



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:

Картунчиков А. М.

Группа:

ББМО-01-23

Москва, 2024 г.

Скопируем проект по ссылке в локальную среду выполнения

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

Сменим директорию исполнения на вновь созданную папку

"EEL6812_DeepFool_Project" проекта

```
[2] %cd /content/EEL6812_DeepFool_Project
```

```
↗ /content/EEL6812_DeepFool_Project
```

Выполним импорт библиотек

```
[3] 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
```

```
[4] from google.colab import drive
drive.mount('/content/drive')
```

```
↗ Mounted at /content/drive
```

Установим случайное рандомное значение в виде переменной rand_seed для варианта 14

```
rand_seed={"14"}
```

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

[7] device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")

[8] 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))])

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):
(unknown open SSL: CRYPTO: VERIFY: failed: certificate has expired ( ssl.c:1097))
```

Загрузим 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

[10] 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()])
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_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%|██████████| 170M/170M [00:04<00:00, 40.2MB/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

```
[11] batch_size = 64
     workers = 4

[12] 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:617: UserWarning: This DataLoader will create 4 worker processes on CUDA device 0. This requires that your system has GPU devices with enough free memory to avoidOOMs. You may want to increase the number of workers to 4 to avoid OOMs. Please see the documentation on DataLoader for more details (https://pytorch.org/docs/stable/data_loader.html).
warnings.warn(
```

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/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('someprint')

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()
```

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)

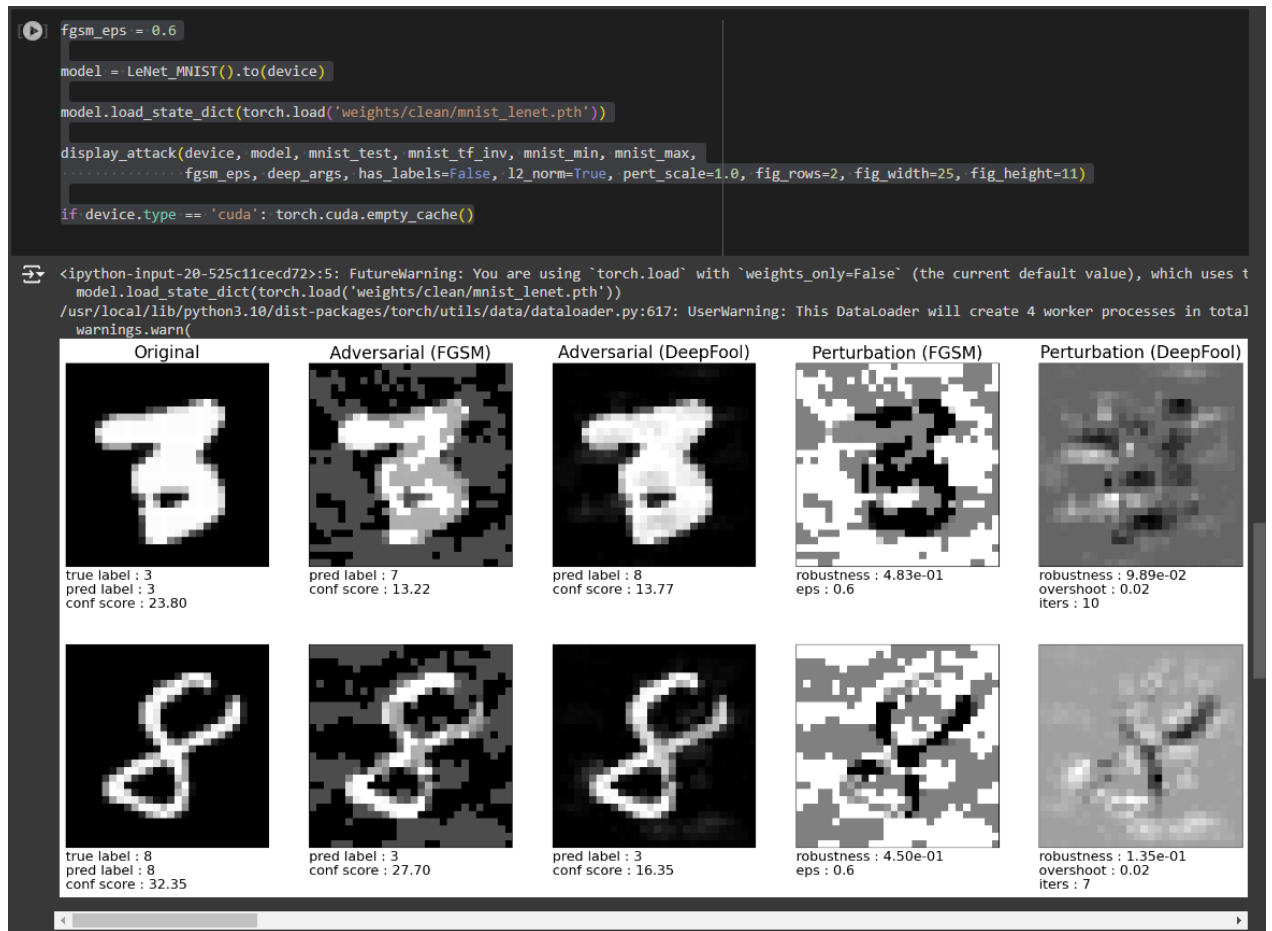
if device.type == 'cuda': torch.cuda.empty_cache()
```

```
⚙ FGSM Test Error : 91.71%
  FGSM Robustness : 8.90e-02
  FGSM Time (All Images) : 0.40 s
  FGSM Time (Per Image) : 40.08 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
  <ipython-input-18-b0414236475d>:5: FutureWarning: You are using `torch.load` with `weights_only=False` (the current default value), which uses the default loading mechanism that is vulnerable to the deserialization of untrusted data. To use a safer loading mechanism that does not execute arbitrary code, you should set `weights_only=True` in `torch.load`. This will become the default for `torch.load` in a future PyTorch release.
    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



FCNet на MNIST

```
fgsm_eps = 0.2



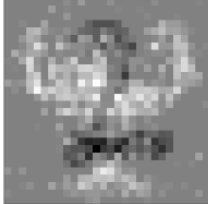
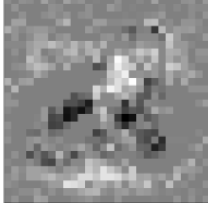
model = FC_500_150().to(device)

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)




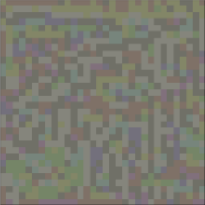




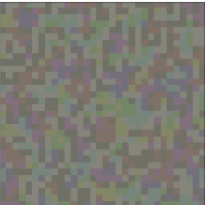
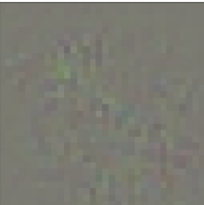
if device.type == 'cuda': torch.cuda.empty_cache()
```

