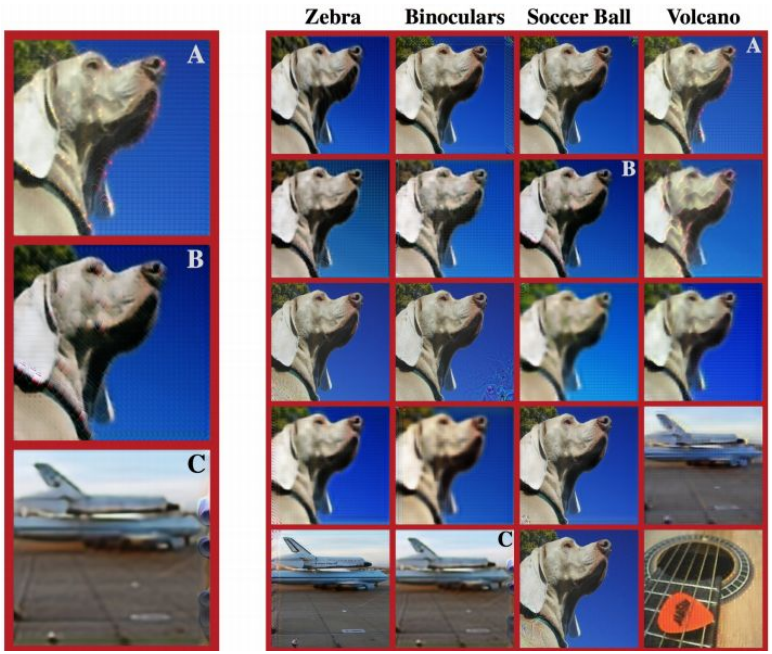


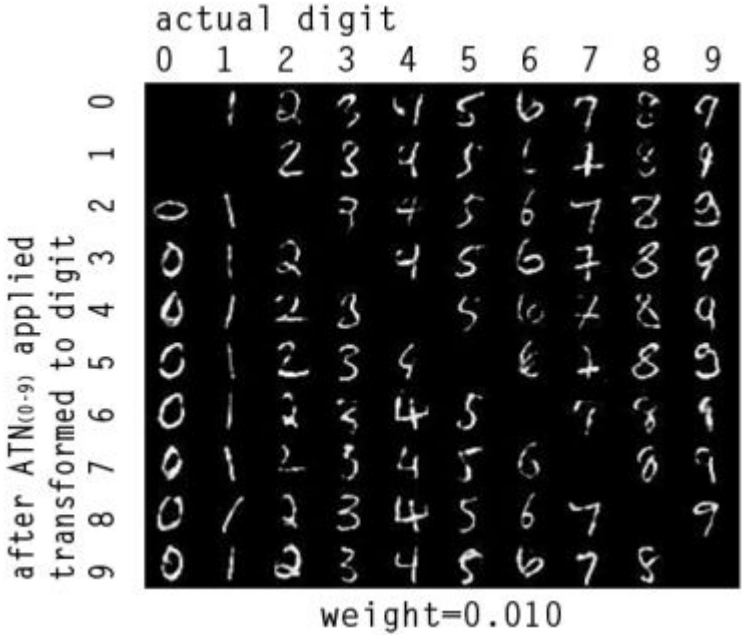
Illustration of AAE.

Adversarial Transformation Networks: Learning to Generate Adversarial Examples (Baluja & Fischer, 2017)

Adversarial Transformation Network (ATN)



Generated adversaries that fool Inception Resnet V2 (Baluja & Fischer, 2017)



Generated adversarial MNIST (Baluja & Fischer, 2017)

Adversarial Transformation Networks: Learning to Generate Adversarial Examples (Baluja & Fischer, 2017)

This categorization is only based on methods covered in this presentation and does not encompasses every single attack or defense method that exist as of today.

Attack & Defense Methods

Attack strategies:

Whitebox:

- Direct gradient step(s): FGSM, BIM, R+FGSM, DAG
- Gradient-based greedy algorithm: JSMA
- Iterative optimization: L-BFGS, C&W, UAP, Adversarial Eyeglasses, RP_2, EOT
- Parameterized optimization: ATN

Blackbox:

- Decision boundary approximation: SBA
- Evolutionary algorithm: One Pixel Attack
- Finite difference method: ZOO

Defense strategies:

Data augmentation:

- Adversarial training
- Ensemble adversarial training
- PGD adversarial training

Gradient regularization:

- DCN
- Defensive distillation

Input manipulation:

- PixelDefend

Detection methods:

- Adversary detector networks
- Feature squeezing

Substitute Blackbox Attack (SBA)

- Blackbox attack by approximating decision boundaries of victim
- Assuming attacker has access to softmax probability
- Procedures:
  - Train a substitute model on a dataset **labelled by the victim**
  - Attack the substitute model using any whitebox methods
  - The generated adversaries should be transferable to the victim model
- Successfully attacked Google, Amazon, and MetaMind image recognition models

Practical Black-Box Attacks against Machine Learning (Papernot et al., 2016)

This categorization is only based on methods covered in this presentation and does not encompasses every single attack or defense method that exist as of today.

Attack & Defense Methods

Attack strategies:

Whitebox:

- Direct gradient step(s): FGSM, BIM, R+FGSM, DAG
- Gradient-based greedy algorithm: JSMA
- Iterative optimization: L-BFGS, C&W, UAP, Adversarial Eyeglasses, RP_2, EOT
- Parameterized optimization: ATN

Blackbox:

- Decision boundary approximation: SBA
- Evolutionary algorithm: One Pixel Attack
- Finite difference method: ZOO

Defense strategies:

Data augmentation:

- Adversarial training
- Ensemble adversarial training
- PGD adversarial training

Gradient regularization:

- DCN
- Defensive distillation

Input manipulation:

- PixelDefend

Detection methods:

- Adversary detector networks
- Feature squeezing

One Pixel Attack

- Blackbox attack using **differential evolution algorithm**
- Able to find adversarial examples by just modifying a **single pixel**
- Fitness function: target misclassification class probability

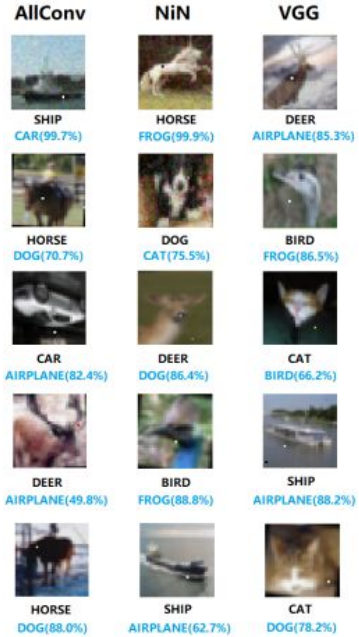


Illustration of one-pixel attack (Su et al., 2017)

One Pixel Attack for Fooling Deep Neural Networks (Su et al., 2017)

This categorization is only based on methods covered in this presentation and does not encompasses every single attack or defense method that exist as of today.

