



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»

Выполнил: Козлов Ф.С.

Проверил: Спирин А.А.

Москва 2024

Скопируем проект в локальную среду выполнения.

```
[ ] !git clone https://github.com/ewatson2/EEL6812_DeepFool_Project.git  
fatal: destination path 'EEL6812_DeepFool_Project' already exists and is not an empty directory.
```

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

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

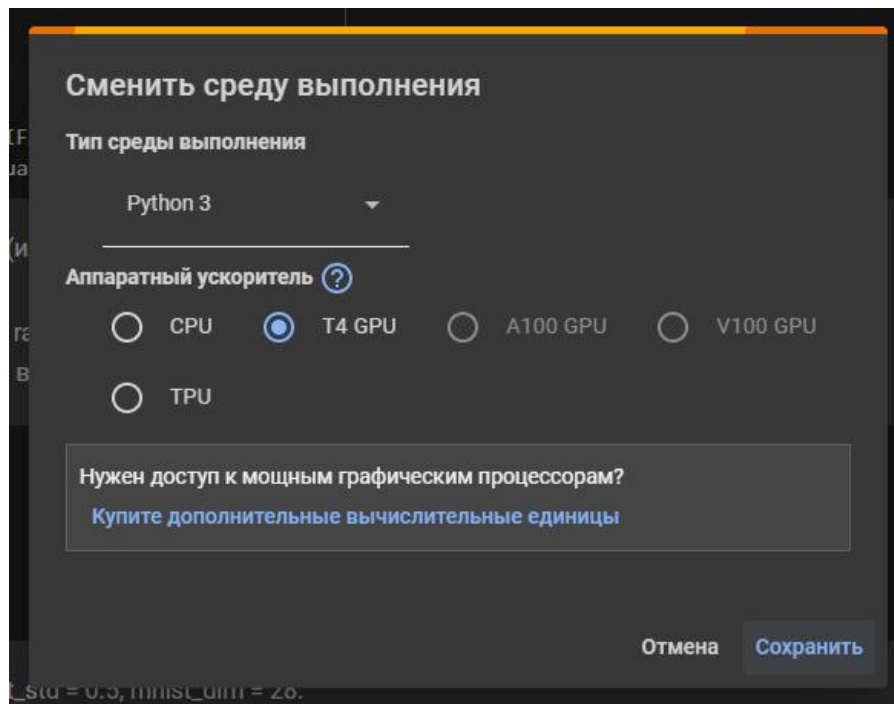
Выполним импорт стандартных и вспомогательных библиотек.

```
[ ] import numpy as np  
import os  
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
```

Установим случайное значение в виде переменной `rand_seed={11}`. Установим указанное значение для `np.random.seed` и `torch.manual_seed`.

```
[ ] rand_seed = 11  
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')
```

В работе используем в качестве устройства видеокарту (T4 GPU).



Загрузим датасет MNIST с параметрами $\text{mnist_mean} = 0.5$, $\text{mnist_std} = 0.5$, $\text{mnist_dim} = 28$.

```
[ ] 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( 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)
```

Загрузим датасет CIFAR-10 с параметрами $\text{cifar_mean} = [0.491, 0.482, 0.447]$, $\text{cifar_std} = [0.202, 0.199, 0.201]$, $\text{cifar_dim} = 32$.

```
[ ] 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']

Files already downloaded and verified
Files already downloaded and verified
```

Выполним настройку и загрузку DataLoader с параметрами `batch_size = 64` `workers = 4`.

```
#DataLoader
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)
```

Произведем обучение параметров.

```
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)
```

