Adversarial Examples

Minhan Li, Xin Shi

Lehigh University

November 13, 2019

Overview

1. What is Adversarial Examples
2. Attack (How to generate adversarial examples)
3. Defense

Background

- Machine learning model, training dataset, testing dataset
- The performance of machine learning models in computer vision is impressive.
  - Have achieved human and even above-human accuracy in many tasks
  - ImageNet challenge. In just seven years, the winning accuracy in classifying objects in the dataset rose from 71.8% to 97.3%

Error rate history on ImageNet

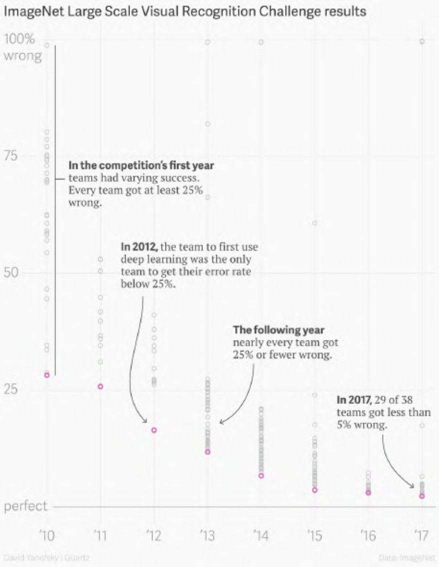


Figure: From https://qz.com/1034972/the-data-that-changed-the-direction-of-ai-research-and-possibly-the-world/

What is Adversarial Examples

- Setup: A trained CNN to classify images
- An adversarial example is an instance with **small, intentional** perturbations that cause a machine learning model to make a false prediction.

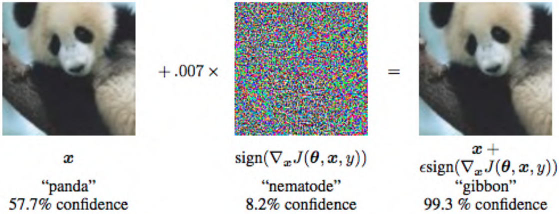
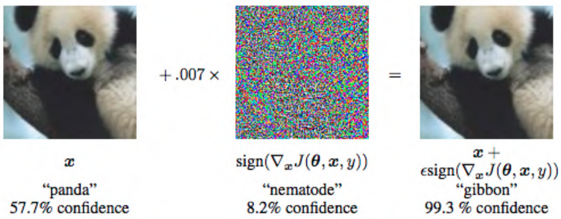


Figure: From Explaining and Harnessing Adversarial Examples by Goodfellow et al.

What is Adversarial Examples (Cont'd)

- Targeted attack

$$argmin_x \quad (\|y_{goal} - \hat{y}(x, w)\|_2^2 + \lambda \|x - x_{target}\|_2^2)$$



What is Adversarial Examples (Cont'd)

- Untargeted attack

$$argmin_x \quad \|y_{goal} - \hat{y}(x, w)\|_2^2$$

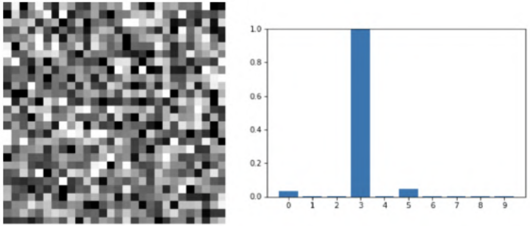


Figure: From Tricking Neural Networks: Create your own Adversarial Examples by Daniel Geng and Rishi Veerapaneni

Why do we need to care about Adversarial Examples

- Security risk: adversarial examples can be transferred from one model to another
  - facial recognition, self-driving cars, biometric recognition
  - existence of 2D picture objects in the physical world *demo*
  - existence of 3D adversarial objects in the physical world[1]
- Understanding of ML models

[1] Synthesizing robust adversarial examples, Athalye et al.