Attack & Defense Methods

Attack strategies:

Whitebox:

- Direct gradient step(s): FGSM, BIM, R+FGSM, DAG
- Gradient-based greedy algorithm: JSMA
- Iterative optimization: L-BFGS, C&W, UAP, Adversarial Eyeglasses, RP_2, EOT
- Parameterized optimization: ATN

Blackbox:

- Decision boundary approximation: SBA
- Evolutionary algorithm: One Pixel Attack
- Finite difference method: ZOO

Defense strategies:

Data augmentation:

- Adversarial training
- Ensemble adversarial training
- PGD adversarial training

Gradient regularization:

- DCN
- Defensive distillation

Input manipulation:

- PixelDefend

Detection methods:

- Adversary detector networks
- Feature squeezing

Zeroth Order Optimization (ZOO)

- Blackbox attack via finite difference approximation

$$\hat{g}_i := \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}_i} \approx \frac{f(\mathbf{x} + h\mathbf{e}_i) - f(\mathbf{x} - h\mathbf{e}_i)}{2h}$$

- ***f*** is victim model, $\boldsymbol{e}_i$ is a vector where only the ***i***-th element is 1, $\boldsymbol{x}_i$ is the ***i***-th element of input ***x***
- Suggest to use stochastic **coordinate** descent (Why?)

Algorithm 2 ZOO-ADAM: Zeroth Order Stochastic Coordinate Descent with Coordinate-wise ADAM

Require: Step size η, ADAM states $M \in \mathbb{R}^p, v \in \mathbb{R}^p, T \in \mathbb{Z}^p$, ADAM hyper-parameters $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$

$M \leftarrow \mathbf{0}, v \leftarrow \mathbf{0}, T \leftarrow \mathbf{0}$
while not converged **do**
  Randomly pick a coordinate $i \in \{1, \cdots, p\}$
  Estimate $\hat{g}_i$ using (6)
  $T_i \leftarrow T_i + 1$
  $M_i \leftarrow \beta_1 M_i + (1 - \beta_1)\hat{g}_i, \quad v_i \leftarrow \beta_2 v_i + (1 - \beta_2)\hat{g}_i^2$
  $\hat{M}_i = M_i / (1 - \beta_1^{T_i}), \quad \hat{v}_i = v_i / (1 - \beta_2^{T_i})$
  $\delta^* = -\eta \frac{\hat{M}_i}{\sqrt{\hat{v}_i + \epsilon}}$
  Update $\mathbf{x}_i \leftarrow \mathbf{x}_i + \delta^*$
end while

ZOO: Zeroth Order Optimization based Black-box Attacks to Deep Neural Networks without Training Substitute Models (Chen et al., 2017)

Agenda

Intro

Adversarial Attacks

Real World Adversaries

Adversarial Defenses and Circumvention Strategy

Takeaways

This categorization is only based on methods covered in this presentation and does not encompasses every single attack or defense method that exist as of today.

Attack & Defense Methods

Attack strategies:

Whitebox:

- Direct gradient step(s): FGSM, BIM, R+FGSM, DAG
- Gradient-based greedy algorithm: JSMA
- Iterative optimization: L-BFGS, C&W, UAP, Adversarial Eyeglasses, RP_2, EOT
- Parameterized optimization: ATN

Blackbox:

- Decision boundary approximation: SBA
- Evolutionary algorithm: One Pixel Attack
- Finite difference method: ZOO

Defense strategies:

Data augmentation:

- Adversarial training
- Ensemble adversarial training
- PGD adversarial training

Gradient regularization:

- DCN
- Defensive distillation

Input manipulation:

- PixelDefend

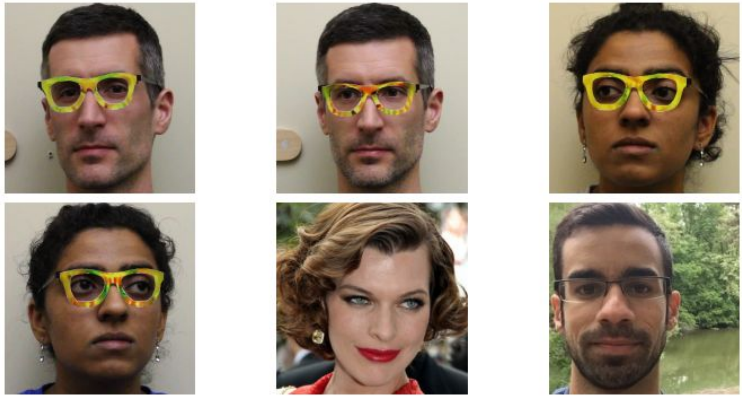
Detection methods:

- Adversary detector networks
- Feature squeezing

Adversarial Eyeglasses

- Printable adversarial eyeglasses to fool face recognition models
- Optimization-based whitebox attack with **binary masking** operation to constraint the location of perturbations
- Printable adversaries remains a challenging problem (e.g. Non-printable Score and Total Variation loss were used during optimization to make the eyeglasses printable)

$$\underset{r}{\operatorname{argmin}} \Big(\big(\sum_{x \in X} softmaxloss(x + r, c_t) \big) + \kappa_1 \cdot TV(r) + \kappa_2 \cdot NPS(r) \Big)$$



Example of adversarial eyeglasses (Sharif et al., 2016). Top row depicts an input given to a model, while bottom row depicts the person from the target misclassification class.

Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition (Sharif et al., 2016)

Adversarial Road Signs

- Similar attack method to the adversarial eyeglasses with different masks, they called this method as RP_2
- Perceptually different, but **inconspicuous**

“STOP” signs misclassified as a “speed limit” sign (Eykholt et al., 2017).

