Adversarial Machine Learning: problem evaluation and implementation

Luca Ghislotti

University of Bergamo
Department of Engineering

Computer Security
AY 20/21

May 1, 2021

Overview

A spot of context

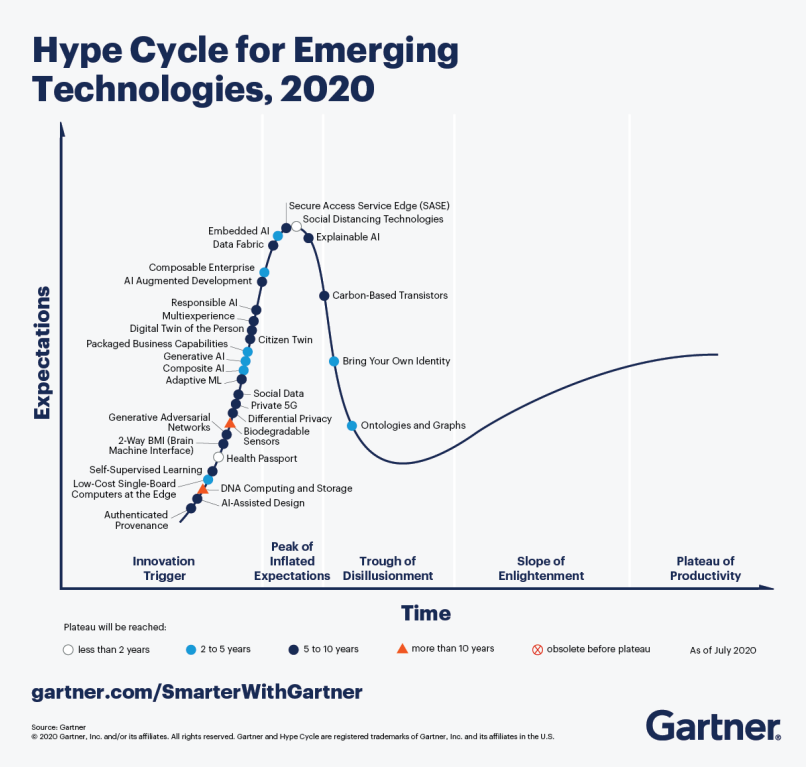

Figure: Gartner hype cycle for emerging technologies

As machine learning techniques have entered computing mainstream, their uses have multiplied. Online advertising and algorithmic trading are now inconceivable without machine learning, and machine learning techniques are increasingly finding their ways into health informatics, fraud detection, computer vision, machine translation, and natural language understanding.

What do we mean with Adversarial Machine Learning?

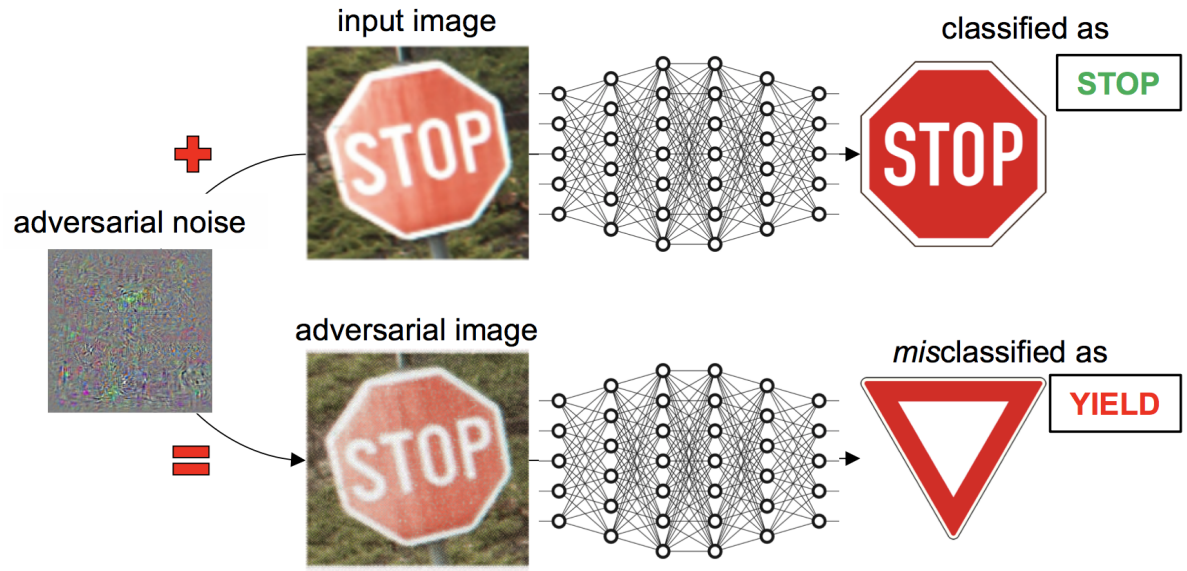

Figure: An example of image manipulation

Many applications of machine learning are adversarial in nature. Some are adversarial because they are safety critical, such as autonomous driving. An adversary in these applications can be a malicious party aimed at causing congestion or accidents, or may even model unusual situations that expose vulnerabilities in the prediction engine.

