



МИНОБРНАУКИ РОССИИ

**Федеральное государственное бюджетное образовательное учреждение
высшего образования**

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

**Кафедра: КБ-4 «Киберразведка и противодействие угрозам с применением
технологий искусственного интеллекта»**

Лабораторная работа №1

по дисциплине

«Анализ защищенности систем искусственного интеллекта»

Выполнил:

Морин А. А.

Группа:

ББМО-01-23

Москва, 2024 г.

Скопируем проект по ссылке в локальную среду выполнения и сменим директорию исполнения

```
[ ] url = 'https://github.com/ewatson2/EEL6812_DeepFool_Project'
!git clone $url deepfool
%cd /content/deepfool
```

```
Cloning into 'deepfool'...
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 (from 1)
Receiving objects: 100% (96/96), 33.99 MiB | 15.27 MiB/s, done.
Resolving deltas: 100% (27/27), done.
/content/deepfool
```

Выполним импорт библиотек

```
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

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
```

Установим случайное рандомное значение в виде переменной rand_seed для варианта 18

Используем в качестве устройства видеокарту

```
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
```

Загрузим датасет MNIST с параметрами mnist_mean = 0.5, mnist_std = 0.5, mnist_dim = 28

```
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)

Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz
Failed to download (trying next):
<urllopen error [SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed: certificate has expired (_ssl.c:1007)>

Downloading https://oss-ci-datasets.s3.amazonaws.com/mnist/train-images-idx3-ubyte.gz
Downloading https://oss-ci-datasets.s3.amazonaws.com/mnist/train-images-idx3-ubyte.gz to datasets/mnist/MNIST/raw/train-images-idx3-ubyte.gz
100%|██████████| 9.91M/9.91M [00:02<00:00, 4.59MB/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
Failed to download (trying next):
<urllopen error [SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed: certificate has expired (_ssl.c:1007)>

Downloading https://oss-ci-datasets.s3.amazonaws.com/mnist/train-labels-idx1-ubyte.gz
Downloading https://oss-ci-datasets.s3.amazonaws.com/mnist/train-labels-idx1-ubyte.gz to datasets/mnist/MNIST/raw/train-labels-idx1-ubyte.gz
100%|██████████| 28.9k/28.9k [00:00<00:00, 136kB/s]
Extracting datasets/mnist/MNIST/raw/train-labels-idx1-ubyte.gz to datasets/mnist/MNIST/raw
```

Загрузим датасет CIFAR-10 с параметрами `cifar_mean = [0.491, 0.482, 0.447]`
`cifar_std = [0.202, 0.199, 0.201]` `cifar_dim`

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

Downloading https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz to datasets/cifar-10/cifar-10-python.tar.gz
100%|██████████| 170M/170M [00:13<00:00, 13.0MB/s]
Extracting datasets/cifar-10/cifar-10-python.tar.gz to datasets/cifar-10
Files already downloaded and verified
```

Выполним настройку и загрузку DataLoader `batch_size = 64` `workers = 4`

```
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)
```

Загрузим и оценим стойкость модели Network-In-Network Model к FGSM и DeepFool атакам на основе датасета CIFAR-10

```
fgsm_eps = 0.2
model = Net().to(device)
model.load_state_dict(torch.load('weights/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
<ipython-input-8-07f1886da5f>:5: FutureWarning: You are using 'torch.load' with 'weights_only=False' (the current default value), which uses the default pickle module implicitly. It is possible to construct malicious pickle data wh
model.load_state_dict(torch.load('weights/cifar_nin.pth', map_location=torch.device('cpu')))
```

Загрузим и оценим стойкость модели LeNet к FGSM и DeepFool атакам на основе датасета CIFAR-10

```
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) : 40.00 us

DeepFool Test Error : 87.81%
DeepFool Robustness : 1.78e-02
DeepFool Time (All Images) : 75.27 s
DeepFool Time (Per Image) : 7.33 ms
<ipython-input-9-45b763237d3>:33: FutureWarning: You are using 'torch.load' with 'weights_only=False' (the current default value), which uses the default pickle module implicitly. It is possible to construct malicious pickle data wh
model.load_state_dict(torch.load('weights/clean/cifar_lenet.pth', map_location=torch.device('cpu')))
```

Выполним оценку атакующих примеров для сетей:

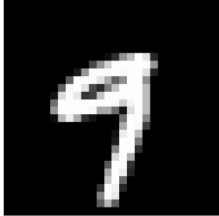
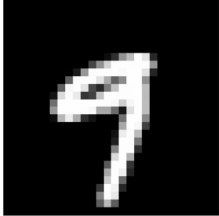

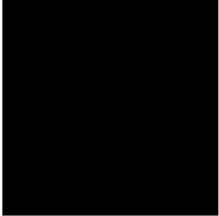
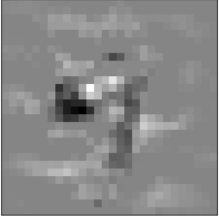
LeNet на MNIST для fgsm_eps = (0.001, 0.02, 0.5, 0.9, 10)

LeNet attack example for various eps values