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('')
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
```

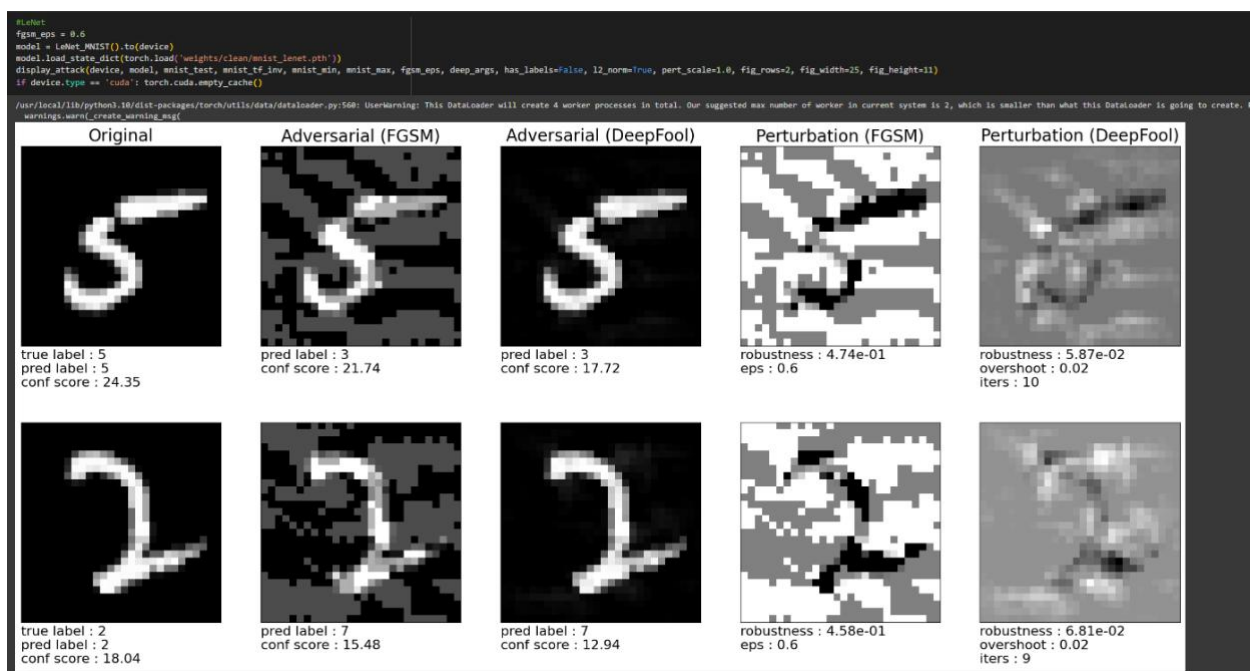
Загрузим и оценим стойкость модели LeNet к FGSM и DeepFool атакам на основе датасета CIFAR-10.

```
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()

