МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт кибербезопасности и цифровых технологий

Отчет по лабораторной работе №1

по дисциплине: «Анализ защищенности систем искусственного интеллекта»

Выполнил:
Студент группы ББМО-02-23
ФИО: Ионов М.С.

Москва 2024

1. Клонируем репозиторий и загружаем нужные библиотеки.

```
[1] !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
Receiving objects: 100% (96/96), 33.99 MiB | 32.14 MiB/s, done.
Resolving deltas: 100% (27/27), done.
```

```
[2] %cd EEL6812_DeepFool_Project
```

```
/content/EEL6812_DeepFool_Project
```

```
[3] import numpy as np
    import os, torch
    from torch.utils.data import DataLoader, random_split
    from torchvision import datasets
    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, model_train, model_eval, evaluate_attack, display_attack
```

2. Устанавливаем случайное число и выбираем устройство выполнения.

```
# Устанавливаем случайное число
rand_seed = 43
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')
```

3. Загружаем dataset «MNIST» и отредактируем его.

```
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
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, 212835494.90it/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%|██████████| 28881/28881 [00:00<00:00, 21972736.05it/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, 67904713.90it/s]Extracting datasets/mnist/MNIST/raw/t10k-images-idx3-ubyte.gz to datasets/mnist/MNIST/raw
```

4. Загружаем dataset «CIFAR-10» и отредактируем его.

```
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 [00:07<00:00, 21840047.81it/s]
Extracting datasets/cifar-10/cifar-10-python.tar.gz to datasets/cifar-10
Files already downloaded and verified
```

5. Отредактируем гиперпараметры.

```
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 worke
  warnings.warn(_create_warning_msg(
```

6. Зададим параметры на модель.

```python
train_model = True

epochs = 50
epochs_nin = 100

lr = 0.004
lr_nin = 0.01
lr_scale = 0.5

momentum = 0.9

print_step = 5

deep_batch_size = 10
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)
```

7. Загрузим и оценим стойкость модели.

```python
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
```

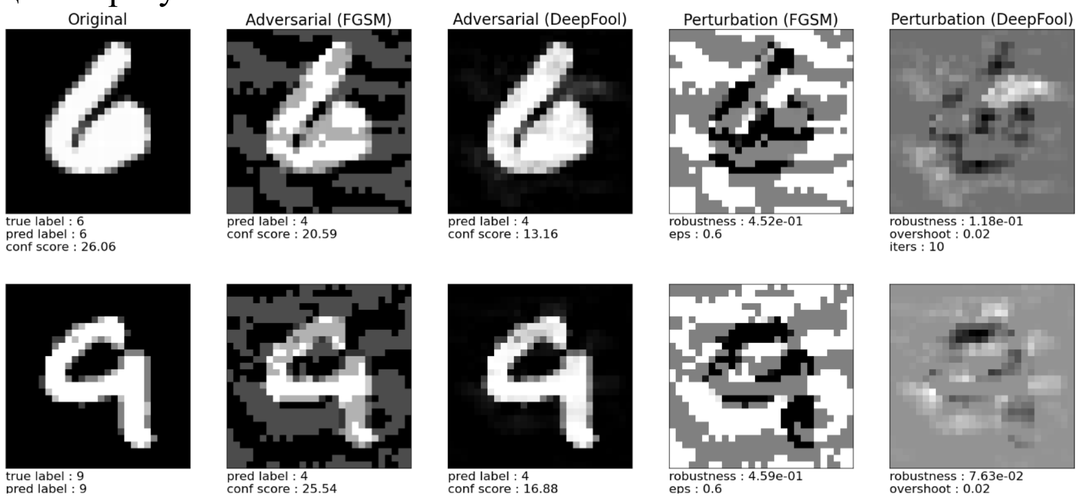
```python
fgsm_eps = 0.1
model = LeNet_CIFAR().to(device)
model.load_state_dict(torch.load('weights/clean/cifar_lenet.pth', map_location=torch.device('cpu')))
evaluate_attack('cifar_lenet_fgsm.csv', 'results', device, model, cifar_loader_test, cifar_min, cifar_max, fgsm_eps, is_fgsm=True)
print('')
evaluate_attack('cifar_lenet_deepfool.csv',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

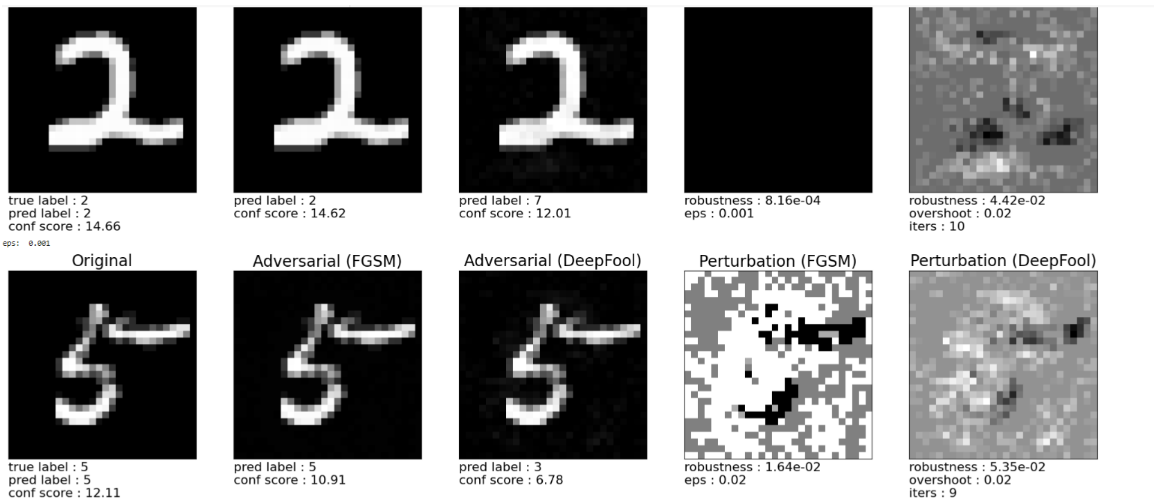
```
FGSM Test Error : 91.71%
FGSM Robustness : 8.90e-02
FGSM Time (All Images) : 0.40 s
FGSM Time (Per Image) : 40.08 us

