Trustworthy Artificial Intelligence

L1. Introduction to Adversarial ML

Dr. Oleg Y. Rogov, PhD., Sr. Research Scientist
Laboratory of Computational Intelligence, Head of AIRI Reliable and Secure Intelligent Systems Lab
February 4, 2025

RSI LAB

Contents

About myself

Ph.D., Sr. Research Scientist at the Computational Intelligence Lab, Skoltech. Head of AIRI Reliable and Secure Intelligent Systems Lab (RSI). Chief of the SAIL Lab at MTUCI. Lecturer at MIPT.

Used to work at the RAS, BCG Gamma Moscow Office, DOM.RF bank Analytics Center, The Keldysh Institue of Applied Mathematics.

Course Breakdown: by areas

This course is 3 credits aimed mainly for DS MSc and PhD Students. Common theme: provable mathematical guarantees for topics covered:

Robustness

Adversarial attacks and defenses, certification (relaxations, branch and bound, certified training, randomized smoothing). Practical issues.

Interpretatbiltiy

Logic + DL, LLMs, methods for building fair systems for tabular, NLP and visual data. Visualization and explainability.

Privacy and Security

Attacks, differential privacy, secure synthetic data, data minimization, federated learning vulnerabilities. Perspective methods and PQC.

Grading

This course is 3 credits aimed mainly for DS MSc and PhD Students. Common theme: provable mathematical guarantees for topics covered:

Homework

2 Homeworks, evaluated as a score from the Kaggle-hosted competition.

Final Project

Topics will be published during week 2. Mentorship with AIRI, UoS, Sber.

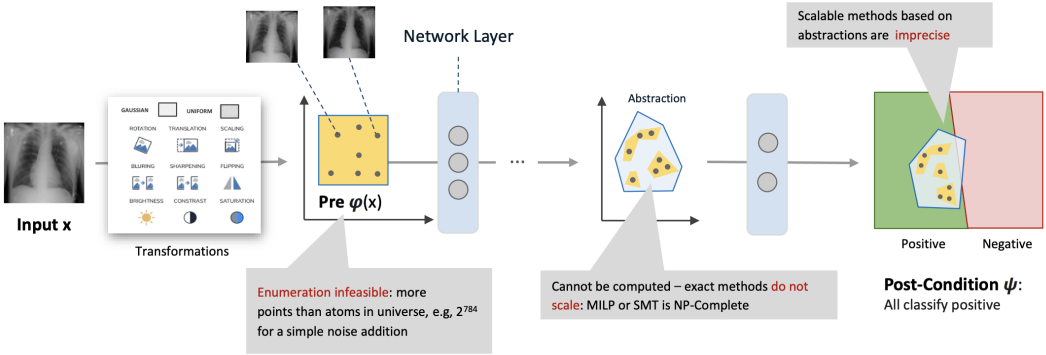
Test

Test with problem solving. In-class. No PC/phones etc.

Introduction

Adversaries arise in many practical applications of machine learning – essentially in any application where there is a party who has an incentive to make a classifier predict in a particular manner. For example, a malware developed may be incentivized to ensure that his malware is classified as benign, and an adversary might want to play audio clips that sends hidden commands to a home assistant device. As we have seen before, the standard statistical learning framework does not take adversaries into account, which we will address next.

Is it hard to certify robustness of AI models?



Besides verification: Provable Defenses of Deep Models

However, an observation here is that if a network is not trained to be provably defended, then it can be difficult to prove properties about it.

Question: can we train the network to be more provable? How?

Adversarial Examples vs. Networks

x

"panda"
57.7% confidence

$+ .007 \times$

$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"nematode"
8.2% confidence

$=$

$x + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"gibbon"
99.3 % confidence

Adversarial Examples IRL

Figure: Robust Physical-World Attacks on Deep Learning Visual Classification, CVPR'18

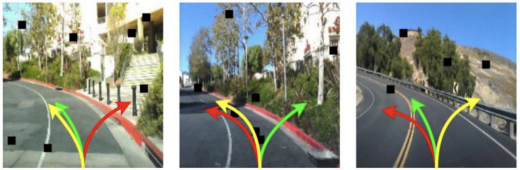
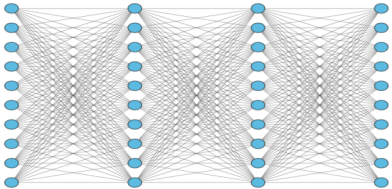


