AI safety

Baptiste COMBELLES, **Clément CONTET**, **Guillaume COULAUD**, **Joceran GOUNEAU** & **Thibault MOUSSET**

Under the direction of:
Guillaume DUPONT, **Aurélie HURAULT** & **Philippe QUÉINNEC**

March 9, 2023

ENSEEIHT – Département Sciences du Numérique – Troisième année 2022-2023

Table of Contents

Introduction

- IA models are spreading fast
- In addition to the performance issue, 3 subsidiary points:
  - Ensure that deployed models actually do what we expect them to do
  - Make sure that models are robust to malevolent actors
  - Ensure that models respect norms in order to be able to deploy them in critical contexts

Table of Contents

Deep Feedforward Neural Networks or multilayer perceptrons

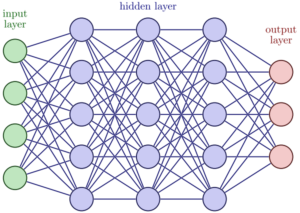
- Goal: approximate a function (e.g. classifier)
- Approximation learned from data using a criterion: the **loss function** $\mathcal{L}$
- Learning with backward propagation and gradient descent algorithm.

Two simple models commonly used

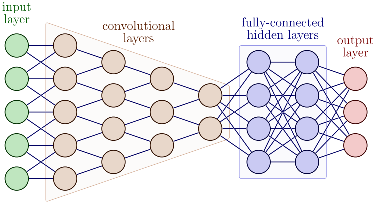
- **Fully connected neural networks:** A sequence of fully connected layers that connect every neuron in one layer to every neuron in the next layer:

$$\begin{cases} z^0 = x \\ z^{\ell+1} = \sigma(W^\ell z^\ell + b^\ell). \end{cases}$$

- **Convolutional neural networks:** Convolution operation in a layer. It can be fully connected.



A fully connected neural network, *https://tikz.net/neural_networks/*



A convolutional neural network, *https://tikz.net/neural_networks/*

A basic training loop

X is an image, y its label.

There are **2 phases:**

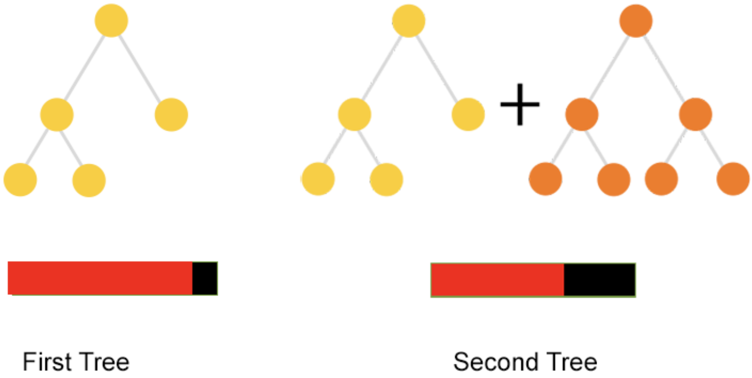
- Training phase:
  Learn the model's weights
- Evaluation phase:
  Compute the accuracy

Algorithm: A basic training loop

for *epoch = 1,...,N* **do**
    for *X,y in Training data* **do**
        Apply the model to X
        Compute the loss value
        Update the model's weight
        Compute the accuracy
    end
    for *X,y in Evaluation data* **do**
        Apply the model to X
        Compute the accuracy
    end
end

Decision Tree

Network in a tree structure, consisting of a root node, branches, internal nodes, and leaf nodes: **random forest** and **gradient boosting decision tree**.



A gradient boosting decision tree, *https://catboost.ai/*

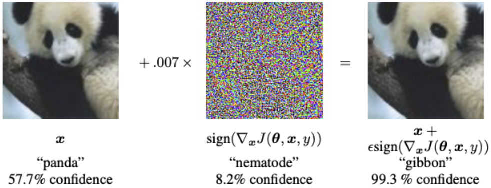
Table of Contents

Threat model

- What is the adversary's **knowledge**
  - **White-box**: access to all information (architecture, parameters, gradient,...)
  - **Black-box**: no information available, can only manipulate the input and see the corresponding output
- What is the adversary's **goal**
  - **Poisoning** attacks: insert fake samples in the training set
  - **Evasion** attacks: craft an example not recognized by the classifier
  - **Targeted** or **non-targeted** attacks

Adversarial attacks in image classification

- Generate a fake image, the **adversarial example** from an existing image
- The adversarial example must be similar to the human eye
- Wrongly classified by the model: panda + perturbation = gibbon



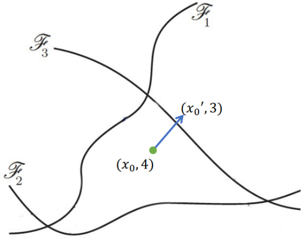
A demonstration of fast adversarial example generation applied to GoogLeNet, *https://arxiv.org/abs/1412.6572*

