МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российскийтехнологическийуниверситет»
РТУ МИРЭА

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

Кафедра КБ-4 «Интеллектуальные системы информационной безопасности»

Лабораторная работа №1

по дисципилне

«Анализ защищенности систем искусственного интеллекта»

Группа:
ББМО-02-22
Выполнила:
Волкова Е.А.

Проверил:
Спирин А.А.

Москва, 2023

Скопируем проект по ссылке в локальную среду выполнения Google Colab.

```
[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 | 24.77 MiB/s, done.
Resolving deltas: 100% (27/27), done.
```

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

```
[2]  %cd 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
     import os
```

Выполним импорт вспомогательных библиотек из локальных файлов проекта.

```
[4]  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
```

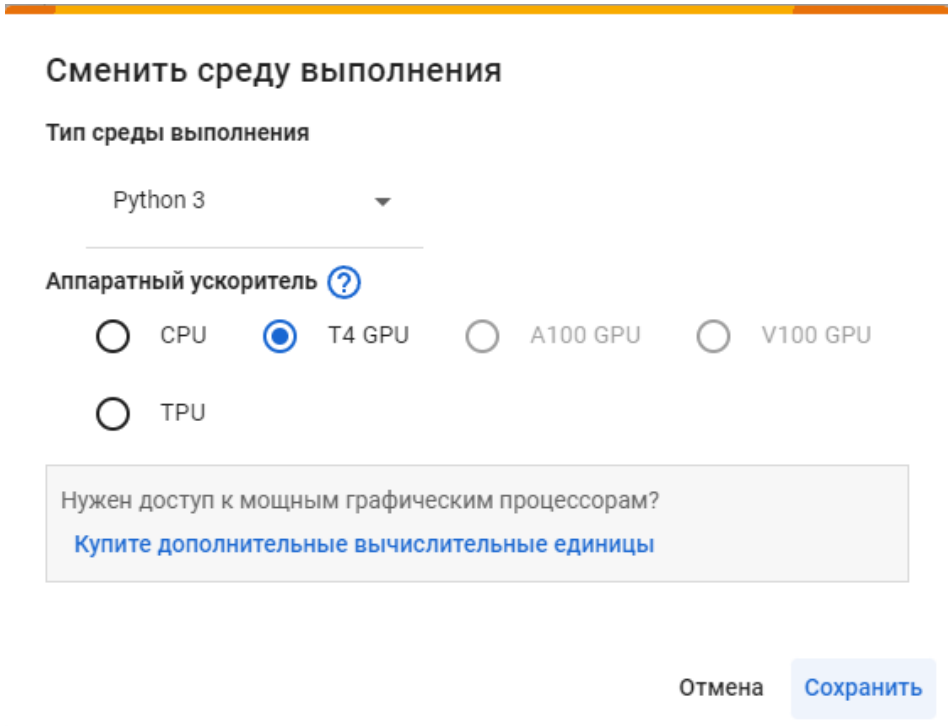
Установим случайное рандомное значение в виде переменной rand_seed={"Порядковый номер ученика группы в Гугл-таблице"}.

```
[5]  rand_seed = 33 # Порядковый номер
```

Установим указанное значение для np.random.seed и torch.manual_seed.

```
[6]  np.random.seed(rand_seed)
     torch.manual_seed(rand_seed)
```

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



Загрузим датасет MNIST с параметрами mnist_mean = 0.5, mnist_std = 0.5, mnist_dim = 28.

```
[8]  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, 142119748.56it/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, 26163216.81it/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, 37963734.06it/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, 5755446.76it/s]
Extracting datasets/mnist/MNIST/raw/t10k-labels-idx1-ubyte.gz to datasets/mnist/MNIST/raw
```

Загрузим датасет CIFAR-10.

```
[9] 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 [00:03<00:00, 43472285.06it/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.

```
[10] 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
  warnings.warn(_create_warning_msg(
```

Установим параметры для обучения.