Why do we have adversarial examples

- Overfitting, nonlinearity, insufficient regularization
- Local linearity
- Data perspective
  - Non-robust features learnt by neural network[2]
  - CNN can exploit the high-frequency image components that are not perceivable to human[3]
    - low frequencies in images mean pixel values that are changing slowly over space, while high frequency content means pixel values that are rapidly changing in space.

[2] Adversarial Examples Are Not Bugs, They Are Features, Ilyas et al.

[3] High Frequency Component Helps Explain the Generalization of Convolutional Neural Networks, Wang et al.

Overfitting, nonlinearity, insufficient regularization

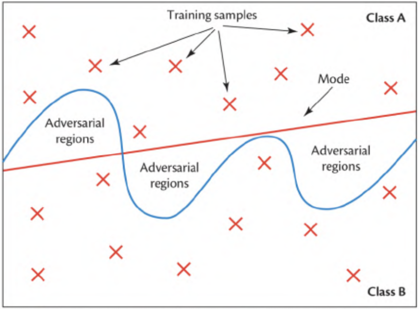


Figure: From McDaniel, Papernot, and Celik, IEEE Security & Privacy Magazine

Non-robust features explanation

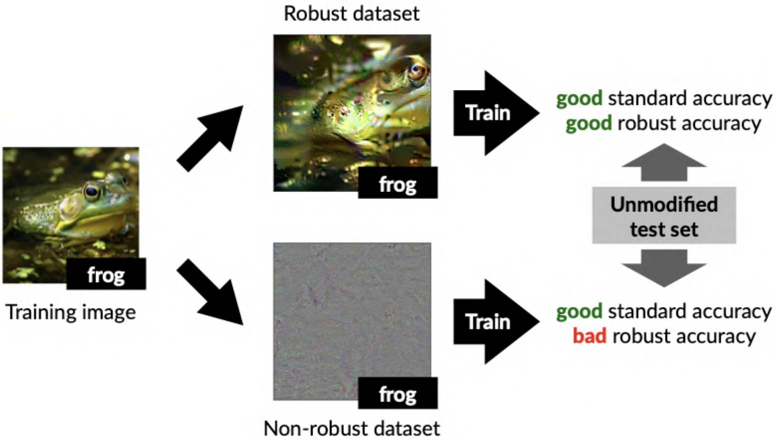


Figure: we disentangle features into combinations of robust/non-robust features. From Adversarial Examples Are Not Bugs, They Are Features, Andrew et al.

How to generate adversarial examples (attack)

x is the input, y is the ground truth label, w is the parameters of the model. Based on the gradient information $\nabla_x J(x, y, w)$.

- Whitebox attack
  - Box-constrained L-BFGS
  - Fast Gradient Sign Method
  - Basic Iterative Method
  - ...
- Blackbox attack
  - Transferability of adversaries
  - Gradient estimation

Attack with L-BFGS

- Smoothness prior means for a small enough radius $\epsilon > 0$ in the vicinity of a given training input, an $x + r$ satisfying $\|r\| < \epsilon$ will get assigned correct label with high probability.
- In [Szegedy et al. 2014], it is pointed out that this smoothness assumption does not hold for neural network.
- Using a simple optimization procedure to find adversarial examples.

Attack with L-BFGS

- Settings
  We denote $f : \mathbb{R}^m \rightarrow \{1 \cdots k\}$ a classifier mapping image pixel value vectors (normalized to range $[0, 1]$) to a discrete label set. Also, f has an associated continuous loss function loss_f.
- For a given $x \in \mathbb{R}^m$ and target label $y \in \{1 \cdots k\}$, we try to solve the following constrained optimization problem.

$$\begin{aligned} &\min_{r \in \mathbb{R}^m} \|r\|_2 \\ s.t.&f(x + r) = y, \\ &x + r \in [0, 1]^m \end{aligned} \tag{1}$$

$x + r$ will be the resulting adversarial example.