Some attacks

- Fast Gradient Sign Method (FGSM):

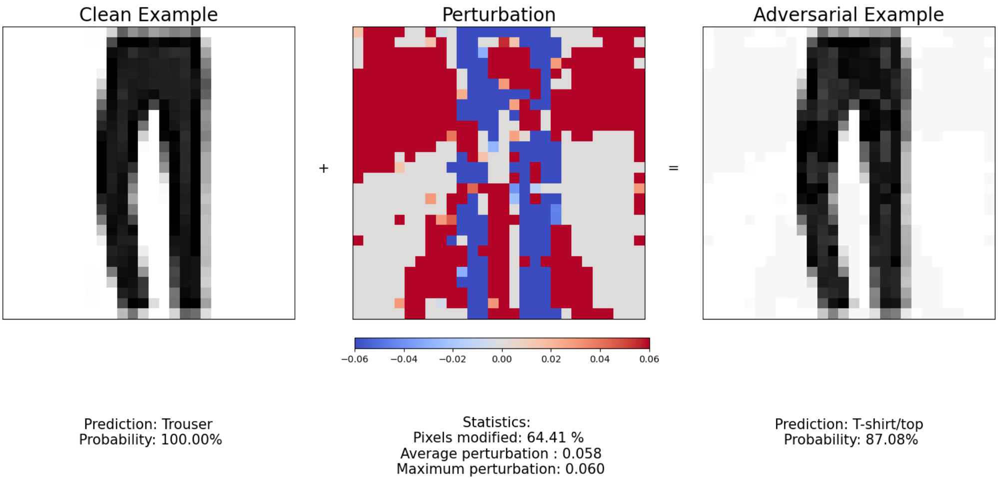
$$x' = x - \varepsilon\,\mathrm{sign}(\nabla_x \mathcal{L}(\theta, x, t))$$

- Projected Gradient Descent (PGD): iterative version of FGSM
- Not all attacks use information from the gradient of the loss function: Deepfool



Example of class separation by hyperplanes

Example of an adversarial attack (PGD)



Trouser + perturbation = T-shirt

Table of Contents

Some approaches

- Make the gradient information less interesting
- Train a model using adversarial examples
- Detect when an input is an adversarial example

Adversarial training

Adversarial training

- Aims to improve the classifier's robustness.
- Replace the data with adversarial examples, often with FGSM or PGD.

Algorithm: An adversarial training loop

for *epoch = 1,...,N* **do**
  for *X,y in Training data* **do**
    Compute the adversarial example X'
    Apply the model to X'
    Compute the loss value
    Update the model's weight
    Compute the accuracy
  end
  for *X,y in Evaluation data* **do**
    Apply the model to X
    Compute the accuracy
  end
end

Table of Contents

Formal Verification

- **What is it?**
  The act of proving or disproving the correctness of intended algorithms underlying a system with respect to a certain formal specification or property, using formal methods of mathematics.[1]
- **What properties? Robustness to input perturbation**
  Ensuring that all points within a ball of a certain radius centered around a given input are classified similarly to the original input.

[1] Wikipedia: `https://en.wikipedia.org/wiki/Formal_verification`

- **SMT** (Satisfiability Modulo Theories)
  Example of an encoding of the model and the (negation of) the verified property as first-order logic formulae[2]:

$$\hat{z}^{\ell+1} = W^{\ell+1} z^{\ell} + b^{\ell+1} \qquad \forall \ell \in [\![0, n-1]\!] \tag{1a}$$

$$z^{\ell} = \max\{0, \hat{z}^{\ell}\} \qquad \forall \ell \in [\![0, n-1]\!] \tag{1b}$$

$$l \leqslant z^{0} \leqslant u \tag{1c}$$

$$z^{n} \leqslant 0 \tag{1d}$$

[2] Bunel et al., A Unified View of Piecewise Linear Neural Network Verification, May 2018

- **MILP** (Mixed Integer Linear Programming)
  Example of an encoding of the model in term of linear equations[3]:

$$\hat{z}^{\ell+1} = W^{\ell+1} z^{\ell} + b^{\ell+1} \qquad \forall \ell \in [\![0, n-1]\!] \tag{2a}$$

$$\delta^{\ell} \in \{0,1\}^{|z^{\ell}|},\ 0 \leqslant z^{\ell} \leqslant u^{\ell} \cdot \delta^{\ell},\ \hat{z}^{\ell} \leqslant z^{\ell} \leqslant \hat{z}^{\ell} - l^{\ell} \cdot (1 - \delta^{\ell}) \qquad \forall i \in [\![0, n-1]\!] \tag{2b}$$

