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:

Филимонов И.М.

Группа:

ББМО-01-22, 2 курс

Проверил:

Спирин А.А.

Москва, 2024 г.

Скопируем проект по ссылке в локальную среду выполнения

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

```
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 | 18.13 MiB/s, done.
Resolving deltas: 100% (27/27), done.
```

Сменим директорию исполнения на вновь созданную папку "EEL6812_DeepFool_Project" проекта

```
%cd /content/EEL6812_DeepFool_Project
```

```
/content/EEL6812_DeepFool_Project
```

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
```

Выполним импорт вспомогательных библиотек из локальных файлов проекта

```
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 10

```
rand_seed = 10

# Установим указанное значение для np.random.seed и torch.manual_seed
np.random.seed(rand_seed)
torch.manual_seed(rand_seed)
```

```
<torch._C.Generator at 0x781914375af0>
```

Используем в качестсве устройства видеокарту

```
use_cuda = torch.cuda.is_available()
device = torch.device('cuda' if use_cuda 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( me

mnist_tf_inv = transforms.Compose([ transforms.Normalize( mean=0.0, std=np.divide(1.0, mnist_std)), transforms.Normalize

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-ubyt
e.gz
100%|██████████| 9912422/9912422 [00:00<00:00, 190126449.62it/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-ubyt
e.gz
100%|██████████| 28881/28881 [00:00<00:00, 39821069.63it/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.g
z
100%|██████████| 1648877/1648877 [00:00<00:00, 75522434.28it/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.g
z
100%|██████████| 4542/4542 [00:00<00:00, 21821911.53it/s]
Extracting datasets/mnist/MNIST/raw/t10k-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

cifar_tf_inv = transforms.Compose([ transforms.Normalize( mean=[0.0, 0.0, 0.0], std=np.divide(1.0, cifar_std)), transfor

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, 29134501.88it/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)
```

```
/usr/local/lib/python3.10/dist-packages/torch/utils/data/dataloader.py:557: UserWarning: This DataLoader will create 4 work
er processes in total. Our suggested max number of worker in current system is 2, which is smaller than what this DataLoade
r is going to create. Please be aware that excessive worker creation might get DataLoader running slow or even freeze, lowe
r the worker number to avoid potential slowness/freeze if necessary.
  warnings.warn(_create_warning_msg(
```

Настроим параметры для обучения

```
batch_size = 10
num_classes = 10
overshoot = 0.02
max_iters = 50
deep_args = [batch_size, num_classes, overshoot, max_iters]
```

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_fg
print('')
evaluate_attack('cifar_nin_deepfool.csv', 'results', device, model, cifar_loader_test, cifar_min, cifar_max, deep_args,

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
```

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

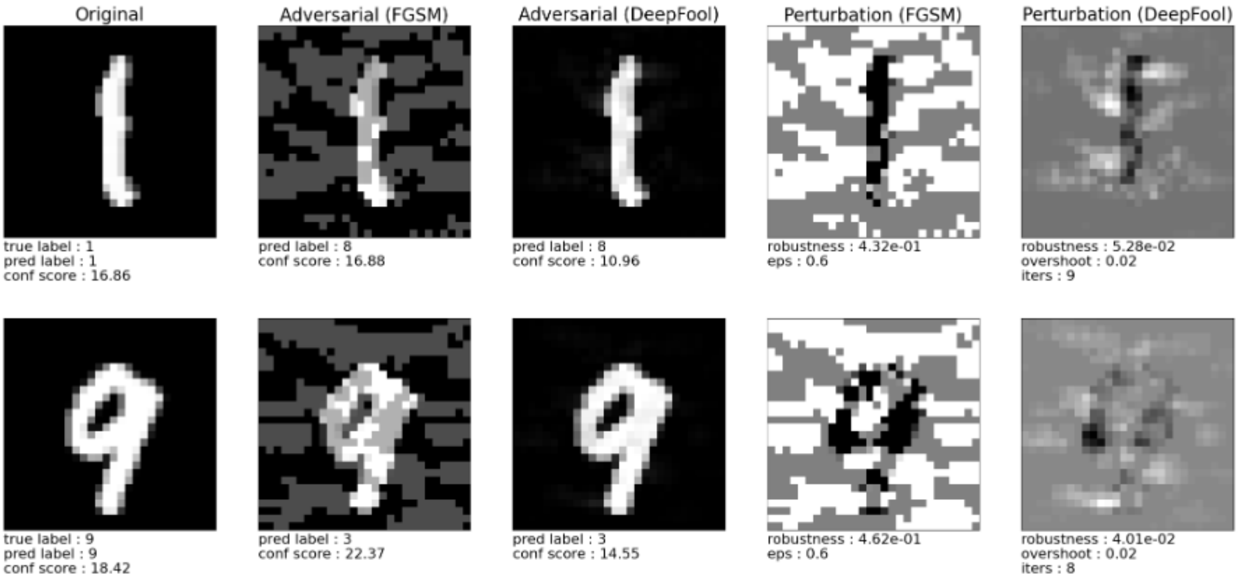
```
fgsm_eps = 0.1
model = LeNet_CIFAR().to(device)
model.load_state_dict(torch.load('weights/clean/cifar_lenet.pth', map_location=torch.device('cpu')))

evaluate_attack('cifar_lenet_fgsm.csv', 'results', device, model, cifar_loader_test, cifar_min, cifar_max, fgsm_eps, is_
print('')
evaluate_attack('cifar_lenet_deepfool.csv', 'results', device, model, cifar_loader_test, cifar_min, cifar_max, deep_args