Robust Physical-World Attacks on Deep Learning Models (Eykholt et al., 2017)

Adversarial Turtle

- 3D printed adversarial turtle
- Optimization-based whitebox attack **over various transformations** (e.g. rotation, etc.) or **Expectation Over Transformation (EOT)**

$$\arg\max_{x'} \mathbb{E}_{t\sim T}\Big[\log P(y_t|t(x')) - \lambda||LAB(t(x')) - LAB(t(x))||_2\Big]$$

- Why are we taking expectation over transformation?

Note: ***x'*** is the adversarial counterpart of original input ***x***, y_t is the target misclassification class, ***T*** is a set of possible transformations

3D printed adversarial turtle misclassified as rifle (Athalye et al., 2018).

Synthesizing Robust Adversarial Examples (Athalye et al., 2018)

Agenda

Intro

Adversarial Attacks

Real World Adversaries

Adversarial Defenses and Circumvention Strategy

Takeaways

This categorization is only based on methods covered in this presentation and does not encompasses every single attack or defense method that exist as of today.

Attack & Defense Methods

Attack strategies:

Whitebox:

- Direct gradient step(s): FGSM, BIM, R+FGSM, DAG
- Gradient-based greedy algorithm: JSMA
- Iterative optimization: L-BFGS, C&W, UAP, Adversarial Eyeglasses, RP_2, EOT
- Parameterized optimization: ATN

Blackbox:

- Decision boundary approximation: SBA
- Evolutionary algorithm: One Pixel Attack
- Finite difference method: ZOO

Defense strategies:

Data augmentation:

- Adversarial training
- Ensemble adversarial training
- PGD adversarial training

Gradient regularization:

- DCN
- Defensive distillation

Input manipulation:

- PixelDefend

Detection methods:

- Adversary detector networks
- Feature squeezing

Adversarial Training

- Include adversarial examples as part of the training set
- Training loss becomes:

$$\tilde{J}(\boldsymbol{\theta}, \boldsymbol{x}, y) = \alpha J(\boldsymbol{\theta}, \boldsymbol{x}, y) + (1 - \alpha) J(\boldsymbol{\theta}, \boldsymbol{x} + \epsilon \text{sign}\left(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y)\right))$$

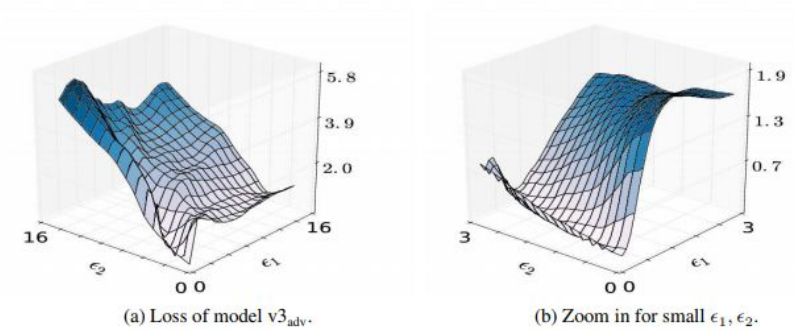
- In other words, new adversarial examples are generated **per training iteration** based on the state of the model at that iteration
- **NOT** the same as the Generative Adversarial Nets (GAN)
- Robust to adversaries included during adversarial training

Note: $J(\cdot)$ denotes the classification loss (e.g. cross-entropy), $\boldsymbol{\theta}$ denotes the model's parameter, α denotes a constant that weigh the importance on classifying normal versus adversarial examples where $\alpha \in [0, 1]$.

Explaining and Harnessing Adversarial Examples (Goodfellow et al., 2015)

BEWARE: Gradient Masking

- When a defense method prevents a model to reveal meaningful gradients
- At the origin, local gradient towards ϵ_1 is larger compared to ϵ_2 direction
- But loss actually higher in ϵ_2 direction for higher ϵ values
- Many directions orthogonal to ϵ_1 with higher loss at larger ϵ
- Often **unintentional**



(a) Loss of model $v3_{adv}$. (b) Zoom in for small ϵ_1, ϵ_2.

Illustration of loss surface of a model trained with FGSM adversarial training (Tramèr et al., 2018). Here, ϵ_1 is the direction given by calculating ***dL/dx***, and ϵ_2 is direction orthogonal to ϵ_1.

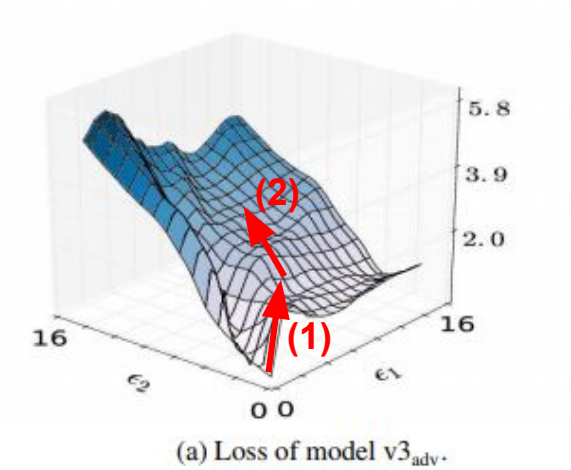
Ensemble Adversarial Training: Attacks and Defenses (Tramèr et al., 2018)

REVISIT: R+FGSM

$$x^{\text{adv}} = x' + (\varepsilon - \alpha) \cdot \texttt{sign}\left(\nabla_{x'} J(x', y_{\text{true}})\right), \quad \text{where} \quad x' = x + \alpha \cdot \texttt{sign}(\mathcal{N}(\mathbf{0}^d, \mathbf{I}^d))$$

(2) (1)

- R+FGSM: add small random perturbation **(1) before** calculating the gradient **(2)** (i.e. random start)
- Can circumvent naive implementation of FGSM adversarial training



(a) Loss of model $v3_{adv}$.

Illustration of how R+FGSM circumvents FGSM adversarial training.

