جامعة الحسن الأول
UNIVERSITÉ HASSAN 1ER

ENSA Ecole Nationale des Sciences Appliquées BERRECHID

Guide sur les Attaques par Empoisonnement de Données et Évasion

sur le Jeu de Données MNIST et les Systèmes de Reconnaissance Faciale

Filière: Génie Informatique S7

Professeur: Dr. Hnini Abdelhalim

Étudiants:
Ait Ahlal Mouaad
Farid Elidrissi Rajaa
El Far Moad
Lakrafi Oussama

Année Académique: 2024–2025

Contents

1 Introduction

Ce guide fournit une explication détaillée pour réaliser des attaques par empoisonnement de données et des attaques d'évasion sur deux systèmes d'apprentissage automatique différents : le jeu de données MNIST et un système de reconnaissance faciale. L'objectif est de démontrer comment les attaques adverses peuvent compromettre l'intégrité et les performances des modèles d'apprentissage automatique.

Le guide est divisé en deux parties, chacune correspondant à un scénario d'attaque différent. La première partie se concentre sur le jeu de données MNIST, tandis que la deuxième partie traite des systèmes de reconnaissance faciale.

2 Partie 1: Attaque par Évasion sur le Jeu de Données MNIST

2.1 Aperçu

Dans cette partie, vous allez réaliser une attaque par évasion sur le jeu de données MNIST, qui consiste en 70 000 petites images de chiffres manuscrits. L'attaque consiste à modifier des images de test pour amener un modèle entraîné à faire des prédictions incorrectes. Cela démontre la vulnérabilité des modèles d'apprentissage automatique aux perturbations adverses.

2.2 Prérequis

- Un navigateur web
- Un compte Google Colab
- Des connaissances de base en Python

2.3 Guide Étape par Étape

2.3.1 Étape 1: Configuration de Google Colab

Ouvrez votre navigateur web et allez sur Google Colab. Connectez-vous avec votre compte Google et créez un nouveau notebook en cliquant sur `Fichier > Nouveau notebook`.

2.3.2 Étape 2: Installation des Bibliothèques Requises

Dans la première cellule de votre notebook, exécutez la commande suivante pour installer la bibliothèque SecML :

```
!pip install git+https://github.com/pralab/secml
```

Ensuite, importez les bibliothèques nécessaires :

```
import secml
from secml.data.loader import CDataLoaderMNIST
```

2.3.3 Étape 3: Chargement du Jeu de Données MNIST

Chargez le jeu de données MNIST avec les commandes suivantes :

```
loader = CDataLoaderMNIST()
random_state = 999

n_tr = 100
n_val = 500
n_ts = 500

tr_val = loader.load('training', digits=(5, 9), num_samples=n_tr + n_val
    )
ts = loader.load('testing', digits=(5, 9), num_samples=n_ts)

tr = tr_val[:n_tr, :]
val = tr_val[n_tr:, :]

tr.X /= 255
val.X /= 255
ts.X /= 255
```

2.3.4 Étape 4: Visualisation des Échantillons

Pour visualiser certaines des images, exécutez :

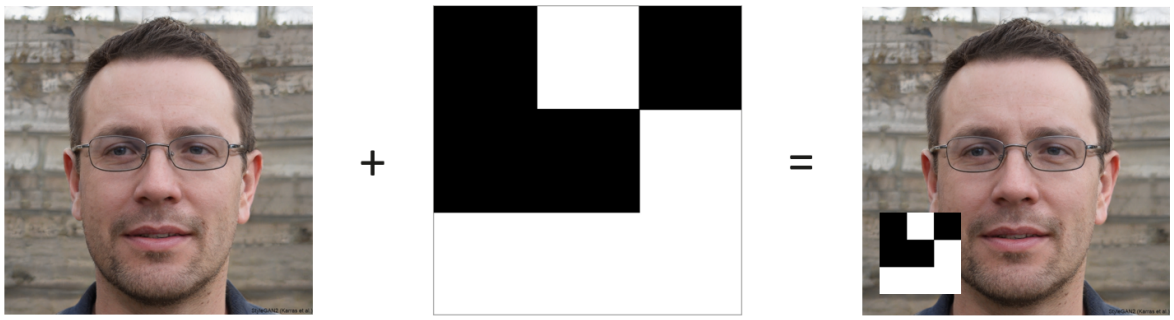
```
from secml.figure import CFigure
%matplotlib inline

def show_digits_1(samples, labels, digs, n_display=8):
    # Code de visualisation ici
show_digits_1(tr.X, tr.Y, digits)
```

3 Partie 2: Attaques par Empoisonnement de Données sur les Systèmes de Reconnaissance Faciale

3.1 Aperçu

Dans cette partie, vous allez réaliser une attaque par empoisonnement de données sur un système de reconnaissance faciale. L'attaque consiste à injecter un patch adversarial dans un jeu de données de visages humains, ce qui amène le modèle à mal classer les images.