```
[11] 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()
```

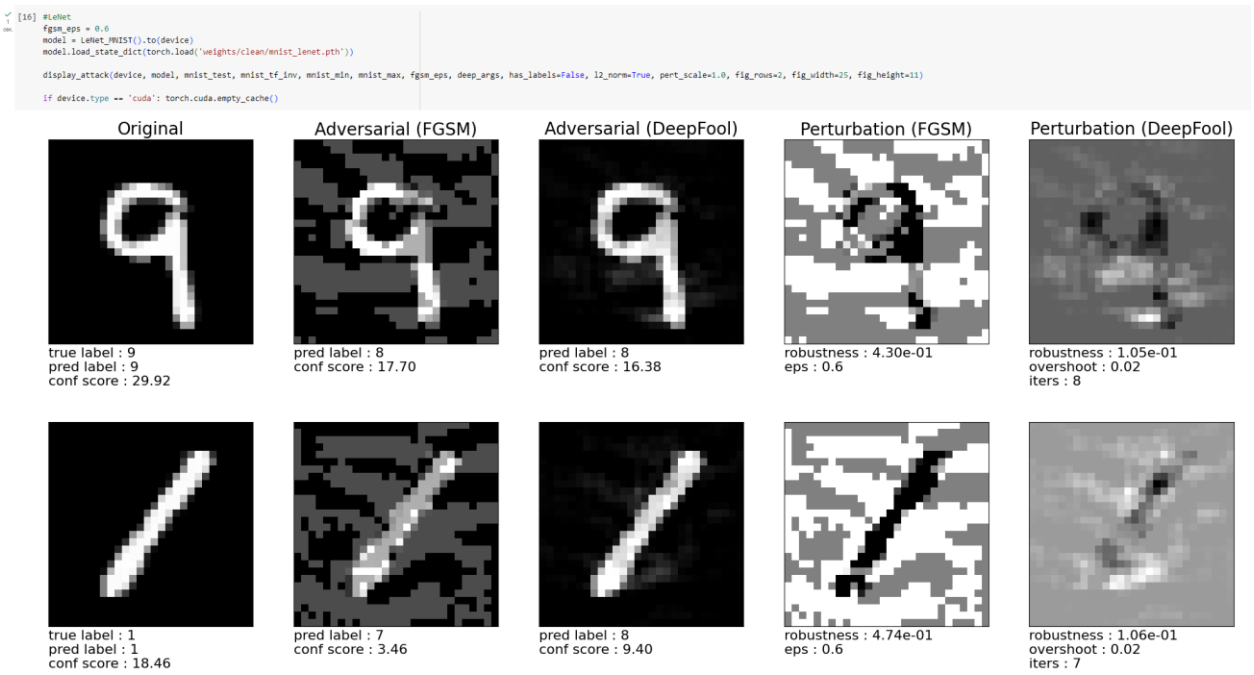
```
FGSM Test Error : 81.29%
FGSM Robustness : 1.77e-01
FGSM Time (All Images) : 0.67 s
FGSM Time (Per Image) : 67.07 us

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.

