



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

**«МИРЭА – Российский технологический университет»
РТУ МИРЭА**

ИКБ направление «Киберразведка и противодействие угрозам с применением технологий
искусственного интеллекта» 10.04.01

Кафедра КБ-4 «Интеллектуальные системы информационной безопасности»

Лабораторная работа №1

по дисциплине: «Анализ защищенности систем искусственного
интеллекта»

Группа:

ББМО-01-22

Выполнил:

Гребенник Г.С

Проверил:

к.т.н. Спирин А.А.

Москва, 2023

Содержание

Цель работы:	3
--------------------	---

Цель работы:

Целью данной работы является отразить отличия для $\text{fgsm_eps}=(0.001, 0.02, 0.5, 0.9, 10)$ и выявить закономерность/обнаружить отсутствие влияние параметра eps для сетей FC LeNet на датасете MNIST, NiN LeNet на датасете CIFAR.

Ход работы:

1. Скопировать проект по ссылке в локальную среду выполнения Jupyter (Google Colab):

https://github.com/ewatson2/EEL6812_DeepFool_Project

```
✓ 2 сек. [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 | 25.90 MiB/s, done.
Resolving deltas: 100% (27/27), done.
```

2. Сменим директорию исполнения на вновь созданную папку "EEL6812_DeepFool_Project" проекта:

```
✓ 0 сек. %cd /content/EEL6812_DeepFool_Project

/content/EEL6812_DeepFool_Project
```

3. Импортируем библиотеки:

```
✓ 12 сек. import numpy as np
import json, torch, os
from torch.utils.data import DataLoader, random_split
from torchvision import datasets, models
from torchvision.transforms import transforms
```

4. Импортируем вспомогательные библиотеки из локальных файлов проекта:

```
✓ 0 сек. 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
```

5. Установим случайное значение в виде переменной `rand_seed`=(порядковый номер в списке гугл-таблицы 11) и установим его для `np.random.seed` и `torch.manual_seed`:

```
✓ 0 сек. rand_seed = 11
np.random.seed(rand_seed)
torch.manual_seed(rand_seed)

<torch._C.Generator at 0x787bb422e3d0>
```

6. Используем в качестве устройства видеокарту (Среды выполнения --> Сменить среду выполнения --> T4 GPU)

```
✓ 0
CEK. ▶ use_cuda = torch.cuda.is_available()
      device = torch.device('cuda' if use_cuda else 'cpu')
```

7. Загрузим датасет MNIST с параметрами:

```
✓ 1
CEK. ▶ 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, 132257617.99it/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, 42563490.45it/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, 44599088.12it/s]Extracting datasets/mnist/MNIST/raw/t10k-images-idx3-ubyte.gz to datasets/mnist/MNIST/raw

Downloading <http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz>
Downloading <http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz> to datasets/mnist/MNIST/raw/t10k-labels-idx1-ubyte.gz
100%|██████████| 4542/4542 [00:00<00:00, 18126097.78it/s]
Extracting datasets/mnist/MNIST/raw/t10k-labels-idx1-ubyte.gz to datasets/mnist/MNIST/raw

8. Загрузим датасет CIFAR-10 с параметрами:

```
✓ 9
CEK. ▶ 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_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=

      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 [00:04<00:00, 34575494.82it/s]
Extracting datasets/cifar-10/cifar-10-python.tar.gz to datasets/cifar-10
Files already downloaded and verified

9. Выполним настройку гиперпараметров и загрузку DataLoader:

```
✓ 0
CEK. ▶ 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 o
warnings.warn(_create_warning_msg)

10. Зададим параметры deep_args:

```

0
cek.
▶ 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]

```

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

```

0
cek.
▶ 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

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

```

0
cek.
▶ 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

13. Выполним оценку атакующих примеров для сетей:

13.1. LeNet на датасет MNIST:

```

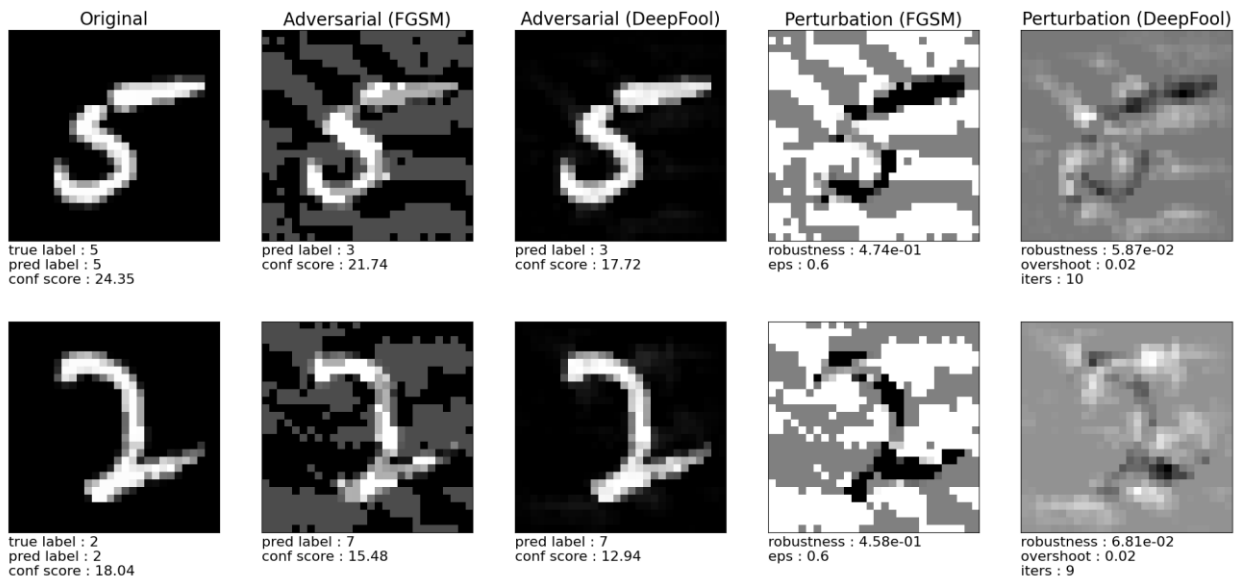
0
cek.
▶ fgsm_eps = 0.6
  model = LeNet_MNIST().to(device)
  model.load_state_dict(torch.load('weights/clean/mnist_lenet.pth', map_location=device))

