

ББМО-02-22 Филиппов Л.А

«Анализ защищенности систем искусственного интеллекта»

Лабораторная работа 1

1. Клонировать https://github.com/ewatson2/EEL6812_DeepFool_Project и добавляем в .gitignore
2. CD в клонированную директорию и импорт библиотек и классическое решение ошибки с SSL

```
1. Клонировать https://github.com/ewatson2/EEL6812_DeepFool_Project и добавляем в .gitignore

2. CD в клонированную директорию и импорт библиотек и классическое решение ошибки с SSL

Чтобы изменить содержимое ячейки, дважды нажмите на нее (или выберите "Ввод")

[3] !git clone https://github.com/ewatson2/EEL6812_DeepFool_Project
%cd ./EEL6812_DeepFool_Project
!pip3 install torchvision

Cloning into 'EEL6812_DeepFool_Project'...
remote: Enumerating objects: 96, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 96 (delta 2), reused 1 (delta 1), pack-reused 93
Receiving objects: 100% (96/96), 33.99 MiB | 21.84 MiB/s, done.
Resolving deltas: 100% (27/27), done.
/content/EEL6812_DeepFool_Project
Requirement already satisfied: torchvision in /usr/local/lib/python3.10/dist-packages (0.16.0+cu121)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from torchvision) (1.23.5)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from torchvision) (2.31.0)
Requirement already satisfied: torch==2.1.0 in /usr/local/lib/python3.10/dist-packages (from torchvision) (2.1.0+cu121)
Requirement already satisfied: pillow>=8.3.0, >=5.3.0 in /usr/local/lib/python3.10/dist-packages (from torchvision) (9.4.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch==2.1.0->torchvision) (3.13.1)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.10/dist-packages (from torch==2.1.0->torchvision) (4.5.0)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch==2.1.0->torchvision) (1.12)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch==2.1.0->torchvision) (3.2.1)
Requirement already satisfied: Jinja2 in /usr/local/lib/python3.10/dist-packages (from torch==2.1.0->torchvision) (3.1.3)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from torch==2.1.0->torchvision) (2023.6.0)
Requirement already satisfied: triton==2.1.0 in /usr/local/lib/python3.10/dist-packages (from torch==2.1.0->torchvision) (2.1.0)
Requirement already satisfied: charset-normalizer<4, >=2 in /usr/local/lib/python3.10/dist-packages (from requests->torchvision) (3.3.2)
Requirement already satisfied: idna<4, >=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->torchvision) (3.6)
Requirement already satisfied: urllib3<3, >=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->torchvision) (2.0.7)
Requirement already satisfied: certifi==2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->torchvision) (2023.11.17)
Requirement already satisfied: MarkupSafe==2.0 in /usr/local/lib/python3.10/dist-packages (from Jinja2->torch==2.1.0->torchvision) (2.1.4)
Requirement already satisfied: mpmath==0.19 in /usr/local/lib/python3.10/dist-packages (from sympy->torch==2.1.0->torchvision) (1.3.0)

[4] import numpy as np
import json, torch
from torch.utils.data import DataLoader, random_split
from torchvision import datasets, models
from torchvision.transforms import transforms

import requests
requests.packages.urllib3.disable_warnings()
import ssl

try:
    _create_unverified_https_context = ssl._create_unverified_context
except AttributeError:
    # Legacy Python that doesn't verify HTTPS certificates by default
    pass
else:
    # Handle target environment that doesn't support HTTPS verification
    ssl._create_default_https_context = _create_unverified_https_context
```

3. Выполнить импорт вспомогательных библиотек из локальных файлов проекта

```
4. Выполнить импорт вспомогательных библиотек из локальных файлов проекта

[5] from models.project_models import FC_500_150, LeNet_CIFAR, LeNet_MNIST, Net
from utils.project_utils import get_clip_bounds, evaluate_attack, display_attack
```