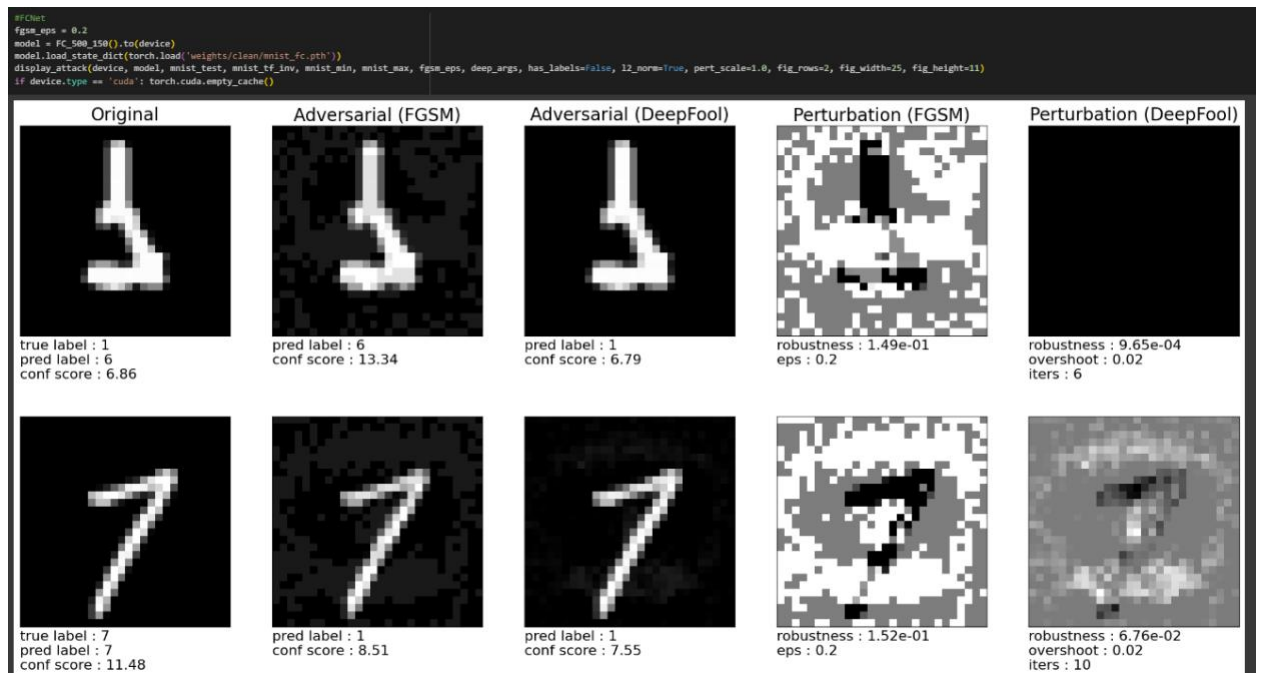
FGSM Test Error : 91.71%
FGSM Robustness : 8.90e-02
FGSM Time (All Images) : 0.40 s
FGSM Time (Per Image) : 40.08 us

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

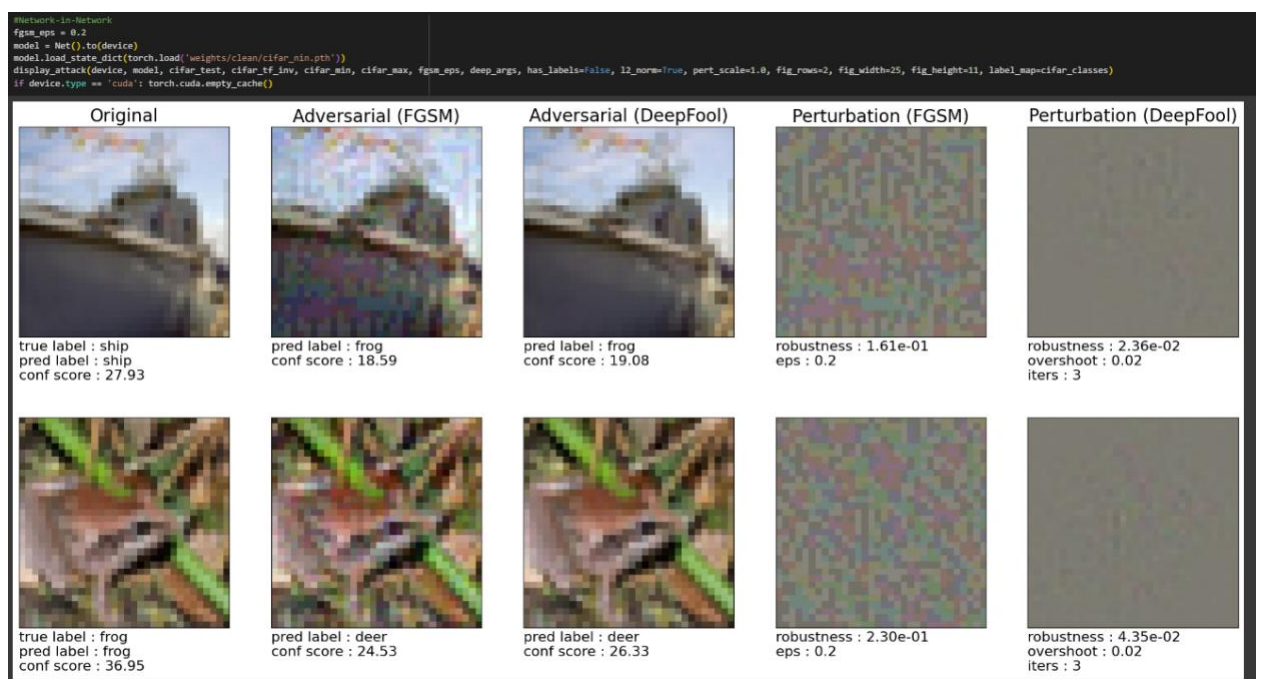
Выполним оценку атакующих примеров для сетей (LeNet).