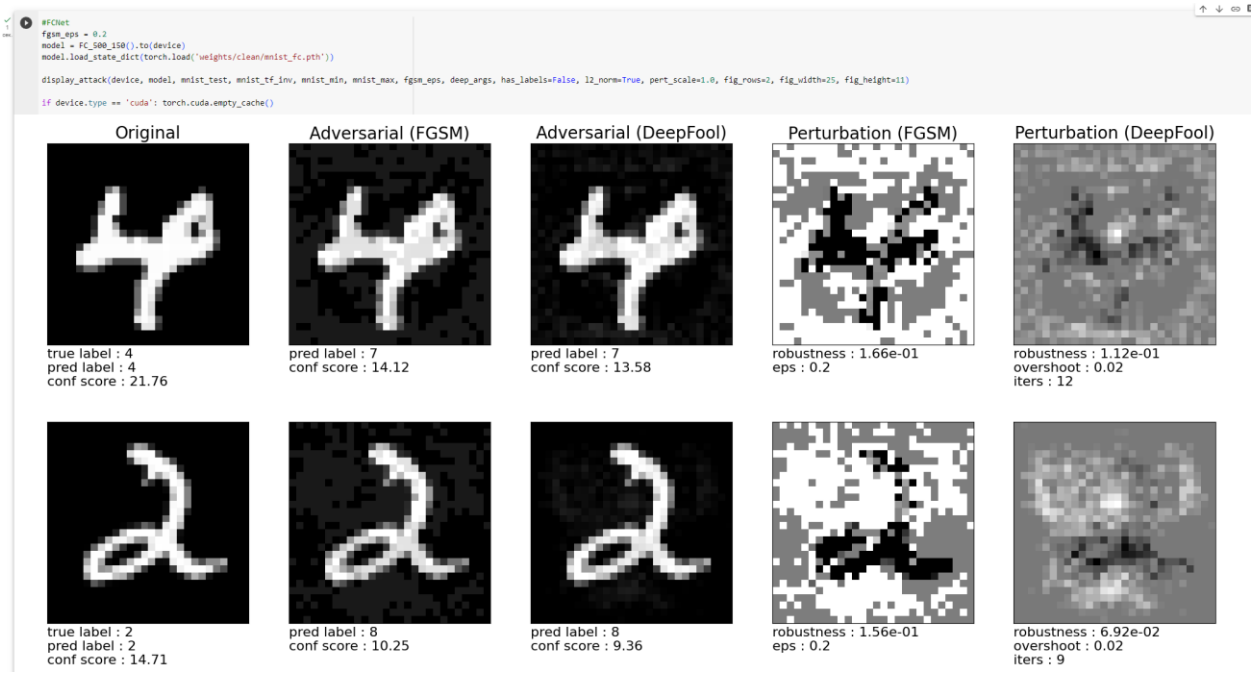
```
[13] fgsm_eps = 0.1
model = LeNet_CIFAR().to(device)
model.load_state_dict(torch.load('weights/clean/cifar_lenet.pth', map_location=torch.device('cpu')))

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).

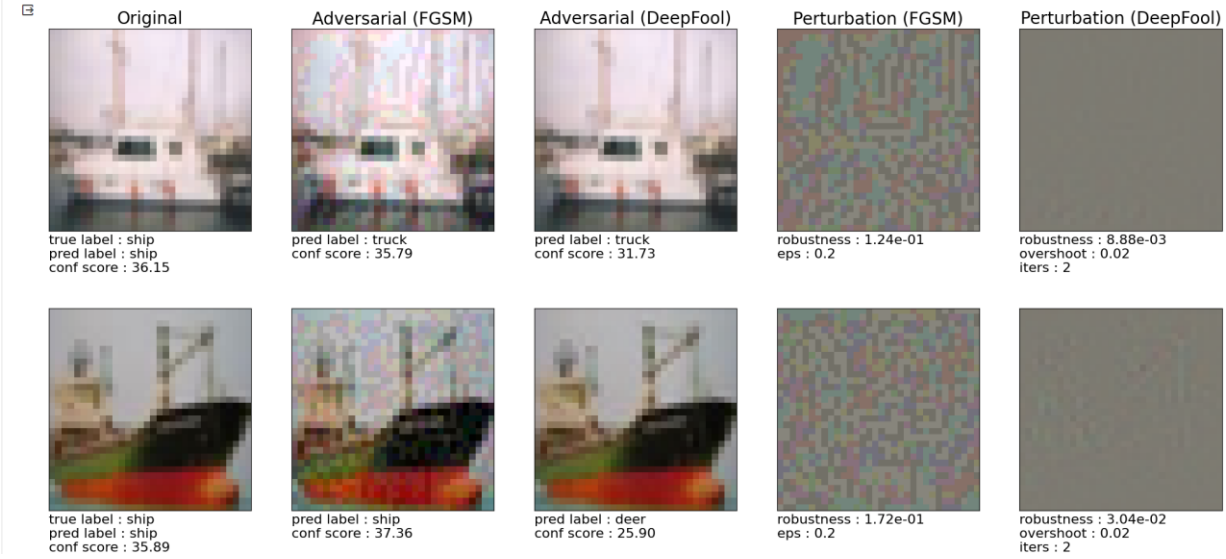


Выполним оценку атакующих примеров для сетей (FCNet).



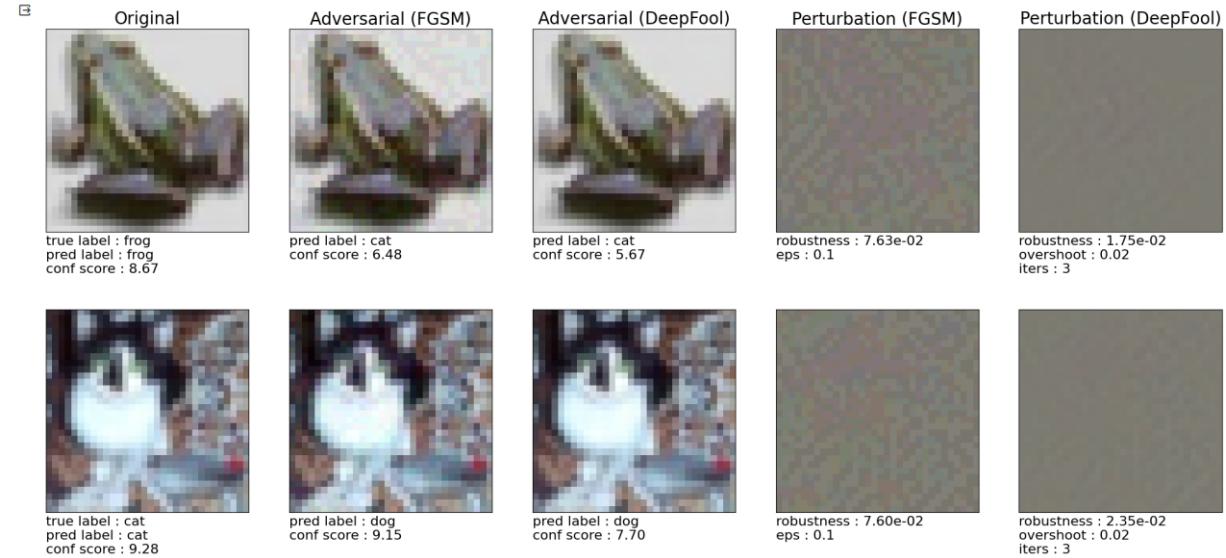
Выполним оценку атакующих примеров для сетей (Network-in-Network).