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
```

Выполним оценку атакующих примеров для сетей:

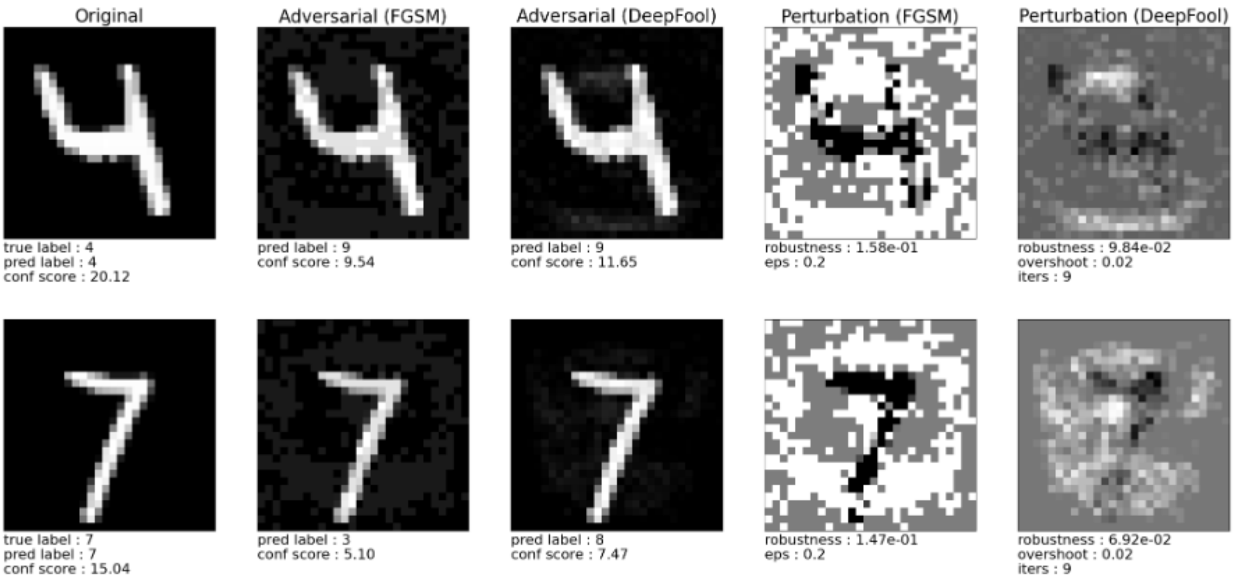
LeNet на MNIST

```
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_

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



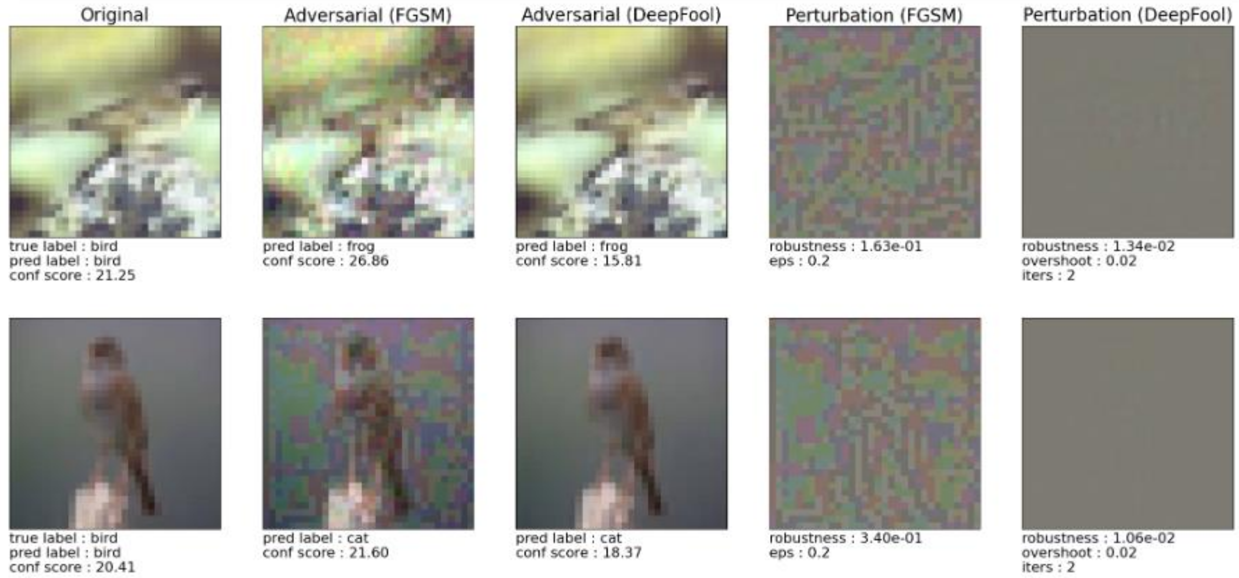
FCNet на MNIST

```
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_

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



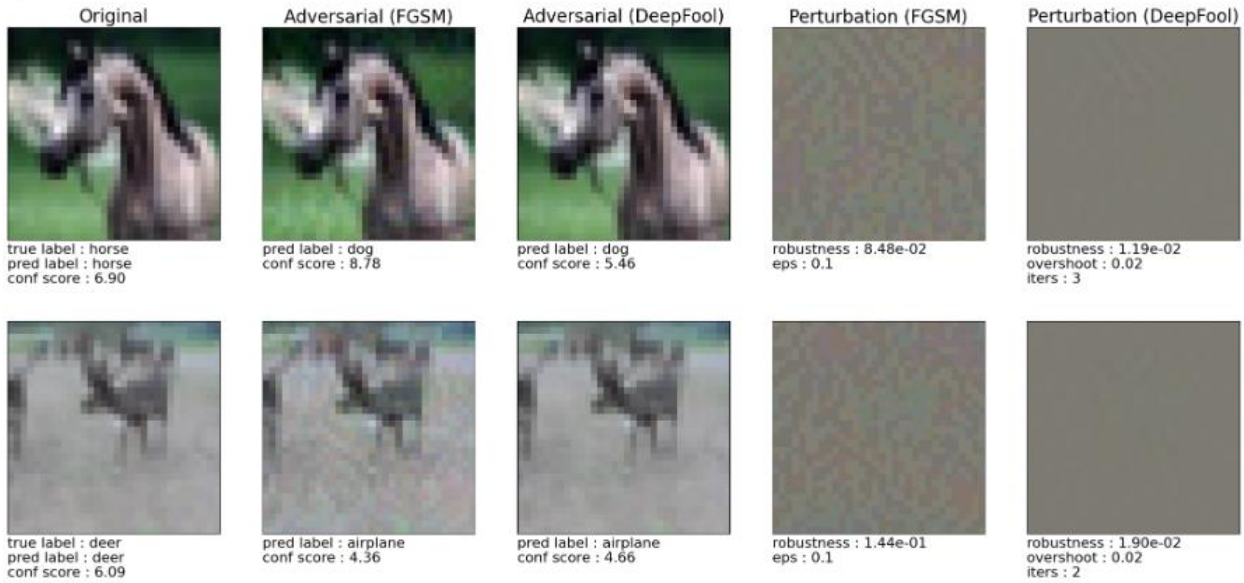
Network-in-Network на CIFAR