<ipython-input-21-6500a8777172>:5: FutureWarning: You are using `torch.load` with `weights_only=False` (the current default value), which uses t
model.load_state_dict(torch.load('weights/clean/mnist_fc.pth'))

Original	Adversarial (FGSM)	Adversarial (DeepFool)	Perturbation (FGSM)	Perturbation (DeepFool)
 <p>true label : 2 pred label : 2 conf score : 15.11</p>	 <p>pred label : 1 conf score : 10.56</p>	 <p>pred label : 8 conf score : 9.78</p>	 <p>robustness : 1.60e-01 eps : 0.2</p>	 <p>robustness : 7.58e-02 overshoot : 0.02 iters : 9</p>
 <p>true label : 2 pred label : 2 conf score : 15.74</p>	 <p>pred label : 3 conf score : 16.29</p>	 <p>pred label : 3 conf score : 12.41</p>	 <p>robustness : 1.65e-01 eps : 0.2</p>	 <p>robustness : 6.47e-02 overshoot : 0.02 iters : 11</p>

Network-in-Network на CIFAR

```
> ipython-input-22-77adf8c089e0::: FutureWarning: You are using `torch.load` with `weights_only=False` (the current default value), which uses the
model.load_state_dict(torch.load('weights/clean/cifar_nin.pth'))
```

Original	Adversarial (FGSM)	Adversarial (DeepFool)	Perturbation (FGSM)	Perturbation (DeepFool)
 <p>true label : truck pred label : truck conf score : 34.54</p>	 <p>pred label : frog conf score : 20.32</p>	 <p>pred label : automobile conf score : 24.70</p>	 <p>robustness : 1.49e-01 eps : 0.2</p>	 <p>robustness : 2.99e-02 overshoot : 0.02 iters : 3</p>
 <p>true label : bird pred label : bird conf score : 41.46</p>	 <p>pred label : bird conf score : 39.99</p>	 <p>pred label : frog conf score : 17.57</p>	 <p>robustness : 2.12e-01 eps : 0.2</p>	 <p>robustness : 7.69e-02 overshoot : 0.02 iters : 3</p>

LeNet на CIFAR

```
fsgm_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, fsgm_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()
```

<ipython-input-23-79fd989ba557>:5: FutureWarning: You are using `torch.load` with `weights_only=False` (the current default value), which uses t
model.load_state_dict(torch.load('weights/clean/cifar_lenet.pth'))

Original	Adversarial (FGSM)	Adversarial (DeepFool)	Perturbation (FGSM)	Perturbation (DeepFool)
				
true label : airplane pred label : automobile conf score : 9.30	pred label : automobile conf score : 15.60	pred label : airplane conf score : 4.54	robustness : 6.33e-02 eps : 0.1	robustness : 1.42e-02 overshoot : 0.02 iters : 4
				
true label : horse pred label : horse conf score : 9.53	pred label : dog conf score : 8.07	pred label : dog conf score : 6.92	robustness : 7.61e-02 eps : 0.1	robustness : 2.21e-02 overshoot : 0.02 iters : 2