```
#Network-in-Network
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()
```



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

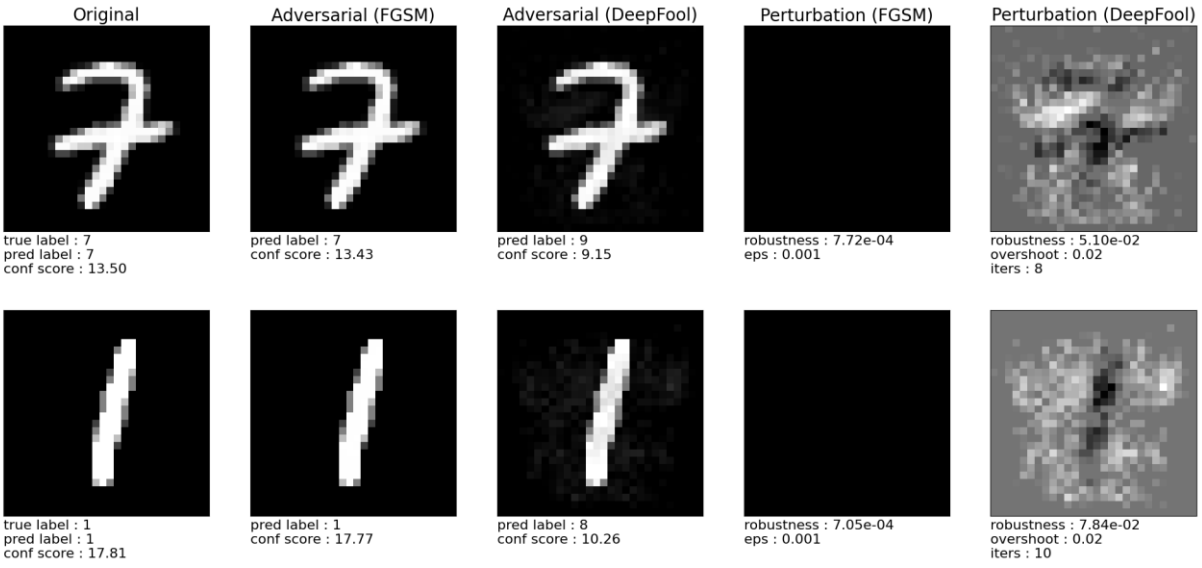
```
#LeNet CIFAR-10
fgsm_eps = 0.1
model = LeNet_CIFAR().to(device)
model.load_state_dict(torch.load('weights/clean/cifar_lenet.pth'))

