



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение  
высшего образования

«МИРЭА – Российский технологический университет»

**РТУ МИРЭА**

---

Институт кибербезопасности и цифровых технологий  
Кафедра КБ-4 «Интеллектуальные системы информационной  
безопасности»

**Отчёт по лабораторной работе №1**

**по дисциплине «Анализ защищенности систем искусственного  
интеллекта»**

Выполнил:  
Евдокимов А.М.  
Группа: ББМО-02-23

Москва - 2024

## Цель лабораторной работы

В данной лабораторной работе необходимо выявить закономерность или обнаружить отсутствие влияния параметра `fgsm_eps` на стойкость моделей к атаке. Закономерности или их отсутствие необходимо выявить для сети FC LeNet на датасете MNIST и для сети NiN LeNet на датасете CIFAR.

## Выполнение лабораторной работы

1. Скопировать проект по ссылке в локальную среду выполнения Jupyter (Google Colab) [https://github.com/ewatson2/EEL6812\\_DeepFool\\_Project](https://github.com/ewatson2/EEL6812_DeepFool_Project)

```
!git clone https://github.com/ewatson2/EEL6812_DeepFool_Project

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 (from 1)
Receiving objects: 100% (96/96), 33.99 MiB | 22.76 MiB/s, done.
Resolving deltas: 100% (27/27), done.
```

2. Сменить директорию исполнения на вновь созданную папку "EEL6812\_DeepFool\_Project" проекта.

```
%cd EEL6812_DeepFool_Project/

/content/EEL6812_DeepFool_Project/EEL6812_DeepFool_Project/EEL6812_DeepFool_Project
```

3. Выполнить 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

import ssl
ssl._create_default_https_context = ssl._create_unverified_context
```

4. Выполнить импорт вспомогательных библиотек из локальных файлов проекта

5. Установить значение в виде переменной `rand_seed`. Установить указанное значение для `np.random.seed` и `torch.manual_seed`.

```
rand_seed = 5
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. Использовать в качестве устройства видеокарту (Среды выполнения -> Сменить среду выполнения --> T4 GPU)

**Сменить среду выполнения**

Тип среды выполнения

Python 3

Аппаратный ускоритель ?

☐ CPU ☒ T4 GPU ☐ A100 GPU ☐ L4 GPU

☐ TPU v2-8

Нужен доступ к мощным графическим процессорам?  
[Купите дополнительные вычислительные единицы](#)

Отмена Сохранить

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

```
mnist_mean = 0.5 #среднее значение
mnist_std = 0.5 #средне кв. отклонение
mnist_dim = 28 #кол-во выборки

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):
HTTP Error 403: Forbidden

Downloading https://ossci-datasets.s3.amazonaws.com/mnist/train-images-idx3-ubyte.gz
Downloading https://ossci-datasets.s3.amazonaws.com/mnist/train-images-idx3-ubyte.gz to datasets/mnist/MNIST/raw/train-images-idx3-ubyte.gz
100% | 9912422/9912422 [00:00<00:00, 14536631.60it/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):
HTTP Error 403: Forbidden
```

8. 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)

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%|██████████| 170498071/170498071 [00:05<00:00, 29571150.99it/s]
Extracting datasets/cifar-10/cifar-10-python.tar.gz to datasets/cifar-10
Files already downloaded and verified
```

## 9. Выполнить 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)

/usr/local/lib/python3.10/dist-packages/torch/utils/data/dataloader.
warnings.warn(_create_warning_msg(
```