  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=2)

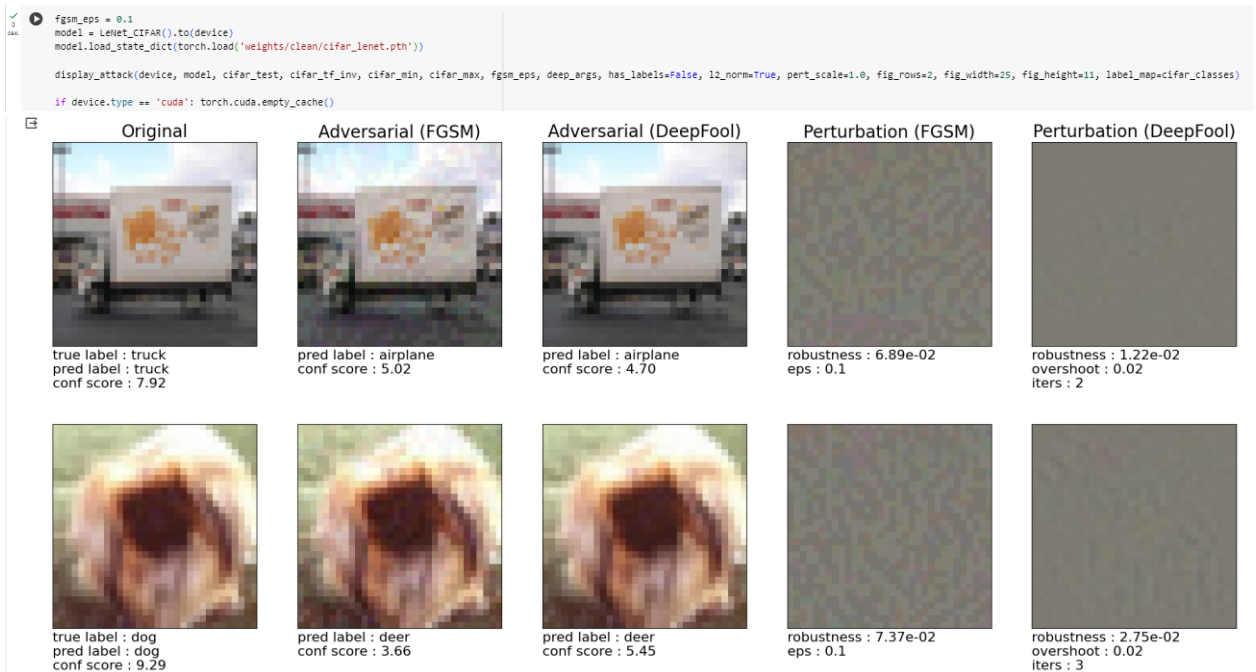
  if device.type == 'cuda': torch.cuda.empty_cache()

```

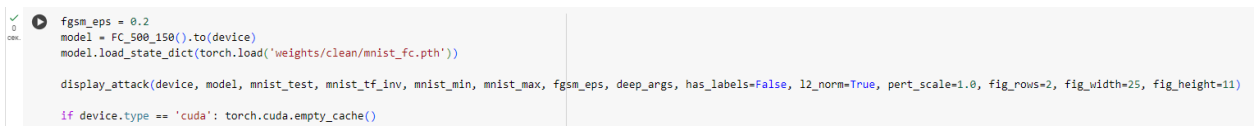
/usr/local/lib/python3.10/dist-packages/torch/utils/data/dataloader.py:557: UserWarning: This DataLoader will create 4 worker processes in total. Our suggested max number
 warnings.warn(_create_warning_msg(

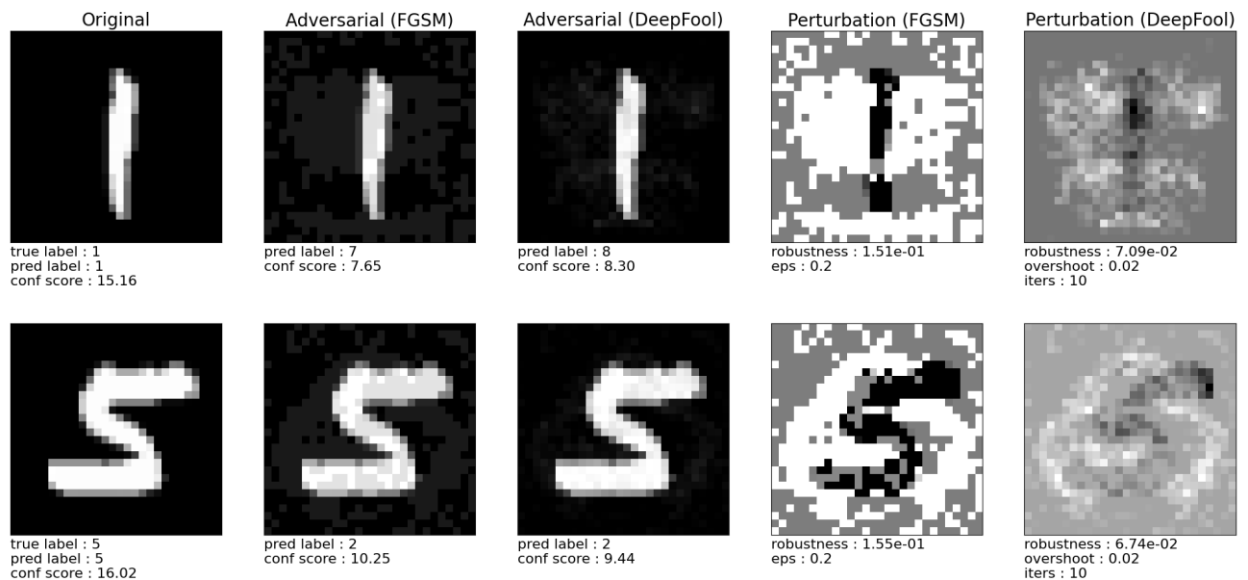


13.2. LeNet на датасете CIFAR-10:



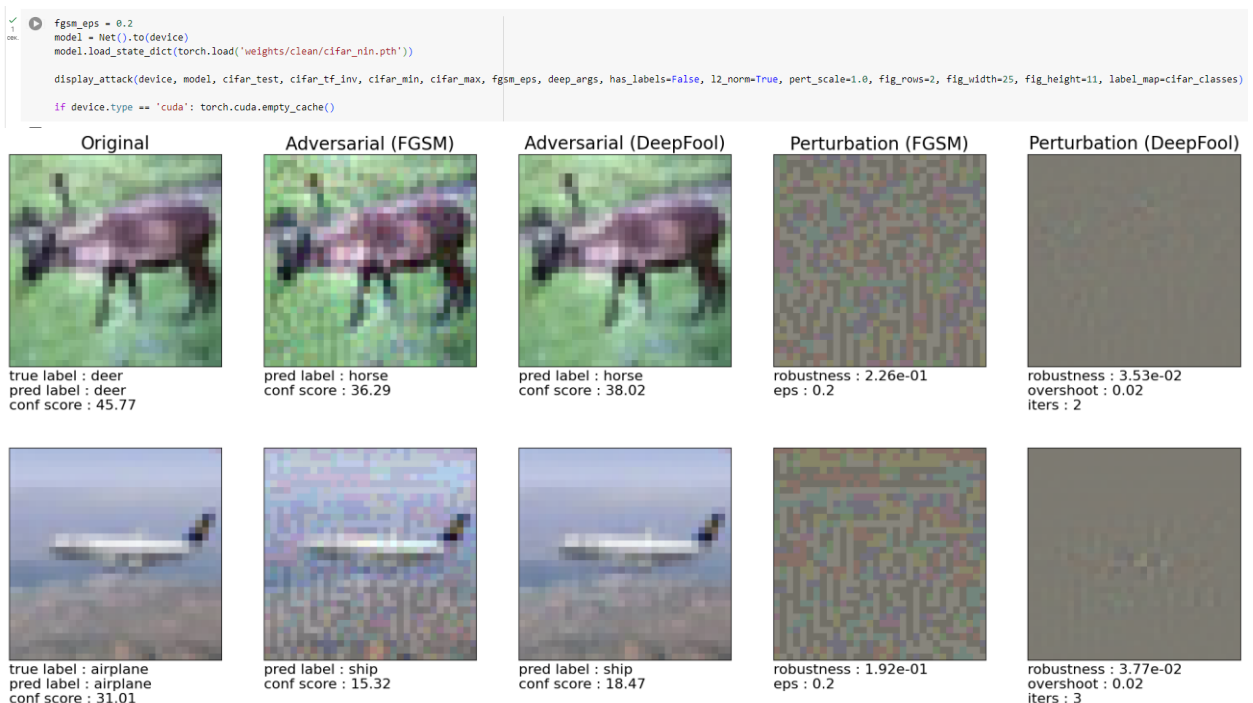
13.3. FCNet на датасете MNIST:





13.4.

Network-in-Network на датасете CIFAR:



14. Отразим отличия для $fgsm_eps=(0.001, 0.02, 0.5, 0.9, 10)$ и выявим закономерности/обнаружим отсутствие влияние параметра eps для сетей FC LeNet на датасете MNIST, NiN LeNet на датасете CIFAR: Для выполнения данной задачи реализуем просто цикл который будет перебирать уже известный нам массив значений:


```

fgsm_epsf = [0.001, 0.02, 0.5, 0.9, 10]

for fgsm_eps in fgsm_epsf:
    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()

    for fgsm_eps in fgsm_epsf:
        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)

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

    for fgsm_eps in fgsm_epsf:
        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()