```
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_

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



LeNet на CIFAR

```
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_

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

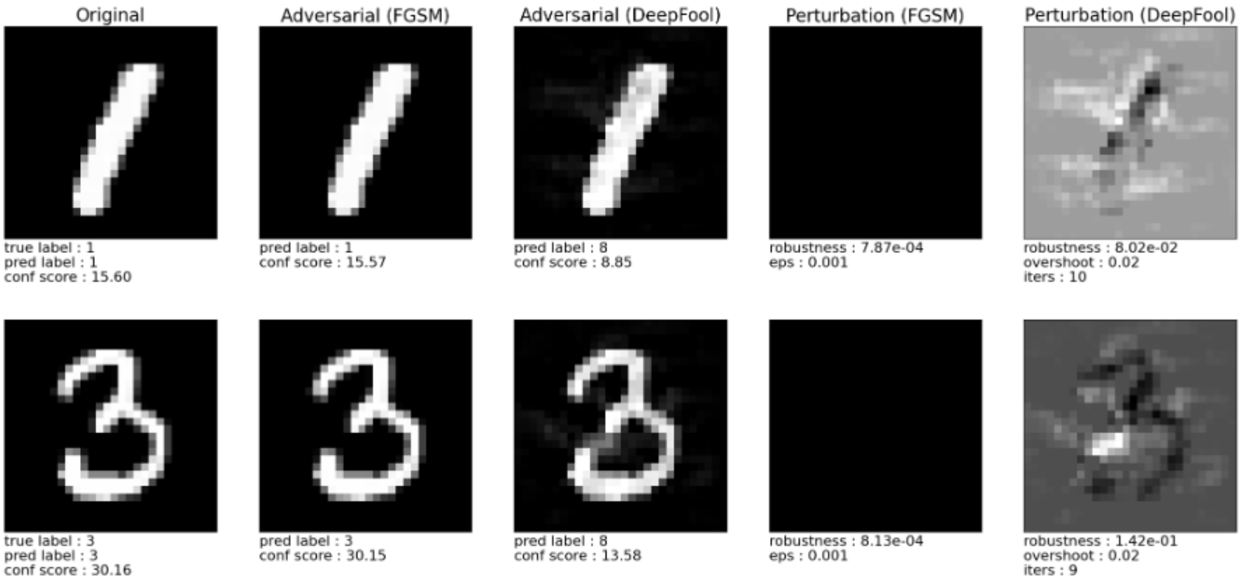


Отразим отличия для fgsm_eps = (0.001, 0.02, 0.5, 0.9, 10)

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

for fgsm_eps in fgsm_eps_arr:
  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, l

  if device.type == 'cuda': torch.cuda.empty_cache()
  print("eps: ", fgsm_eps)
```

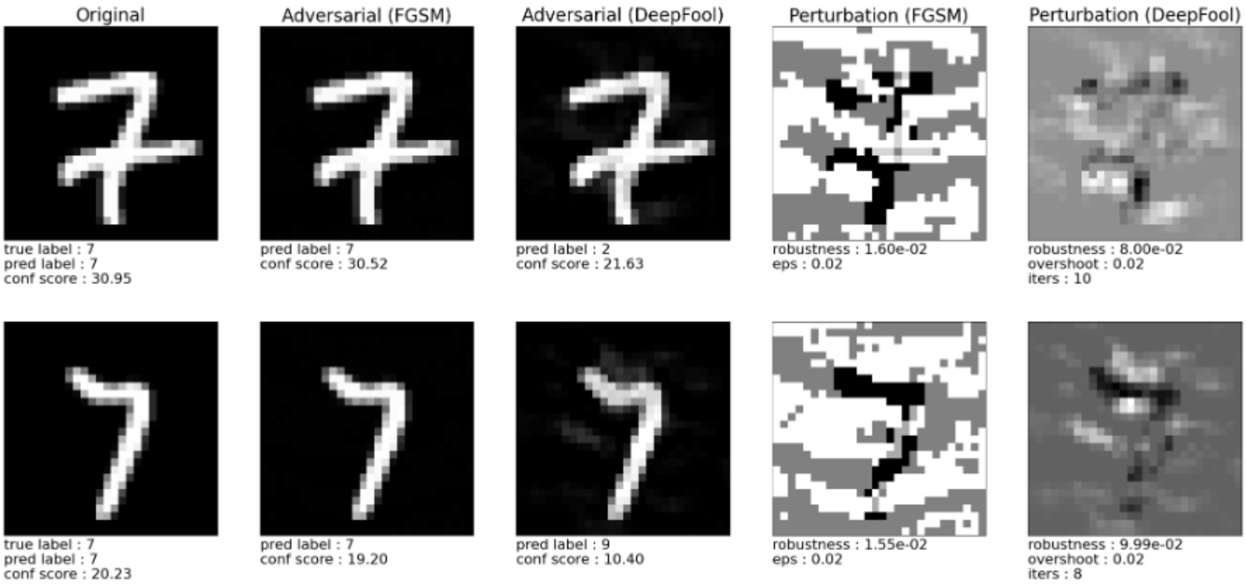
```
/usr/local/lib/python3.10/dist-packages/torch/utils/data/dataloader.py:557: UserWarning: This DataLoader will create 4 work
er processes in total. Our suggested max number of worker in current system is 2, which is smaller than what this DataLoade
r is going to create. Please be aware that excessive worker creation might get DataLoader running slow or even freeze, lowe
r the worker number to avoid potential slowness/freeze if necessary.
  warnings.warn(_create_warning_msg(
```