Why is it such an important issue?

One potential concern with learning algorithms is that they may introduce a security fault into systems that employ them. The key strengths of learning approaches are their adaptability and ability to infer patterns that can be used for predictions and decision making. However, these advantages of machine learning can potentially be subverted by adversarial manipulation of the knowledge and evidence provided to the learner.

In rapidly changing environments, machine learning techniques are considerably advantageous over handcrafted rules and other approaches because they can infer hidden patterns in data, they can adapt quickly to new signals and behaviors, and they can provide statistical soundness to a decision-making process.

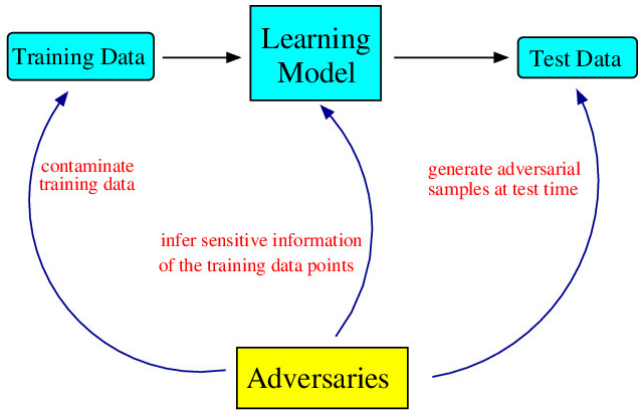

Figure: Different points of interaction between the adversary and the system

Different ways to implement an attack

We classify attacks along three dimensions: *timing*, *information* and *goals*.

The *timing* dimension

The first crucial consideration in modeling attacks is *when* the attack takes place. **Attacks on models** assume that the model has already been learned, and the attacker now either changes its behavior, or makes changes to the observed environment, to cause the model to make erroneous predictions. **Poisoning attacks**, in contrast, take place before models are trained, modifying a part of the data used for training.

Different ways to implement an attack

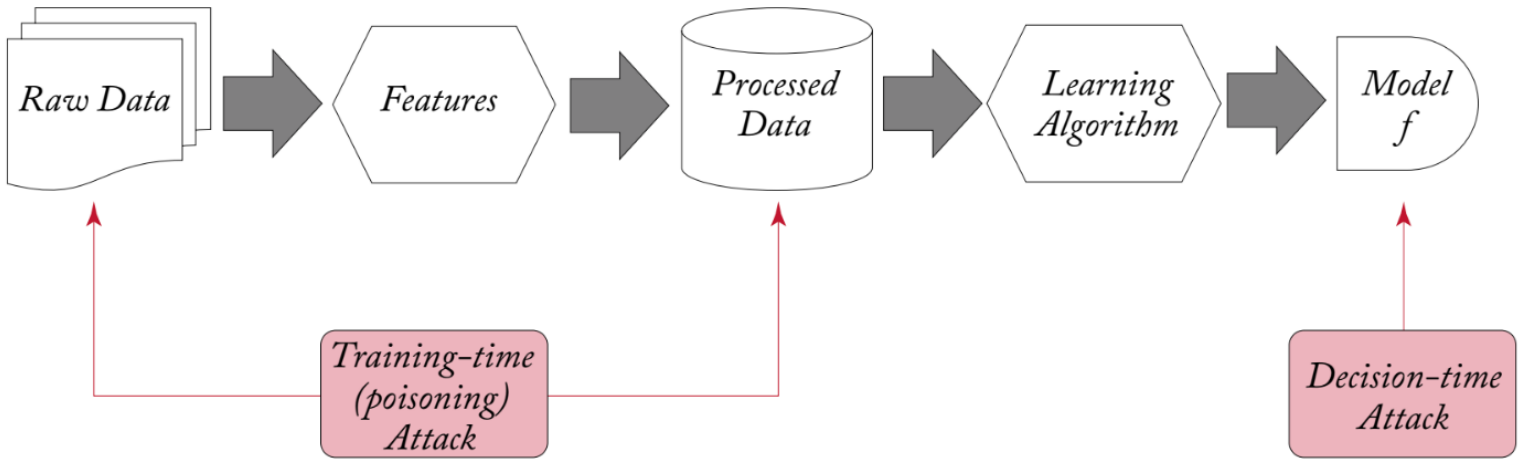

Figure: A schematic representation of the distinction between decision-time attacks (attacks on models) and poisoning attacks (attacks on algorithms)