Attack with L-BFGS

- Solve the aforementioned problem exactly can be hard. Instead, we approximately optimize the corresponding penalty function using a box-constrained L-BFGS.
-
$$\begin{aligned} &\min_{r \in \mathbb{R}^m} c\|r\|_2 + \text{loss}_f(x + r, y) \\ &s.t. x + r \in [0, 1]^m, \end{aligned} \tag{2}$$

  Here the scalar c is the number that makes the resulting minimizer r satisfy $f(x + r) = y$, which can be found using binary search.

Properties of the resulting adversarial example

- Cross model generalization: Many misclassified by different network
- Cross training-set generalization: Many misclassified by network trained on a disjoint training set.

Conclusion:

It suggests that adversarial examples are universal and not the results of overfitting or specific to training set.

Fast Gradient Sign Method[4]

- Linearity brings adversarial examples
  - Linear behavior in high-dimensional spaces is sufficient to cause adversarial examples
  - Dropout, pretraining and model averaging do not significantly increase robustness
  - Models that are easy to optimize are easy to perturb.

[4] Explaining and Harnessing Adversarial Examples by Goodfellow et al.

Fast Gradient Sign Method: For linear model

Considering linear model:

$$w^T x$$

perturbation on the input: $\tilde{x} = x + \eta$. And $\|\eta\|_\infty \leq \epsilon$.
Then

$$w^T \tilde{x} = w^T x + w^T \eta.$$

To maximize deviation, set $\eta = sign(w)$. Then $w^T \eta = nm\epsilon$

Fast Gradient Sign Method: For nonlinear model

$J(x, y, w)$ is the cost function to train the neural network. Assume there is local linearity regarding to x for the current w and y. Then to maximize $J(x + \eta, y, w)$ where $\|\eta\|_\infty \leq \epsilon$, set

$$\eta = \epsilon \textit{sign}(\nabla_x J(x, y, w)).$$

This is the fast gradient sign method to generate adversarial examples. The gradient can be efficiently computed using back propagation.

Fast Gradient Sign Method: Numerical result

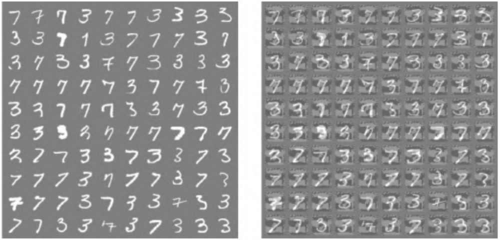


Figure: The fast gradient sign method applied to logistic regression. The logistic regression model has a 1.6% error rate on the 3 versus 7 discrimination task. The logistic regression model has an error rate of 99% on these examples.

Fast Gradient Sign Method: Defense

Adversarial objective function based on the fast gradient sign method:

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

For a maxout network, the error rate on adversarial examples decrease from 89.4% to 17.9%.

An optimization view on adversarial robustness

Training problem:

$$\min_w \rho(w), \quad \text{where} \quad \rho(w) = \mathbf{E}_{(x,y)\sim D}[J(w,x,y)]$$

Min-max problem:

$$\min_w \rho(w), \quad \text{where} \quad \rho(w) = \mathbf{E}_{(x,y)\sim D}[max_{\delta \in S} J(w, x+\delta, y)]$$

- Attack: $max_{\delta \in S} J(w, x+\delta, y)$
  - Constrained nonconvex problem (robust optimization)
  - Projected gradient descent:

$$x^{t+1} = \Pi_{x+S}(x^t + \alpha sgn(\nabla_x) J(w, x, y))$$

- Defense: min-max problem

How to defend

- Adversarial Training: Incorporating adversarial examples into the training data
  - Feeding the model with both the original data and the adversarial examples data
  - Learning with a modified objective function
- Defensive distillation
- Parseval networks
  - Lipschitz constant is bounded
- and more ...