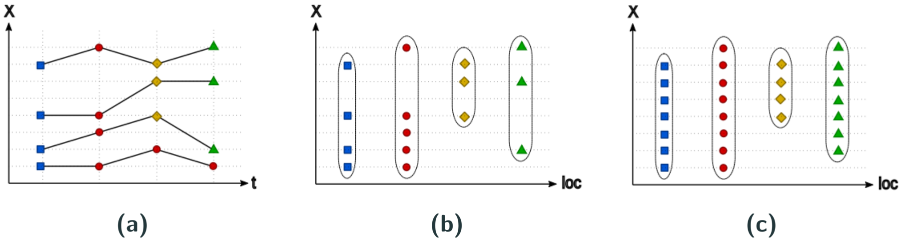
$$l \leqslant z^{0} \leqslant u \tag{2c}$$

$$\min\ z^{n} \tag{2d}$$

[3] Tjeng et al., Evaluating Robustness of Neural Networks with Mixed Integer Programming, February 2019

Formal Verification Methods

- **Static Analysis by Abstract Interpretation**
  Abstract program semantics until the semantics becomes computable:



From Trace Semantics (a), to State Semantics (b), to Interval Semantics (c).[4]

[4] Urban & Miné, A Review of Formal Methods applied to Machine Learning, April 2021.

Table of Contents

Explained Image

Figure 1: Basic Image of a Cat and a Mouse

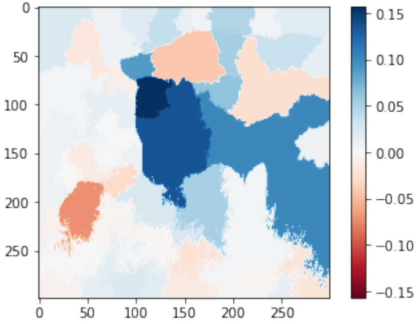


Figure 2: Heatmap of weights for the top class explanation (Black Bear)

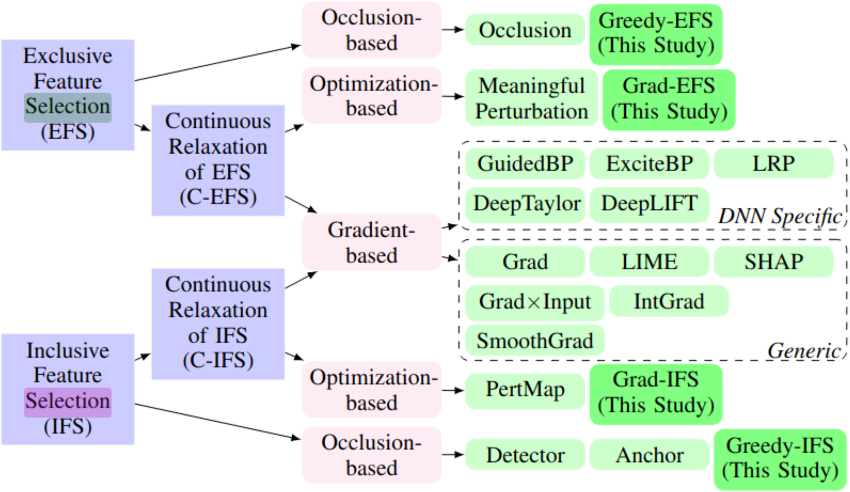
From Lime's Github repository

Elimination of biases

Samples of learning dataset

AI: "That's a wolf then !"

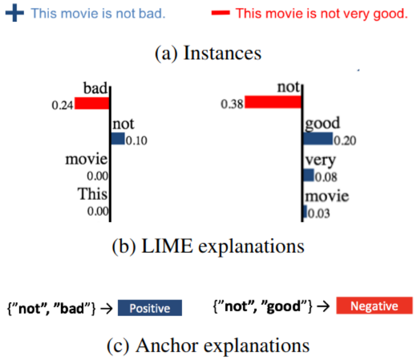
A variety of tools



From "Feature attribution as feature selection", Hara, Ikeno, Soma, Maehara - First figure

Feature attribution VS Feature selection

- LIME and Anchor comparison

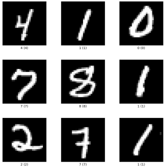


From "Anchors: High-Precision Model-Agnostic Explanations", T. Ribeiro, Singh, Guestrin - First figure

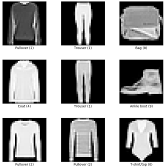
Table of Contents

The datasets

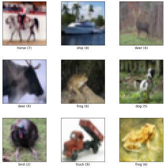
3 datasets: MNIST, FashionMNIST, and CIFAR10