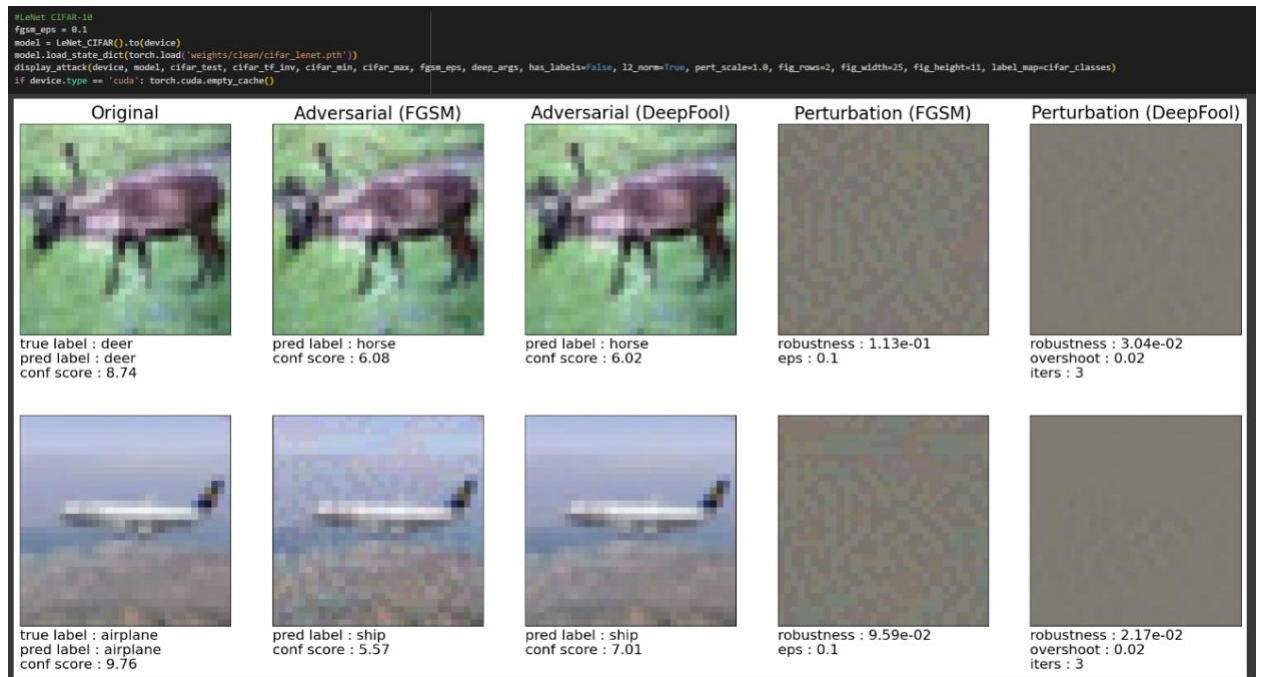
Выполним оценку атакующих примеров для сетей (FCNet).



Выполним оценку атакующих примеров для сетей (Network-in-Network).