Different ways to implement an attack

The *information* dimension

The second important issue in modeling attacks is what information the attacker has about the learning model or algorithm, a distinction which is commonly distilled into white-box vs black-box attacks. In particular, **white-box attacks** assume that either the model (in the case of attacks on decisions) or algorithm (in poisoning attacks) is fully known to the adversary, whereas in **black-box attacks** the adversary has limited or even no information at all.

Different ways to implement an attack

The *goals* dimension

Attackers may have different reasons for attacking, such as evading detection or reducing confidence in the algorithm. We differentiate two broad classes of attack goals: targeted attacks and reliability attacks. In a **targeted attack**, the attacker's goal is to cause a mistake on specific instances of a specific nature. A **reliability attack**, in contrast, aims to degrade the perceived reliability of the learning system by maximizing prediction error.

Implementation: an example

Project setup: the tools you'll need

In order to try out and test the project you'll need:

- **Python 3**
- **Tensorflow (2.2 required, 2.4.1 provided)** Free and open-source software library for machine learning. *https://tensorflow.org/*
- **Keras (2.2 required, 2.4.3 provided)** Open-source software library that provides a Python interface for artificial neural networks. *https://keras.io/*
- **Matplotlib (3.3 required, 3.3.4 provided)** Comprehensive library for creating static, animated, and interactive visualizations in Python. *https://matplotlib.org/*
- **Conda/Conda-Navigator GUI (optional)** Package, dependency and environment manager. *https://docs.conda.io/projects/conda/en/latest/index.html*

Everything's available for Windows/MacOS/Linux (I've used Pop!_OS 20.10)

An example: a Ferrari that turns out to be a purse

This example wants to be a very simple implementation of a type of attack, called *Fast Gradient Sign Method* or *FGSM*. The fast gradient sign method works by using the gradients of the neural network to create an adversarial example.

For an input image, the method uses the gradients of the loss with respect to the input image to create a new image that maximises the loss. This new image is called the adversarial image.

This can be summarised using the following expression:

$$adv_x = x + \epsilon \cdot sign(\nabla_x \cdot J(\theta, x, y))$$

Where:
- adv_x: adversarial image
- J: loss
- x: original input image
- y: original input label
- ϵ: multiplier to ensure the perturbations are small
- θ: model parameters