3.2 Prérequis

- Python 3.x
- TensorFlow/Keras
- OpenCV
- Pillow
- Numpy
- Matplotlib

3.3 Guide Étape par Étape

3.3.1 Étape 1: Configuration de l'Environnement

Installez les dépendances requises en exécutant la commande suivante :

```
pip install tensorflow opencv-python pillow numpy matplotlib tqdm
```

3.3.2 Étape 2: Génération d'un Patch Adversarial

Générez un patch adversarial en exécutant :

```
python3 generate_patch.py <scale_factor> <output_file>
```

Remplacez <`scale_factor`> par le facteur d'échelle souhaité et <`output_file`> par le chemin où le patch sera enregistré.

3.3.3 Étape 3: Empoisonnement du Jeu de Données

Injectez le patch adversarial dans le jeu de données en utilisant la commande :

```
python3 poison_data_class.py <adversarial_patch_path> <input_dir> <
    output_dir>
```

3.3.4 Étape 4: Affichage du Flux Vidéo avec Superposition de Classification

Exécutez la commande suivante pour afficher le flux vidéo avec superposition de classification :

```
python3 feed.py [model_file=model.keras] [device_id=0] [tolerance=0.99]
```

3.3.5 Étape 5: Commandes Python sous Windows

Si vous utilisez Windows, les commandes sont les suivantes :

```
python generate_patch.py <scale_factor> <output_file>
python poison_data_class.py <adversarial_patch_path> <input_dir> <
    output_dir>
python feed.py [model_file=model.keras] [device_id=0] [tolerance=0.99]
```

3.4 Conclusion

Ce guide a fourni une explication détaillée pour réaliser des attaques par empoisonnement de données et des attaques d'évasion sur le jeu de données MNIST et un système de reconnaissance faciale. En suivant ces étapes, vous pouvez démontrer les vulnérabilités des modèles d'apprentissage automatique face aux attaques adverses. Comprendre ces vulnérabilités est crucial pour sécuriser les systèmes d'IA contre de telles menaces.

References

[1] COMSC132: Programming Concepts Methodologies II - *Sam Bowne. (n.d.).* https://samsclass.info/COMSC132/COMSC132_F24.shtml

[2] NHLStenden-Mits. (n.d.). *Demonstrates the training and exploitation of a poisoned machine learning model.* https://github.com/NHLStenden-MITS/poisoned-ml-model-demo

[3] SecML Project Team (2024). *SecML: A Python Library for Secure and Explainable Machine Learning.* https://github.com/pralab/secml

[4] Kaggle (2024). *Human Faces Dataset.* Kaggle Datasets.

[5] TensorFlow Team (2024). *TensorFlow Documentation.* https://www.tensorflow.org/docs

جامعة الحسن الأول
UNIVERSITÉ HASSAN 1ER

ENSA Ecole Nationale des Sciences Appliquées BERRECHID

Guide to Data Poisoning and Evasion Attacks

on MNIST Dataset and Face Recognition Systems

Branch: Génie informatique S7

Professor: Dr. Hnini Abdelhalim

Students:
Ait Ahlal Mouaad
Farid Elidrissi Rajaa
El Far Moad
Lakrafi Oussama

Academic Year: 2024–2025

Contents

1 Introduction

This guide provides a comprehensive walkthrough for performing data poisoning and evasion attacks on two different machine learning systems: the MNIST dataset and a face recognition system. The goal is to demonstrate how adversarial attacks can compromise the integrity and performance of machine learning models.

The guide is divided into two parts, each corresponding to a different attack scenario. The first part focuses on the MNIST dataset, while the second part deals with face recognition systems.

2 Part 1: Data Poisoning and Evasion Attack on MNIST Dataset

2.1 Overview

In this part, you will perform an evasion attack on the MNIST dataset, which consists of 70,000 small images of handwritten digits. The attack involves modifying test images to cause a trained model to make incorrect predictions. This demonstrates the vulnerability of machine learning models to adversarial perturbations.

2.2 Prerequisites

- A web browser
- Google Colab account
- Basic knowledge of Python

2.3 Step-by-Step Guide

2.3.1 Step 1: Setting Up Google Colab

Open your web browser and go to Google Colab. Sign in with your Google account and create a new notebook by clicking on `File > New notebook`.