Defensive Distillation[6]

Knowledge Distillation[5]: a way to transfer knowledge from a large neural networks to a smaller one

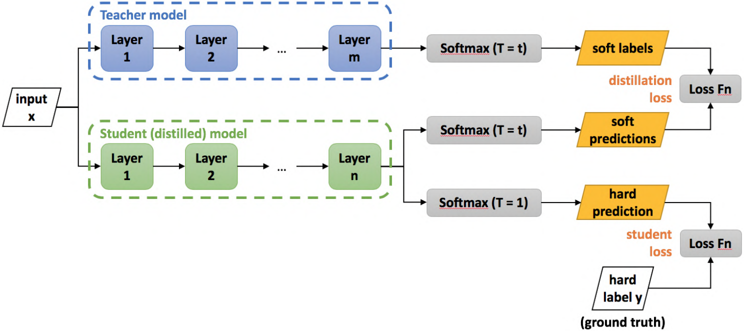


Figure: From: https://medium.com/neuralmachine/knowledge-distillation-dc241d7c2322

[5] Distilling the Knowledge in a Neural Network, Hinton et al. 2015
[6] Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks, Papernot et al. 2016

Defensive Distillation: Softmax temperature

The output of a normal softmax function has the correct class at a very high probability, with all other class probabilities very close to 0.

Softmax function with temperature:

$$F(X) = \left[\frac{e^{\frac{z_i(X)}{T}}}{\sum_{i=0}^{m-1} e^{\frac{z_i(X)}{T}}} \right]_{i \in 0,\dots,m-1}$$

Denote $g(X) = \sum_{i=0}^{m-1} e^{\frac{z_i(X)}{T}}$, then

$$\begin{aligned}
\left.\frac{\partial F_i(X)}{\partial X_j}\right|_T &= \frac{\partial}{\partial X_j}\left(\frac{e^{z_i/T}}{\sum_{l=0}^{N-1} e^{z_l/T}}\right) \\
&= \frac{1}{g^2(X)}\left(\frac{\partial e^{z_i(X)/T}}{\partial X_j} g(X) - e^{z_i(X)/T}\frac{\partial g(X)}{\partial X_j}\right) \\
&= \frac{1}{g^2(X)}\frac{e^{z_i/T}}{T}\left(\sum_{l=0}^{N-1}\frac{\partial z_i}{\partial X_j}e^{z_l/T} - \sum_{l=0}^{N-1}\frac{\partial z_l}{\partial X_j}e^{z_l/T}\right) \\
&= \frac{1}{T}\frac{e^{z_i/T}}{g^2(X)}\left(\sum_{l=0}^{N-1}\left(\frac{\partial z_i}{\partial X_j} - \frac{\partial z_l}{\partial X_j}\right)e^{z_l/T}\right)
\end{aligned}$$

Defensive Distillation (Cont'd)

Denote $g(X) = \sum_{i=0}^{m-1} e^{\frac{z_i(X)}{T}}$, then

$$
\begin{aligned}
\left.\frac{\partial F_i(X)}{\partial X_j}\right|_T &= \frac{\partial}{\partial X_j}\left(\frac{e^{z_i/T}}{\sum_{l=0}^{N-1} e^{z_l/T}}\right) \\
&= \frac{1}{g^2(X)}\left(\frac{\partial e^{z_i(X)/T}}{\partial X_j} g(X) - e^{z_i(X)/T}\frac{\partial g(X)}{\partial X_j}\right) \\
&= \frac{1}{g^2(X)}\frac{e^{z_i/T}}{T}\left(\sum_{l=0}^{N-1}\frac{\partial z_i}{\partial X_j} e^{z_l/T} - \sum_{l=0}^{N-1}\frac{\partial z_l}{\partial X_j} e^{z_l/T}\right) \\
&= \frac{1}{T}\frac{e^{z_i/T}}{g^2(X)}\left(\sum_{l=0}^{N-1}\left(\frac{\partial z_i}{\partial X_j} - \frac{\partial z_l}{\partial X_j}\right) e^{z_l/T}\right)
\end{aligned}
$$