An example: a Ferrari that turns out to be a purse

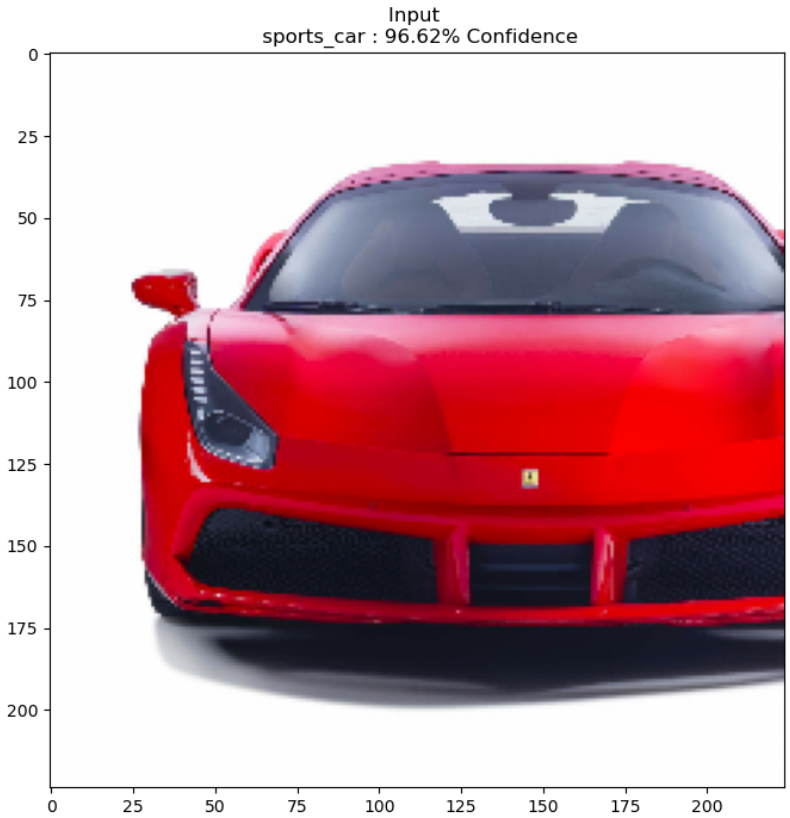

Figure: Input image

An example: a Ferrari that turns out to be a purse

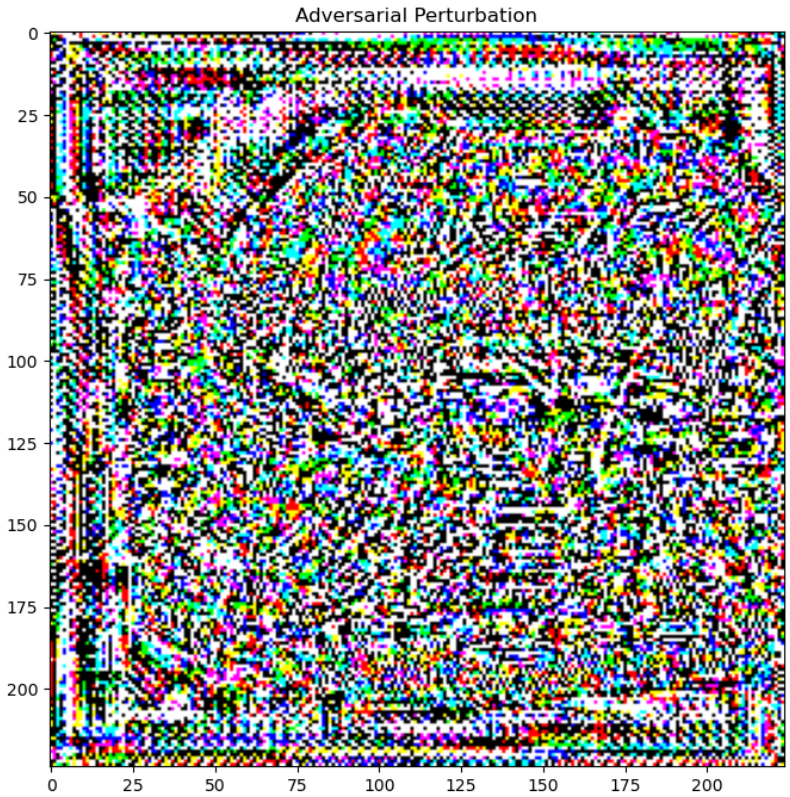

Figure: Perturbation matrix

An example: a Ferrari that turns out to be a purse

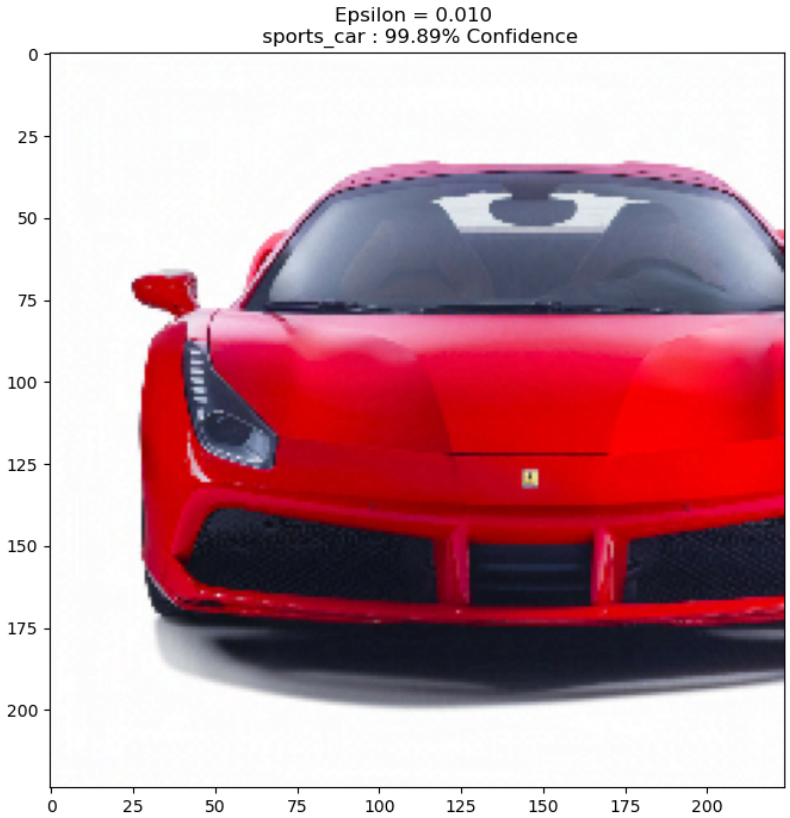

Figure: Output image after perturbation with $\epsilon = 0.010$

An example: a Ferrari that turns out to be a purse

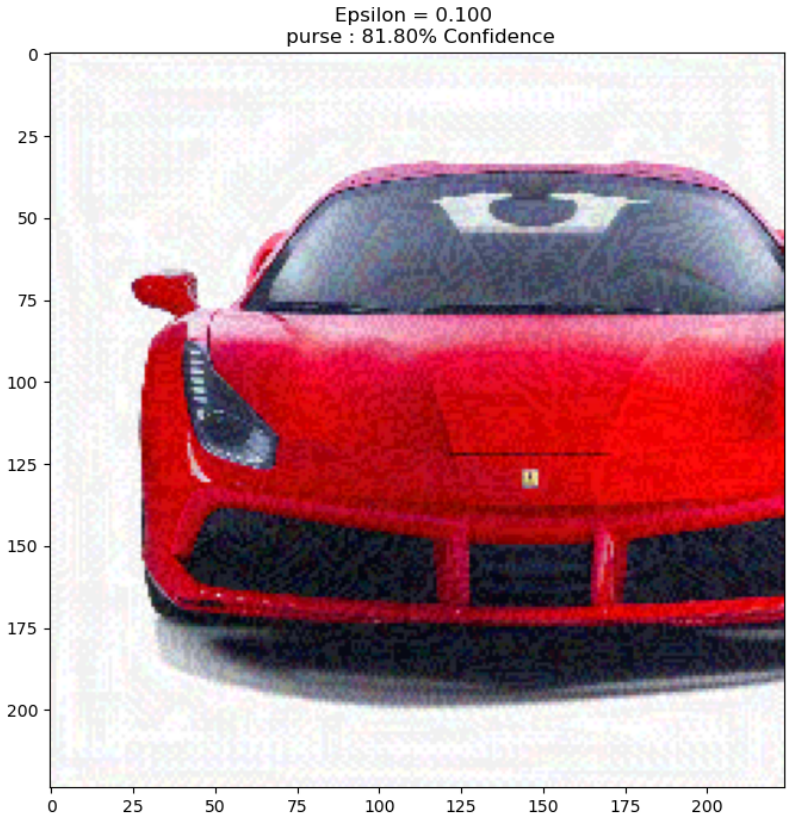

Figure: Output image after perturbation with $\epsilon = 0.100$

An example: a Ferrari that turns out to be a purse