Ensemble Adversarial Training: Attacks and Defenses (Tramèr et al., 2018)

Ensemble Adversarial Training

- Adversarial training but **strictly** collect adversarial examples from various different models
- More robust to future blackbox attacks
- Reduces the model reliance to gradient masking
- Still vulnerable to adversarial examples, although arguably more robust compared to standard adversarial training
- Other claim: related to robust optimization definition as defined in PGD adversarial training (next slide)

Ensemble Adversarial Training: Attacks and Defenses (Tramèr et al., 2018)

PGD Adversarial Training

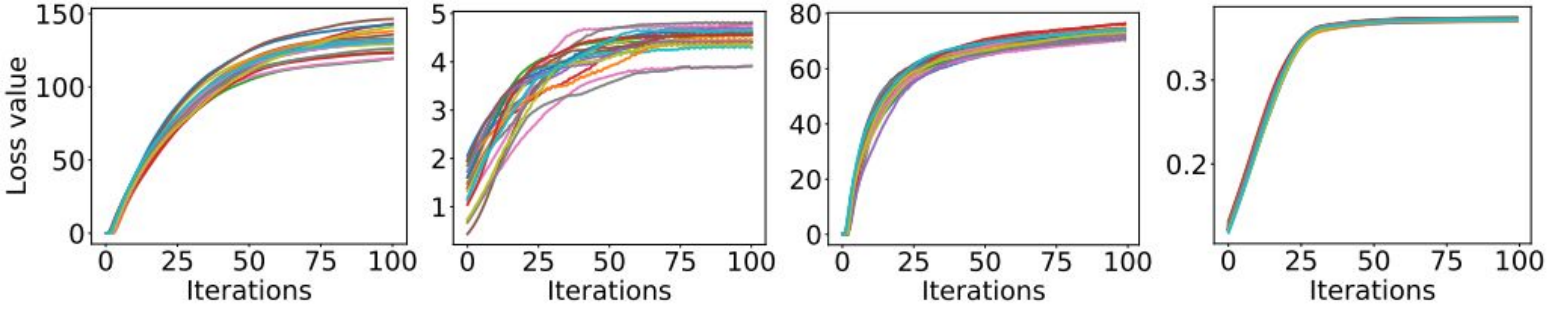
- Adversarial training from robust optimization perspective

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

- In other words: find a set of parameter ***θ*** that **minimizes** the loss in the **worst-case** scenario (Minimax formulation)
- Empirically showed that adversaries generated using BIM, which they called as Projected Gradient Descent (PGD) method, are the worst-case adversaries (see next slide)

Towards Deep Learning Models Resistant to Adversarial Attacks (Madry et al., 2018)

PGD Adversarial Training



(a) MNIST Natural training (b) MNIST Adversarial training (c) CIFAR10 Natural training (d) CIFAR10 Adversarial training

Cross-entropy loss values plateau to similar values when evaluated against different sets of PGD adversarial examples, where each set was generated at random starting points (i.e. using R+BIM) (Madry et al., 2018). They argue that there are many local maxima with similar values, so the local maxima approximates the global maxima (similar to argument of many local minima with similar values).

Towards Deep Learning Models Resistant to Adversarial Attacks (Madry et al., 2018)

PGD Adversarial Training

Model \ Adversary	Natural	FGSM	FGSM random	PGD (7 steps)	PGD (20 steps)
Simple (natural training)	92.7%	27.5%	19.6%	1.2%	0.8%
Simple (FGSM training)	87.4%	90.9%	90.4%	0.0%	0.0%
Simple (PGD training)	79.4%	51.7%	55.9%	47.1%	43.7%
Wide (natural training)	95.2%	32.7%	25.1%	4.1%	3.5%
Wide (FGSM training)	90.3%	95.1%	95.0%	0.0%	0.0%
Wide (PGD training)	87.3%	56.1%	60.3%	50.0%	45.8%

Classifier accuracy on CIFAR10 against various whitebox attacks (Madry et al., 2018). Note that “simple” and “wide” denote whether the model is a low or high capacity model, respectively.

Towards Deep Learning Models Resistant to Adversarial Attacks (Madry et al., 2018)

PGD Adversarial Training

Target \ Source	Simple (natural training)	Simple (FGSM training)	Simple (PGD training)	Wide (natural training)	Wide (FGSM training)	Wide (PGD training)
Simple (natural training)	6.6%	71.6%	71.8%	1.4%	51.4%	75.6%
Simple (FGSM training)	66.3%	40.3%	58.4%	65.4%	26.8%	66.2%
Simple (PGD training)	78.1%	78.2%	57.7%	77.9%	78.1%	65.2%
Wide (natural training)	10.9%	79.6%	79.1%	0.0%	51.3%	79.7%
Wide (FGSM training)	67.6%	51.7%	67.4%	56.5%	0.0%	71.6%
Wide (PGD training)	86.4%	86.8%	72.1%	86.0%	86.3%	64.2%

Classifier accuracy on CIFAR10 against various blackbox attacks (Madry et al., 2018). Note that “simple” and “wide” denote whether the model is a low or high capacity model, respectively.

Towards Deep Learning Models Resistant to Adversarial Attacks (Madry et al., 2018)

This categorization is only based on methods covered in this presentation and does not encompasses every single attack or defense method that exist as of today.

Attack & Defense Methods

Attack strategies:

Whitebox:

- Direct gradient step(s): FGSM, BIM, R+FGSM, DAG
- Gradient-based greedy algorithm: JSMA
- Iterative optimization: L-BFGS, C&W, UAP, Adversarial Eyeglasses, RP_2, EOT
- Parameterized optimization: ATN

Blackbox:

- Decision boundary approximation: SBA
- Evolutionary algorithm: One Pixel Attack
- Finite difference method: ZOO

Defense strategies:

Data augmentation:

- Adversarial training
- Ensemble adversarial training
- PGD adversarial training

Gradient regularization:

- DCN
- Defensive distillation

Input manipulation:

- PixelDefend

Detection methods:

- Adversary detector networks
- Feature squeezing

Deep Contractive Network (DCN)

- Motivation: realization that L-BFGS works by exploiting the gradients
- Add penalty term to regularize gradients at each layer

$$J_{DCN}(\theta) = \sum_{i=1}^{m} (L(t^{(i)}, y^{(i)}) + \sum_{j=1}^{H+1} \lambda_j \parallel \frac{\partial h_j^{(i)}}{\partial h_{j-1}^{(i)}} \parallel_2)$$

- Intuitively, why do we want to regularize the gradients?