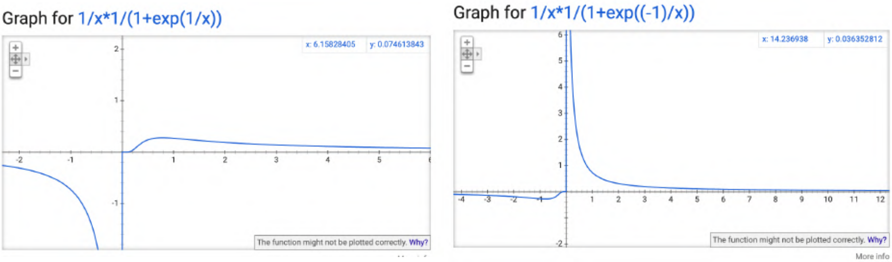


Figure: From google.com

Defensive Distillation (Cont'd)

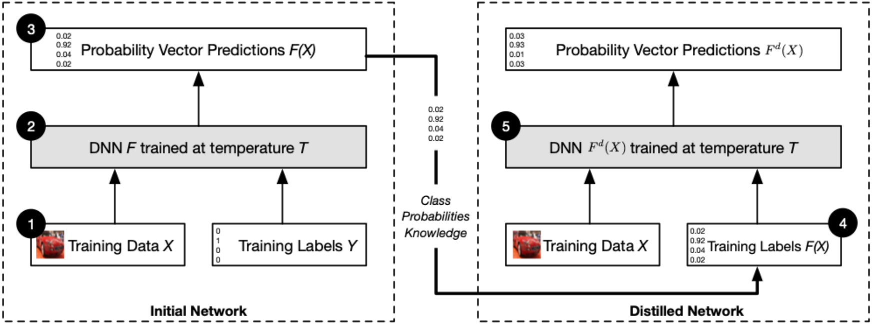
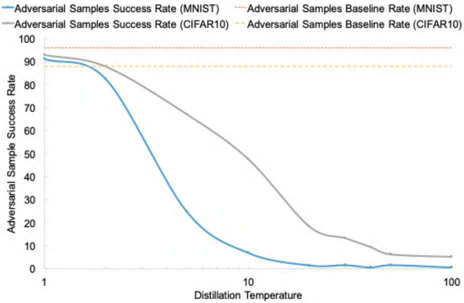


Figure: An overview of the defense mechanism based on a transfer of knowledge contained in probability vectors through distillation

- Reduce the gradient exploited by the adversaries
- Smooth the model

Defensive Distillation (Cont'd)



Distillation Temperature	MNIST Adversarial Samples Success Rate (%)	CIFAR10 Adversarial Samples Success Rate (%)
1	91	92.78
2	82.23	87.67
5	24.67	67
10	6.78	47.56
20	1.34	18.23
30	1.44	13.23
40	0.45	9.34
50	1.45	6.23
100	0.45	5.11
No distillation	95.89	87.89

Figure: An exploration of the temperature parameter space: for 900 targets against the MNIST and CIFAR10 based models and several distillation temperatures

Adversarial Training

A lot of methods have been proposed

- adversarial retraining [Grosse, 2017]
- critical path identification [Wang, 2018]
- build subnetwork as adversary detector [Metzen, 2017]
- and more $\cdots$

Subnetwork as Adversary Detector

Key idea:
instead of making the model robust, consider branching off the main network and add an subnetwork as the "adversary detection network".

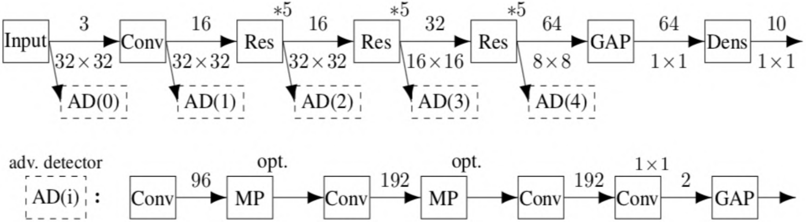


Figure: Example ResNet with adversary detection network

The detector outputs $p_{adv} \in [0, 1]$, can be interpreted as the probability of the input being adversarial.