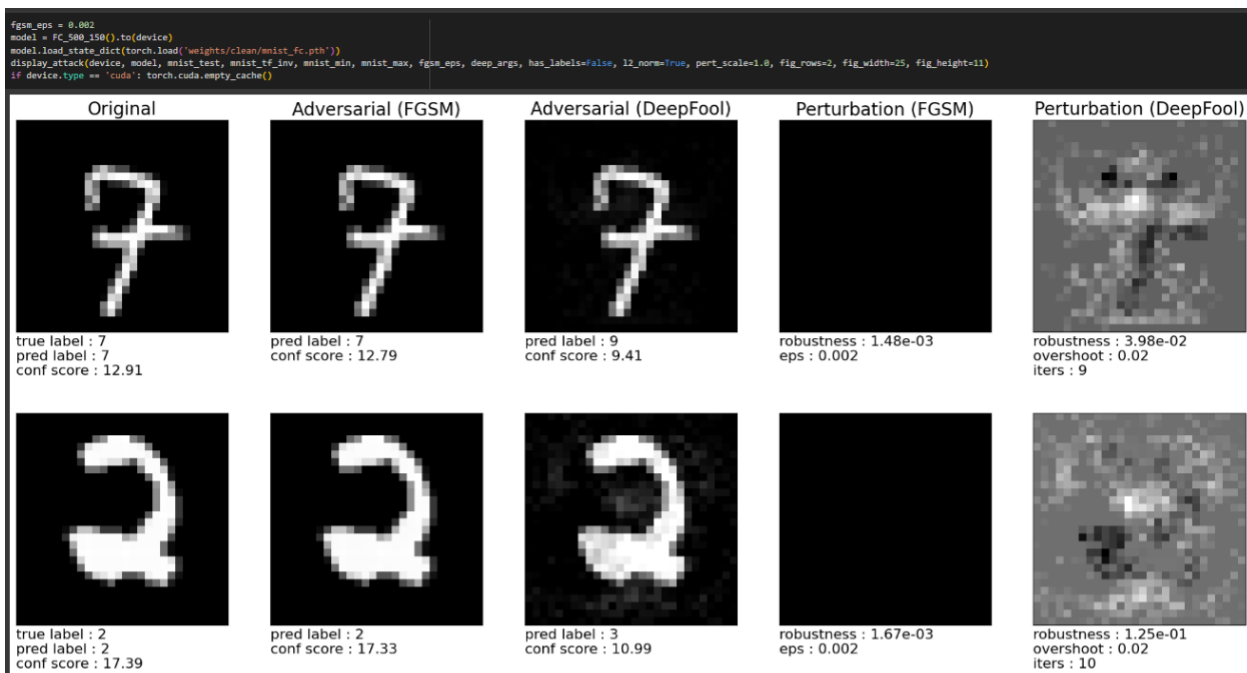


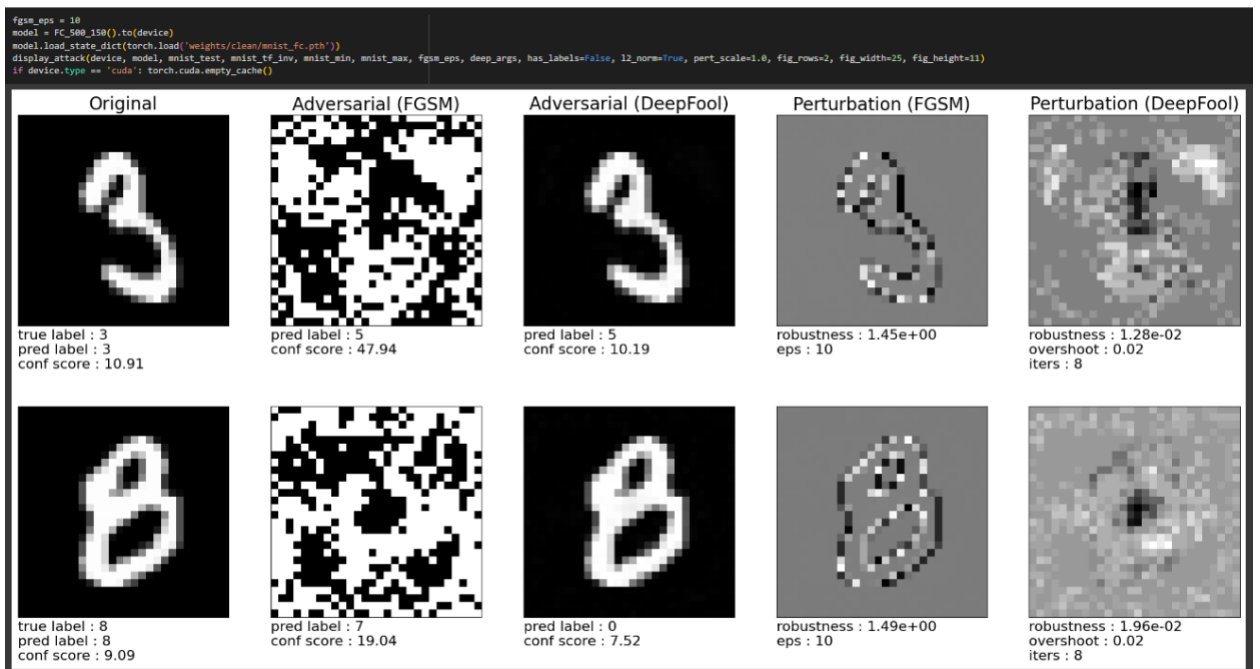
Выполним оценку атакующих примеров для сетей (LeNet CIFAR-10).