Note: ***L(t,y)*** is the classification loss given input-label pair ***(t,y)***, h_j denotes the activation nodes at the ***j***-th layer, ***H*** denotes the number of layers, and ***m*** denotes the number of training data.

Towards Deep Neural Network Architectures Robust to Adversarial Examples (Gu & Rigazio et al., 2014)

Deep Contractive Network (DCN)

Model Name	DCN error	DCN adv. distortion	orig. error	orig. adv. distortion
N100-100-10	2.3%	**0.107**	1.8%	0.084
N200-200-10	2.0%	**0.102**	1.6%	0.087
AE400-10	2.0%	**0.106**	2.0%	0.098
ConvNet	1.2%	**0.106**	0.9%	0.095

Results of using DCN as a defense against L-BFGS attack. They show how the perturbations needed to create adversarial examples that fool DCN are higher compared to standard model (measured by average distortion e.g. ||**x - x'**||).

Towards Deep Neural Network Architectures Robust to Adversarial Examples (Gu & Rigazio et al., 2014)

Defensive Distillation

- Analogy of “teacher” and “student” networks
- Procedure:
  - Train “teacher” network with modified softmax function where the temperature ***T*** is set to be larger than 1
  - Train “student” network using the teacher’s prediction as training label with softmax temperature ***T***
  - During test time, set ***T*** back to 1
- Showed how ***dL/dx*** approaches 0 as ***T*** increases, to the point where they are smaller than the precision of floating points by the end of training process
- Circumventable by **C&W attacks** and **JSMA that includes the softmax temperature**

Softmax function with temperature T:

$$F(X) = \left[\frac{e^{z_i(X)/T}}{\sum_{l=0}^{N-1} e^{z_l(X)/T}} \right]_{i \in 0..N-1}$$

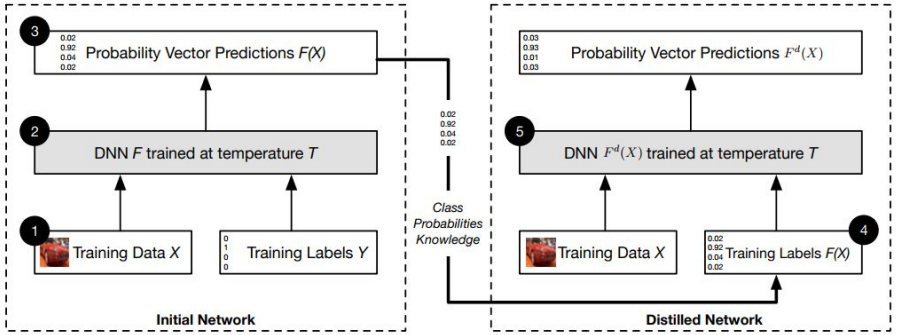


Illustration of defensive distillation (Papernot et al., 2015)

Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks (Papernot et al., 2015)

This categorization is only based on methods covered in this presentation and does not encompasses every single attack or defense method that exist as of today.

Attack & Defense Methods

Attack strategies:

Whitebox:

- Direct gradient step(s): FGSM, BIM, R+FGSM, DAG
- Gradient-based greedy algorithm: JSMA
- Iterative optimization: L-BFGS, C&W, UAP, Adversarial Eyeglasses, RP_2, EOT
- Parameterized optimization: ATN

Blackbox:

- Decision boundary approximation: SBA
- Evolutionary algorithm: One Pixel Attack
- Finite difference method: ZOO

Defense strategies:

Data augmentation:

- Adversarial training
- Ensemble adversarial training
- PGD adversarial training

Gradient regularization:

- DCN
- Defensive distillation

Input manipulation:

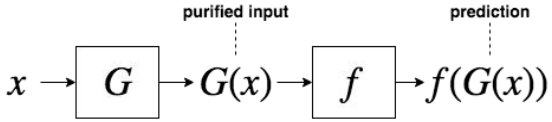
- PixelDefend

Detection methods:

- Adversary detector networks
- Feature squeezing

PixelDefend

- Idea: assuming adversaries have different distribution than the training data, train a generative model ***G*** (e.g. PixelCNN (Oord et al., 2016; Salimans et al., 2017)) strictly on clean training data to model the distribution of non-adversarial examples
- During test time pass the input to ***G*** before classifying it, with the hope that if input is adversarial, ***G*** will still output data that comes from the training data distribution



- The whole pipeline is fully differentiable, thus still vulnerable to future attacks (e.g. vulnerable to Backward Pass Differentiable Approximation, Simultaneous Perturbation Stochastic Approximation - not covered in this presentation)

PixelDefend: Leveraging Generative Models to Understand and Defend against Adversarial Examples (Song et al., 2017)
Conditional Image Generation with PixelCNN Decoders (Oord et al., 2016)
PixelCNN++: Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications (Salimans et al., 2017)

Details will be skipped since background knowledge in PixelCNN is needed.

PixelDefend

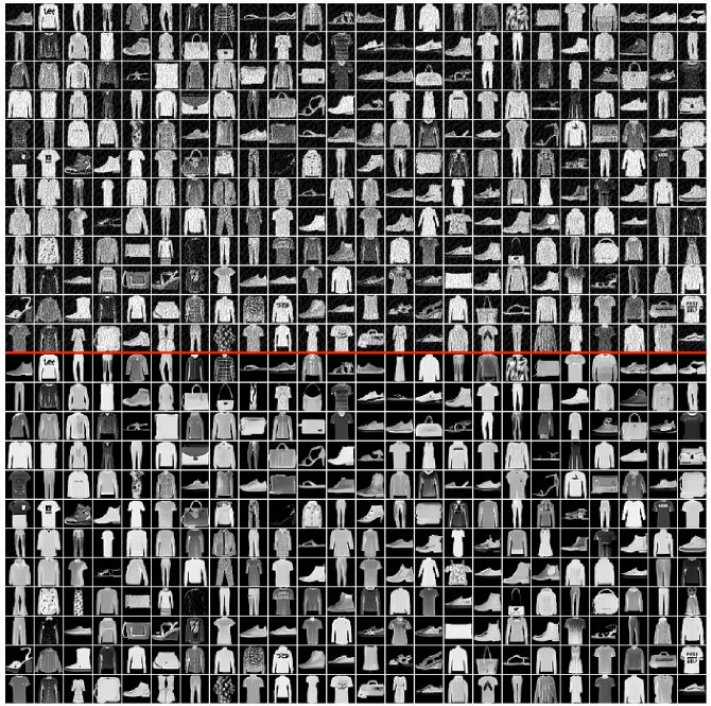


Table 1: **Fashion MNIST** ($\epsilon_{\text{attack}} = 8/25$, $\epsilon_{\text{defend}} = 32$)