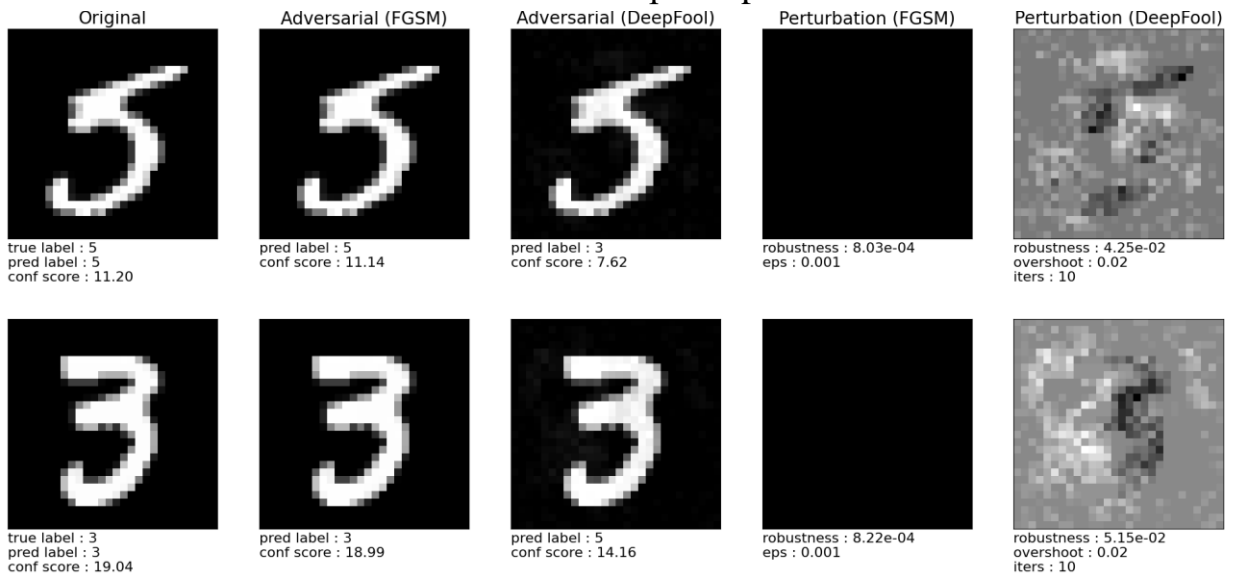
    for fgsm_eps in fgsm_epsf:
        model = Net().to(device)
        model.load_state_dict(torch.load('weights/clean/cifar_nin.pth'))

        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)