Figure: DeepXplore: Automated Whitebox Testing of Deep Learning Systems, SOSP'17

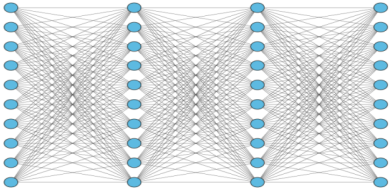
Adversarial Geometric Perturbations

I_o



7

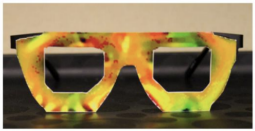
$I = rotate(I_o, -35)$



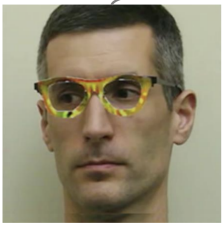
3

Real World Impersonation/Dodging Attacks

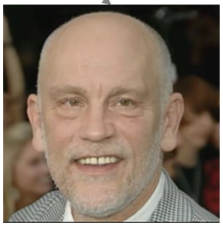
Real glasses



100% success



=



Lujo Bauer

John Malkovich

Cracking down RL is also real

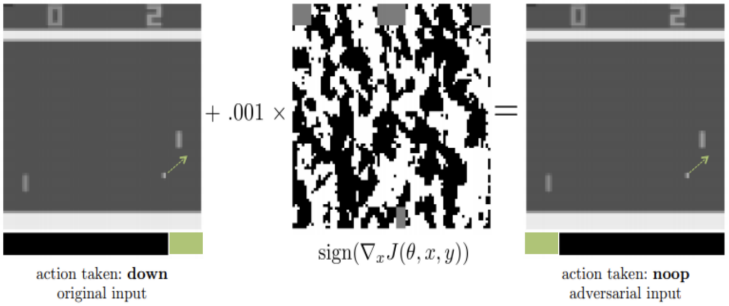


Figure: An agent (DQN) plays the game by selecting actions from a given state (image) that the game produces. An attacker can perturb the image slightly so that the DQN agent chooses the wrong action: here, it wrongly picks noop (do nothing) in the right image, instead of moving the paddle down (left image).

NLP models are vulnerable too

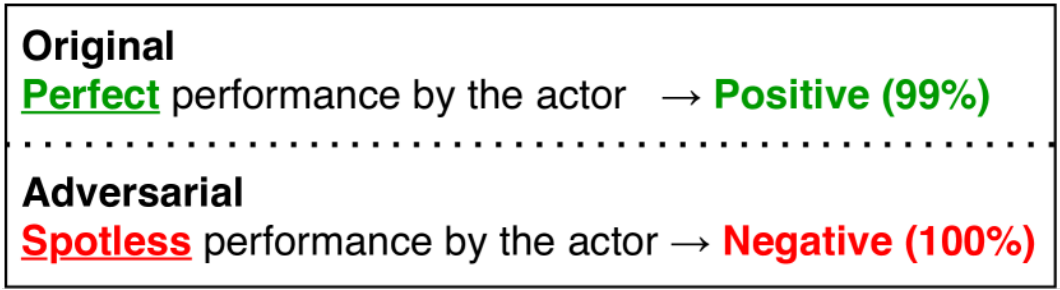


Figure: TextAttack is a Python framework for adversarial attacks, data augmentation, and adversarial training in NLP. It builds attacks from four components: a goal function, a set of constraints, a transformation, and a search method.

Adversarial Attacks on Audio Processing: S2T

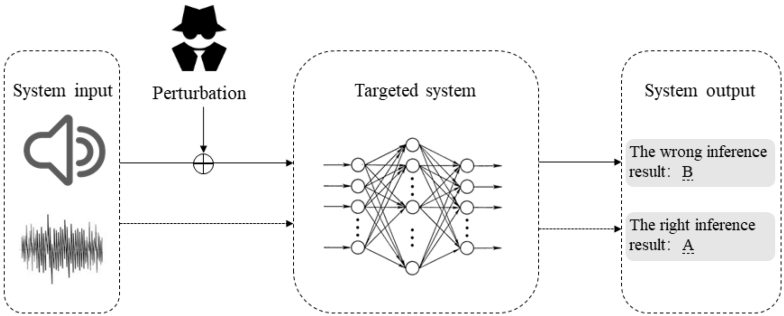


Figure: Modern adversarial attacks against automatic speech recognition (ASR) systems are vicious and often undetectable to the naked ear. An adversary might play what appears to be classical music but in fact is transcribed by a smartphone as "Hey WhatEverAssistand, send 1337 USD to 123-456-7890".