## 10. Загрузить и оценить 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/clean/cifar_nin.pth'))
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-64-885f11da824e>:3: FutureWarning: You are using `torch
model.load_state_dict(torch.load('weights/clean/cifar_nin.pth'))
```

## 11. Загрузить и оценить 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'))

evaluate_attack('cifar_lenet_fgsm.csv', 'results',
               device, model, cifar_loader_test,
               cifar_min, cifar_max, fgsm_eps, is_fgsm=True)

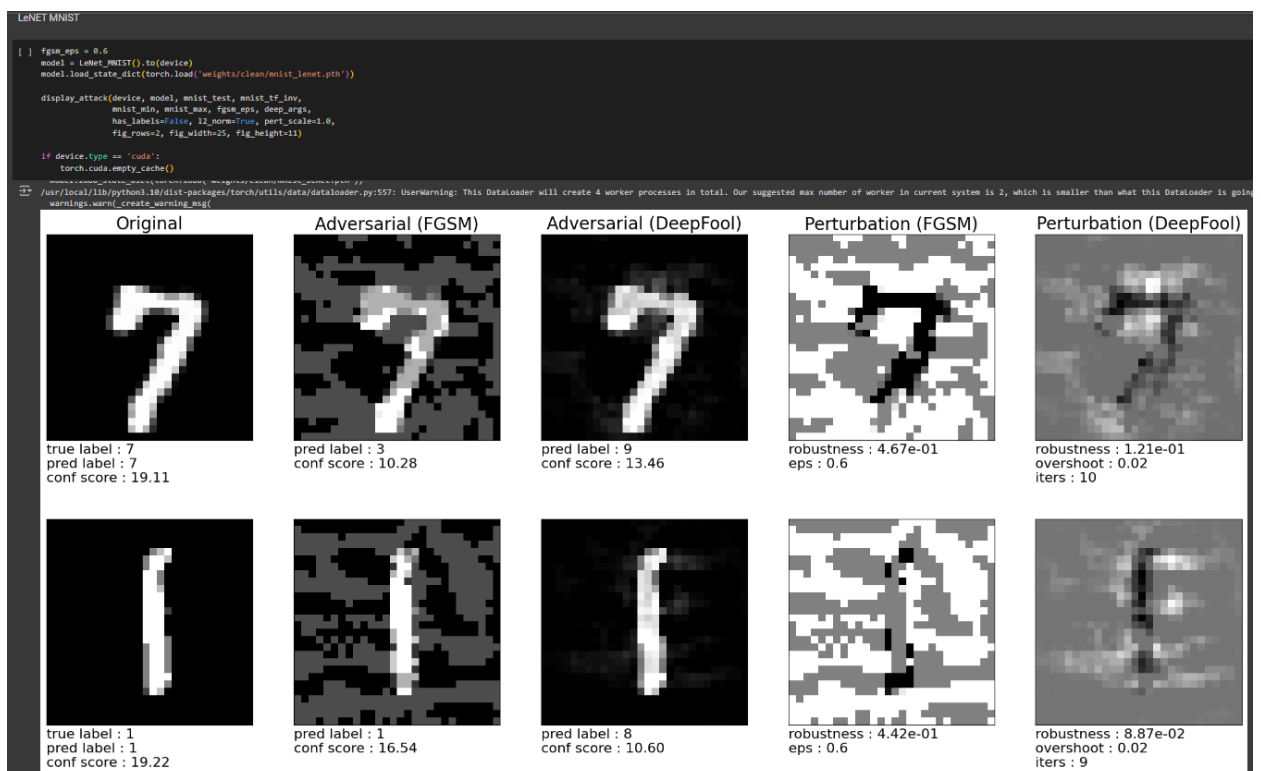
print('')
evaluate_attack('cifar_lenet_deepfool.csv', 'results',
               device, model, cifar_loader_test,
               cifar_min, cifar_max, deep_args, is_fgsm=False)

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:







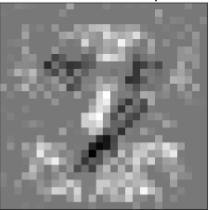




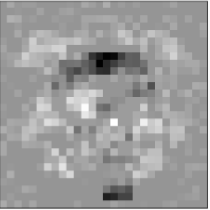
FCNet