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()
```



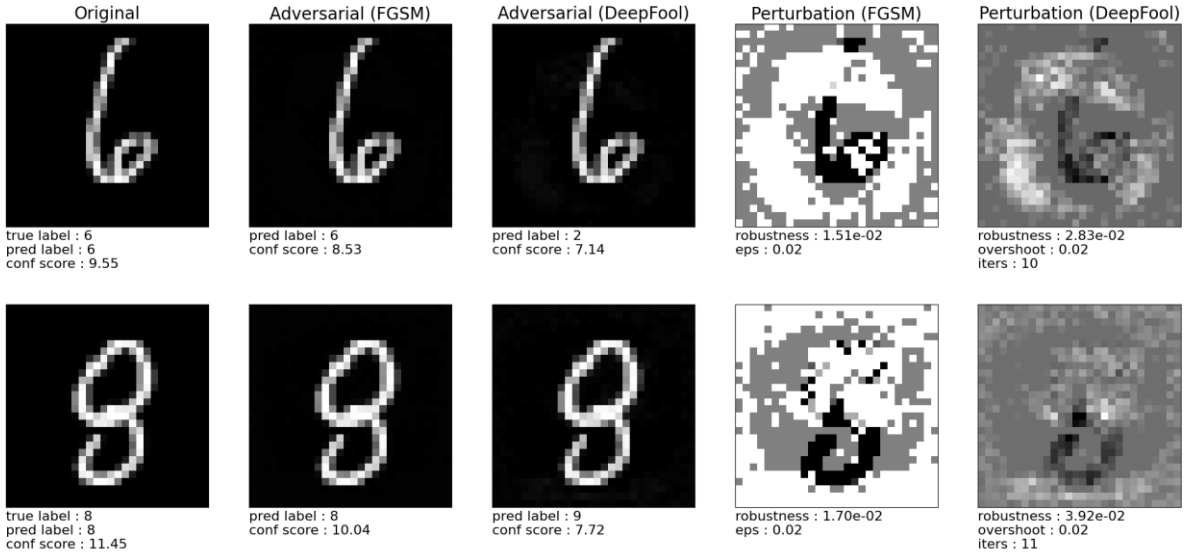
Отразим отличия для fgsm_eps=(0.001, 0.02, 0.5, 0.9, 10).

Проверим влияние параметра для FC на датасете MNIST.

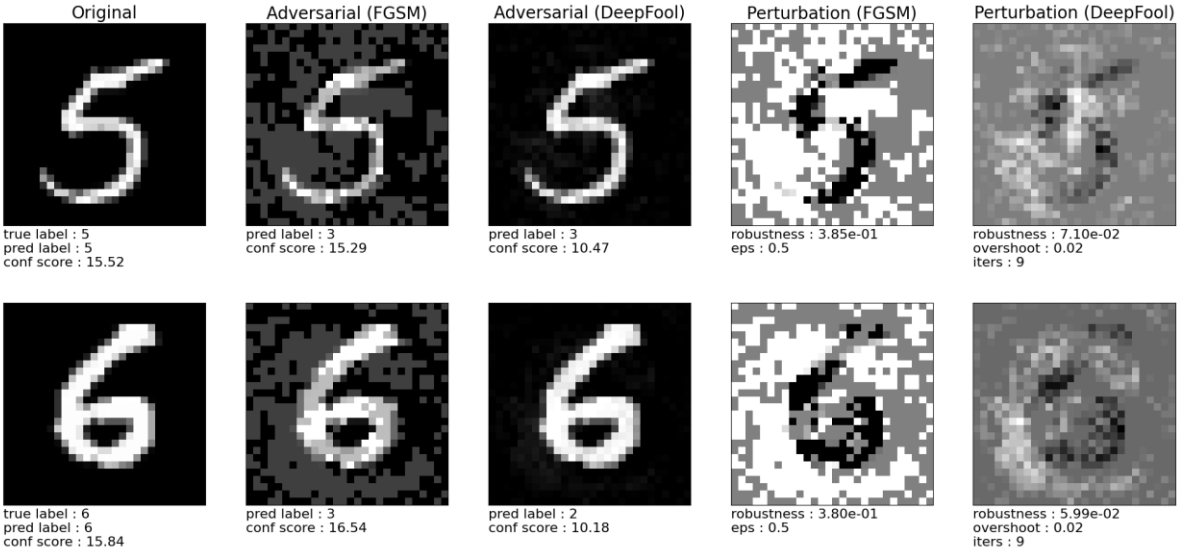
```
fgsm_eps = 0.001
model = FC_500_150().to(device)
model.load_state_dict(torch.load('weights/clean/mnist_fc.pth'))