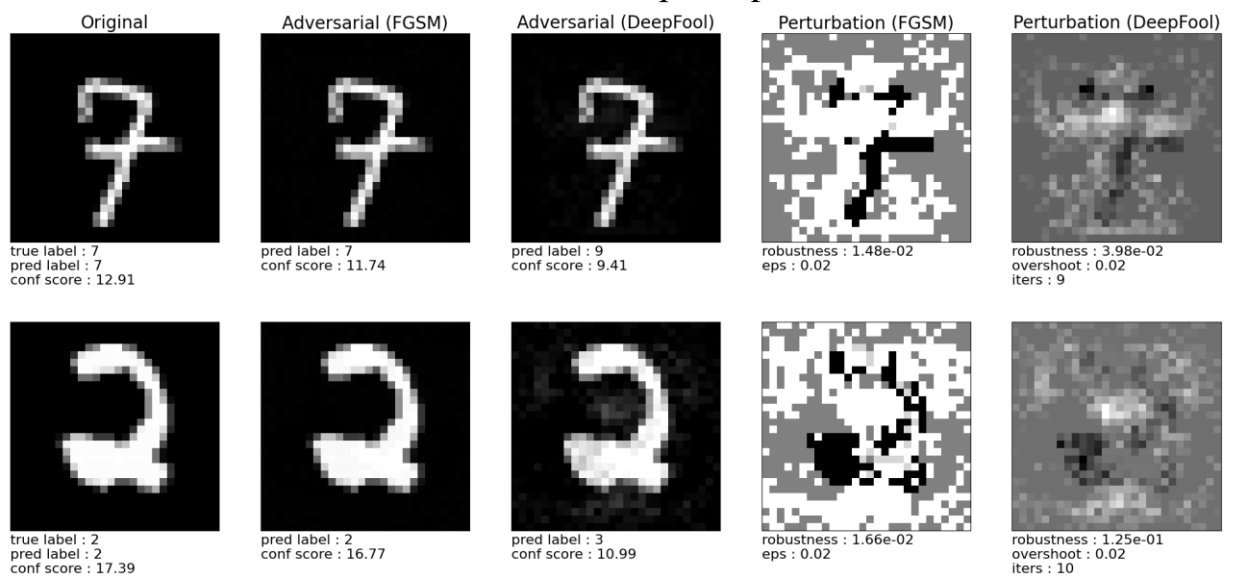
        if device.type == 'cuda':
            torch.cuda.empty_cache()

```

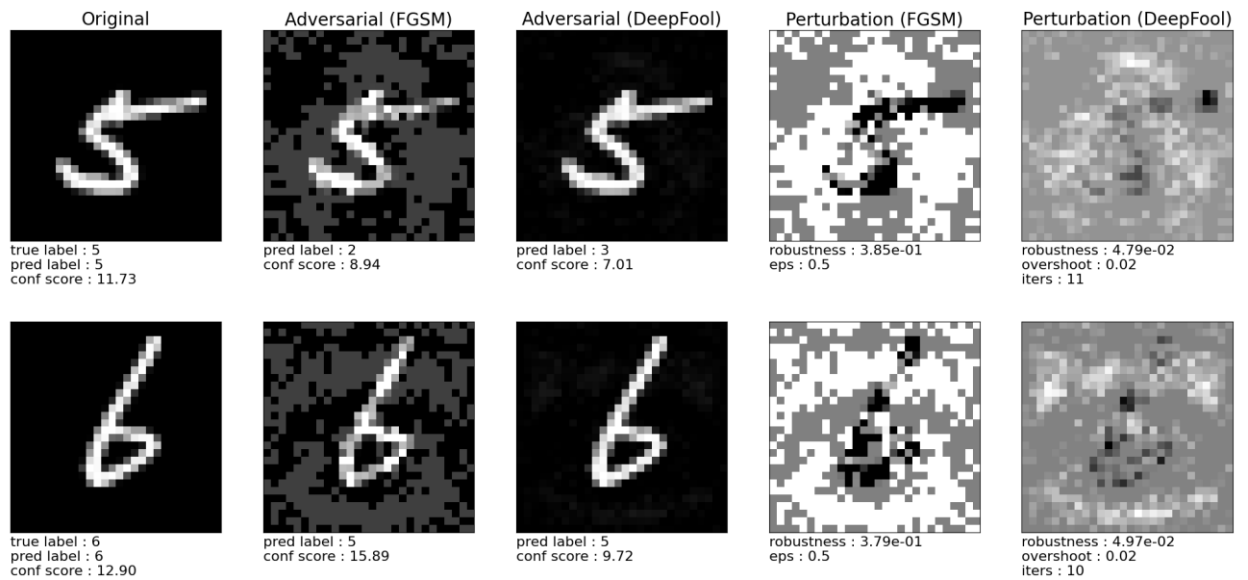
14.1. FC на MNIST при Esp 0.001



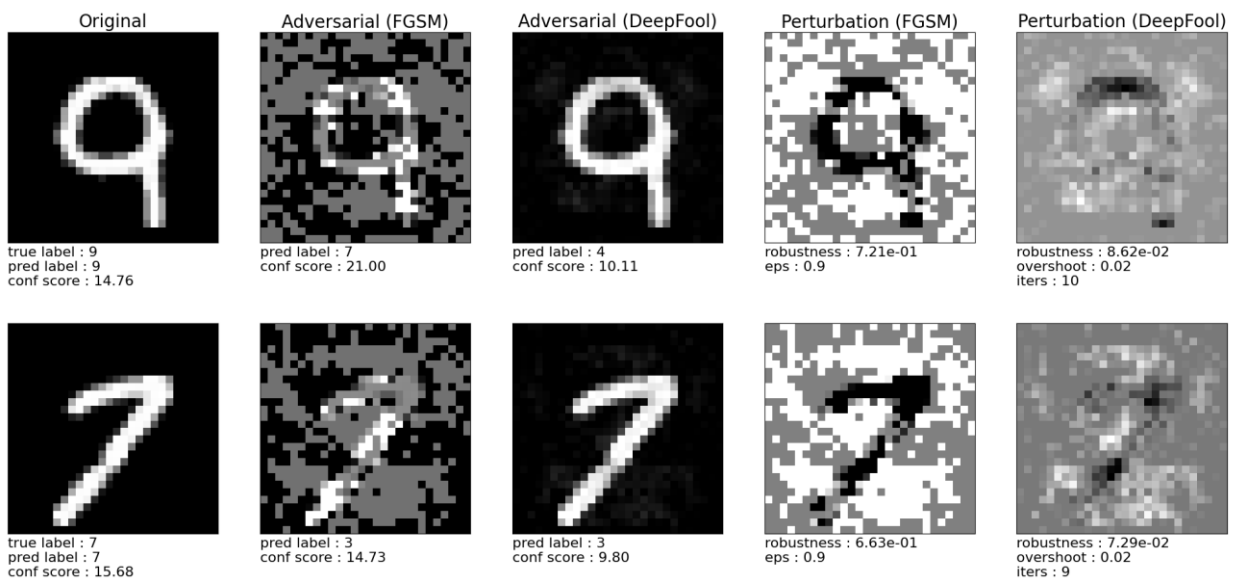