NETWORK	TRAINING TECHNIQUE	CLEAN	RAND	FGSM	BIM	DEEP FOOL	CW	STRONGEST ATTACK
ResNet	Normal	**93/93**	89/71	38/24	00/00	06/06	20/01	00/00
VGG	Normal	92/92	91/87	73/58	36/08	49/14	43/23	36/08
ResNet	Adversarial FGSM	**93/93**	**92**/89	85/85	51/00	63/07	67/21	51/00
	Adversarial BIM	92/91	**92/91**	84/79	76/63	82/72	81/70	76/63
	Label Smoothing	**93/93**	91/76	73/45	16/00	29/06	33/14	16/00
	Feature Squeezing	84/84	84/70	70/28	56/25	83/83	83/83	56/25
	Adversarial FGSM + Feature Squeezing	88/88	87/82	80/77	70/46	86/82	84/85	70/46
ResNet	Normal + ***PixelDefend***	88/88	88/89	85/74	83/76	87/87	87/87	83/74
VGG	Normal + ***PixelDefend***	89/89	89/89	87/82	85/83	88/88	88/88	85/82
ResNet	Adversarial FGSM + ***PixelDefend***	90/89	91/90	**88**/82	**85**/76	**90**/88	**89/88**	**85**/76
	Adversarial FGSM +***Adaptive PixelDefend***	91/91	91/**91**	**88/88**	**85/84**	89/**90**	**89**/84	**85/84**

Left: F-MNIST images before (above red line) and after (below red line) the images passed into ***G***.
Right: Results of using PixelDefend against various attacks on F-MNIST dataset, in comparison with other defenses.

PixelDefend: Leveraging Generative Models to Understand and Defend against Adversarial Examples (Song et al., 2017)

Details will be skipped since background knowledge in PixelCNN is needed.

PixelDefend

Table 2: **CIFAR-10** ($\epsilon_{\text{attack}} = 2/8/16$, $\epsilon_{\text{defend}} = 16$)

NETWORK	TRAINING TECHNIQUE	CLEAN	RAND	FGSM	BIM	DEEP FOOL	CW	STRONGEST ATTACK
ResNet	Normal	**92/92/92**	**92**/87/76	33/15/11	10/00/00	12/06/06	07/00/00	07/00/00
VGG	Normal	89/89/89	89/88/80	60/46/30	44/02/00	57/25/11	37/00/00	37/00/00
ResNet	Adversarial FGSM	91/91/91	90/**88**/84	**88/91/91**	24/07/00	45/00/00	20/00/07	20/00/00
	Adversarial BIM	87/87/87	87/87/86	80/52/34	74/32/06	79/48/25	76/42/08	74/32/06
	Label Smoothing	**92/92/92**	91/**88**/77	73/54/28	59/08/01	56/20/10	30/02/02	30/02/01
	Feature Squeezing	84/84/84	83/82/76	31/20/18	13/00/00	75/75/75	78/78/78	13/00/00
	Adversarial FGSM + Feature Squeezing	86/86/86	85/84/81	73/67/55	55/02/00	**85/85/85**	83/83/83	55/02/00
ResNet	Normal + ***PixelDefend***	85/85/88	82/83/84	73/46/24	71/46/25	80/80/80	78/78/78	71/46/24
VGG	Normal + ***PixelDefend***	82/82/82	82/82/84	80/62/52	80/61/48	81/76/76	81/79/79	80/61/48
ResNet	Adversarial FGSM + ***PixelDefend***	88/88/86	86/86/**87**	81/68/67	**81**/69/**56**	**85/85/85**	**84/84/84**	**81**/69/**56**
	Adversarial FGSM + ***Adaptive PixelDefend***	90/90/90	86/87/**87**	81/70/67	**81/70/56**	82/81/82	81/80/81	**81/70/56**

Left: CIFAR10 images before (above red line) and after (below red line) the images passed into ***G***.
Right: Results of using PixelDefend against various attacks on CIFAR10 dataset, in comparison with other defenses.

PixelDefend: Leveraging Generative Models to Understand and Defend against Adversarial Examples (Song et al., 2017)

This categorization is only based on methods covered in this presentation and does not encompasses every single attack or defense method that exist as of today.

Attack & Defense Methods

Attack strategies:

Whitebox:

- Direct gradient step(s): FGSM, BIM, R+FGSM, DAG
- Gradient-based greedy algorithm: JSMA
- Iterative optimization: L-BFGS, C&W, UAP, Adversarial Eyeglasses, RP_2, EOT
- Parameterized optimization: ATN

Blackbox:

- Decision boundary approximation: SBA
- Evolutionary algorithm: One Pixel Attack
- Finite difference method: ZOO

Defense strategies:

Data augmentation:

- Adversarial training
- Ensemble adversarial training
- PGD adversarial training

Gradient regularization:

- DCN
- Defensive distillation

Input manipulation:

- PixelDefend

Detection methods:

- Adversary detector networks
- Feature squeezing

Adversary Detector Networks

- Train a network to differentiate between adversarial and non-adversarial examples
- **Metzen et al. (2017):** take the output of a network at a certain layer, and use it as the input to train the discriminator network
- **Gong et al. (2017):** train a separate classifier using adversarial examples generated using various attack methods
- **Grosse et al. (2017):** augment classifier with an extra node that represents “adversarial” class
- All these are circumventable by C&W attacks (Carlini & Wagner, 2017)

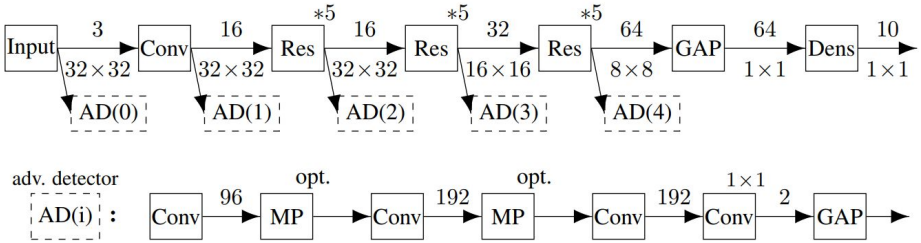


Illustration of detection method proposed by Metzen et al. (2017).