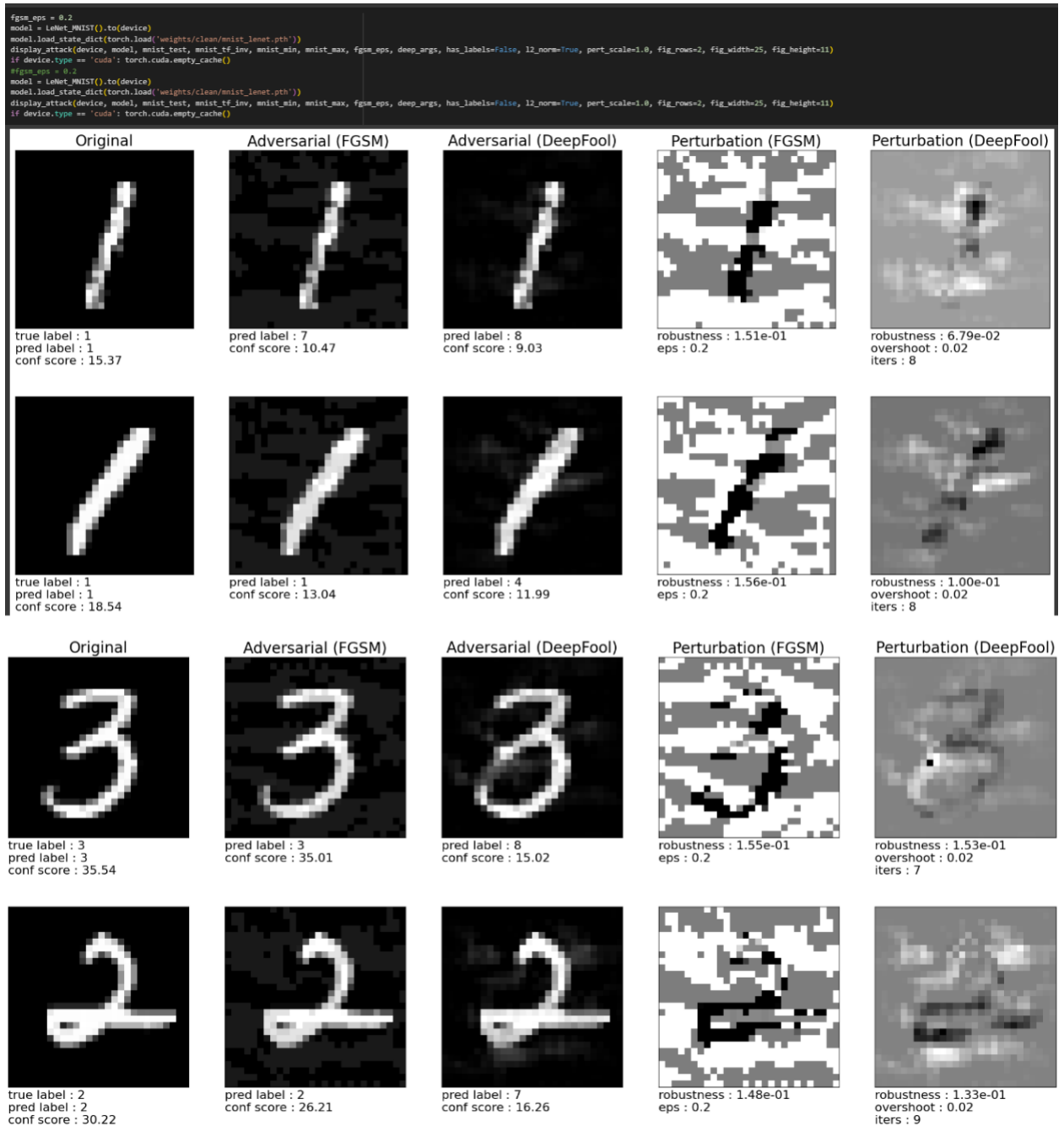
Отражаем отличия для fgsm_eps=(0.001, 0.02, 0.5, 0.9, 10).