14.1. FC на MNIST при Esp 0.02



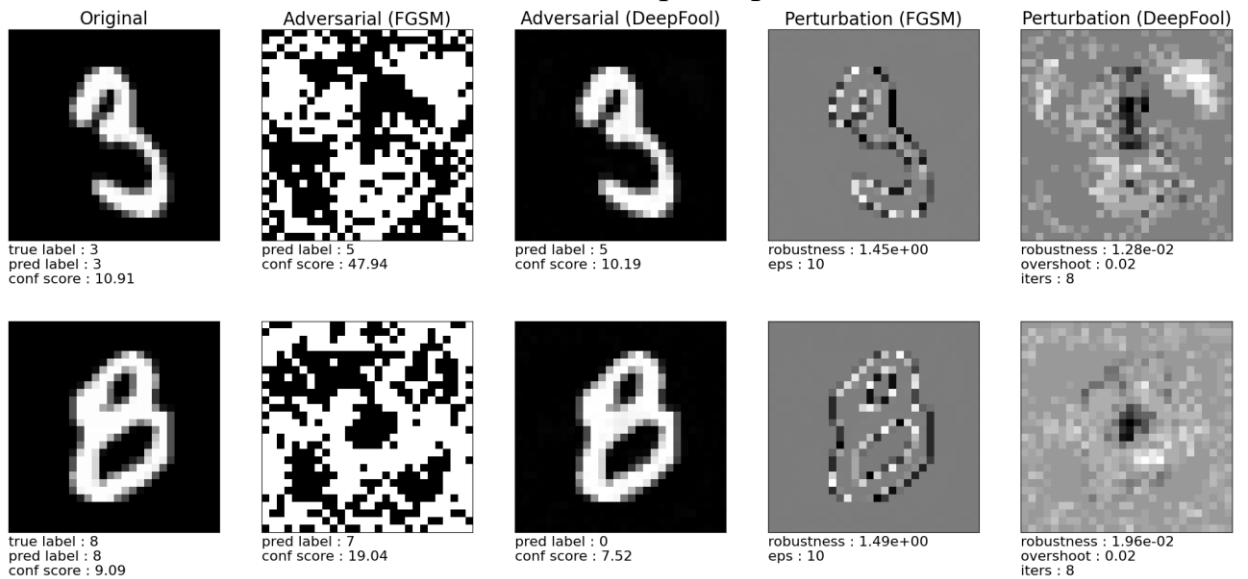
14.2. FC на MNIST при Esp 0.5



14.3. FC на MNIST при Esp 0.9



14.4. FC на MNIST при Esp 10

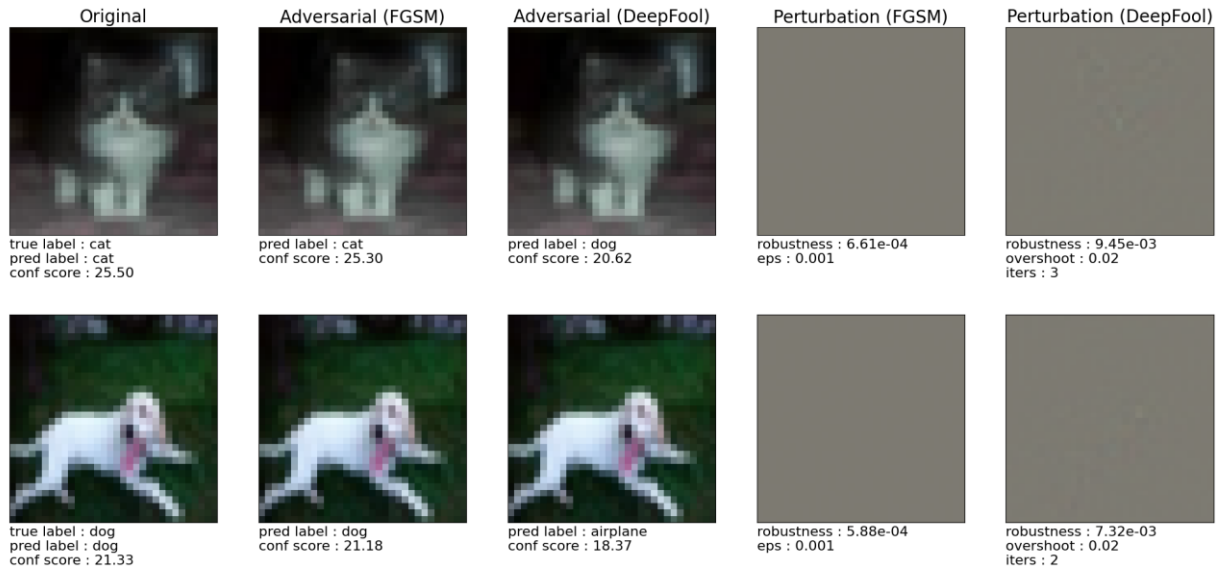