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

8. Выполним оценку атакующих примеров для сетей.

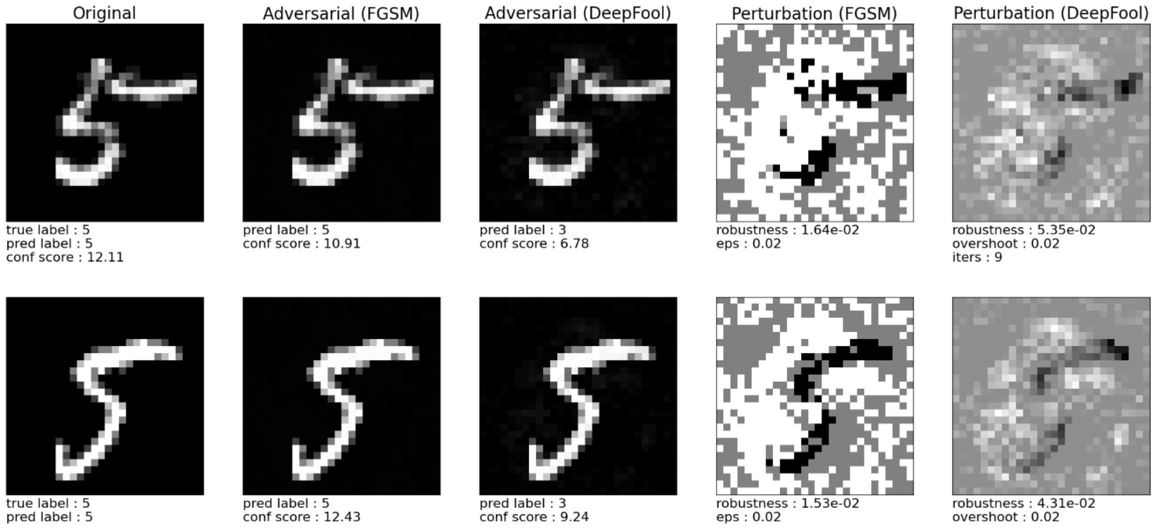
```
fgsm_eps = 0.6
model = LeNet_MNIST().to(device)
model.load_state_dict(torch.load('weights/clean/mnist_lenet.pth'))
display_attack(device, model, mnist_test, mnist_tf_inv, mnist_min, mnist_max, fgsm_eps, deep_args, has_labels=False, l2_norm=True, pert_scale=1.0, fig_rows=2, fig_width=25, fig_height=11)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

fgsm_eps_arr = [0.001, 0.02, 0.5, 0.9, 10]
for fgsm_eps in fgsm_eps_arr:
  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)
  if device.type == 'cuda': torch.cuda.empty_cache()
  print("eps: ", fgsm_eps)

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

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

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

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

9. Оценим результат атаки.

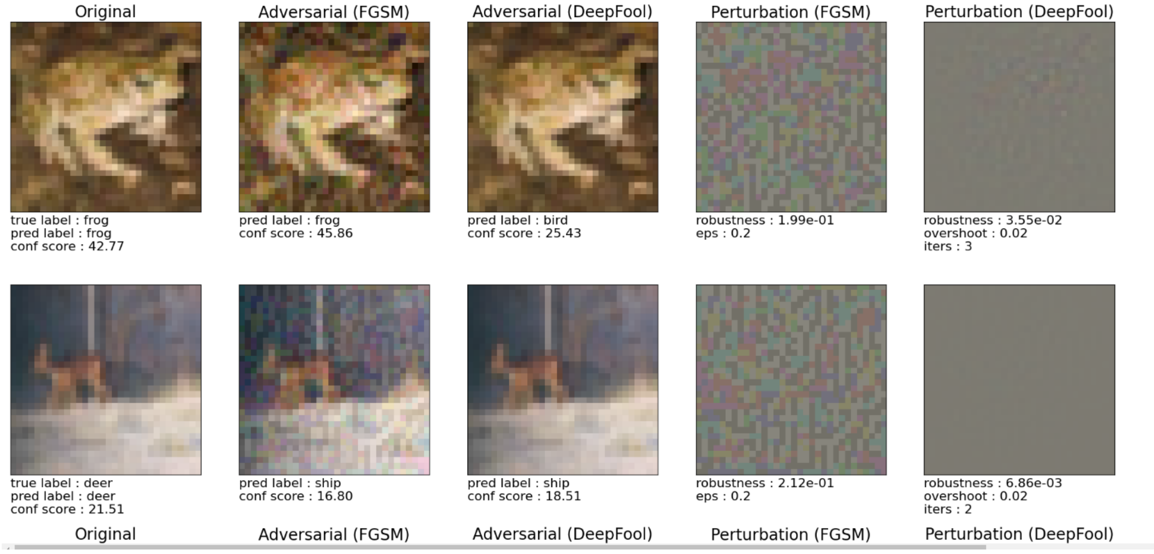




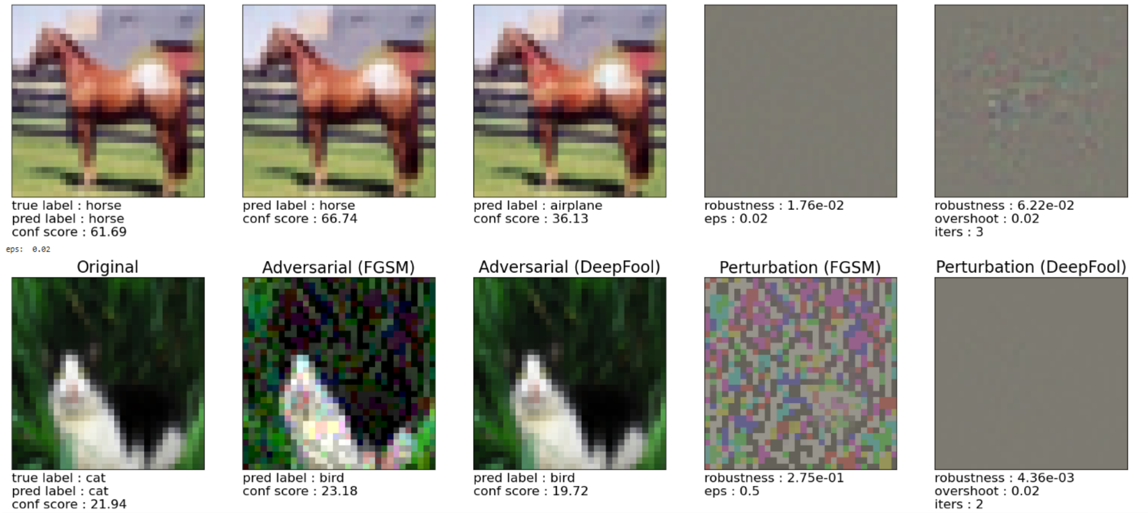
true label : 2
pred label : 2
conf score : 14.66
pred label : 2