On Detecting Adversarial Perturbations (Metzen et al., 2017)
Adversarial and Clean Data Are Not Twins (Gong et al., 2017)
On the (Statistical) Detection of Adversarial Examples (Grosse et al., 2017)
Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods (Carlini & Wagner, 2017)

Feature Squeezing

- “Squeeze” input features (e.g. pixels) to limit attack dimensionality
- Measure the output difference between model with and without squeezer
- Options for squeezers: bit-depth reduction, binary filter, median smoothing, etc.
- Suggest to use multiple squeezers
- Best squeezers are yet to be determined

Note: ***g*** denotes the classifier model, ***x*** is the input, ***d*** is a distance metric (in this case is L_1 distance), ***T*** is a chosen threshold value

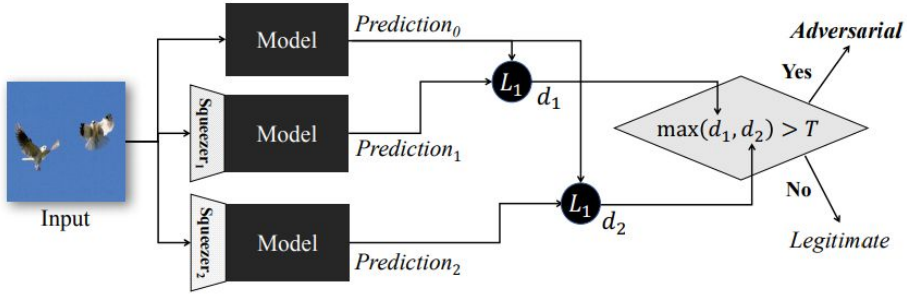


Illustration of feature squeezing (Xu et al., 2017).

$$score^{(\mathbf{x},\mathbf{x}_{squeezed})} = \|g(\mathbf{x}) - g(\mathbf{x}_{squeezed})\|_1$$

$$score^{joint} = \max\left(score^{(\mathbf{x},\mathbf{x}_{sq1})}, score^{(\mathbf{x},\mathbf{x}_{sq2})}, \dots\right)$$

Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks (Xu et al., 2017)

Feature Squeezing

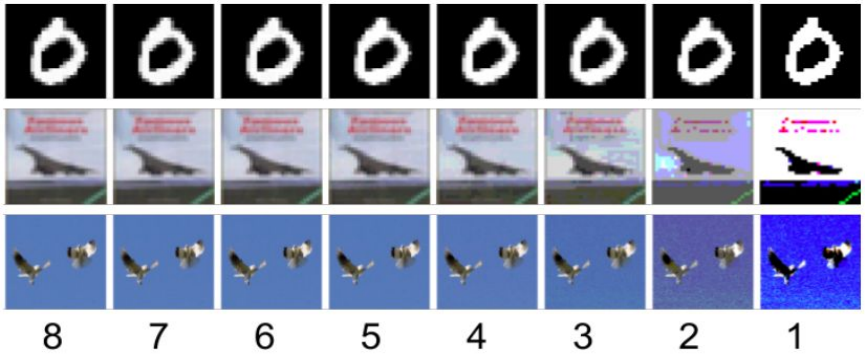


Illustration of the effect of using bit-depth reduction (from 8-bits to 1-bit) as a feature squeezer (Xu et al., 2017).

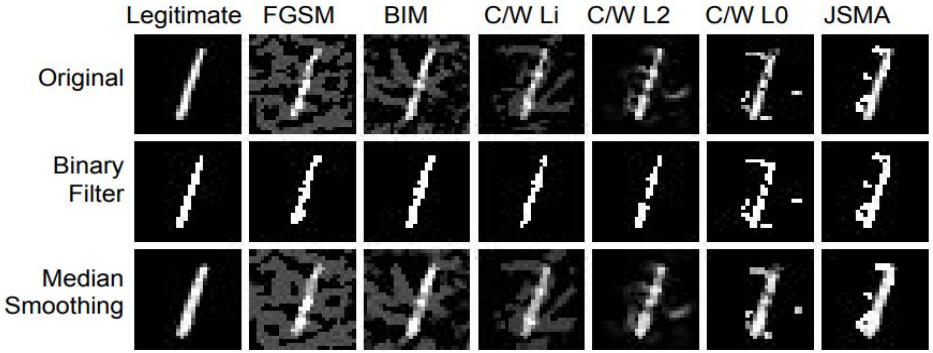


Illustration of the effect of using binary filter and median smoothing as feature squeezers (Xu et al., 2017).

Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks (Xu et al., 2017)