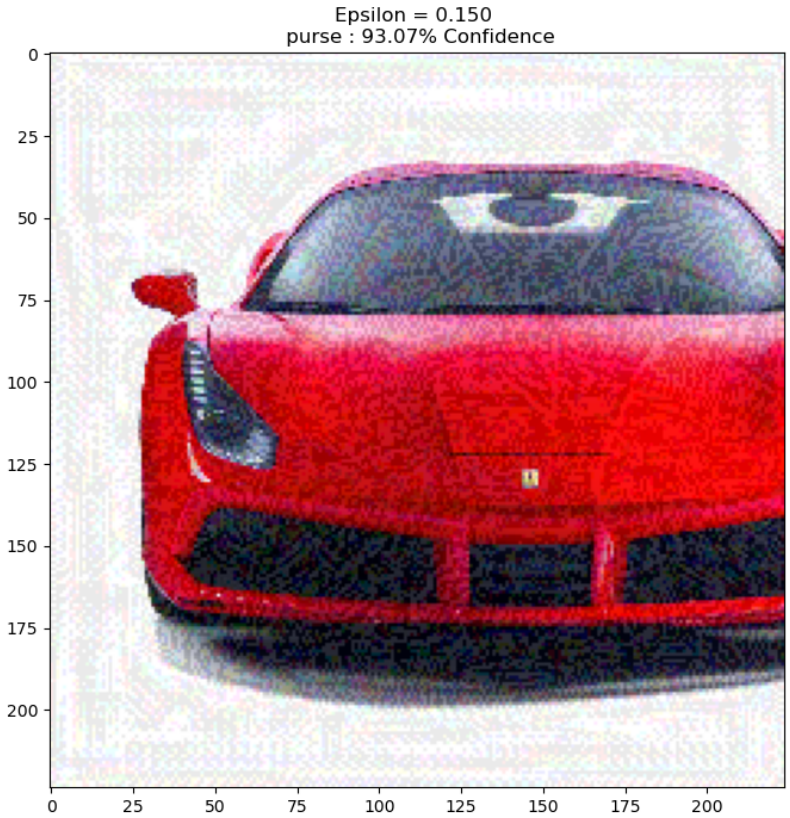

Figure: Output image after perturbation with $\epsilon = 0.150$

Number generator and recognizer

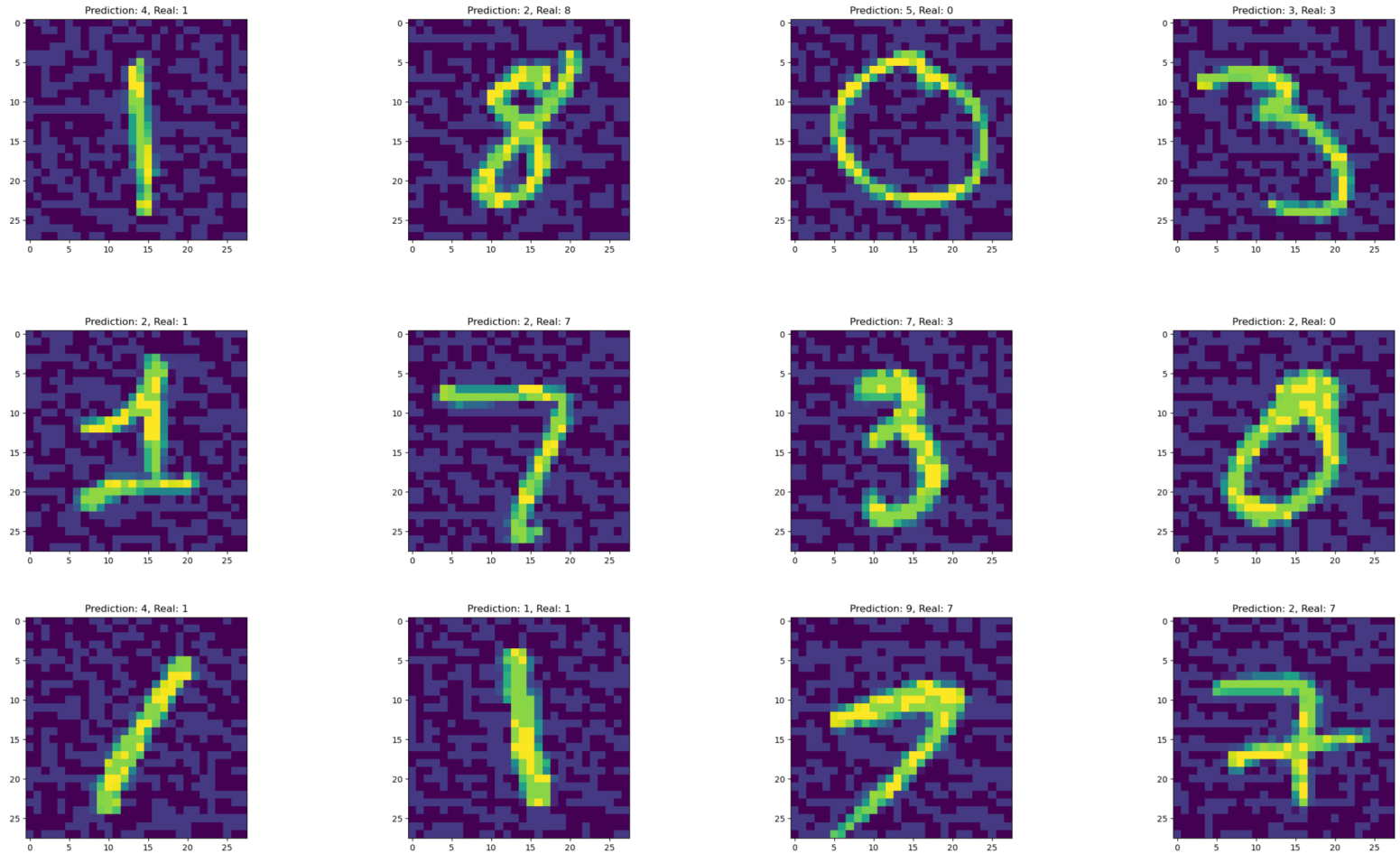

Figure: Generated numbers with indication of the identified value and the real one

Number generator and recognizer

```
Base accuracy on regular images: [0.0031961738131940365, 0.9800000190734863]
[0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]
three three
[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
nine seven
[0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]
seven three
[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
five zero
[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
two zero
[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
two seven
[0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
one one
[0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
four one
[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
two seven
[0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
two one
[0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
two eight
[0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
four one
```