conf score : 14.62
pred label : 7
conf score : 12.01
robustness : 8.16e-04
eps : 0.001
robustness : 4.42e-02
overshoot : 0.02
iters : 10
eps: 0.001
Original
Adversarial (FGSM)
Adversarial (DeepFool)
Perturbation (FGSM)
Perturbation (DeepFool)
true label : 5
pred label : 5
conf score : 12.11
pred label : 5
conf score : 10.91
pred label : 3
conf score : 6.78
robustness : 1.64e-02
eps : 0.02
robustness : 5.35e-02
overshoot : 0.02
iters : 9

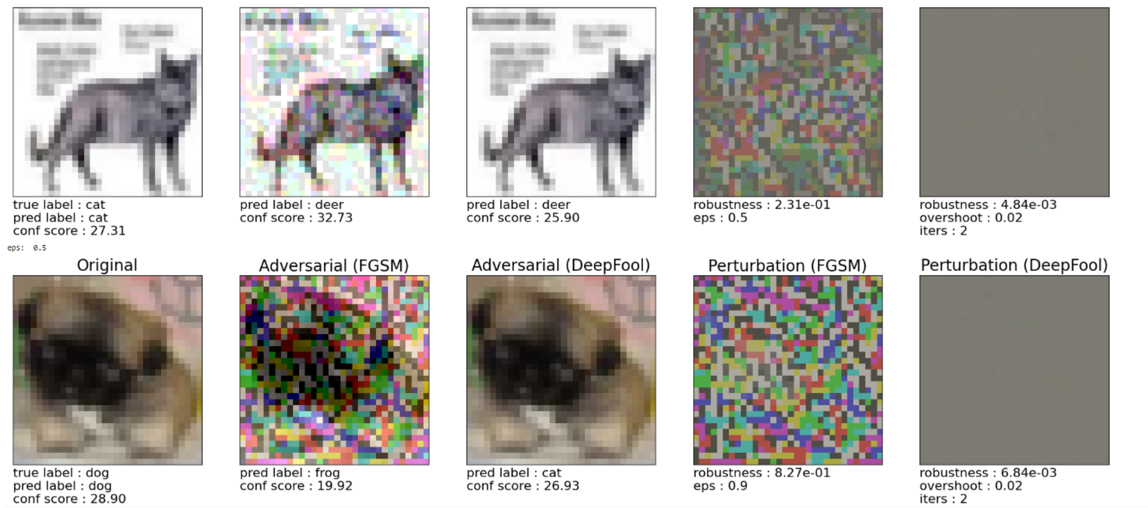


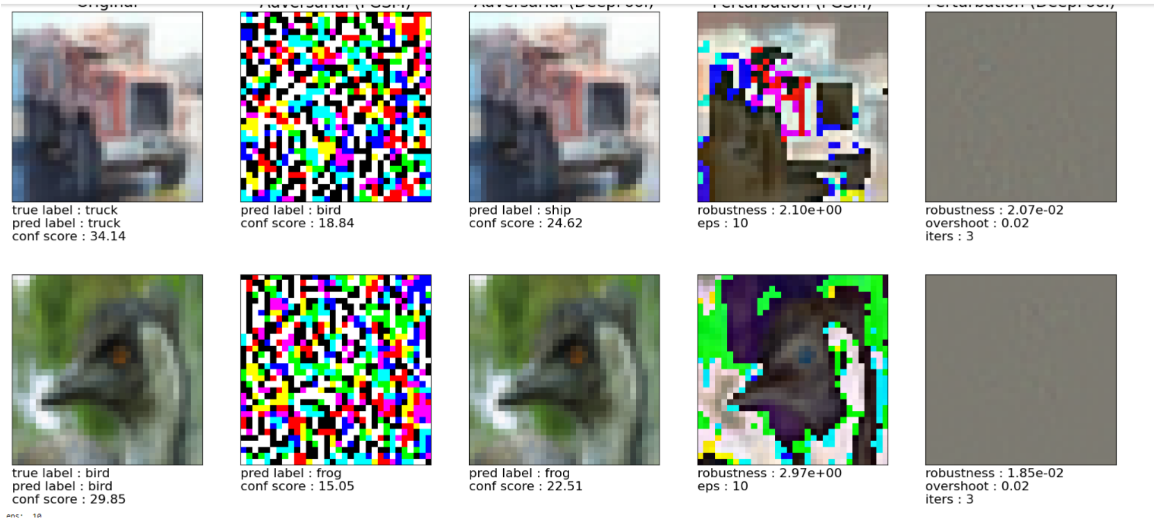
Original
Adversarial (FGSM)
Adversarial (DeepFool)
Perturbation (FGSM)
Perturbation (DeepFool)
true label : 5
pred label : 5
conf score : 12.11
pred label : 5
conf score : 10.91
pred label : 3
conf score : 6.78
robustness : 1.64e-02
eps : 0.02
robustness : 5.35e-02
overshoot : 0.02
iters : 9
true label : 5
pred label : 5
pred label : 5
conf score : 12.43
pred label : 3
conf score : 9.24
robustness : 1.53e-02
eps : 0.02
robustness : 4.31e-02
overshoot : 0.02



Original
Adversarial (FGSM)
Adversarial (DeepFool)
Perturbation (FGSM)
Perturbation (DeepFool)
true label : frog
pred label : frog