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()
```



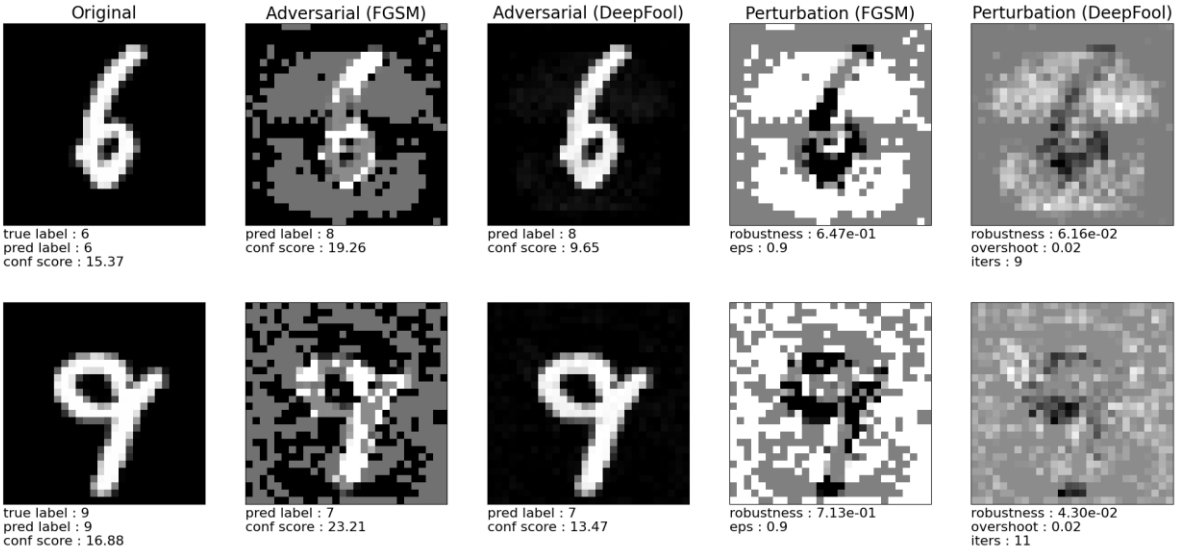

```
[22] fgsm_eps = 0.02
     model = FC_500_150().to(device)
     model.load_state_dict(torch.load('weights/clean/mnist_fc.pth'))