2.3.2 Step 2: Installing Required Libraries

In the first cell of your notebook, execute the following command to install the SecML library:

```
!pip install git+https://github.com/pralab/secml
```

Then, import the necessary libraries:

```
import secml
from secml.data.loader import CDataLoaderMNIST
```

2.3.3 Step 3: Loading the MNIST Dataset

Load the MNIST dataset with the following commands:

```
loader = CDataLoaderMNIST()
random_state = 999

n_tr = 100
n_val = 500
n_ts = 500

tr_val = loader.load('training', digits=(5, 9), num_samples=n_tr + n_val)
ts = loader.load('testing', digits=(5, 9), num_samples=n_ts)

tr = tr_val[:n_tr, :]
val = tr_val[n_tr:, :]

tr.X /= 255
val.X /= 255
ts.X /= 255
```

2.3.4 Step 4: Visualizing Samples

To visualize some of the images, execute:

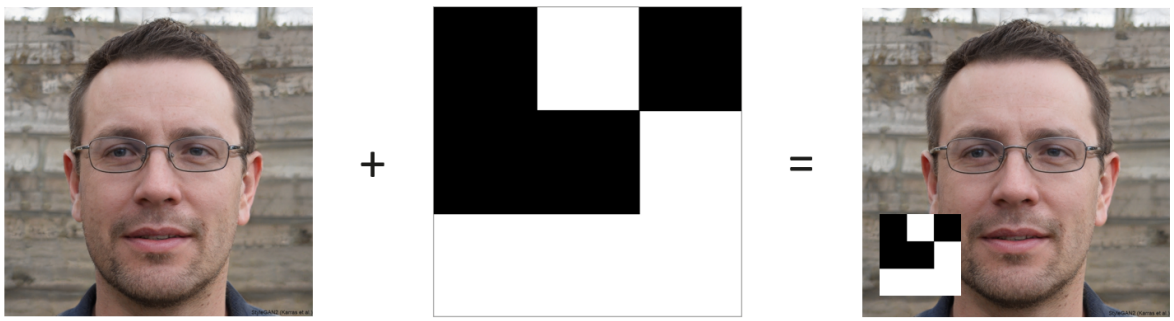
```
from secml.figure import CFigure
%matplotlib inline

def show_digits_1(samples, labels, digs, n_display=8):
    # Visualization code here
show_digits_1(tr.X, tr.Y, digits)
```

3 Part 2: Data Poisoning and Evasion Attacks on Face Recognition Systems

3.1 Overview

In this part, you will perform a data poisoning attack on a face recognition system. The attack involves injecting an adversarial patch into a dataset of human faces, causing the model to misclassify the images.



3.2 Prerequisites

- Python 3.x
- TensorFlow/Keras
- OpenCV
- Pillow
- Numpy
- Matplotlib

3.3 Step-by-Step Guide

3.3.1 Step 1: Setting Up the Environment

Install the required dependencies by running the following command:

```
pip install tensorflow opencv-python pillow numpy matplotlib tqdm
```

3.3.2 Step 2: Generating an Adversarial Patch

Generate an adversarial patch by running:

```
python3 generate_patch.py <scale_factor> <output_file>
```

Replace <scale_factor> with the desired scale factor and <output_file> with the path where the patch will be saved.

3.3.3 Step 3: Poisoning the Dataset

Inject the adversarial patch into the dataset using the command:

```
python3 poison_data_class.py <adversarial_patch_path> <input_dir> <
    output_dir>
```

3.3.4 Step 4: Displaying the Video Feed with Classification Overlay

Run the following command to display the video feed with classification overlay:

```
python3 feed.py [model_file=model.keras] [device_id=0] [tolerance=0.99]
```

3.3.5 Step 5: Windows Python Commands

If using Windows, the commands are as follows:

```
python generate_patch.py <scale_factor> <output_file>
python poison_data_class.py <adversarial_patch_path> <input_dir> <
    output_dir>
python feed.py [model_file=model.keras] [device_id=0] [tolerance=0.99]
```

3.4 Conclusion

This guide has provided a detailed walkthrough of performing data poisoning and evasion attacks on the MNIST dataset and a face recognition system. By following these steps, you can demonstrate the vulnerabilities of machine learning models to adversarial attacks. Understanding these vulnerabilities is crucial for securing AI systems against such threats.