Feature Squeezing

	Configuration			L_∞ Attacks				L_2 Attacks			L_0 Attacks				Overall
	Squeezer	Parameters	Threshold	FGSM	BIM	CW_∞		Deep	CW_2		CW_0		JSMA		Detection
						Next	LL	Fool	Next	LL	Next	LL	Next	LL	Rate
MNIST	Bit Depth	1-bit	0.0005	**1.000**	**0.979**	**1.000**	**1.000**	-	**1.000**	**1.000**	0.556	0.563	**1.000**	**1.000**	0.903
		2-bit	0.0002	0.615	0.064	0.615	0.755	-	0.963	0.958	0.378	0.396	0.969	**1.000**	0.656
	Median Smoothing	2x2	0.0029	0.731	0.277	**1.000**	**1.000**	-	0.944	**1.000**	0.822	0.938	0.938	**1.000**	0.868
		3x3	0.0390	0.385	0.106	0.808	0.830	-	0.815	0.958	0.889	**1.000**	0.969	**1.000**	0.781
	Best Attack-Specific Single Squeezer		-	**1.000**	**0.979**	**1.000**	**1.000**	-	**1.000**	**1.000**	0.889	**1.000**	**1.000**	**1.000**	-
	Best Joint Detection (1-bit, 2x2)		0.0029	**1.000**	**0.979**	**1.000**	**1.000**	-	**1.000**	**1.000**	**0.911**	0.938	**1.000**	**1.000**	**0.982**
CIFAR-10	Bit Depth	1-bit	1.9997	0.063	0.075	0.000	0.000	0.019	0.000	0.000	0.000	0.000	0.000	0.000	0.013
		2-bit	1.9967	0.083	0.175	0.000	0.000	0.000	0.000	0.000	0.000	0.018	0.000	0.000	0.022
		3-bit	1.7822	0.125	0.250	0.755	0.977	0.170	0.787	0.939	0.365	0.214	0.000	0.000	0.409
		4-bit	0.7930	0.125	0.150	0.811	0.886	0.642	0.936	0.980	0.192	0.179	0.041	0.000	0.446
		5-bit	0.3301	0.000	0.050	0.377	0.636	0.509	0.809	0.878	0.096	0.018	0.041	0.038	0.309
	Median Smoothing	2x2	1.1296	0.188	**0.550**	**0.981**	**1.000**	0.717	0.979	**1.000**	**0.981**	**1.000**	**0.837**	**0.885**	0.836
		3x3	1.9431	0.042	0.250	0.660	0.932	0.038	0.681	0.918	0.750	0.929	0.041	0.077	0.486
	Non-local Mean	11-3-2	0.2770	0.125	0.400	0.830	0.955	0.717	0.915	0.939	0.077	0.054	0.265	0.154	0.484
		11-3-4	0.7537	0.167	0.525	0.868	0.977	0.679	0.936	**1.000**	0.250	0.232	0.245	0.269	0.551
		13-3-2	0.2910	0.125	0.375	0.849	0.977	0.717	0.915	0.939	0.077	0.054	0.286	0.173	0.490
		13-3-4	0.8290	0.167	0.525	0.887	0.977	0.642	0.936	**1.000**	0.269	0.232	0.224	0.250	0.547
	Best Attack-Specific Single Squeezer		-	0.188	**0.550**	**0.981**	**1.000**	0.717	0.979	**1.000**	**0.981**	**1.000**	**0.837**	**0.885**	-
	Best Joint Detection (5-bit, 2x2, 13-3-2)		1.1402	**0.208**	**0.550**	**0.981**	**1.000**	**0.774**	**1.000**	**1.000**	**0.981**	**1.000**	**0.837**	**0.885**	**0.845**
ImageNet	Bit Depth	1-bit	1.9942	0.151	0.444	0.042	0.021	0.048	0.064	0.000	0.000	0.000	-	-	0.083
		2-bit	1.9512	0.132	0.511	0.500	0.354	0.286	0.170	0.306	0.218	0.191	-	-	0.293
		3-bit	1.4417	0.132	0.556	**0.979**	**1.000**	0.476	0.787	**1.000**	0.836	**1.000**	-	-	0.751
		4-bit	0.7996	0.038	0.089	0.813	**1.000**	0.381	0.915	**1.000**	0.727	**1.000**	-	-	0.664
		5-bit	0.3528	0.057	0.022	0.688	0.958	0.310	**0.957**	**1.000**	0.473	**1.000**	-	-	0.606
	Median Smoothing	2x2	1.1472	0.358	0.422	0.958	**1.000**	0.714	0.894	**1.000**	**0.982**	**1.000**	-	-	0.816
		3x3	1.6615	0.264	0.444	0.917	0.979	0.500	0.723	0.980	0.909	**1.000**	-	-	0.749
	Non-local Mean	11-3-2	0.7107	0.113	0.156	0.813	0.979	0.357	0.936	0.980	0.418	0.830	-	-	0.618
		11-3-4	1.0387	0.208	0.467	0.958	**1.000**	0.548	0.936	**1.000**	0.673	0.957	-	-	0.747
		13-3-2	0.7535	0.113	0.156	0.813	0.979	0.357	0.936	0.980	0.418	0.851	-	-	0.620
		13-3-4	1.0504	0.226	0.444	0.958	**1.000**	0.548	0.936	**1.000**	0.709	0.957	-	-	0.751
	Best Attack-Specific Single Squeezer		-	0.358	0.556	**0.979**	**1.000**	0.714	**0.957**	**1.000**	**0.982**	**1.000**	-	-	-
	Best Joint Detection (5-bit, 2x2, 11-3-4)		1.2128	**0.434**	**0.644**	**0.979**	**1.000**	**0.786**	0.915	**1.000**	**0.982**	**1.000**	-	-	**0.859**

Detection success rate of feature squeezing (Xu et al., 2017).

Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks (Xu et al., 2017)

Agenda

Intro

Adversarial Attacks

Real World Adversaries

Adversarial Defenses and Circumvention Strategy

Takeaways

This categorization is only based on methods covered in this presentation and does not encompasses every single attack or defense method that exist as of today.

Attack & Defense Methods

Attack strategies:

Whitebox:

- Direct gradient step(s): FGSM, BIM, R+FGSM, DAG
- Gradient-based greedy algorithm: JSMA
- Iterative optimization: L-BFGS, C&W, UAP, Adversarial Eyeglasses, RP_2, EOT
- Parameterized optimization: ATN

Blackbox:

- Decision boundary approximation: SBA
- Evolutionary algorithm: One Pixel Attack
- Finite difference method: ZOO

Defense strategies:

Data augmentation:

- Adversarial training
- Ensemble adversarial training
- PGD adversarial training

Gradient regularization:

- DCN
- Defensive distillation

Input manipulation:

- PixelDefend

Detection methods:

- Adversary detector networks
- Feature squeezing

Takeaway Messages

- Arms race between adversarial attacks and defenses: attackers are winning
- Beware of **gradient masking**, often case it is unintentional and may give false robustness
    - 7 out of 9 defenses accepted to ICLR 2018 were successfully attacked just few days after acceptance decision date (Athalye et al., 2018)
- Always evaluate new defenses with fair evaluations and **adapted adversaries** (always assume attacker is aware of the new defense strategy used during evaluation)
- Although most attacks focus on virtual world adversaries, there are works that aim to generate adversarial examples in the physical world (e.g. the adversarial eyeglasses, adversarial turtle, etc.)
- The field is very empirical, need more works that can provide guarantee on adversarial robustness (e.g. by providing upper bound of a proposed defense method)

Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples (Athalye et al., 2018)

Thank You.

Questions?