     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()
```



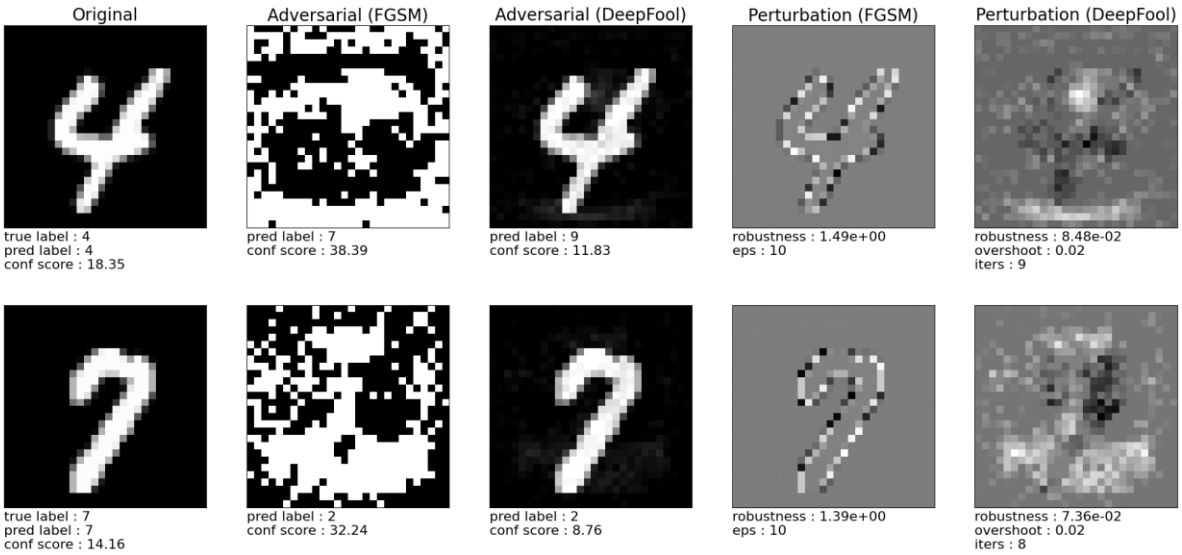
```
[23] fgsm_eps = 0.5
     model = FC_500_150().to(device)
     model.load_state_dict(torch.load('weights/clean/mnist_fc.pth'))

     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()
```




```
[24] fgsm_eps = 0.9
     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()
```



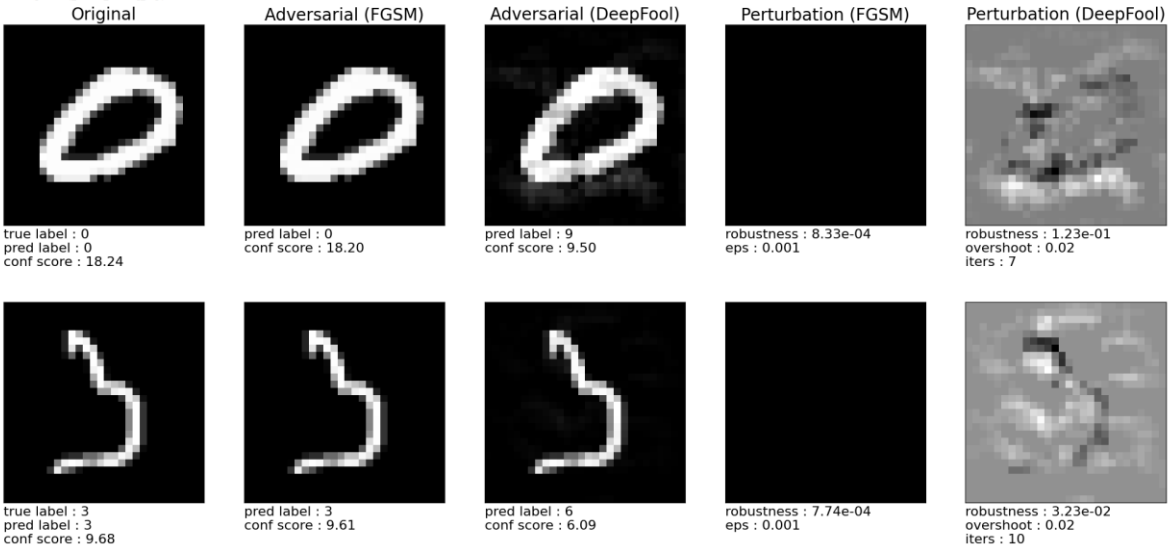
```
[25] fgsm_eps = 10
     model = FC_500_150().to(device)
     model.load_state_dict(torch.load('weights/clean/mnist_fc.pth'))