Отразим отличия для $fsgm_eps = (0.001, 0.02, 0.5, 0.9, 10)$

```
fsgm_eps_arr = [0.001, 0.02, 0.5, 0.9, 10]

for fsgm_eps in fsgm_eps_arr:
    print("eps: ", fsgm_eps)

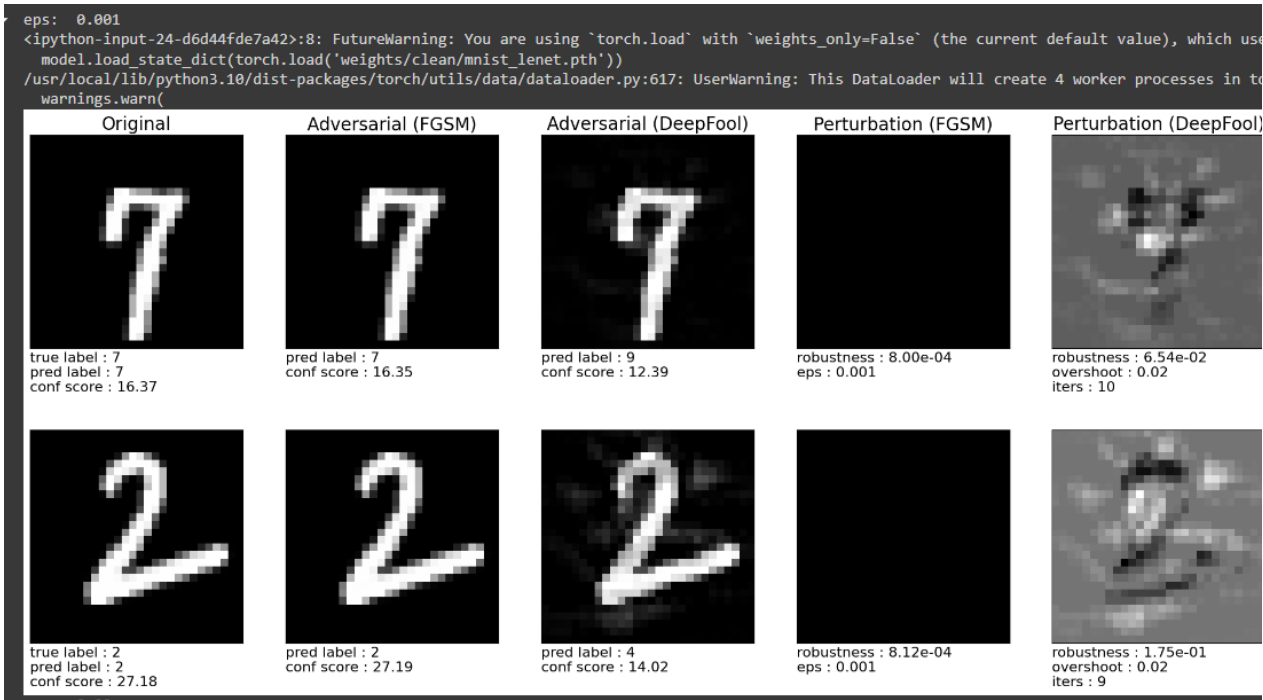
    model = LeNet_MNIST().to(device)