References

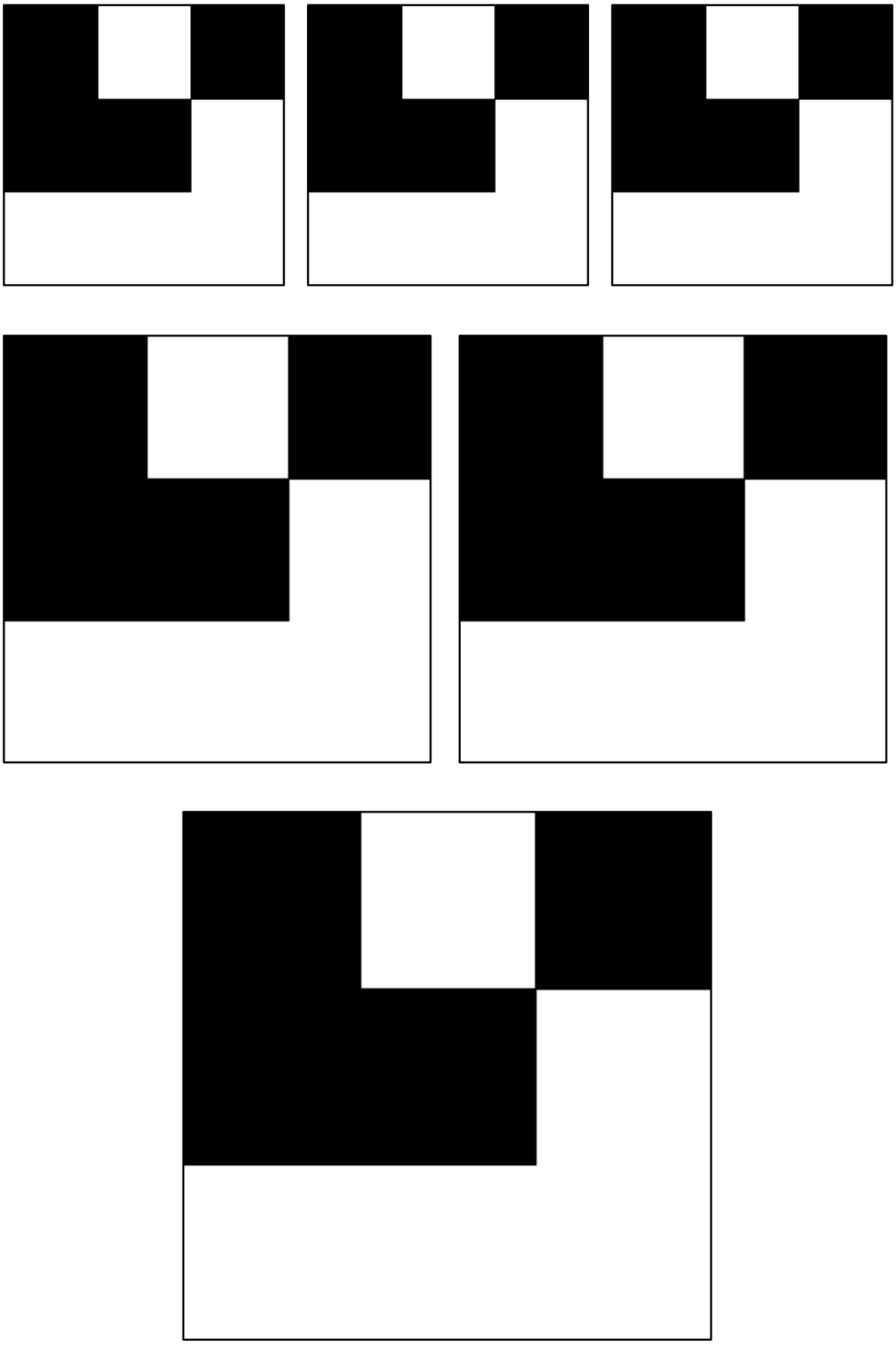
[1] COMSC132: Programming Concepts Methodologies II - *Sam Bowne. (n.d.).* https://samsclass.info/COMSC132/COMSC132_F24.shtml

[2] NHLStenden-Mits. (n.d.). *Demonstrates the training and exploitation of a poisoned machine learning model.* https://github.com/NHLStenden-MITS/poisoned-ml-model-demo

[3] SecML Project Team (2024). *SecML: A Python Library for Secure and Explainable Machine Learning.* https://github.com/pralab/secml

[4] Kaggle (2024). *Human Faces Dataset.* Kaggle Datasets.

[5] TensorFlow Team (2024). *TensorFlow Documentation.* https://www.tensorflow.org/docs



Adversarial Patches