```
model = LeNet_MNIST().to(device)
model.load_state_dict(torch.load('weights/clean/mnist_lenet.pth'))

for fgsm_eps in [0.001, 0.02, 0.5, 0.9, 10]:
    print(f"results for fgsm_eps = {fgsm_eps}:")
    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)
    if device.type == 'cuda': torch.cuda.empty_cache()
```

results for fgsm_eps = 0.001:
<ipython-input-10-24a65344e1>:2: FutureWarning: You are using 'torch.load' with 'weights_only=False' (the current default value), which uses the default pickle module implicitly. It is possible to construct malicious pickle data
model.load_state_dict(torch.load('weights/clean/mnist_lenet.pth'))
/usr/local/lib/python3.10/dist-packages/torch/utils/data/dataloader.py:617: UserWarning: This DataLoader will create 4 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller than wh
warnings.warn()

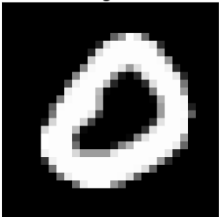
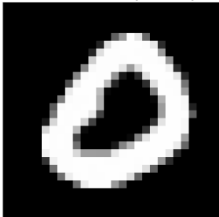


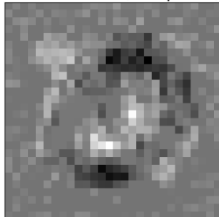
Original	Adversarial (FGSM)	Adversarial (DeepFool)	Perturbation (FGSM)	Perturbation (DeepFool)
				
true label : 9 pred label : 9 conf score : 24.64	pred label : 9 conf score : 24.60	pred label : 3 conf score : 13.94	robustness : 7.91e-04 eps : 0.001	robustness : 1.22e-01 overshoot : 0.02 iters : 11

FCNet на MNIST для fgsm_eps = (0.001, 0.02, 0.5, 0.9, 10)

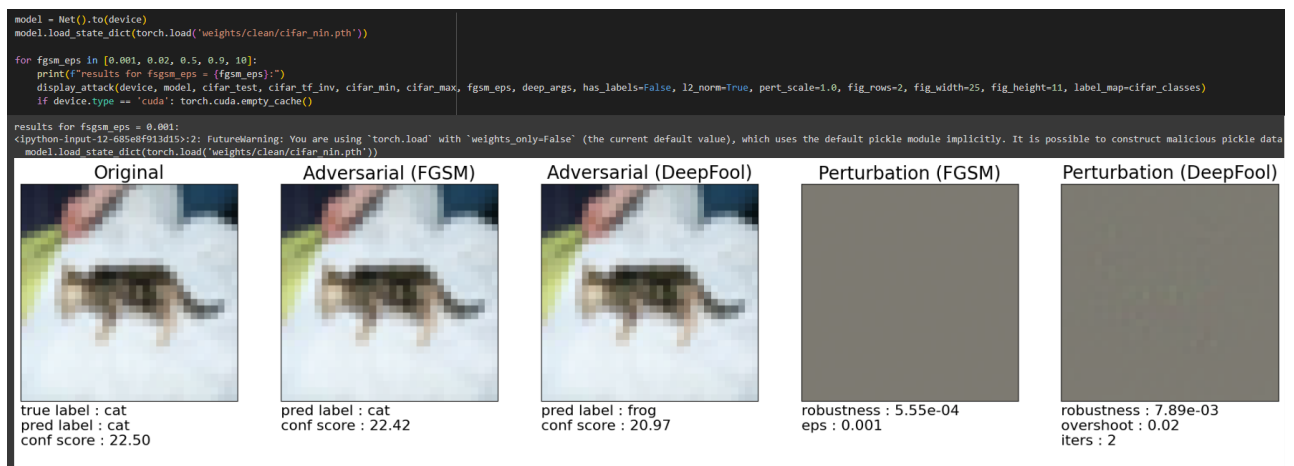
```
model = FC_500_150().to(device)
model.load_state_dict(torch.load('weights/clean/mnist_fc.pth'))