```
[ ] fgsm_eps = 0.2
model = FCNet_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_fgsm,
               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()
```

`>>> c:\python-input-67-f2c8eac42d72>3: 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 which will execute arbitrary code during unpickling. Please use 'torch.load(..., weights_only=True)' instead to disable this warning.`



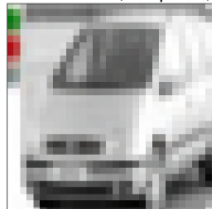

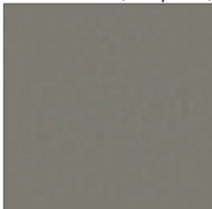
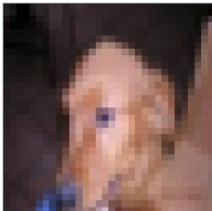
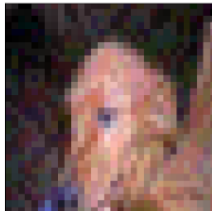
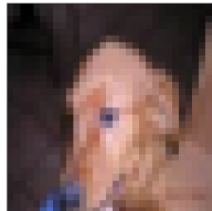
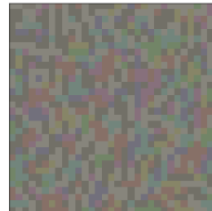
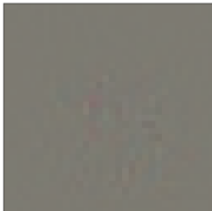
Original	Adversarial (FGSM)	Adversarial (DeepFool)	Perturbation (FGSM)	Perturbation (DeepFool)
				
true label : 7 pred label : 7 conf score : 20.36	pred label : 3 conf score : 13.08	pred label : 3 conf score : 12.71	robustness : 1.50e-01 eps : 0.2	robustness : 9.64e-02 overshoot : 0.02 iters : 9
				
true label : 9 pred label : 9 conf score : 17.05	pred label : 7 conf score : 12.22	pred label : 4 conf score : 10.85	robustness : 1.65e-01 eps : 0.2	robustness : 7.18e-02 overshoot : 0.02 iters : 10

```
fgsm_eps = 0.2
model = Net().to(device)
model.load_state_dict(torch.load('weights/clean/cifar_nin.pth'))

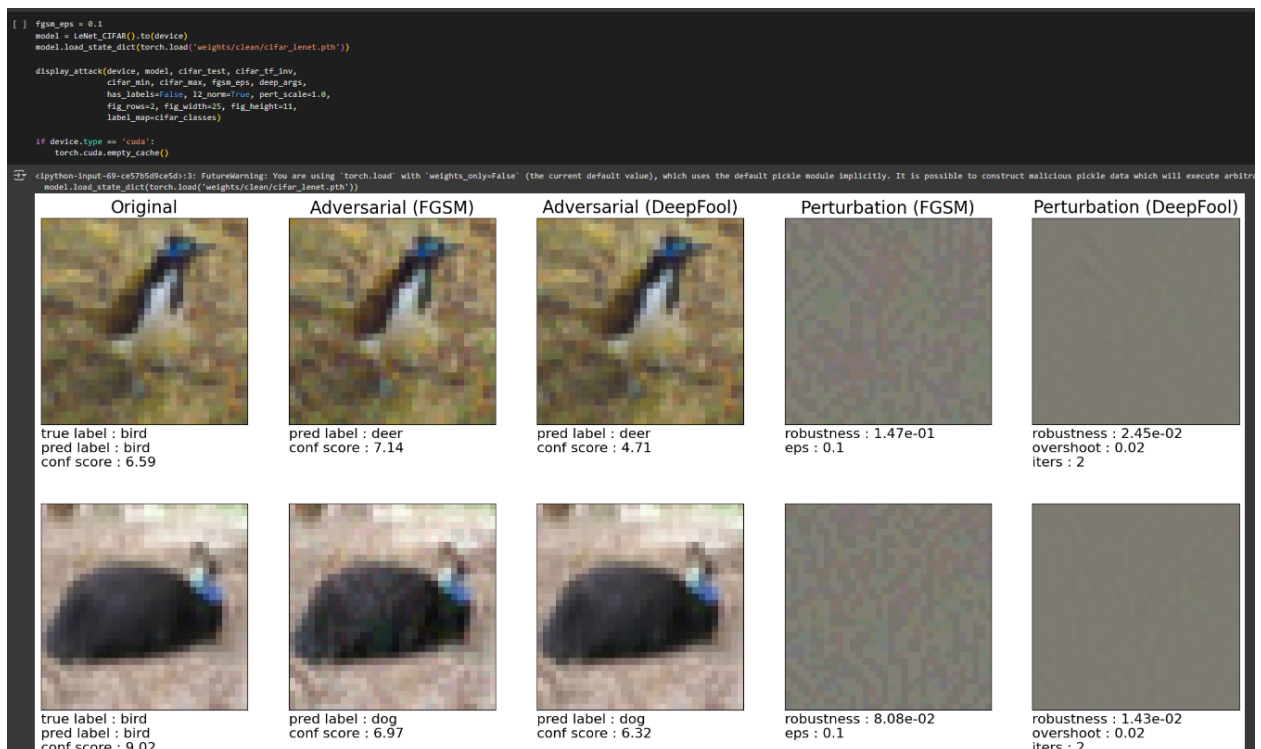
display_attack(device, model, cifar_test, cifar_tf_inv,
               cifar_min, cifar_max, fgsm_eps, deep_fgsm,
               has_labels=False, l2_norm=True, pert_scale=1.0,
               fig_rows=2, fig_width=25, fig_height=11,
               label_map=cifar_classes)