    model.load_state_dict(torch.load('weights/clean/mnist_lenet.pth'))

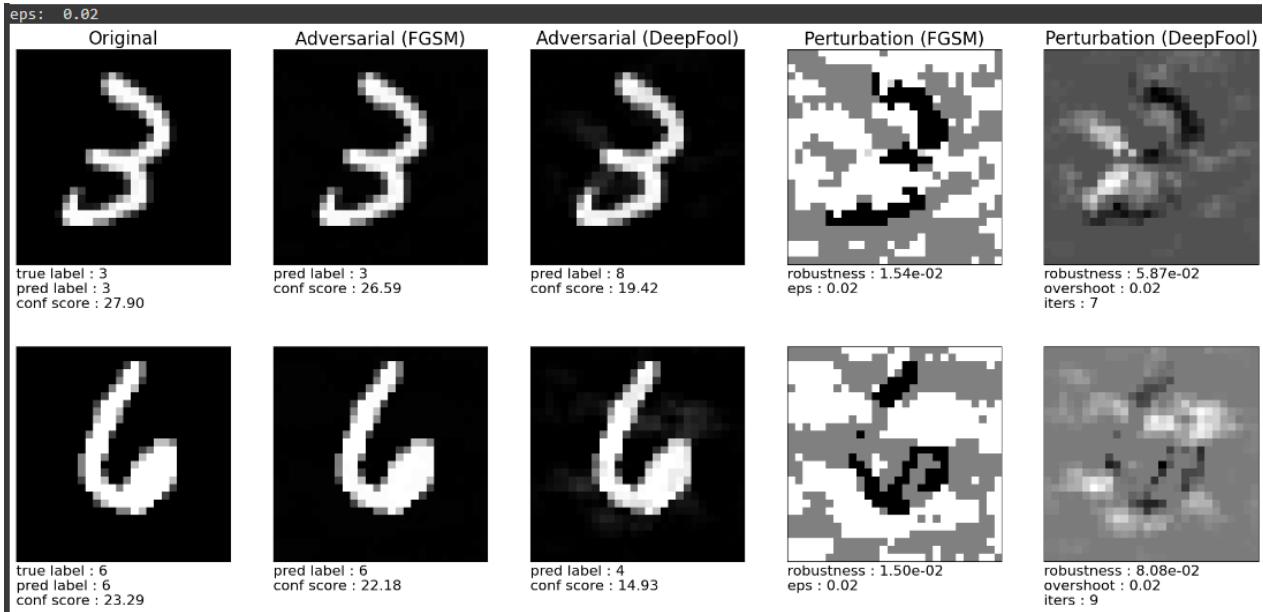
    display_attack(device, model, mnist_test, mnist_tf_inv, mnist_min, mnist_max,
                  fsgm_eps, deep_args, has_labels=False, l2_norm=True, pert_scale=1.0, fig_rows=2, fig_width=25, fig_height=11)