```

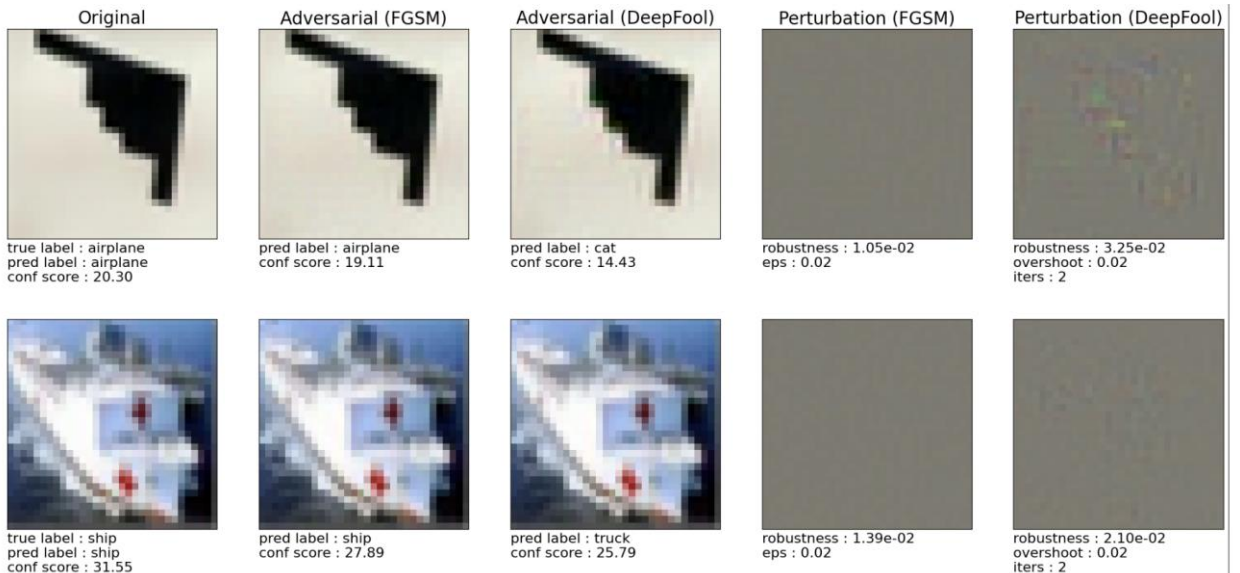
/var/local/lib/python3.10/dist-packages/torch/utils/data/dataloader.py:557: UserWarning: This DataLoader will create 4 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller than what this DataLoader is going to create. Please be aware that excessive worker creation
warnings.warn(_create_warning_msg(
FGSM Batches Complete : (157 / 157)
FGSM Test Error : 3.87%
FGSM Robustness : 8.89e-04
FGSM Time (All Images) : 0.51 s
FGSM Time (Per Image) : 98.71 us
/var/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 be aware that excessive worker creation
warnings.warn(_create_warning_msg(
FGSM Batches Complete : (157 / 157)
FGSM Test Error : 5.54%
FGSM Robustness : 1.08e-02
FGSM Time (All Images) : 0.67 s
FGSM Time (Per Image) : 67.22 us
/var/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 be aware that excessive worker creation