Subnetwork as Adversary Detector

General procedure:

1. train the classification network on regular(no adversarial) data,
2. generate adversarial examples for each data points using existing attacking methods, assign original with label zero and adversarial with label 1
3. fix the weights of network and train the detector, based on cross-entropy of p_{adv} and the labels.
4. for specific classification network, detector network maybe attached at different places.

Subnetwork as Adversary Detector

The attack methods used for generating adversarial examples are:

1. Fast Gradient Sign Method

$$x^{adv} = x + \epsilon sign(\nabla_x J(x, y, w))$$

2. Basic Iterative Method (iterative version of fast method)

$$x_0^{adv} = x, x_{n+1}^{adv} = \text{Clip}_x^{\epsilon}\{x_n^{adv} + \alpha \text{sgn}(\nabla_x J_{cls}(x_n^{adv}, y_{true}))\} \rightarrow l_\infty \text{ norm}$$

$$x_0^{adv} = x, x_{n+1}^{adv} = Proj_x^{\epsilon}\{x_n^{adv} + \alpha \frac{\nabla_x J_{cls}(x_n^{adv}, y_{true})}{\|\nabla_x J_{cls}(x_n^{adv}, y_{true})\|_2}\} \rightarrow l_2 \text{ norm}$$

3. DeepFool Method
   Iteratively perturbs an image x_0^{adv}.

Subnetwork as Adversary Detector

Experiment details:

- Network: a 32-layer Residual Network
- Data: CIFAR 10, 45000 data points for training and 5000 for testing
- Optimization: Adam with learning rate 0.0001 and $\beta_1 = 0.99, \beta_2 = 0.999$.
- Detector was trained for 20 epochs
- Benchmark: test accuracy of 91.3% on non-adversarial data

Subnetwork as Adversary Detector

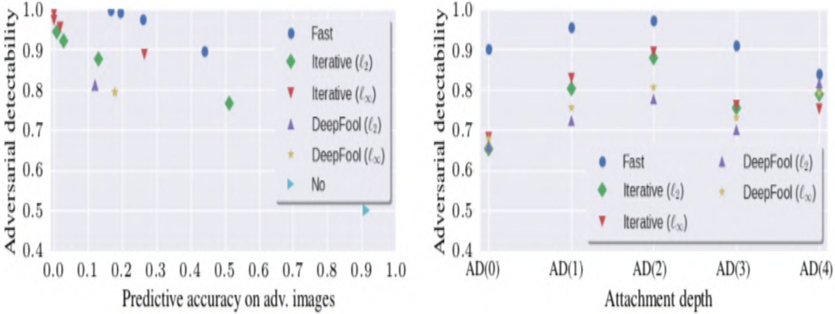


Figure: Example ResNet with adversary detection network

Subnetwork as Adversary Detector

The generalizability of trained detectors

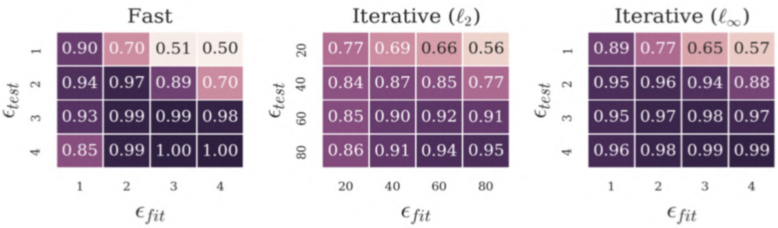


Figure 3: Transferability on CIFAR10 of detector trained for adversary with maximal distortion ϵ_{fit} when tested on the same adversary with distortion ϵ_{test}. Different plots show different adversaries. Numbers correspond to the accuracy of detector on unseen test data.