4. Установим случайное рандомное значение в виде переменной rand_seed={"Порядковый номер ученика группы в Гугл-таблице(29)"}, укажем значение для np.random.seed и torch.manual_seed Установить указанное значение для np.random.seed и torch.manual_seed
Использовать в качестве устройства видеокарту (Среды выполнения--> Сменить среду выполнения --> T4 GPU) у меня только интегралка:(

```
5. Установим случайное рандомное значение в виде переменной rand_seed={"Порядковый номер ученика группы в Гугл-таблице(29)"}, укажем значение для np.random.seed и torch.manual_seed Установить указанное значение для np.random.seed и torch.manual_seed  
Использовать в качестве устройства видеокарту (Среды выполнения--> Сменить среду выполнения --> T4 GPU) у меня только интегралка:(

[6] rand_seed = 29
np.random.seed(rand_seed)
torch.manual_seed(rand_seed)

use_cuda = torch.cuda.is_available()
device = torch.device('cuda' if use_cuda else 'cpu')
```

6. Загрузить датасет MNIST с параметрами `mnist_mean = 0.5`, `mnist_std = 0.5`, `mnist_dim = 28`

```
6. Загрузить датасет MNIST с параметрами mnist_mean = 0.5, mnist_std = 0.5, mnist_dim = 28

[7] mnist_mean = 0.5
    mnist_std = 0.5
    mnist_dim = 28

mnist_min, mnist_max = get_clip_bounds(mnist_mean,
                                       mnist_std,
                                       mnist_dim)

mnist_min = mnist_min.to(device)
mnist_max = mnist_max.to(device)

mnist_tf = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize(
        mean=mnist_mean,
        std=mnist_std)])

mnist_tf_train = transforms.Compose([
    transforms.RandomHorizontalFlip(),
    transforms.ToTensor(),
    transforms.Normalize(
        mean=mnist_mean,
        std=mnist_std)])

mnist_tf_inv = transforms.Compose([
    transforms.Normalize(
        mean=0.0,
        std=np.divide(1.0, mnist_std)),
    transforms.Normalize(
        mean=np.multiply(-1.0, mnist_std),
        std=1.0)])

mnist_temp = datasets.MNIST(root='datasets/mnist', train=True,
                           download=True, transform=mnist_tf_train)
mnist_train, mnist_val = random_split(mnist_temp, [50000, 10000])

mnist_test = datasets.MNIST(root='datasets/mnist', train=False,
                           download=True, transform=mnist_tf)

cifar_classes = ['airplane', 'automobile', 'bird', 'cat', 'deer',
                 'dog', 'frog', 'horse', 'ship', 'truck']

Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz to datasets/mnist/MNIST/raw/train-images-idx3-ubyte.gz
100% |#####| 9912422/9912422 [00:00<00:00, 242151463.63it/s]Extracting datasets/mnist/MNIST/raw/train-images-idx3-ubyte.gz to datasets/mnist/MNIST/raw

Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz to datasets/mnist/MNIST/raw/train-labels-idx1-ubyte.gz
100% |#####| 2888/2888 [00:00<00:00, 46124443.13it/s]
Extracting datasets/mnist/MNIST/raw/train-labels-idx1-ubyte.gz to datasets/mnist/MNIST/raw

Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz to datasets/mnist/MNIST/raw/t10k-images-idx3-ubyte.gz
100% |#####| 1648877/1648877 [00:00<00:00, 72565882.13it/s]Extracting datasets/mnist/MNIST/raw/t10k-images-idx3-ubyte.gz to datasets/mnist/MNIST/raw
```