warnings.warn(_create_warning_msg(
FGSM Batches Complete : (157 / 157)
FGSM Test Error : 99.21%
FGSM Robustness : 3.86e-01
FGSM Time (All Images) : 0.82 s
FGSM Time (Per Image) : 82.06 us
/var/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 be aware that excessive worker creation
warnings.warn(_create_warning_msg(
FGSM Batches Complete : (157 / 157)
FGSM Test Error : 99.87%
FGSM Robustness : 1.47e-01
FGSM Time (All Images) : 0.40 s
FGSM Time (Per Image) : 40.06 us
/var/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 be aware that excessive worker creation
warnings.warn(_create_warning_msg(

```

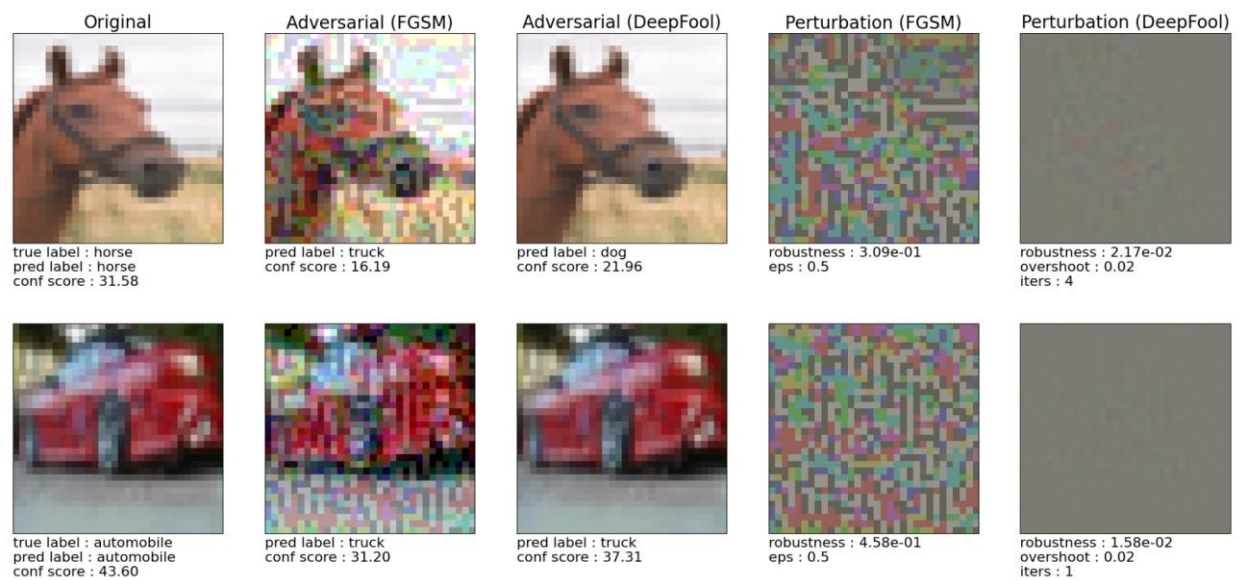
14.2. Esp 0.001



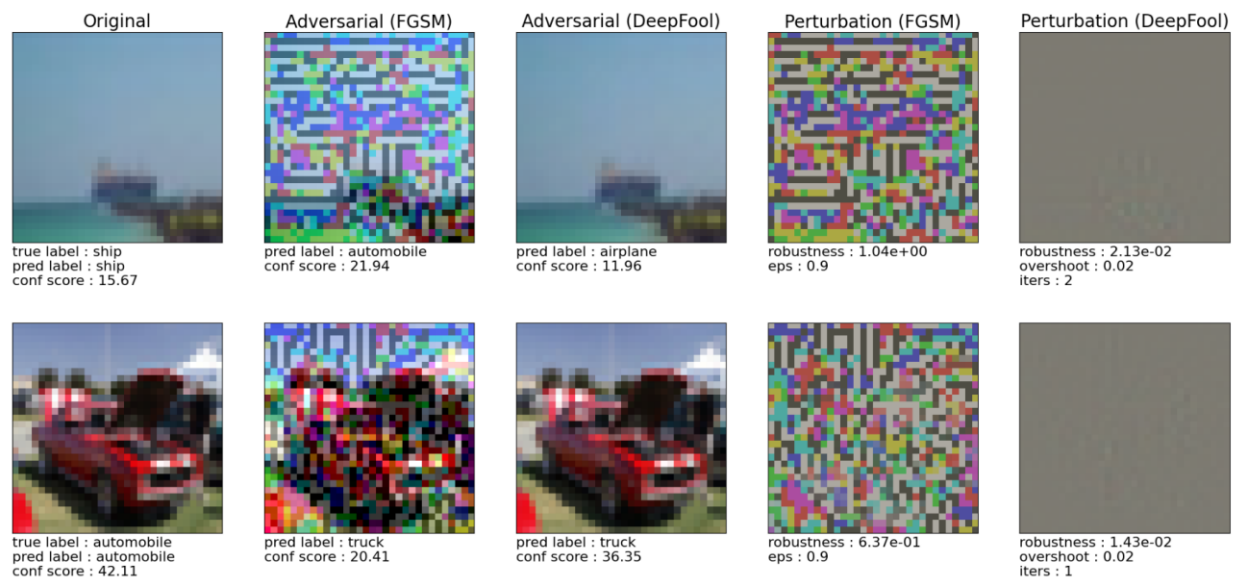
14.5. Esp 0.02



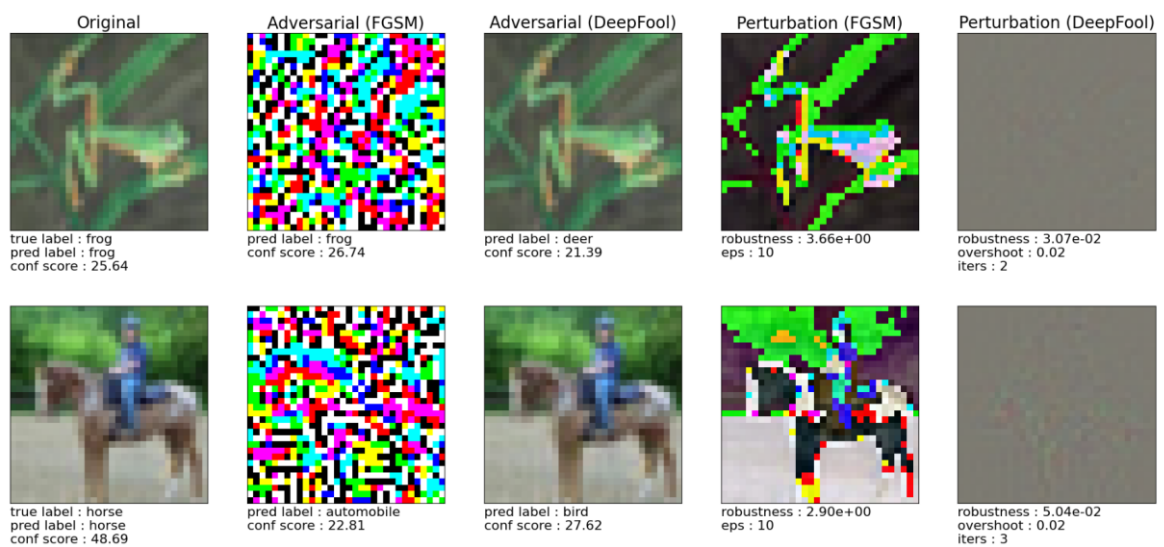
14.6. Esp 0.5



14.7. Esp 0.9



14.8. Esp 10