Figure: Example ResNet with adversary detection network

Adversaries need to generalize across models, detectors, on the other hand, requires generalizability across adversaries.

Subnetwork as Adversary Detector

The generalizability of trained detectors

Figure 4: Transferability on CIFAR10 of detector trained for one adversary when tested on other adversaries. The maximal distortion ϵ of the adversary (when applicable) has been chosen minimally such that the predictive accuracy of the classifier is below 30%. Numbers correspond to the accuracy of the detector on unseen test data.

Figure: Example ResNet with adversary detection network

Subnetwork as Adversary Detector

Dynamic Adversaries:
Since we add an extra detector, we need to consider the possibility of a strong adversary, which have access to classification network and its gradient but also to the adversary detector and its gradient.
Objective:
Maximize the following cost function

$$(1-\sigma)J_{cls}(x, y_{true}) + \sigma J_{det}(x, 1),$$

then the classifier will try to mis-label input x and make the detector output fail to classify x as adversary at the same time.
Method:

$$x_0^{adv} = x,$$

$$x_{n+1}^{adv} = \text{Clip}_x^{\epsilon}\{x_n^{adv} + \alpha[(1-\sigma)\text{sgn}(\nabla_x J_{cls}(x_n^{adv}, y_{true})) + \sigma\text{sgn}(\nabla_x J_{det}(x_n^{adv}, 1))]$$

Subnetwork as Adversary Detector

Method:

$$x_0^{adv} = x,$$

$$x_{n+1}^{adv} = \text{Clip}_x^{\epsilon}\{x_n^{adv} + \alpha[(1-\sigma)\text{sgn}(\nabla_x J_{cls}(x_n^{adv}, y_{true})) + \sigma \text{sgn}(\nabla_x J_{det}(x_n^{adv}, 1))]\}$$

Dynamic Detector:

1. When training the detector, instead of precomputing a dataset of adversarial examples, we compute adversarial examples on-the-fly for each mini-batch.
2. Let the adversary modify each data point with probability 0.5, where the adversary has σ selected uniform randomly from $[0, 1]$.
3. Training detector this way,both the detector and adversary adapt to each other.

Subnetwork as Adversary Detector

Evaluate dynamic adversaries for $\sigma \in \{0.0, 0.1, \cdots, 1.0\}$

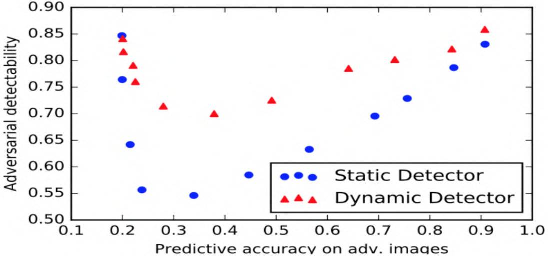


Figure 5: Illustration of detectability versus classification accuracy of a dynamic adversary for different values of σ against a static and dynamic detector. The parameter σ has been chosen as $\sigma \in \{0.0, 0.1, \ldots, 1.0\}$, with smaller values of σ corresponding to lower predictive accuracy, i.e., being further on the left.

Figure: Example ResNet with adversary detection network

A dynamic detector is more robust.

References

Christian Szegedy Wojciech Zaremba Iiya Sutskever (2014)
Intriguing properties of neural networks

I. J. Goodfellow, J. Shlens, and C. Szegedy (2014)
Explaining and harnessing adversarial examples
arXiv preprint arXiv:1412.6572.

IJ. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff (2017)
On detecting adversarial perturbations
arXiv preprint arXiv:1702.04267.

K. Grosse, P. Manoharan, N. Papernot, M. Backes, and P. McDaniel (2017)
On the (statistical) detection of adversarial examples.
arXiv preprint arXiv:1702.06280, 2017.

Y. Wang, H. Su, B. Zhang, and X. Hu (2018)
Interpret neural networks by identifying critical data routing paths.
In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition

The End