Adversarial Attacks

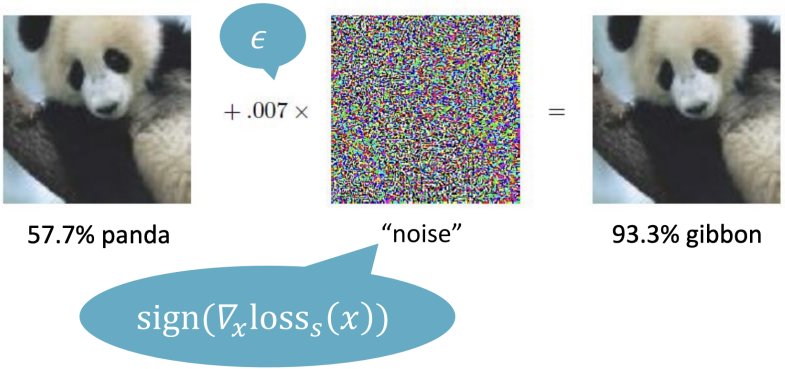


Figure: Explaining and Harnessing Adversarial Examples, ICLR'15

Targeted vs. Untargeted Attacks

Targeted Attack – aims to misclassify the input (e.g., image) to a specific label (e.g. Cat to Dog)

Untargeted Attack – aims to misclassify the input to any wrong label (e.g. Cat to any other animal).

Formulated as a slightly different optimization problem.

Targeted Attack: Problem Statement

Input: - neural network $f : X \to C$ - input $x \in X$ - target label $t \in C$, such that $f(x) \neq t$

Output: - A perturbation η such that $f(x + \eta) = t$. Where the adversarial example is $x' = x + \eta$.

Untargeted Attack: Problem Statement

Input: - neural network $f : X \to C$ - input $x \in X$

Output: - A perturbation η such that $f(x + \eta) \neq f(x)$

Attacks types

White-box attack: the attacker knows the model, the parameters, and the architecture.

Black-box attack: the attacker knows the architecture (e.g., the layers) but not its parameters (e.g., weights).

Note: it was found adversarial examples are transferrable, hence given the same training data as the original network, an attacker can train their own mirror network of the black box original network and then attack the mirror network with white-box techniques. If attack on mirror network succeeds, it will likely succeed on the original.

We will look at white box attacks primarly.

Targeted Fast Gradient Sign Method

1. Compute perturbation: $\eta = \epsilon \cdot \text{sign}\left(\nabla_x \text{loss}_t(x)\right)$, where

$$\nabla_x \text{loss}_t = \left(\frac{\partial \text{loss}_t}{\partial x_1}, \ldots, \frac{\partial \text{loss}_t}{\partial x_n}\right) \quad \text{sign}(g) = \begin{cases} -1, & \text{if } g < 0 \\ 0, & \text{if } g = 0 \\ 1, & \text{if } g > 0 \end{cases}$$

2. Perturb the input: $x' = x - \eta$.
3. Check if $f\left(x'\right) = t$

Here, each x_i is a pixel $x' = x - \eta$. ϵ is a very small constant (e.g, 0.007); As FGSM is 1-step, x' is guaranteed to stay inside the box $[x - \epsilon, x + \epsilon]$, so no need to project. t which is the target, and abad label loss_t is the loss w.r.t target label.

FGSM was designed to be fast, not optimal (may not compute minimal perturbation)

Untargeted version of FGSM

1. Compute perturbation
$$\eta = \epsilon \cdot \text{sign}\left(\nabla_x \text{loss}_s(x)\right)$$
2. Perturb the input:
$$x' = x + \eta$$
3. Check if:
$$f\left(x'\right) \neq s$$

Note: With untargeted FGSM, we do not know what the target (bad) label is that we want. We just want some label different than the correct label s. So we try to 'get away' from the correct label by maximizing the value of the loss

Fast Gradient Sign Method (FGSM)

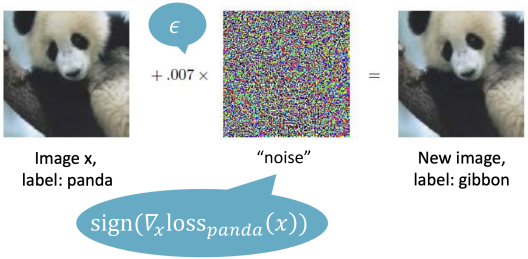


Figure: FGSM attack is incredibly simple. It involves three steps in this order:

1. Calculate the loss after forward propagation. 2. Calculate the gradient with respect to the pixels of the image, 3. Nudge the pixels of the image ever so slightly in the direction of the calculated gradients that maximize the loss calculated above.