```

/user/local/lib/python3.10/dist-packages/torch/utils/data/dataloader.py:157: 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 be aware that excessive worker creation might get DataLoader running slow or
warnings.warn(_create_warning_msg)
FGDQ Batches Complete : (157 / 157)
FGDQ Test Error : 50.126
FGDQ Robustness : 0.53e-04
FGDQ Time (All Steps) : 1.33 s
FGDQ Time (Per Step) : 132.79 us
/user/local/lib/python3.10/dist-packages/torch/utils/data/dataloader.py:157: 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 be aware that excessive worker creation might get DataLoader running slow or
warnings.warn(_create_warning_msg)
FGDQ Batches Complete : (157 / 157)
FGDQ Test Error : 50.706
FGDQ Robustness : 1.78e-02
FGDQ Time (All Steps) : 1.25 s
FGDQ Time (Per Step) : 123.16 us
/user/local/lib/python3.10/dist-packages/torch/utils/data/dataloader.py:157: 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 be aware that excessive worker creation might get DataLoader running slow or
warnings.warn(_create_warning_msg)
FGDQ Batches Complete : (157 / 157)
FGDQ Test Error : 82.408
FGDQ Robustness : 4.40e-01
FGDQ Time (All Steps) : 1.36 s
FGDQ Time (Per Step) : 135.72 us
/user/local/lib/python3.10/dist-packages/torch/utils/data/dataloader.py:157: 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 be aware that excessive worker creation might get DataLoader running slow or
warnings.warn(_create_warning_msg)
FGDQ Batches Complete : (157 / 157)
FGDQ Test Error : 84.408
FGDQ Robustness : 7.77e-02
FGDQ Time (All Steps) : 1.39 s
FGDQ Time (Per Step) : 139.48 us
/user/local/lib/python3.10/dist-packages/torch/utils/data/dataloader.py:157: 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 be aware that excessive worker creation might get DataLoader running slow or
warnings.warn(_create_warning_msg)
FGDQ Batches Complete : (157 / 157)
FGDQ Test Error : 87.538
FGDQ Robustness : 2.46e-03
FGDQ Time (All Steps) : 1.31 s
FGDQ Time (Per Step) : 130.49 us

```

Заключение

При рассмотрении результатов эксперимента, проведенного в рамках данной лабораторий работы, была выявлена закономерность, обозначающая, что при увеличении значения ϵ_{rs} , сети становятся более уязвимыми и допускают больше ошибок классификации, нежели при низком значении ϵ_{rs} .