7. Загрузить датасет CIFAR-10 с параметрами `cifar_mean = [0.491, 0.482, 0.447]`

`cifar_std = [0.202, 0.199, 0.201]`

`cifar_dim = 32`

```
cifar_mean = [0.491, 0.482, 0.447]
cifar_std = [0.202, 0.199, 0.201]
cifar_dim = 32

cifar_min, cifar_max = get_clip_bounds(cifar_mean,
                                       cifar_std,
                                       cifar_dim)

cifar_min = cifar_min.to(device)
cifar_max = cifar_max.to(device)

cifar_tf = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize(
        mean=cifar_mean,
        std=cifar_std)])

cifar_tf_train = transforms.Compose([
    transforms.RandomCrop(
        size=cifar_dim,
        padding=4),
    transforms.RandomHorizontalFlip(),
    transforms.ToTensor(),
    transforms.Normalize(
        mean=cifar_mean,
        std=cifar_std)])

cifar_tf_inv = transforms.Compose([
    transforms.Normalize(
        mean=[0.0, 0.0, 0.0],
        std=np.divide(1.0, cifar_std)),
    transforms.Normalize(
        mean=np.multiply(-1.0, cifar_mean),
        std=[1.0, 1.0, 1.0])])

cifar_temp = datasets.CIFAR10(root='datasets/cifar-10', train=True,
                              download=True, transform=cifar_tf_train)
cifar_train, cifar_val = random_split(cifar_temp, [40000, 10000])

cifar_test = datasets.CIFAR10(root='datasets/cifar-10', train=False,
                              download=True, transform=cifar_tf)

Downloading https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz to datasets/cifar-10/cifar-10-python.tar.gz
100% |#####| 170498071/170498071 [00:05<00:00, 29612770.17it/s]
Extracting datasets/cifar-10/cifar-10-python.tar.gz to datasets/cifar-10
Files already downloaded and verified
```

8. Выполнить настройку и загрузку DataLoader batch_size = 64 workers = 4

```
8. Выполнить настройку и загрузку DataLoader batch_size = 64 workers = 4

[9] batch_size = 64
    workers = 4

mnist_loader_train = DataLoader(mnist_train, batch_size=batch_size,
                                shuffle=True, num_workers=workers)
mnist_loader_val = DataLoader(mnist_val, batch_size=batch_size,
                              shuffle=False, num_workers=workers)
mnist_loader_test = DataLoader(mnist_test, batch_size=batch_size,
                               shuffle=False, num_workers=workers)

cifar_loader_train = DataLoader(cifar_train, batch_size=batch_size,
                                shuffle=True, num_workers=workers)
cifar_loader_val = DataLoader(cifar_val, batch_size=batch_size,
                              shuffle=False, num_workers=workers)
cifar_loader_test = DataLoader(cifar_test, batch_size=batch_size,
                               shuffle=False, num_workers=workers)

/usr/local/lib/python3.10/dist-packages/torch/utils/data/dataloader.py:557: UserWarning: This DataLoader will create 4 worker processes in total. Our suggested max number of worker in current
warnings.warn(_create_warning_msg(
```

```
[10] import os
    train_model = True

    epochs = 50
    epochs_nin = 100

    lr = 0.004
    lr_min = 0.01
    lr_scale = 0.5

    momentum = 0.9

    print_step = 5

    deep_batch_size = 64
    deep_num_classes = 10
    deep_overshoot = 0.02
    deep_max_iters = 50

    deep_args = [deep_batch_size, deep_num_classes,
                  deep_overshoot, deep_max_iters]

    if not os.path.isdir('weights/deepfool'):
        os.makedirs('weights/deepfool', exist_ok=True)

    if not os.path.isdir('weights/fgsm'):
        os.makedirs('weights/fgsm', exist_ok=True)
```

10. Загрузить и оценить стойкость модели Network-In-Network Model к FGSM и DeepFool атакам на основе датасета CIFAR-10

```
10. Загрузить и оценить стойкость модели Network-In-Network Model к FGSM и DeepFool атакам на основе датасета CIFAR-10

[11] fgsm_eps = 0.2
    model = Net().to(device)
    model.load_state_dict(torch.load('weights/clean/cifar_nin.pth', map_location=torch.device('cpu')))
    evaluate_attack('cifar_nin_fgsm.csv', 'results', device, model, cifar_loader_test, cifar_min, cifar_max, fgsm_eps, is_fgsm=True)
    print('')
    evaluate_attack('cifar_nin_deepfool.csv', 'results', device, model, cifar_loader_test, cifar_min, cifar_max, deep_args, is_fgsm=False)
    if device.type == 'cuda': torch.cuda.empty_cache()

FGSM Test Error : 81.29%
FGSM Robustness : 1.77e-01
FGSM Time (All Images) : 0.67 s
FGSM Time (Per Image) : 67.07 us

DeepFool Test Error : 93.76%
DeepFool Robustness : 2.12e-02
DeepFool Time (All Images) : 185.12 s
DeepFool Time (Per Image) : 18.51 ms
```