Importance of Small Perturbations

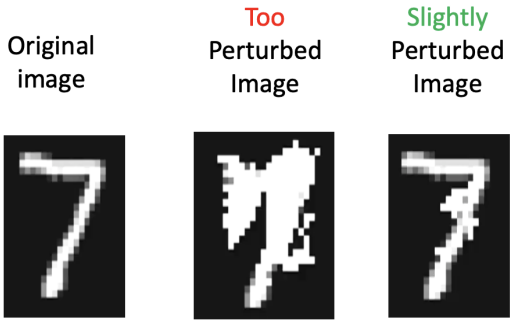


Figure: One needs to get the impact of these. Through notion of distance.

Norm: Notion of Distance

Similarity of $x \sim x'$ is usually captured by an l_p norm:

$$x \sim x' \text{ iff } \|x - x'\|_p < \epsilon$$

$$\text{where } \|x - x'\|_p = \left((|x_1 - x'_1|)^p + \cdots + (|x_n - x'_n|)^p\right)^{\frac{1}{p}}$$

l_0 (when $0^0 = 0$ and we get rid of $1/p$ root) captures the number of changed pixels. l_2 captures the Euclidian distance between x and x'. It can remain small if there are many small changes to many pixels. l_∞ captures maximum noise (change) added to any coordinate. It is the maximum of the absolute values of the entries:

$$\|x - x'\|_\infty = \max\left(|x_1 - x'_1|, \ldots, |x_n - x'_n|\right)$$

This is the most common norm used for adversarial example generation and it is argued that it most naturally captures human vision or audio perceptions.

Targeted Attack with Small Changes

Input: - neural network $f : X \to C$

- input $x \in X$

- target label $t \in C$, such that $f(x) \neq t$.

Output:

- A perturbation $\boldsymbol{\eta}$ such that $f(\boldsymbol{x} + \boldsymbol{\eta}) = \boldsymbol{t}$.

- $\|\eta\|_p$ is minimized.

Optimization Problem

The problem of generating small perturbations can be phrased as an optimization problem:

$p \in \{0, 2, \infty\}$ — This is a hard discrete constraint which is difficult to optimize for with gradient methods.

find		
minimize	$\|\boldsymbol{\eta}\|_p$	
such that		
	$f(\boldsymbol{x} + \boldsymbol{\eta}) = t$	Note: $\boldsymbol{\eta}$ can have negative components.
	$\boldsymbol{x} + \boldsymbol{\eta} \in [0, 1]^n$	

Key insight: Relaxation of the hard constraint.

Problem

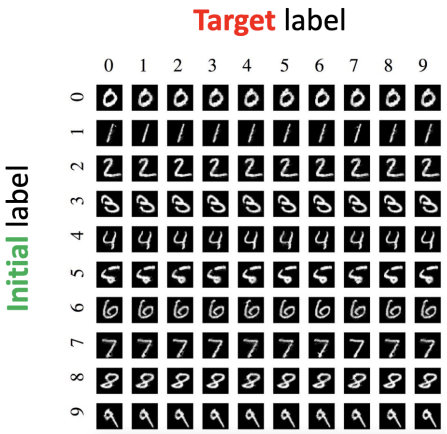
Two steps:

Step 1: Define an objective function $\boldsymbol{obj}_t$ such that: if $\boldsymbol{obj}_t(\boldsymbol{x}+\boldsymbol{\eta}) \leq \mathbf{0}$ then $\boldsymbol{f}(\boldsymbol{x}+\boldsymbol{\eta}) = \boldsymbol{t}$

Step 2: Solve the following optimization problem:

$$\begin{array}{ll} \text{find} & \boldsymbol{\eta} \\ \text{minimize} & \|\boldsymbol{\eta}\|_p + c \cdot \boldsymbol{obj}_t(\boldsymbol{x}+\boldsymbol{\eta}) \\ \text{such that} & \boldsymbol{x}+\boldsymbol{\eta} \in [\mathbf{0},\mathbf{1}]^n \end{array}$$

MNIST application



What we see is that on the MNIST (digit recognition) data set it is not difficult to get a realistic looking image that fools the neural network classifier…

Another attack used during training

So far, we looked at FGSM as well as an attack to minimize the distance to the original input (e.g., image, audio):

- Now, we illustrate another attack, a variant of FGSM applied iteratively with projection.

- The attack uses Projected Gradient Descent (PGD) and is referred to as a PGD attack.

- This is a commonly used attack for adversarial training: training the network to be robust.