LeNet на MNIST, eps: 0,001



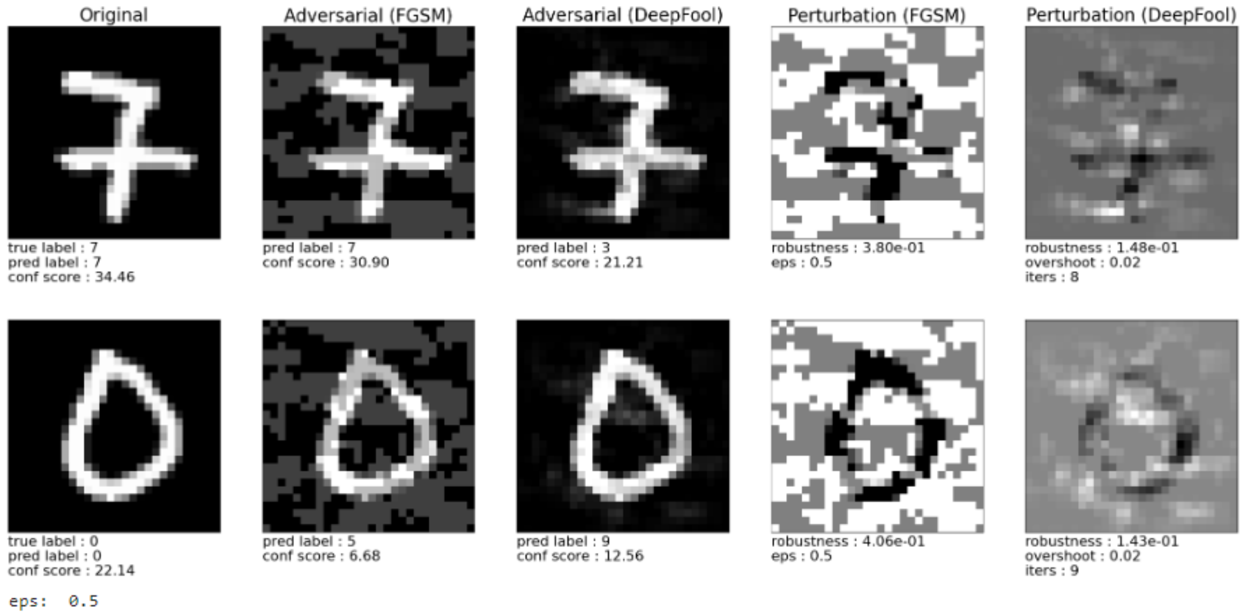

```
eps:  0.001
```

LeNet на MNIST, eps: 0,02

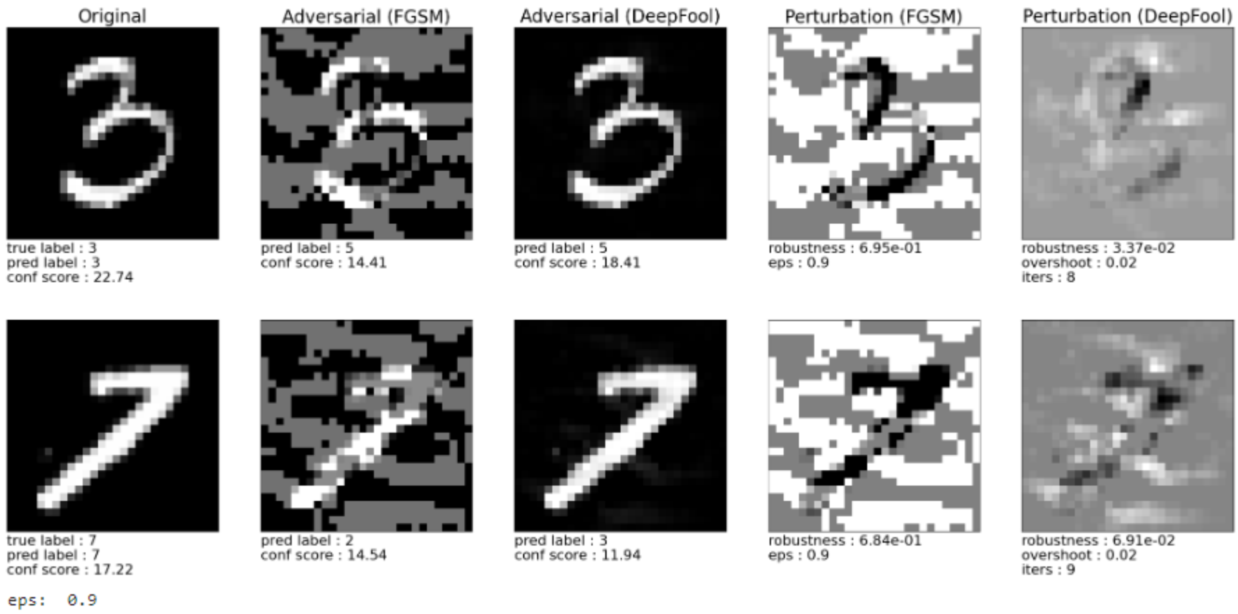



```
eps:  0.02
```

LeNet на MNIST, eps: 0,5

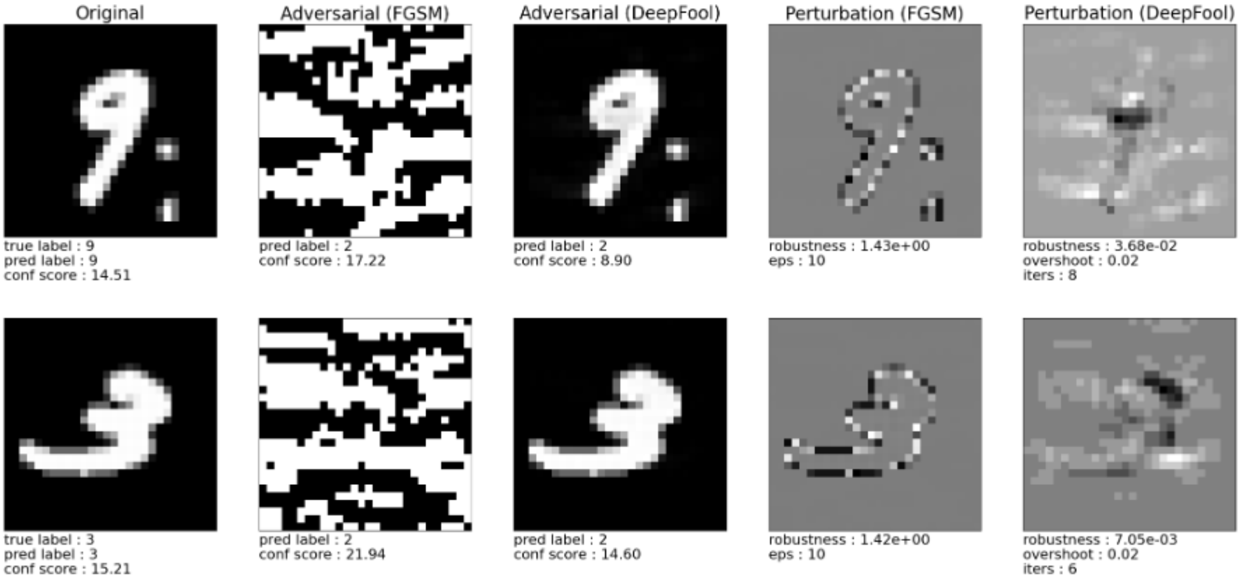


LeNet на MNIST, eps: 0,9



LeNet на MNIST, eps: 10

```
eps:  0.9
```



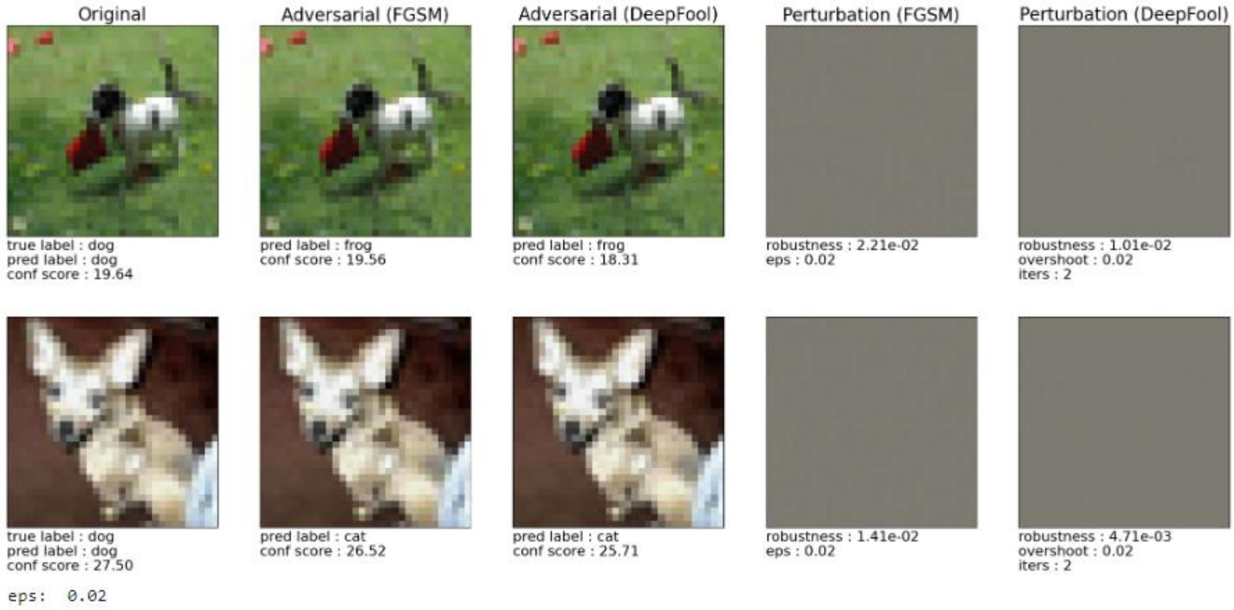

```
eps:  10
```

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