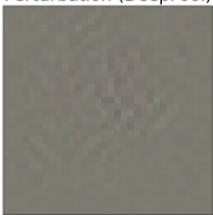



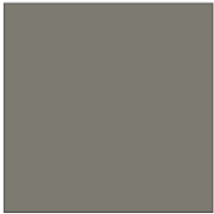
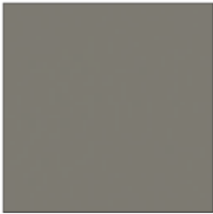



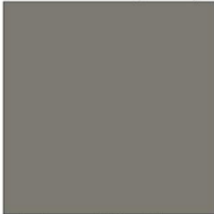












Выявим закономерность/обнаружим отсутствие влияния параметра eps для сетей FC LeNet на датасете MNIST, NiN LeNet на датасете CIFAR.



<pre>fgsm_eps = 0.01 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 = 0.01 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()</pre>				
Original	Adversarial (FGSM)	Adversarial (DeepFool)	Perturbation (FGSM)	Perturbation (DeepFool)
				
true label : frog pred label : frog conf score : 14.23	pred label : frog conf score : 14.06	pred label : cat conf score : 7.32	robustness : 1.13e-03 eps : 0.001	robustness : 4.89e-02 overshoot : 0.02 iters : 4
				
true label : truck pred label : automobile conf score : 11.63	pred label : automobile conf score : 11.76	pred label : truck conf score : 11.30	robustness : 8.44e-04 eps : 0.001	robustness : 1.59e-03 overshoot : 0.02 iters : 1
Original	Adversarial (FGSM)	Adversarial (DeepFool)	Perturbation (FGSM)	Perturbation (DeepFool)
				
true label : ship pred label : ship conf score : 9.22	pred label : ship conf score : 9.08	pred label : frog conf score : 4.53	robustness : 7.52e-04 eps : 0.001	robustness : 2.47e-02 overshoot : 0.02 iters : 2
				
true label : ship pred label : ship conf score : 17.58	pred label : ship conf score : 17.46	pred label : airplane conf score : 7.55	robustness : 6.11e-04 eps : 0.001	robustness : 5.01e-02 overshoot : 0.02 iters : 5

```
fgsm_eps = 0.01
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, label_mapcifar_classes)
evaluate_attack('mnist_fc_fgsm.csv', 'results', device, model, mnist_loader_test, mnist_min, mnist_max, fgsm_eps, is_fgsm=True)
if device.type == 'cuda': torch.cuda.empty_cache()

fgsm_eps = 0.01
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, label_mapcifar_classes)
evaluate_attack('mnist_fc_fgsm.csv', 'results', device, model, mnist_loader_test, mnist_min, mnist_max, fgsm_eps, is_fgsm=True)
if device.type == 'cuda': torch.cuda.empty_cache()
```