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

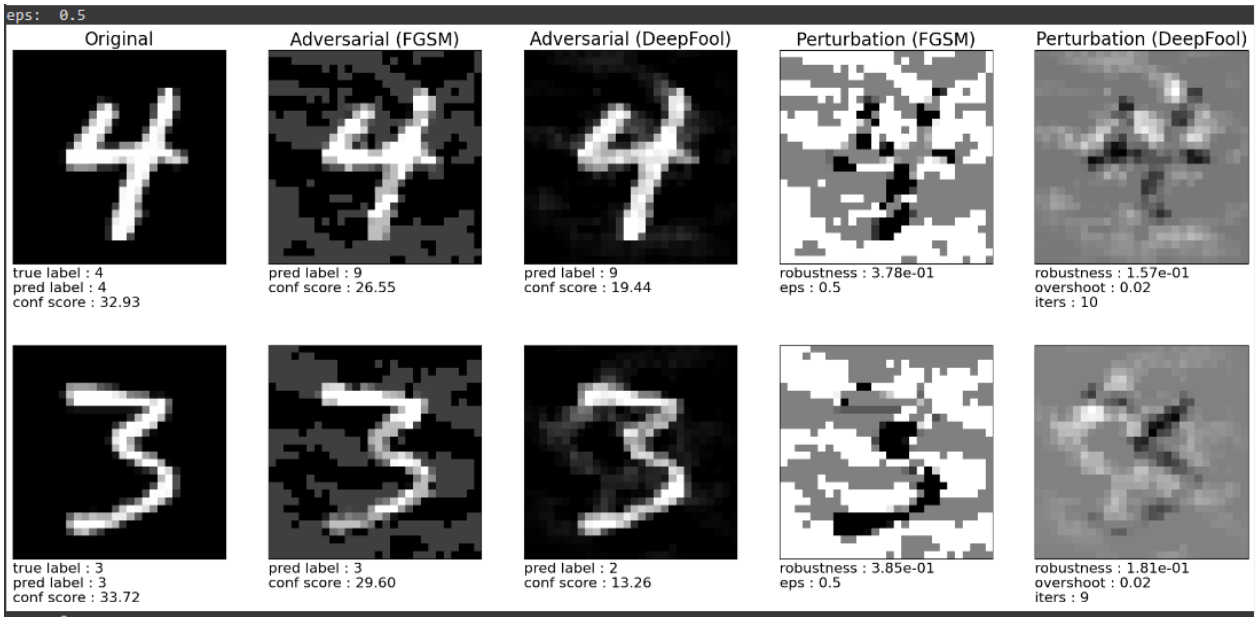

LeNet на MNIST, eps: 0,001



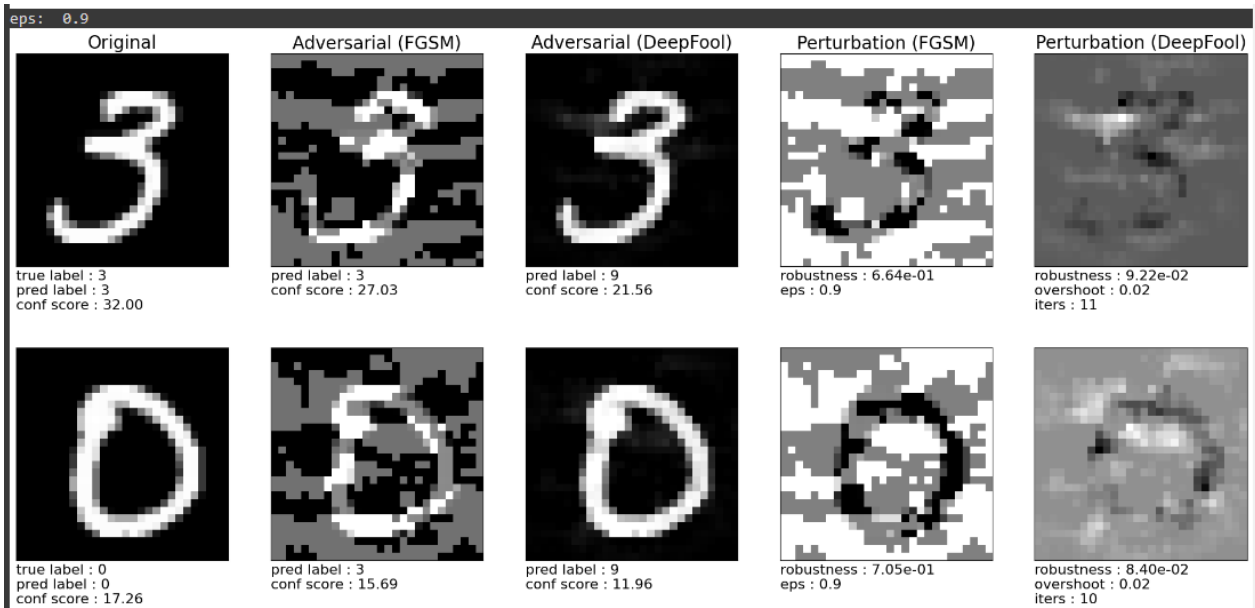
LeNet на MNIST, eps: 0,02



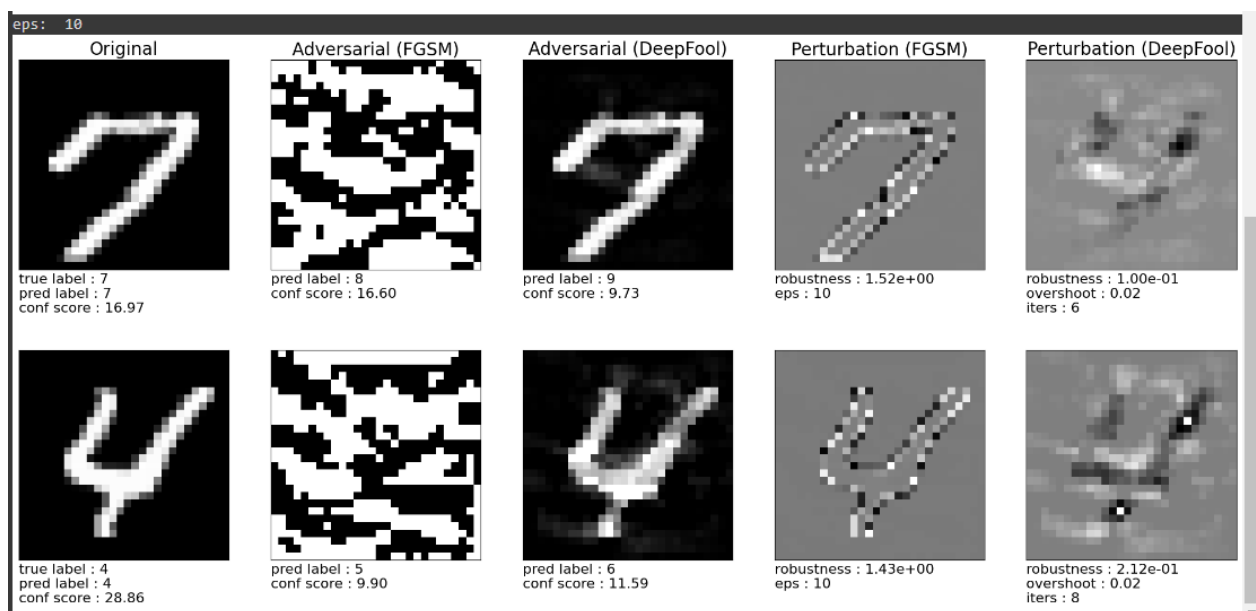
LeNet на MNIST, eps: 0,5