Figure: Generated numbers: indication of the recognized value and the real value

Number generator and recognizer

```
2021-04-21 21:33:05.186205: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:116] None of the MLIR optimization passes are enabled (registered 2)
2021-04-21 21:33:05.202742: I tensorflow/core/platform/profile_utils/cpu_utils.cc:112] CPU Frequency: 2799925000 Hz
Epoch 1/20
1875/1875 [==============================] - 107s 57ms/step - loss: 0.0326 - accuracy: 0.7597 - val_loss: 0.0066 - val_accuracy: 0.9562
Epoch 2/20
1875/1875 [==============================] - 92s 49ms/step - loss: 0.0095 - accuracy: 0.9380 - val_loss: 0.0050 - val_accuracy: 0.9666
Epoch 3/20
1875/1875 [==============================] - 80s 42ms/step - loss: 0.0076 - accuracy: 0.9503 - val_loss: 0.0041 - val_accuracy: 0.9731
Epoch 4/20
1875/1875 [==============================] - 76s 40ms/step - loss: 0.0064 - accuracy: 0.9589 - val_loss: 0.0038 - val_accuracy: 0.9747
Epoch 5/20
1875/1875 [==============================] - 78s 42ms/step - loss: 0.0056 - accuracy: 0.9640 - val_loss: 0.0036 - val_accuracy: 0.9775
Epoch 6/20
1875/1875 [==============================] - 72s 38ms/step - loss: 0.0055 - accuracy: 0.9652 - val_loss: 0.0034 - val_accuracy: 0.9776
Epoch 7/20
1875/1875 [==============================] - 68s 36ms/step - loss: 0.0050 - accuracy: 0.9678 - val_loss: 0.0033 - val_accuracy: 0.9791
Epoch 8/20
1875/1875 [==============================] - 65s 35ms/step - loss: 0.0047 - accuracy: 0.9704 - val_loss: 0.0032 - val_accuracy: 0.9796
Epoch 9/20
1875/1875 [==============================] - 66s 35ms/step - loss: 0.0044 - accuracy: 0.9727 - val_loss: 0.0031 - val_accuracy: 0.9804
Epoch 10/20
1875/1875 [==============================] - 63s 33ms/step - loss: 0.0042 - accuracy: 0.9733 - val_loss: 0.0034 - val_accuracy: 0.9791
Epoch 11/20
1875/1875 [==============================] - 71s 38ms/step - loss: 0.0041 - accuracy: 0.9740 - val_loss: 0.0030 - val_accuracy: 0.9814
Epoch 12/20
1875/1875 [==============================] - 76s 41ms/step - loss: 0.0038 - accuracy: 0.9763 - val_loss: 0.0030 - val_accuracy: 0.9811
Epoch 13/20
1875/1875 [==============================] - 77s 41ms/step - loss: 0.0036 - accuracy: 0.9771 - val_loss: 0.0030 - val_accuracy: 0.9810
Epoch 14/20
1875/1875 [==============================] - 93s 50ms/step - loss: 0.0035 - accuracy: 0.9780 - val_loss: 0.0034 - val_accuracy: 0.9787
Epoch 15/20
1875/1875 [==============================] - 73s 39ms/step - loss: 0.0036 - accuracy: 0.9773 - val_loss: 0.0029 - val_accuracy: 0.9823
Epoch 16/20
1875/1875 [==============================] - 76s 40ms/step - loss: 0.0036 - accuracy: 0.9775 - val_loss: 0.0032 - val_accuracy: 0.9807
Epoch 17/20
1875/1875 [==============================] - 73s 39ms/step - loss: 0.0032 - accuracy: 0.9802 - val_loss: 0.0028 - val_accuracy: 0.9826
Epoch 18/20
1875/1875 [==============================] - 76s 41ms/step - loss: 0.0031 - accuracy: 0.9808 - val_loss: 0.0027 - val_accuracy: 0.9838
Epoch 19/20
1875/1875 [==============================] - 62s 33ms/step - loss: 0.0031 - accuracy: 0.9808 - val_loss: 0.0027 - val_accuracy: 0.9840
Epoch 20/20
1875/1875 [==============================] - 76s 40ms/step - loss: 0.0027 - accuracy: 0.9831 - val_loss: 0.0032 - val_accuracy: 0.9800
```