true label : truck
pred label : truck
conf score : 20.88



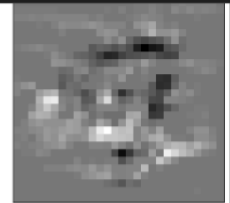
pred label : truck
conf score : 20.16



pred label : ship
conf score : 15.47



robustness : 8.26e-03
eps : 0.01



robustness : 5.09e-02
overshoot : 0.02
iters : 9



true label : bird
pred label : bird
conf score : 20.10



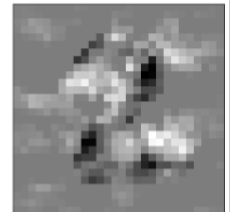
pred label : bird
conf score : 19.60



pred label : ship
conf score : 12.68



robustness : 8.22e-03
eps : 0.01



robustness : 9.82e-02
overshoot : 0.02
iters : 8

FGSM Test Error : 87.80%

FGSM Robustness : 4.58e-01

FGSM Time (All Images) : 0.29 s

FGSM Time (Per Image) : 38.66 us

```
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, label_mapcifar_classes)
evaluate_attack('mnist_lenet_fgsm.csv', 'results', device, model, mnist_loader_test, mnist_min, mnist_max, fgsm_eps, is_fgsm=True)
if device.type == 'cuda': torch.cuda.empty_cache()

fgsm_eps = 0.01
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, label_mapcifar_classes)
evaluate_attack('mnist_fc_fgsm.csv', 'results', device, model, mnist_loader_test, mnist_min, mnist_max, fgsm_eps, is_fgsm=True)
if device.type == 'cuda': torch.cuda.empty_cache()
```



true label : deer
pred label : deer
conf score : 20.87



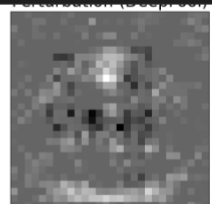
pred label : deer
conf score : 20.31



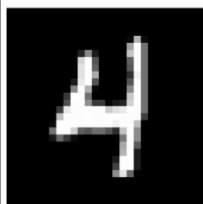
pred label : truck
conf score : 12.85



robustness : 8.40e-03
eps : 0.01



robustness : 9.62e-02
overshoot : 0.02
iters : 9



true label : deer
pred label : deer
conf score : 18.57



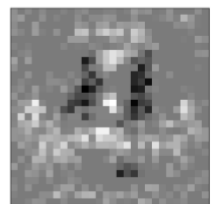
pred label : deer
conf score : 17.98



pred label : bird
conf score : 9.89



robustness : 8.45e-03
eps : 0.01



robustness : 9.64e-02
overshoot : 0.02
iters : 10

FGSM Test Error : 87.80%

FGSM Robustness : 1.56e-01

FGSM Time (All Images) : 0.15 s

FGSM Time (Per Image) : 14.99 us

```

# NiN
fgsm_eps = 0.1
model = Net().to(device)
model.load_state_dict(torch.load('weights/clean/cifar_nin.pth', map_location=torch.device('cpu')))
evaluate_attack('cifar_lenet_nin.csv', 'results', device, model, cifar_loader_test, cifar_min, cifar_max, fgsm_eps, is_fgsm=True)
if device.type == 'cuda': torch.cuda.empty_cache()

print('')

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

FGSM Test Error : 9.28%
FGSM Robustness : 8.93e-06
FGSM Time (All Images) : 1.41 s
FGSM Time (Per Image) : 141.14 us

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

```

Вывод: Параметр eps оказывает влияние на устойчивость сети. При этом, чем больше данный параметр, тем больше доля ошибки классификации и сети более уязвимы к атакам. Чем меньше значение eps, тем более устойчивые к атакам становятся сети.