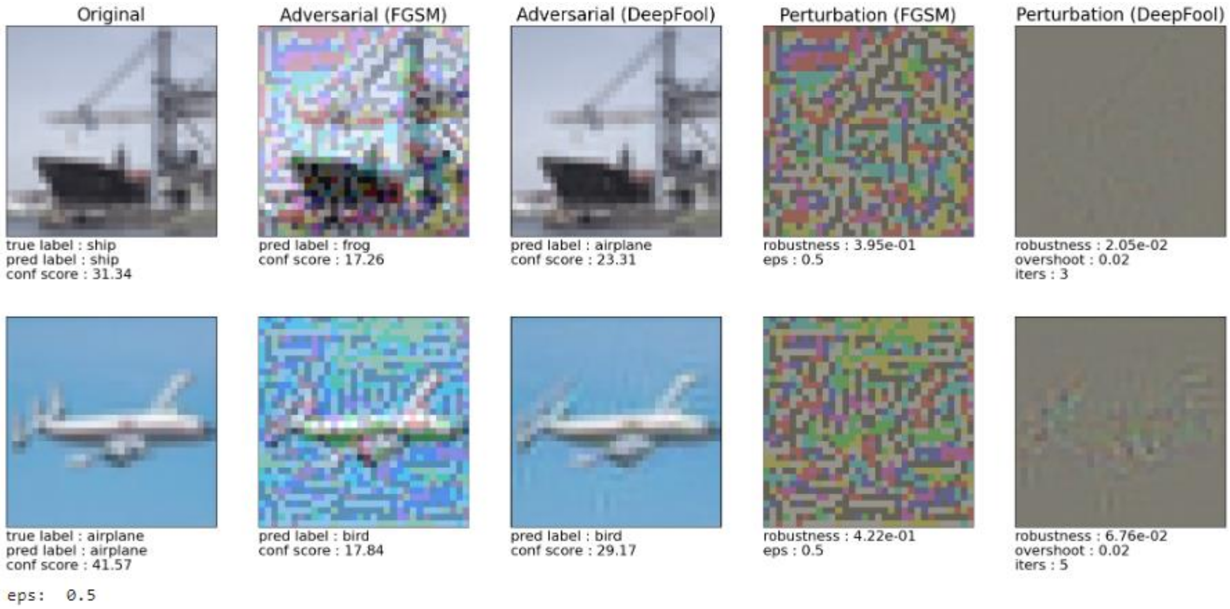
```
for fgsm_eps in fgsm_eps_arr:
  model = Net().to(device)
  model.load_state_dict(torch.load('weights/clean/cifar_nin.pth'))

  display_attack(device, model, cifar_test, cifar_tf_inv, cifar_min, cifar_max, fgsm_eps, deep_args, has_labels=False, l

  if device.type == 'cuda': torch.cuda.empty_cache()
  print("eps: ", fgsm_eps)
```


```
eps:  0.001
```