Attack types

Attack Type	Region	Optimization	Outcome
FGSM (targeted, untargeted)	Change η fixed to $[-\varepsilon, +\varepsilon]$.	Take exactly one ε-sized step	Produced example will be on boundary of region.
PGD (typically untargeted, but can be targeted)	Can be instantiated with any region one can project to.	Take many steps. Uses projection to stay inside region. For special case of l_∞, step size smaller than ε.	Result will be inside region. Tries to maximize loss.
C&W (presented as targeted)	No real restriction, except image has to be in $[0, 1]$ (like all other methods). This restricts the region for the change $\eta : \eta$ has to be bounded s.t. original image $+\eta$ stays in $[0, 1]$.	Aims to produce a change η with small l_∞. Takes many steps, using LBFGS-B to ensure η stays in bounds.	Result will be inside $[0, 1]$, with a hopefully small l_∞ distance from original image.

Can we Avoid Adversarial Examples?

Many works have tried to, but follow-up works showed that all fail.

The main successful defenses in practice now incorporate adversarial examples during training.

Some pretty good experimental defenses exist

Adversarial Accuracy vs. Test Accuracy

Adversarial accuracy refers to a metric on the test set where for each data point we check if the network classifies the point correctly and the network is robust in a region around that point.

Example: [l_∞ ball]: Let $\epsilon = 0.3$, and let the test set T contain 100 examples. For each example $d_i \in T$, lets check if in the l_∞ region of size ≤ 0.3 around d_i, we find an (adversarial) example with a different classification than d_i. For that purpose we typically use a PGD attack. Now suppose, 95 of the 100 examples classify correctly and for 15 of these 95, we find an adversarial example. Then, our adversarial accuracy will be $\frac{80}{100} = 80\%$ and our test accuracy will be $\frac{95}{100} = 95\%$.

Adversarial accuracy and Test accuracy can be at odds: it is possible to raise the adversarial accuracy which tends to lower test accuracy. This trade off is being actively investigated.

Defense as Optimization Problem

$$
\begin{array}{ll}
\text{find} & \theta \\
\text{minimize} & \rho(\theta) \\
\text{where} & \rho(\theta) = \mathbf{E}_{(\mathrm{x},\mathrm{y})\sim D}\left[\max_{x' \in S(x)} L\left(\theta, x', y\right)\right]
\end{array}
$$

D is the underlying distribution.

$\mathbf{E}_{(\mathrm{x},\mathrm{y})\sim D}$ is typically estimated with the empirical risk.

$S(x)$ denotes the perturbation region around point x, that is, we want all points in $S(x)$ to classify the same as x. We can pick $S(x)$ to be (see Madry et.al, 2017):

$$S(x) = \{x' \mid \|x - x'\|_\infty < \epsilon\}$$

PGD Defense in Practice

Step 1: select a mini-batch B of examples from dataset D.

Step 2: compute $B_{\max}$ by applying PGD attack (actually computes an approximation) as follows to every point $(x, y) \in B$:

$$x_{\max} = \underset{x' \in S(x)}{\operatorname{argmax}} L\left(\theta, x', y\right)$$

Note: $x_{\max}$ need not be adversarial example; it just aims to maximize L

Step 3: solve outer problem:

$$\theta' = \theta - \frac{1}{|B_{\max}|} \sum_{(x_{\max}, y) \in B_{\max}} \nabla_\theta L\left(\theta, x_{\max}, y\right)$$

Step 4: goto Step 1. Various stopping criteria, including reaching a certain number of epochs.

Points to Consider when Defending

Model capacity matters: larger networks are more defendable and less easy to be attacked with transferrable examples. Training smaller nets with PGD has negative effects on accuracy.

Training with adversarial examples from PGD attacks (many steps and project) tends to perform better than training with adversarial examples from FGSM attacks (one step, no projection). Even on larger networks, defenses can negatively affect accuracy (e.g. CIFAR).

More research is needed here. By this we mean that after the network is trained, we test its accuracy on the test set. And there, it is more robust yet more points classify incorrectly.

Lecture Summary

Deep Learning is susceptible to adversarial examples.
Defending against Adversarial examples (an optimization problem)
We looked at a way to (experimentally) defend the network by training with adversarial examples, specifically the PGD defense. This results in a minmax nested optimization problem. Adversarial training can lower standard accuracy. Remains a question of research interest, how to avoid this from happening

Generating Adversarial examples (an optimization problem):
- FGSM: targeted and untargeted
- C&W (minimize perturbation)
- PGD

Applicability Many of the techniques shown today are applicable to domains beyond images, but also models for natural language, code, audio processing, financial data and many more.

Thank you.