(a) MNIST
28×28



(b) FashionMNIST
28×28



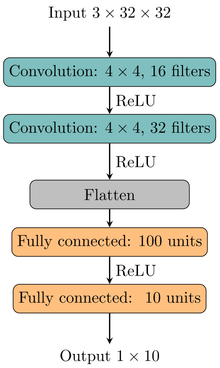
(c) CIFAR10
32×32

Images from the three datasets, MNIST, FashionMNIST, and CIFAR10,
https://www.tensorflow.org/datasets/catalog/

The model

A "small" CNN

- 2 convolution layers
- 2 fully connected layers
- $\sim$ 160.000 parameters



The "small" CNN representation

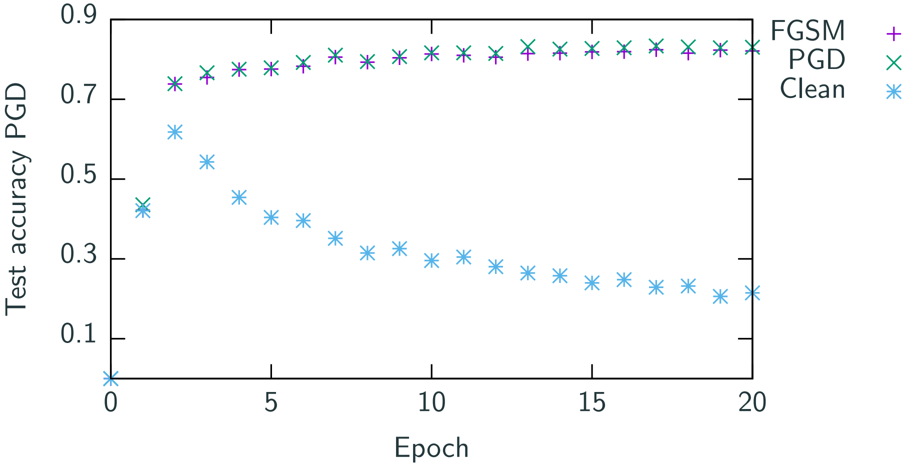
Experiments

We conduced multiple experiments among others:

1. Train the same model with and without adversarial training.
2. Compare the efficiency of attacks according to the training mode of the model.
3. Use the different verification methods on these models.

In the following slides we only present a sample of our results

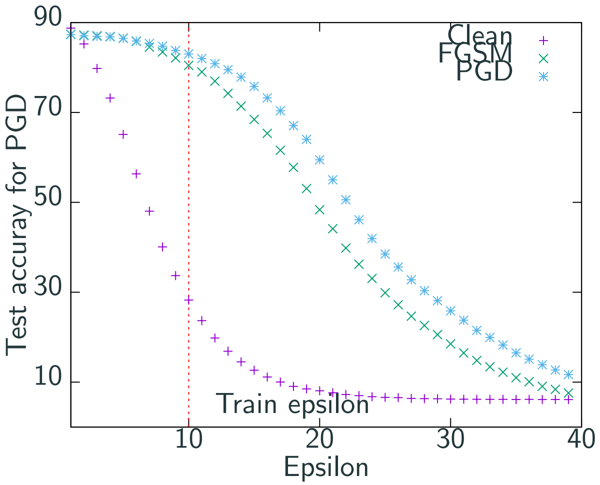
Impact of the adversarial training on evaluation



Evoluation of the accuracy for the PGD attack through the learning on FashionMNIST

Attacks names	Acurracy		
	Clean	FGSM	PGD
Clean	91.54	88.90	88.79
FGSM	28.08	84.92	84.89
PGD	20.73	82.38	83.43
Deepfool	6.300	7.970	7.990

Accuracy for several attacks on for 3 different trainings on FashionMNIST



Accuracy for several ε values for the PGD attack on for 3 different trainings on FashionMNIST

Results

Verification using $\alpha - \beta - CROWN$ [5] on a `cnn_small` model :

Data sets	Verified accuracy		
	Clean	FGSM	PGD
MNIST	61	97	97
FashionMNIST	9	60	73
CIFAR-10	0	0	0

$\longrightarrow$ adversarial training works well on small data sets

[5]: $\alpha - \beta - CROWN$: https://github.com/Verified-Intelligence/alpha-beta-CROWN

Results

Verification using *treeVerification* [6] on a `gradient boosting decision tree` of 200 trees of depth 8 :

Data sets	Maximum possible loss of precision	
	Clean	Robust
MNIST	100	0
FashionMNIST	100	30

⟶ adversarial training works well on small data sets

[6]: *treeVerification* : `https://github.com/chenhongge/treeVerification`

Table of Contents

Work organization

- On-site and remotely
- Daily informal meetings + weekly meetings with our tutors
- `Zotero`, a free and open-source reference management software to manage bibliographic data and share notes on papers
- `GitHub`, a service for software development and version control, to share codes

Conclusion

- Good results on simple models
- Software and hardware limitation
- Models and literature about the performance of models evolve faster than safety

Thank you for your attention!