conf score : 42.77
pred label : frog
conf score : 45.86
pred label : bird
conf score : 25.43
robustness : 1.99e-01
eps : 0.2
robustness : 3.55e-02
overshoot : 0.02
iters : 3
true label : deer
pred label : deer
conf score : 21.51
pred label : ship
conf score : 16.80
pred label : ship
conf score : 18.51
robustness : 2.12e-01
eps : 0.2
robustness : 6.86e-03
overshoot : 0.02
iters : 2
Original
Adversarial (FGSM)
Adversarial (DeepFool)
Perturbation (FGSM)
Perturbation (DeepFool)


true label : horse
pred label : horse
conf score : 61.69
pred label : horse
conf score : 66.74
pred label : airplane
conf score : 36.13
robustness : 1.76e-02
eps : 0.02
robustness : 6.22e-02
overshoot : 0.02
iters : 3
eps: 0.02
Original
Adversarial (FGSM)
Adversarial (DeepFool)
Perturbation (FGSM)
Perturbation (DeepFool)
true label : cat
pred label : cat
conf score : 21.94
pred label : bird
conf score : 23.18
pred label : bird
conf score : 19.72
robustness : 2.75e-01
eps : 0.5
robustness : 4.36e-03
overshoot : 0.02
iters : 2


true label : cat
pred label : cat
conf score : 27.31
pred label : deer
conf score : 32.73
pred label : deer