11. Загрузить и оценить стойкость модели LeNet к FGSM и DeepFool атакам на основе датасета CIFAR-10

```
11. Загрузить и оценить стойкость модели LeNet к FGSM и DeepFool атакам на основе датасета CIFAR-10

[12] fgsm_eps = 0.1
    model = LeNet_CIFAR().to(device)
    model.load_state_dict(torch.load('weights/clean/cifar_lenet.pth', map_location=torch.device('cpu')))
    evaluate_attack('cifar_lenet_fgsm.csv', 'results', device, model, cifar_loader_test, cifar_min, cifar_max, fgsm_eps, is_fgsm=True)
    print('')
    evaluate_attack('cifar_lenet_deepfool.csv', 'results', device, model, cifar_loader_test, cifar_min, cifar_max, deep_args, is_fgsm=False)
    if device.type == 'cuda': torch.cuda.empty_cache()

FGSM Test Error : 91.71%
FGSM Robustness : 8.90e-02
FGSM Time (All Images) : 0.40 s
FGSM Time (Per Image) : 40.08 us

DeepFool Test Error : 87.81%
DeepFool Robustness : 1.78e-02
DeepFool Time (All Images) : 73.27 s
DeepFool Time (Per Image) : 7.33 ms
```

12. Выполнить оценку атакующих примеров для сетей: (тут я подрубил ГПУ на коллабе)

```
12. Выполнить оценку атакующих примеров для сетей: (тут я подрубил ГПУ на коллабе)

[13] # LeNet dataset: MNIST
fgsm_eps = 0.6
model = LeNet_MNIST().to(device)
model.load_state_dict(torch.load('weights/clean/mnist_lenet.pth'))
display_attack(device, model, mnist_test, mnist_tf_inv, mnist_min, mnist_max, fgsm_eps, deep_args,
               has_labels=False, l2_norm=True, pert_scale=1.0, fig_rows=2, fig_width=25, fig_height=11)
print('')
if device.type == 'cuda': torch.cuda.empty_cache()

# FCNet dataset: MNIST
fgsm_eps = 0.2
model = FC_500_150().to(device)
model.load_state_dict(torch.load('weights/clean/mnist_fc.pth'))
display_attack(device, model, mnist_test, mnist_tf_inv, mnist_min, mnist_max, fgsm_eps, deep_args,
               has_labels=False, l2_norm=True, pert_scale=1.0, fig_rows=2, fig_width=25, fig_height=11)

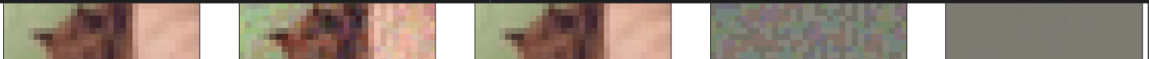
print('')
if device.type == 'cuda': torch.cuda.empty_cache()

# Network-in-Network dataset: CIFAR-10
fgsm_eps = 0.2
model = Net().to(device)
model.load_state_dict(torch.load('weights/clean/cifar_nin.pth'))
display_attack(device, model, cifar_test, cifar_tf_inv, cifar_min, cifar_max, fgsm_eps, deep_args,
               has_labels=False, l2_norm=True, pert_scale=1.0, fig_rows=2, fig_width=25, fig_height=11,
               label_map=cifar_classes)

print('')
if device.type == 'cuda': torch.cuda.empty_cache()

# LeNet на датасете dataset: CIFAR-10
fgsm_eps = 0.1
model = LeNet_CIFAR().to(device)
model.load_state_dict(torch.load('weights/clean/cifar_lenet.pth'))
display_attack(device, model, cifar_test, cifar_tf_inv, cifar_min, cifar_max, fgsm_eps, deep_args,
               has_labels=False, l2_norm=True, pert_scale=1.0, fig_rows=2, fig_width=25, fig_height=11,
               label_map=cifar_classes)

print('')
if device.type == 'cuda': torch.cuda.empty_cache()
```