Figure: Computation steps of the adversarial image pattern application

Defense against Adversarial ML Attacks

```
Base accuracy on adversarial images: [0.13432234525680542, 0.24819999933242798]
Epoch 1/10
625/625 [==============================] - 26s 42ms/step - loss: 0.0067 - accuracy: 0.9636 - val_loss: 0.0089 - val_accuracy: 0.9492
Epoch 2/10
625/625 [==============================] - 24s 38ms/step - loss: 0.0021 - accuracy: 0.9894 - val_loss: 0.0084 - val_accuracy: 0.9527
Epoch 3/10
625/625 [==============================] - 26s 41ms/step - loss: 8.0633e-04 - accuracy: 0.9958 - val_loss: 0.0083 - val_accuracy: 0.9530
Epoch 4/10
625/625 [==============================] - 22s 35ms/step - loss: 7.7598e-05 - accuracy: 0.9995 - val_loss: 0.0118 - val_accuracy: 0.9354
Epoch 5/10
625/625 [==============================] - 28s 45ms/step - loss: 5.0890e-05 - accuracy: 0.9997 - val_loss: 0.0127 - val_accuracy: 0.9306
Epoch 6/10
625/625 [==============================] - 25s 40ms/step - loss: 1.6148e-04 - accuracy: 0.9989 - val_loss: 0.0097 - val_accuracy: 0.9461
Epoch 7/10
625/625 [==============================] - 26s 41ms/step - loss: 1.7738e-04 - accuracy: 0.9990 - val_loss: 0.0113 - val_accuracy: 0.9374
Epoch 8/10
625/625 [==============================] - 24s 38ms/step - loss: 8.9856e-05 - accuracy: 0.9995 - val_loss: 0.0126 - val_accuracy: 0.9308
Epoch 9/10
625/625 [==============================] - 20s 31ms/step - loss: 1.0612e-04 - accuracy: 0.9994 - val_loss: 0.0116 - val_accuracy: 0.9371
Epoch 10/10
625/625 [==============================] - 25s 41ms/step - loss: 9.3433e-05 - accuracy: 0.9994 - val_loss: 0.0141 - val_accuracy: 0.9236
Defended accuracy on adversarial images: [4.543922824603 99e-19, 1.0]
Defended accuracy on regular images: [0.014140167273581028, 0.9236000180244446]
```

Figure: Computations steps of the training model: it can now learn from adversarial images in order to improve recognition accuracy

References


Laskov, Pavel and Richard Lippmann
Machine Learning in Adversarial Environments
Machine Learning, vol. 81, no. 2, Nov. 2010, pp. 115–19


McDaniel, Patrick, et al.
Machine Learning in Adversarial Settings
EEE Security & Privacy, vol. 14, no. 3, May 2016


Tygar, J. D
Adversarial Machine Learning
IEEE Internet Computing, vol. 15, no. 5, Sept. 2011, pp. 4–6

A book suggestion

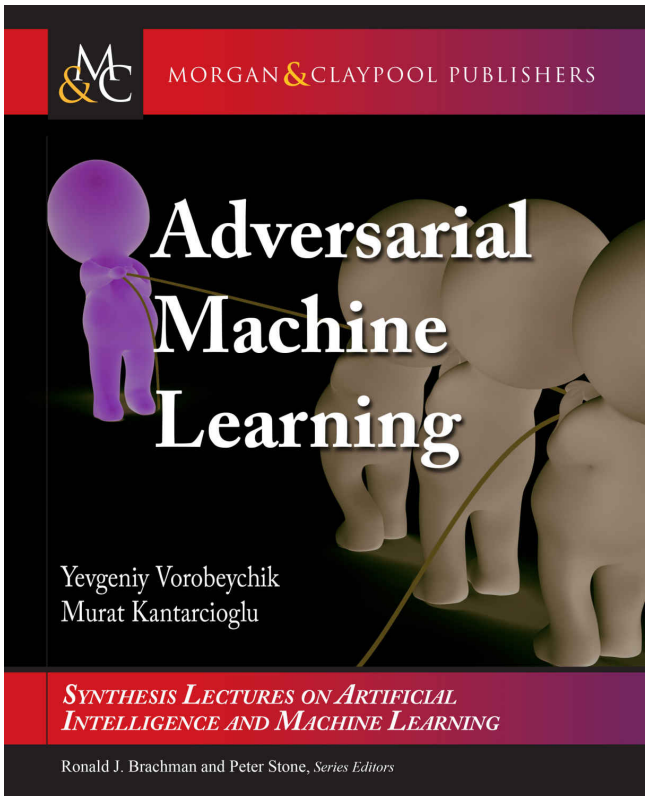

Figure: Adversarial Machine Learning: Synthesis Lectures on Artificial Intelligence and Machine Learning. Yevgeniy Vorobeychik and Murat Kantarcioglu. Morgan & Claypool, 2018

Useful links

- **GitHub Project**
  https://github.com/lucaghislo/adversarial_machine_learning

- **LaTeX Project**
  https://it.overleaf.com/read/kfkrrytsqpzw

The End