conf score : 25.90
robustness : 2.31e-01
eps : 0.5
robustness : 4.84e-03
overshoot : 0.02
iters : 2
eps: 0.5
Original
Adversarial (FGSM)
Adversarial (DeepFool)
Perturbation (FGSM)
Perturbation (DeepFool)
true label : dog
pred label : dog
conf score : 28.90
pred label : frog
conf score : 19.92
pred label : cat
conf score : 26.93
robustness : 8.27e-01
eps : 0.9
robustness : 6.84e-03
overshoot : 0.02
iters : 2


true label : truck
pred label : truck
conf score : 34.14
pred label : bird
conf score : 18.84
pred label : ship
conf score : 24.62
robustness : 2.10e+00
eps : 10
robustness : 2.07e-02
overshoot : 0.02
iters : 3
true label : bird
pred label : bird
conf score : 29.85
pred label : frog
conf score : 15.05
pred label : frog
conf score : 22.51
robustness : 2.97e+00
eps : 10
robustness : 1.85e-02
overshoot : 0.02
iters : 3

Выводы

В ходе выполенния лабораторной работы 1 было обнаружено, что маленькие значения fgsm eps сохраняют устойчивость сетей к атакам, и ошибки классификации остаются низкими. Однако 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 существенно влияет на стойкость сетей к атакам.