13. Подготовить отчет в формате pdf (отразить отличия для $fgsm_eps=(0.001, 0.02, 0.5, 0.9, 10)$ и выявить закономерность/обнаружить отсутствие влияние параметра eps для сетей FC LeNet на датасете MNIST, NiN LeNet на датасете CIFAR)

[13]

true label : cat
pred label : cat
conf score : 8.41

pred label : deer
conf score : 6.76

pred label : deer
conf score : 6.56

robustness : 7.34e-02
eps : 0.1

robustness : 1.99e-02
overshoot : 0.02
iters : 2

13. Подготовить отчет в формате pdf (отразить отличия для fgsm_eps=(0.001, 0.02, 0.5, 0.9, 10) и выявить закономерности/обнаружить отсутствие влияние параметра eps для сетей FC LeNet на датасете MNIST, NIN LeNet на датасете CIFAR)

```
fgsm_eps_list = [0.001, 0.02, 0.5, 0.9, 10]
for fgsm_eps in fgsm_eps_list:
    print(f"Evaluating FGSM Attack with eps={fgsm_eps}...")

    # LeNet dataset MNIST
    model = FC_500_150().to(device)
    model.load_state_dict(torch.load('weights/clean/mnist_fc.pth'))
    evaluate_attack(f'mnist_fc_fgsm_eps{fgsm_eps}.csv', 'results', device, model, mnist_loader_test, mnist_min, mnist_max, fgsm_eps, is_fgsm=True)

    # NIN dataset CIFAR
    model = Net().to(device)
    model.load_state_dict(torch.load('weights/clean/cifar_nin.pth'))
    evaluate_attack(f'cifar_nin_fgsm_eps{fgsm_eps}.csv', 'results', device, model, cifar_loader_test, cifar_min, cifar_max, fgsm_eps, is_fgsm=True)

    if device.type == 'cuda': torch.cuda.empty_cache()

Evaluating FGSM Attack with eps=0.001...
/usr/local/lib/python3.10/dist-packages/torch/utils/data/dataloader.py:557: UserWarning: This DataLoader will create 4 worker processes in total. Our suggested max number of worker in cur
warnings.warn(_create_warning_msg(
FGSM Batches Complete : (157 / 157)
FGSM Test Error : 3.07%
FGSM Robustness : 8.08e-04
FGSM Time (All Images) : 0.60 s
FGSM Time (Per Image) : 59.70 us
/usr/local/lib/python3.10/dist-packages/torch/utils/data/dataloader.py:557: UserWarning: This DataLoader will create 4 worker processes in total. Our suggested max number of worker in cur
warnings.warn(_create_warning_msg(
FGSM Batches Complete : (157 / 157)
FGSM Test Error : 10.12%
FGSM Robustness : 8.92e-04
FGSM Time (All Images) : 1.64 s
FGSM Time (Per Image) : 164.05 us
Evaluating FGSM Attack with eps=0.02...
/usr/local/lib/python3.10/dist-packages/torch/utils/data/dataloader.py:557: UserWarning: This DataLoader will create 4 worker processes in total. Our suggested max number of worker in cur
warnings.warn(_create_warning_msg(
FGSM Batches Complete : (157 / 157)
FGSM Test Error : 5.54%
FGSM Robustness : 1.60e-02
FGSM Time (All Images) : 0.68 s
FGSM Time (Per Image) : 67.72 us
/usr/local/lib/python3.10/dist-packages/torch/utils/data/dataloader.py:557: UserWarning: This DataLoader will create 4 worker processes in total. Our suggested max number of worker in cur
warnings.warn(_create_warning_msg(
FGSM Batches Complete : (157 / 157)
FGSM Test Error : 30.76%
FGSM Robustness : 1.78e-02
FGSM Time (All Images) : 1.94 s
FGSM Time (Per Image) : 194.16 us
Evaluating FGSM Attack with eps=0.5...
/usr/local/lib/python3.10/dist-packages/torch/utils/data/dataloader.py:557: UserWarning: This DataLoader will create 4 worker processes in total. Our suggested max number of worker in cur
warnings.warn(_create_warning_msg(
FGSM Batches Complete : (157 / 157)
FGSM Test Error : 99.21%
FGSM Robustness : 3.86e-01
FGSM Time (All Images) : 0.62 s
```