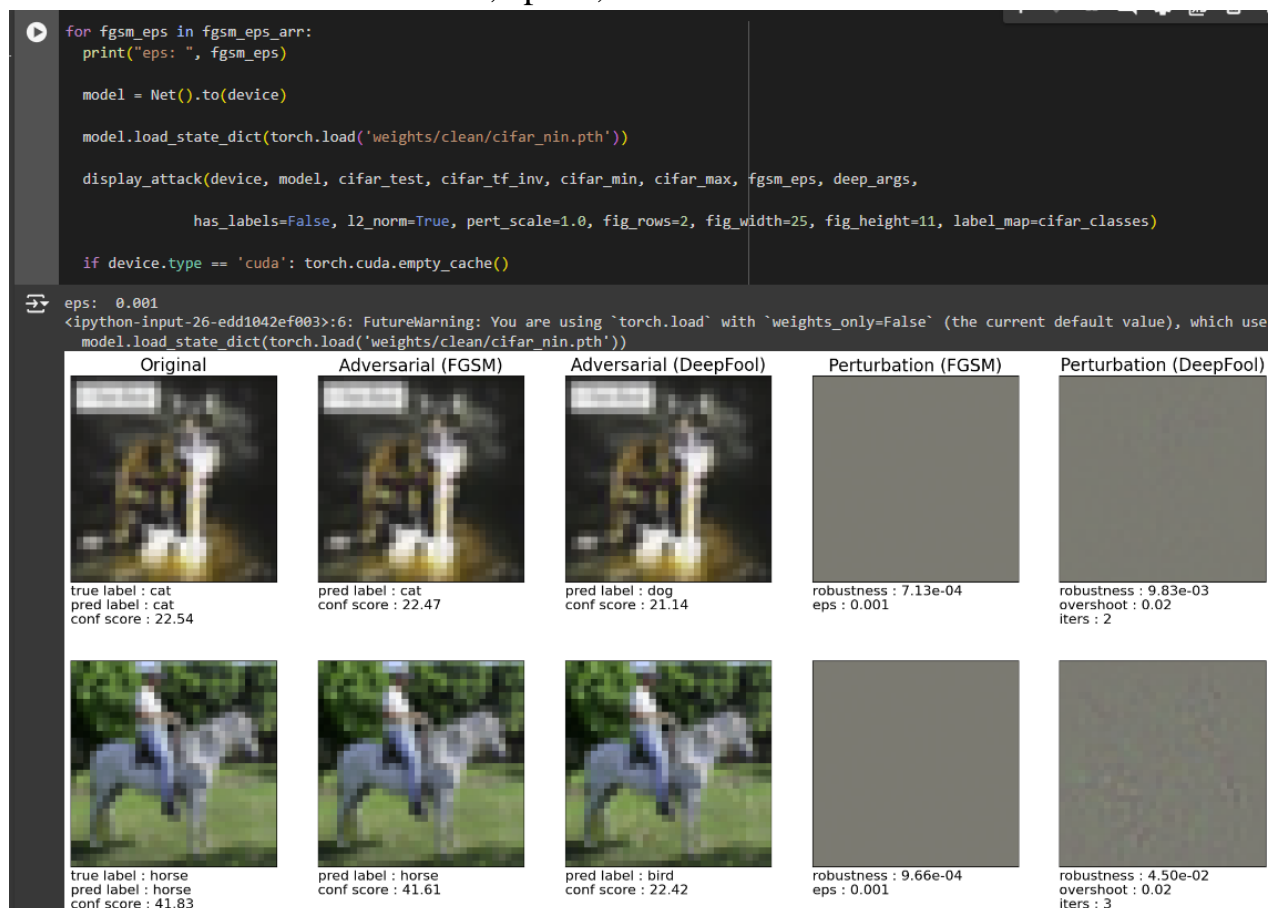
LeNet на MNIST, eps: 0,9



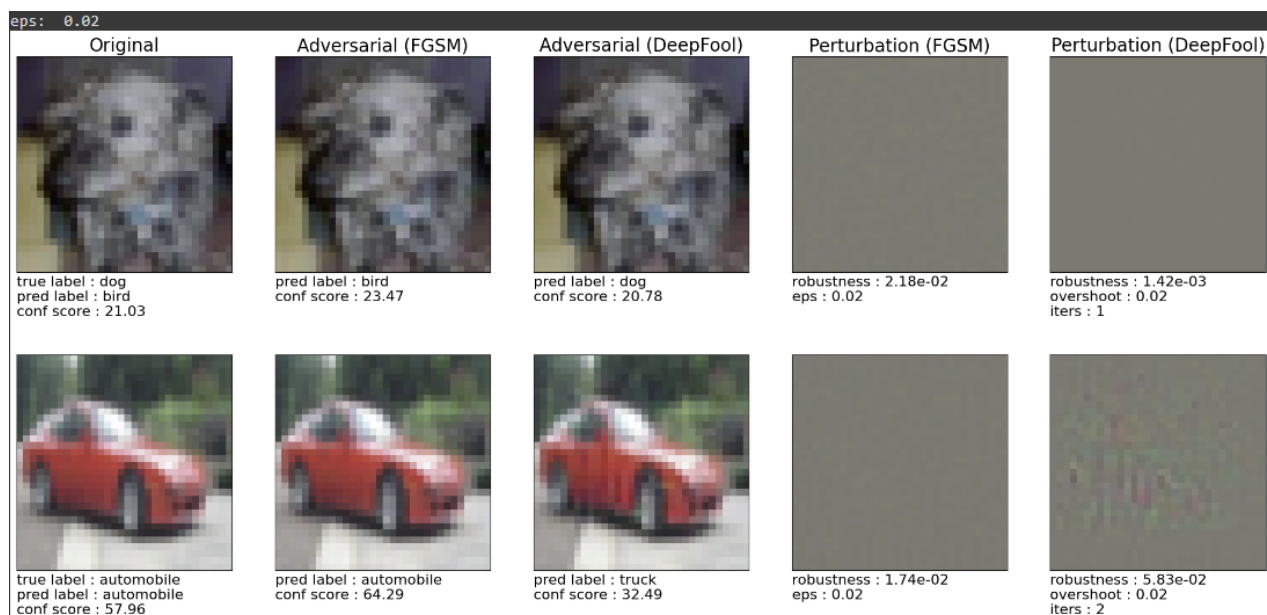
LeNet на MNIST, eps: 10



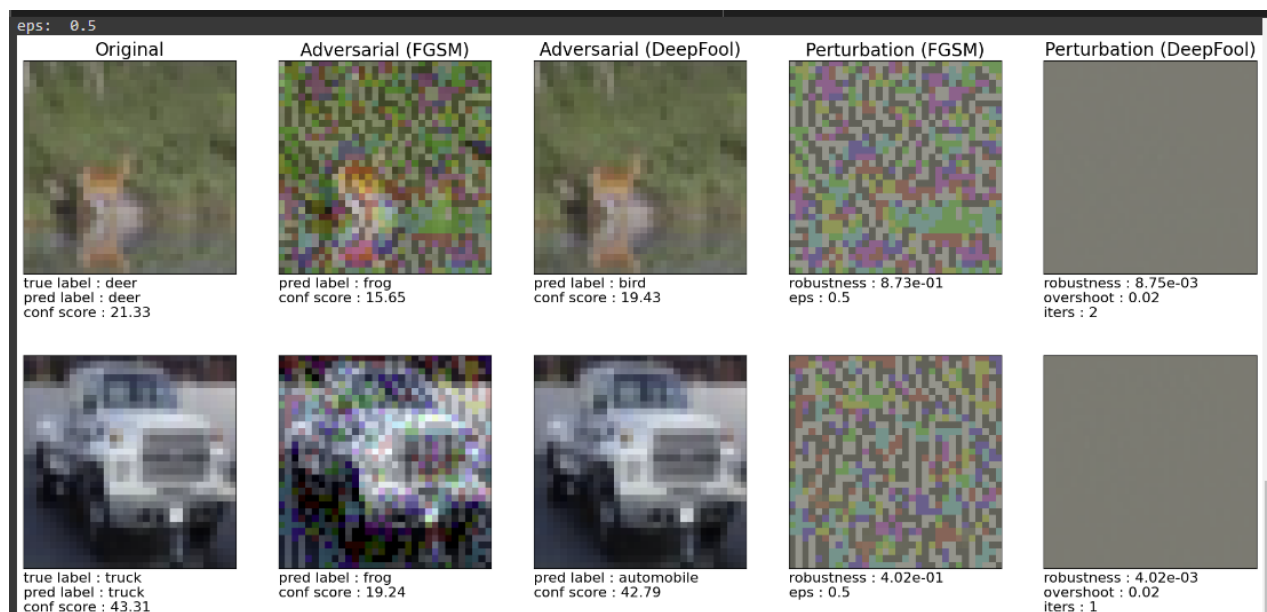
Network-in-Network на CIFAR, eps: 0,001



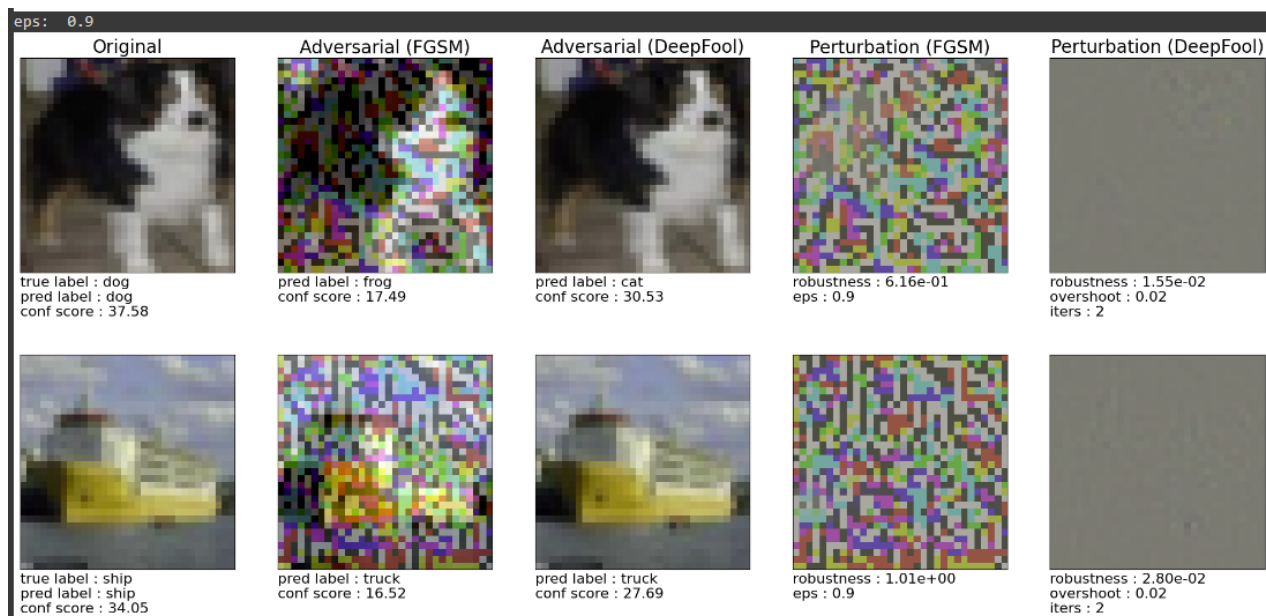