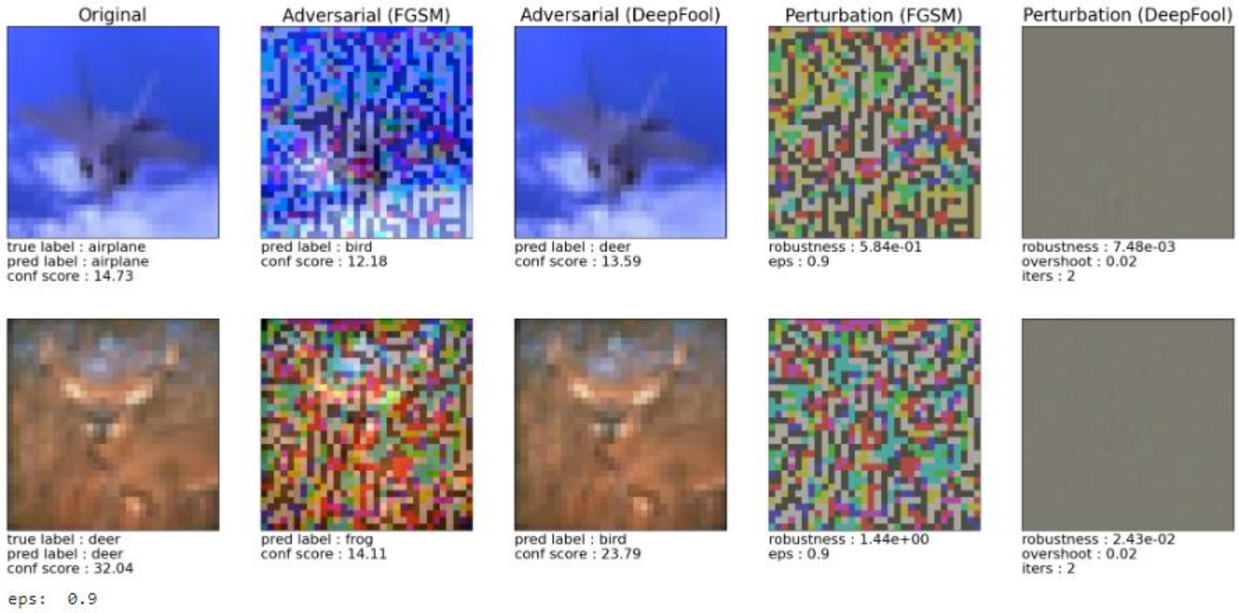
Network-in-Network на CIFAR, eps: 0,02



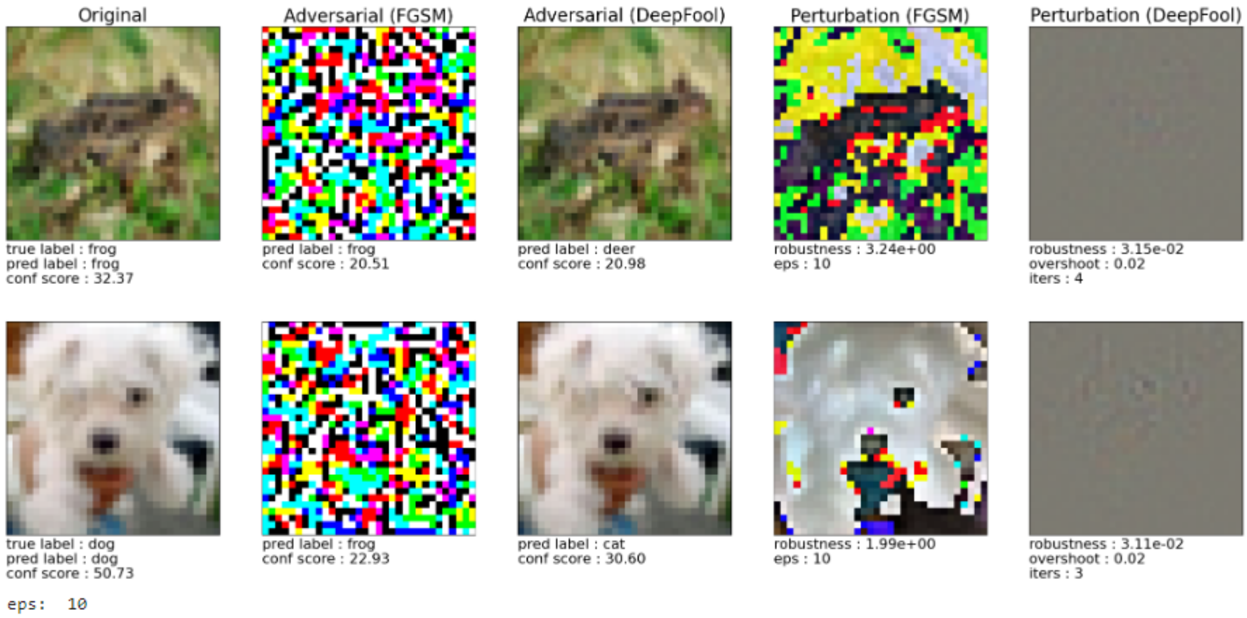
Network-in-Network на CIFAR, eps: 0,05



Network-in-Network на CIFAR, eps: 0,09

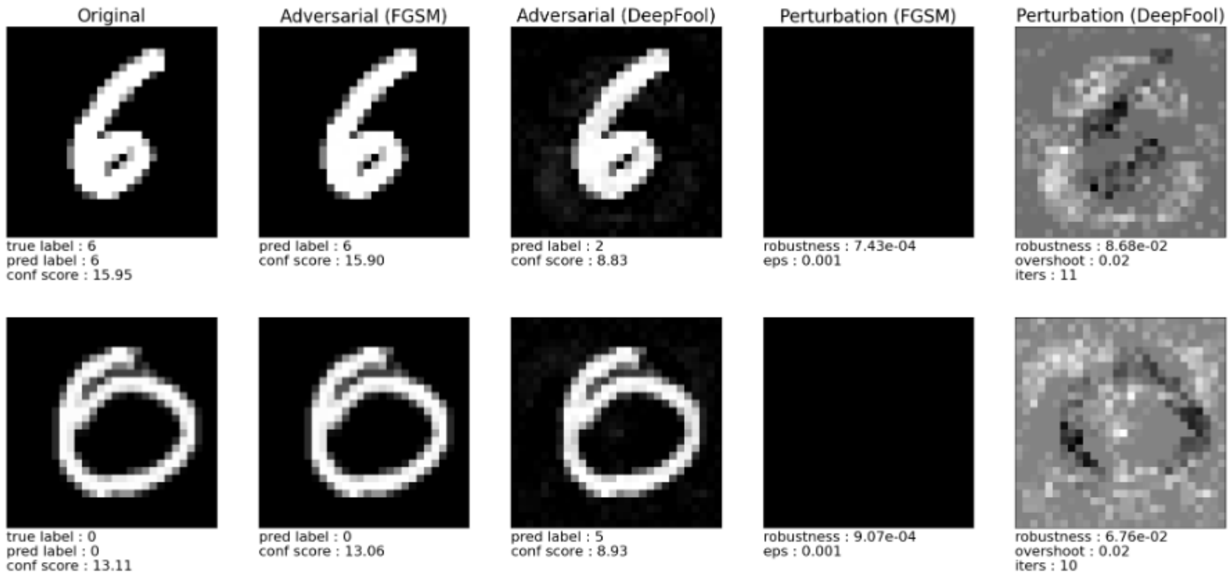