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

`/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 system is 2, which is smaller than what this DataLoader is going to create. Please go to the docs of DataLoader for more details (https://pytorch.org/docs/1.10/data_loader.html). You can also increase the limit of your system's torch.cuda.device_count() by calling torch.cuda.set_device(-1) (warning of this does not apply to Windows platform). And if you are using a CUDA compatible GPU, please use pinned memory instead of pageable memory to get the maximal OOM protection and reduce the risk of CUDA-unsafe operations. (Triggered at: /usr/local/lib/python3.10/dist-packages/torch/utils/data/dataloader.py:557)`

Original	Adversarial (FGSM)	Adversarial (DeepFool)	Perturbation (FGSM)	Perturbation (DeepFool)
				
true label : automobile pred label : automobile conf score : 43.23	pred label : truck conf score : 38.62	pred label : truck conf score : 39.63	robustness : 1.29e-01 eps : 0.2	robustness : 8.77e-03 overshoot : 0.02 iters : 2
				
true label : dog pred label : dog conf score : 29.48	pred label : bird conf score : 21.57	pred label : bird conf score : 19.99	robustness : 1.55e-01 eps : 0.2	robustness : 1.94e-02 overshoot : 0.02 iters : 3





## Результаты:

Маленькие значения eps (например, 0.001, 0.02):

При маленьких значениях eps атака не сильно искажает данные. Градиенты, добавленные к исходным изображениям, настолько малы, что модель может легко отличить атакующие примеры от исходных.

Точность модели остается высокой. Например, при  $\text{eps} = 0.001$  модель может практически не терять точности. Это значит, что такая атака не является эффективной, так как она минимально искажает изображения.

Низкие значения eps создают почти незаметные искажения, и модель сохраняет свою устойчивость.

Средние значения eps (например, 0.02, 0.5):

Как только значение eps увеличивается, модель начинает сталкиваться с трудностями. Атака становится более заметной, и ошибки начинают накапливаться. При  $\text{eps} = 0.02$ , некоторые примеры начинают давать ложные предсказания, хотя точность все еще остается достаточно высокой.

Например, при  $\text{eps} = 0.02$ , модель может потерять несколько процентов точности. При  $\text{eps} = 0.5$  точность может значительно упасть.

Средние значения  $\epsilon_{rs}$  демонстрируют, как легко можно запутать модель с увеличением силы атаки.

Большие значения  $\epsilon_{rs}$  (например, 0.9, 10):

При очень больших значениях  $\epsilon_{rs}$  искажения становятся настолько сильными, что модель практически не может отличить атакующие примеры от случайного шума. Изображения сильно деформированы и больше не представляют собой исходные данные.

При  $\epsilon_{rs} = 0.9$  точность модели может упасть до критического уровня. При экстремально больших значениях, таких как  $\epsilon_{rs} = 10$ , модель перестает предсказывать правильно, и точность падает практически до нуля.

Сильно большие значения  $\epsilon_{rs}$  приводят к разрушительным атакам, которые полностью разрушают представление модели об исходных данных. Это показывает, насколько уязвимы модели при больших искажениях данных.

## **Заключение**

В результате выполнения лабораторной работы было выявлено, что маленькие значения `fgsm_eps` сохраняют стойкость сетей к атакам, и ошибки классификации остаются низкими. При увеличении `fgsm_eps` сети становятся более уязвимыми к атакам и допускают больше ошибок классификации. Для сети FC LeNet на датасете MNIST и для сети NiN LeNet на датасете CIFAR не наблюдается отсутствие влияния параметра `fgsm_eps`. Наоборот, параметр `fgsm_eps` оказывает существенное влияние на стойкость сетей к атакам.

Ссылка на colab: [https://colab.research.google.com/drive/1qJKmab4-YISIF08hKEzgQ\\_MYY4X-6gFZ?usp=sharing](https://colab.research.google.com/drive/1qJKmab4-YISIF08hKEzgQ_MYY4X-6gFZ?usp=sharing)