Network-in-Network на CIFAR, eps: 0,02



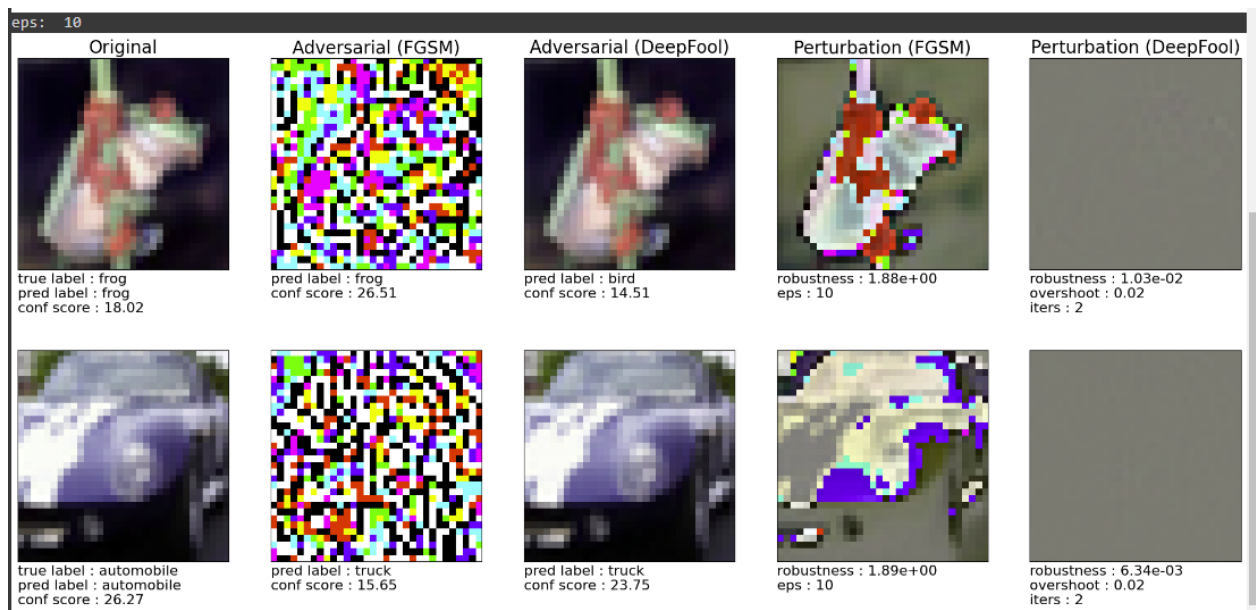
Network-in-Network на CIFAR, eps: 0,05



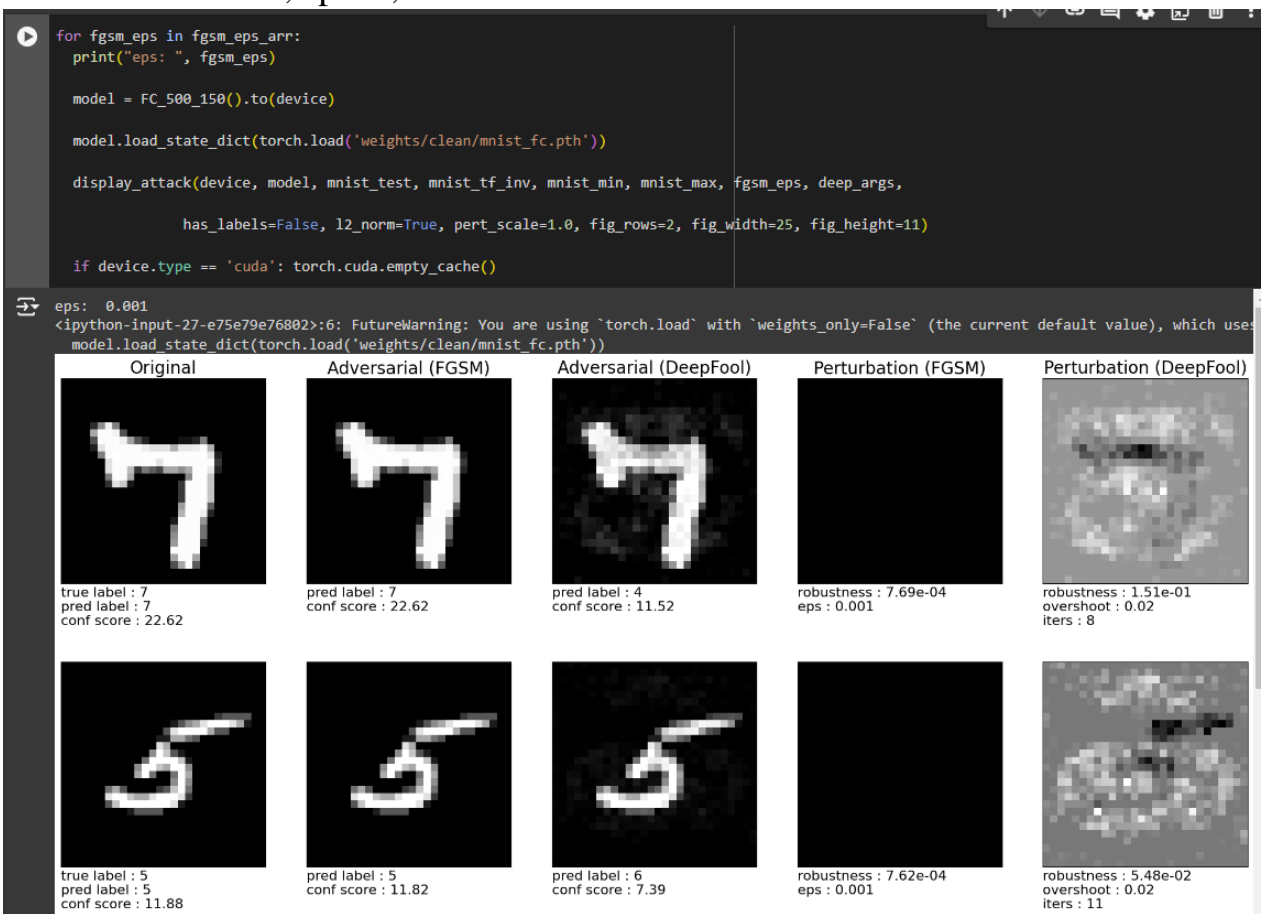
Network-in-Network на CIFAR, eps: 0,9



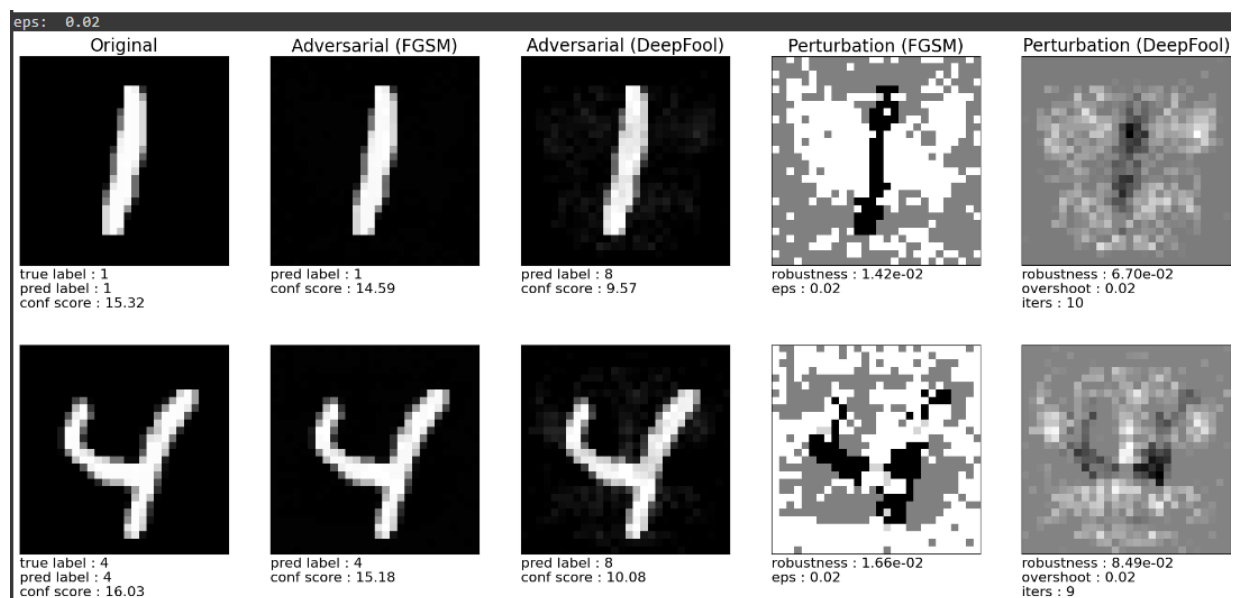
Network-in-Network на CIFAR, eps: 10