Network-in-Network на CIFAR, eps: 10



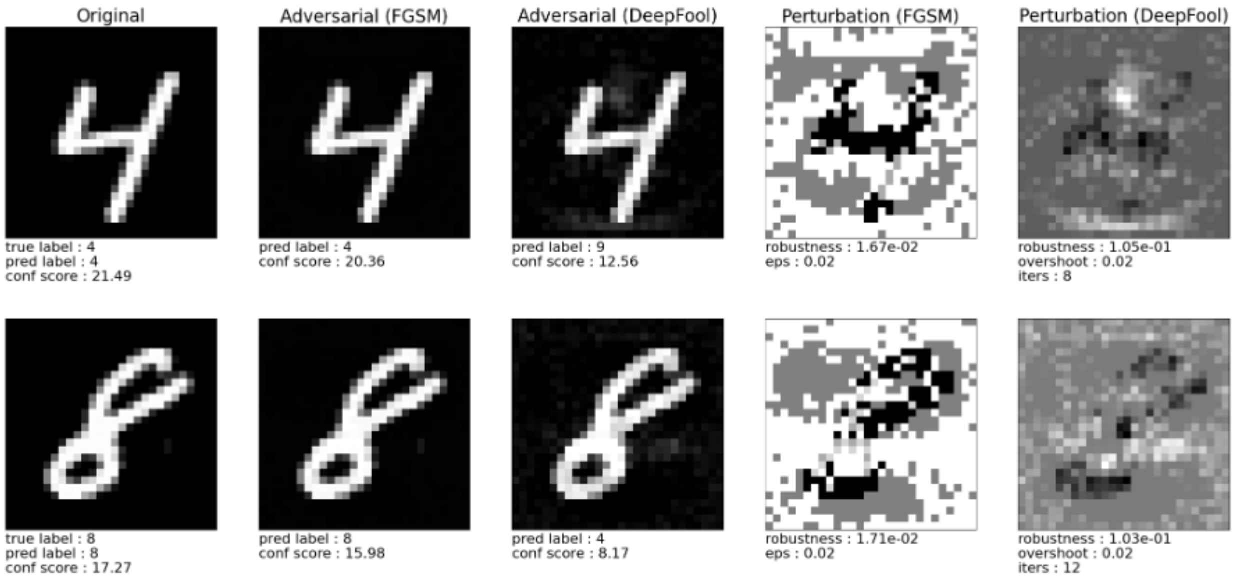
FCNet на MNIST, eps: 0,001

```
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, l

  if device.type == 'cuda': torch.cuda.empty_cache()
  print("eps: ", fgsm_eps)
```