for fgsm_eps in [0.001, 0.02, 0.5, 0.9, 10]:
    print(f"results for fgsm_eps = {fgsm_eps}:")
    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)
    if device.type == 'cuda': torch.cuda.empty_cache()
```

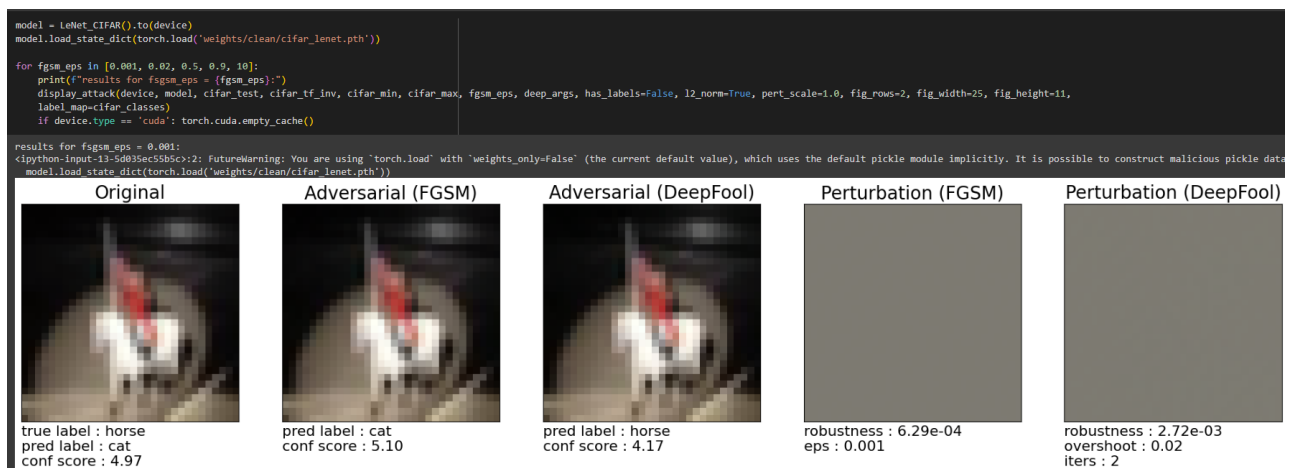
results for fgsm_eps = 0.001:
<ipython-input-11-f377ed2e6379>:2: FutureWarning: You are using 'torch.load' with 'weights_only=False' (the current default value), which uses the default pickle module implicitly. It is possible to construct malicious pickle data
model.load_state_dict(torch.load('weights/clean/mnist_fc.pth'))

Original	Adversarial (FGSM)	Adversarial (DeepFool)	Perturbation (FGSM)	Perturbation (DeepFool)
				
true label : 0 pred label : 0 conf score : 23.20	pred label : 0 conf score : 23.20	pred label : 4 conf score : 12.75	robustness : 8.23e-04 eps : 0.001	robustness : 2.04e-01 overshoot : 0.02 iters : 9

Network-in-Network на CIFAR для fgsm_eps = (0.001, 0.02, 0.5, 0.9, 10)



LeNet на CIFAR для fgsm_eps = (0.001, 0.02, 0.5, 0.9, 10)



Вывод.

Анализируя результаты эксперимента, можно сделать вывод, что с увеличением параметра **fgsm_eps** количество шума на изображениях возрастает. Это явление свидетельствует о том, что изображения становятся более искажёнными, и модель демонстрирует большую подверженность ошибкам в процессе работы.

Иными словами, устойчивость модели к атакам снижается с повышением значения **fgsm_eps**: чем выше этот параметр, тем легче атаке ввести модель в заблуждение и спровоцировать некорректные предсказания.

На диапазоне значений 0.001, 0.02, 0.5, 0.9, 10, модели показывают следующие результаты:

- LeNet + MNIST начинает давать ложные предсказания на значении 0.02;
- FC + MNIST аналогично;
- NiN + CIFAR начинает давать ложные предсказания на значении 0.5;
- LeNet + CIFAR на значении 0.02.

На основе этого можно сделать вывод о том, что модель NiN наиболее устойчива к атаке FGSM.