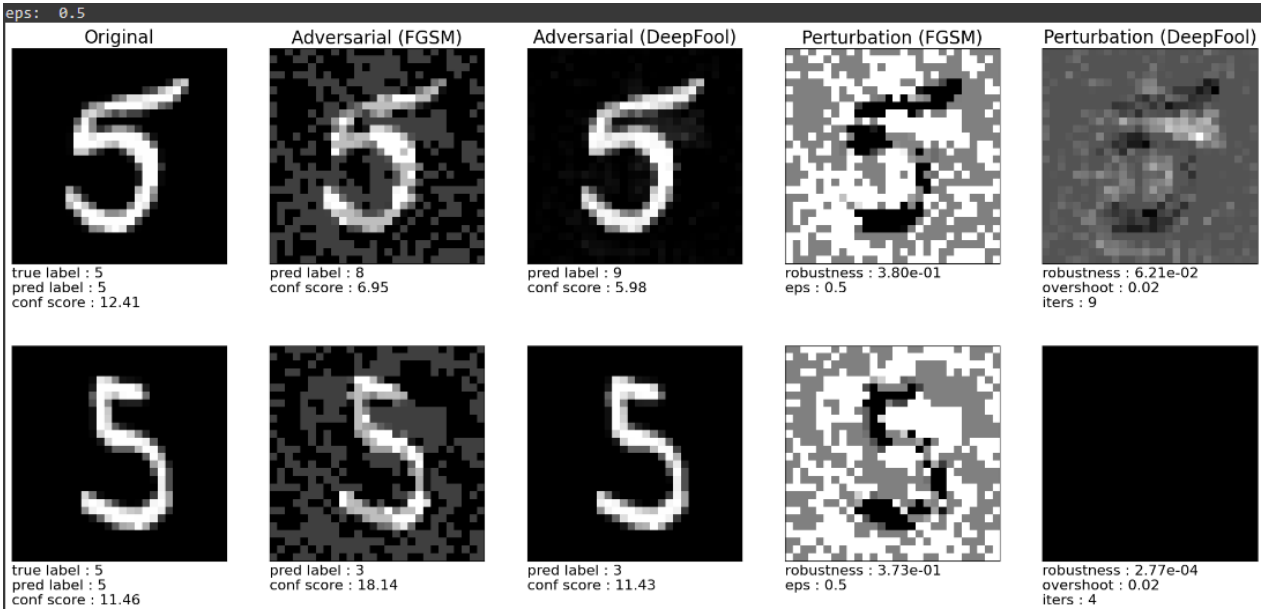
FCNet на MNIST, eps: 0,001



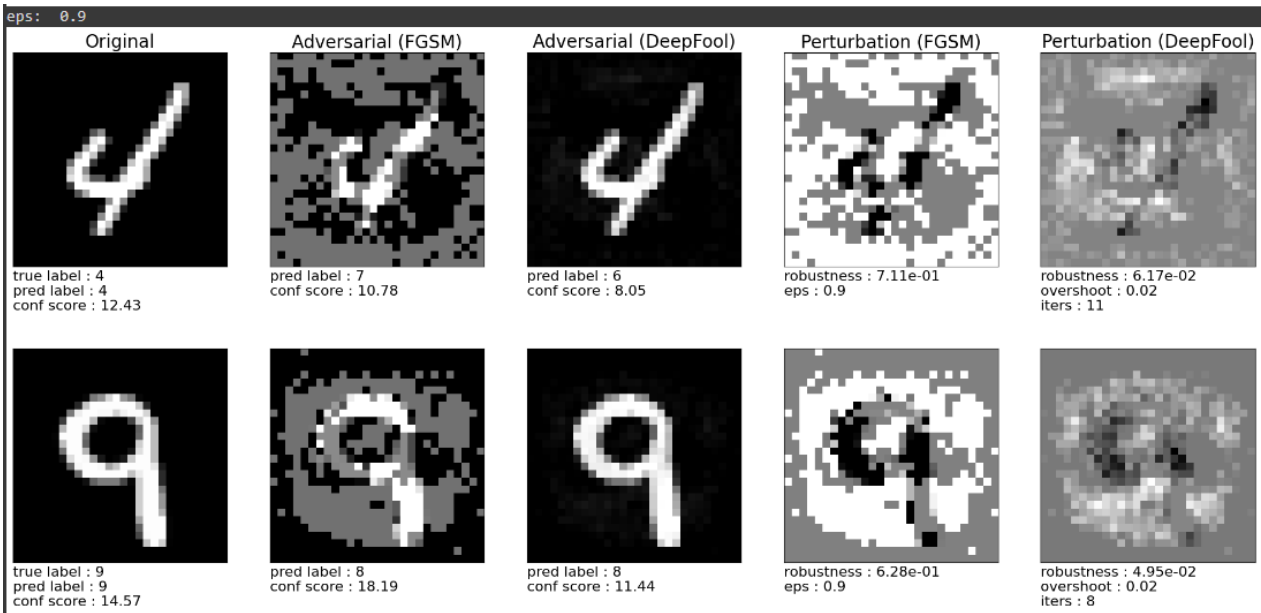
FCNet на MNIST, eps: 0,02



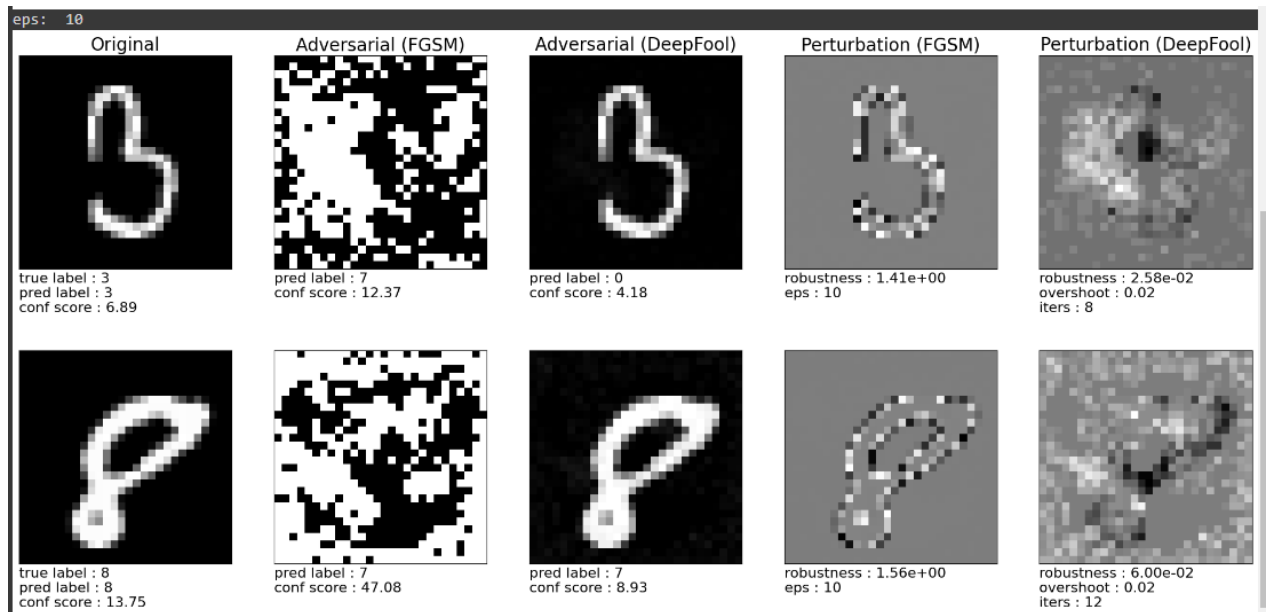
eps: 0,5



FCNet на MNIST, eps: 0,9



eps: 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.