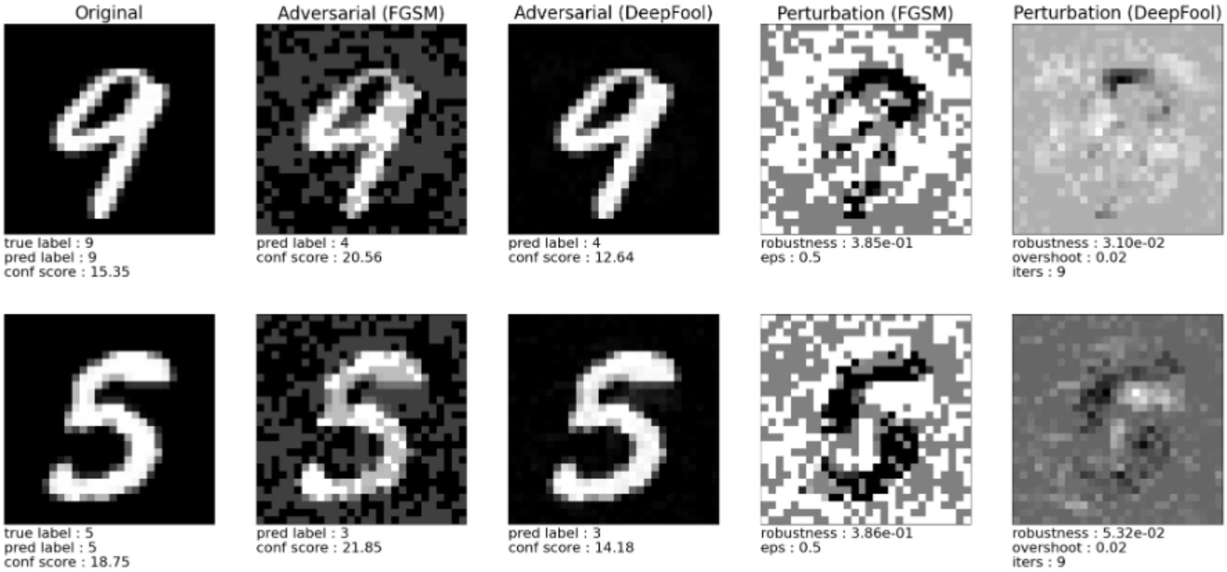

```
eps:  0.001
```

FCNet на MNIST, eps: 0,02



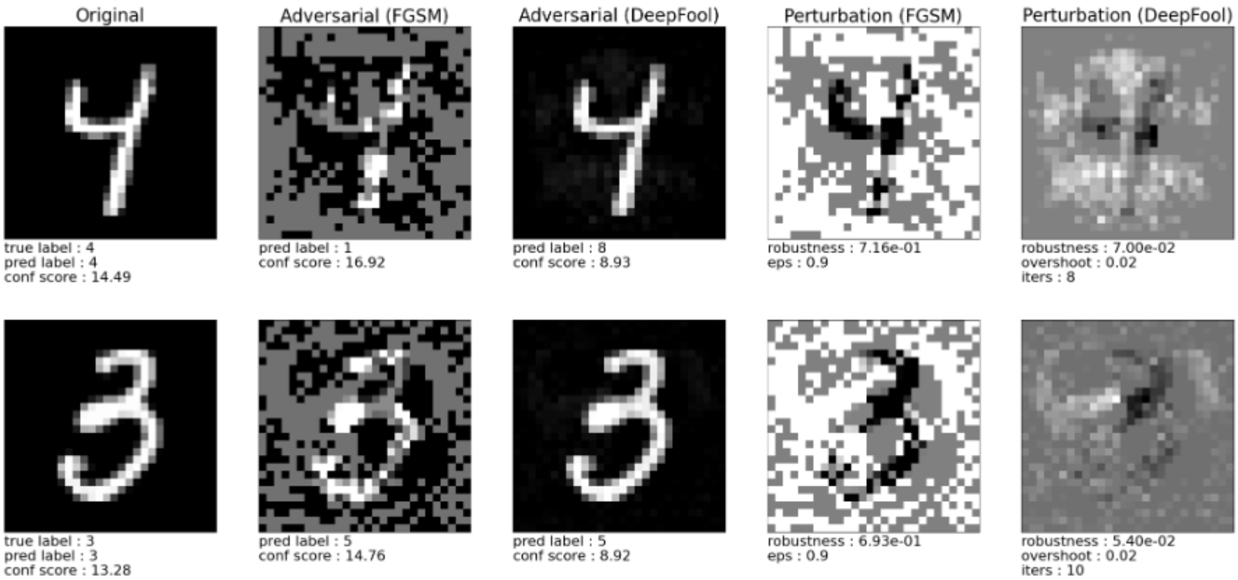

```
eps:  0.02
```

FCNet на MNIST, eps: 0,5



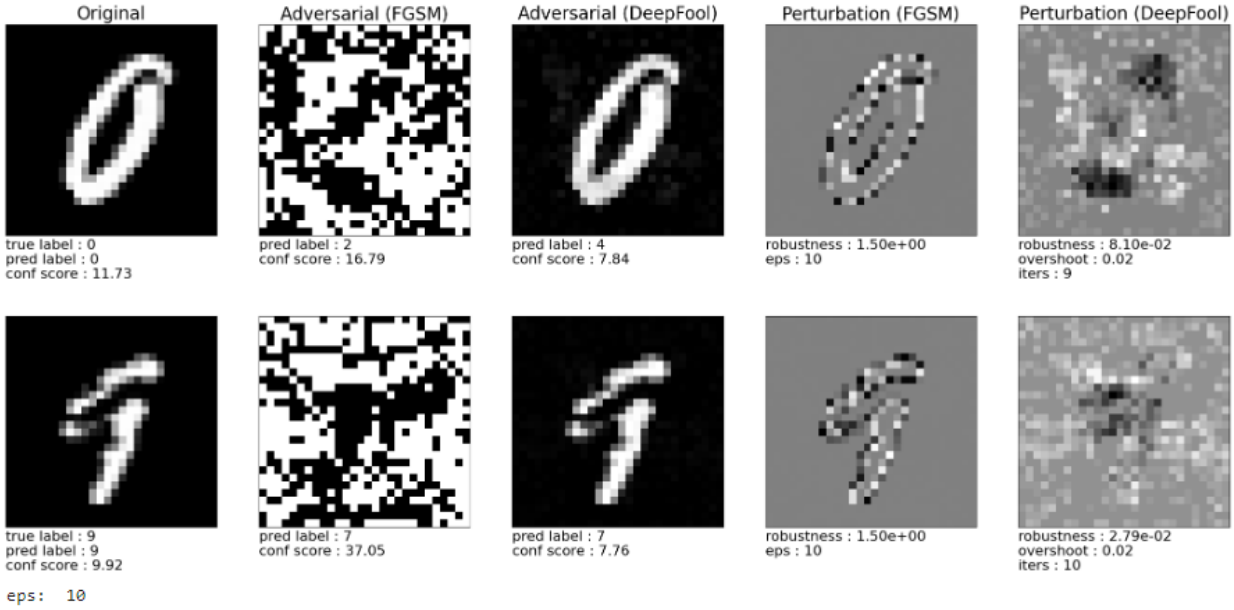

```
eps:  0.5
```

FCNet на MNIST, eps: 0,9




```
eps:  0.9
```

FCNet на MNIST, eps: 10



Вывод.

Наблюдая результат эксперимента, можно сделать вывод, что при увеличении параметра fgsm_eps увеличивается количество шума на изображениях. Это говорит о том, что модель становиться более подвержена ошибкам во время работы, а ее степень устойчивости к атакам меньше, чем если бы был более низкий показатель fgsm_eps.