     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()
```



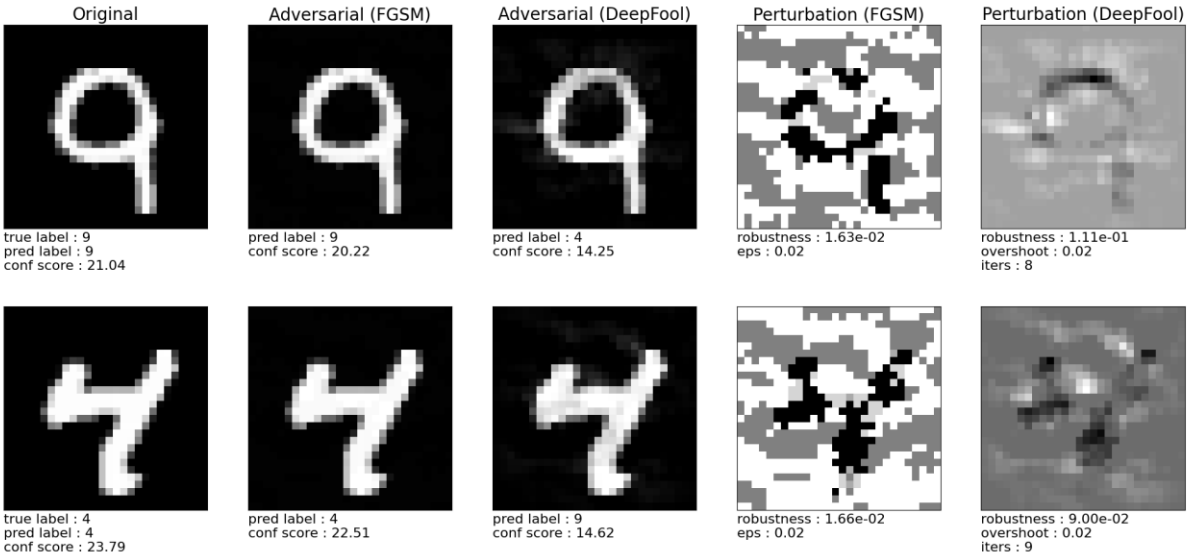
Проверим влияние параметра для LeNet на датасете MNIST.

```
fgsm_eps = 0.001
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()
```

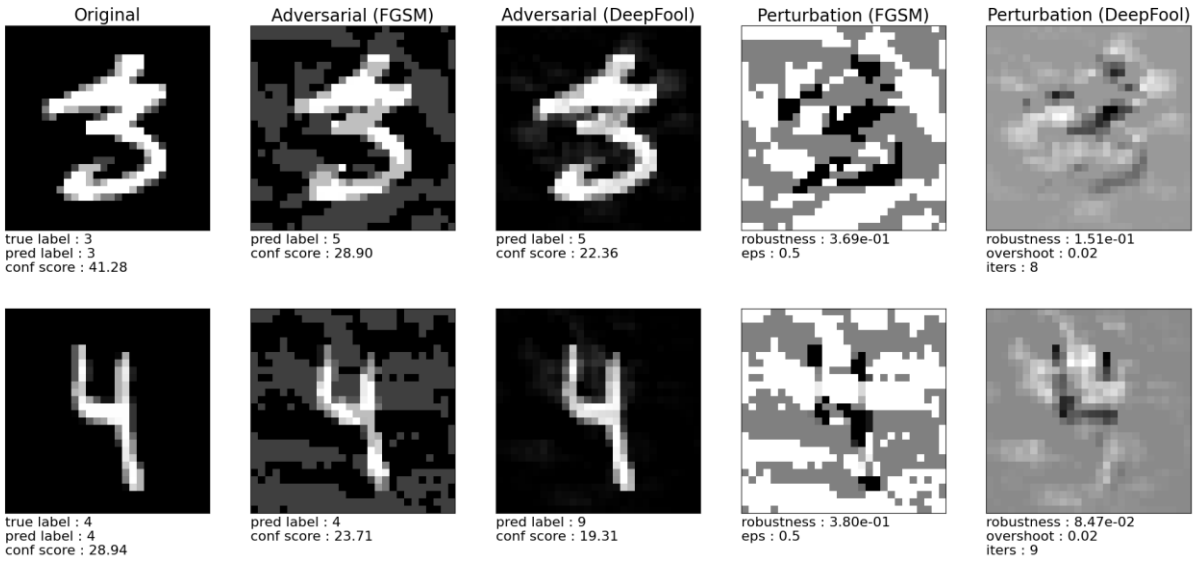
```
/usr/local/lib/python3.10/dist-packages/torch/utils/data/dataloader.py:557: UserWarning: This DataLoader will create 4 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller than what this DataLoader is going to cr
  warnings.warn(_create_warning_msg(
```



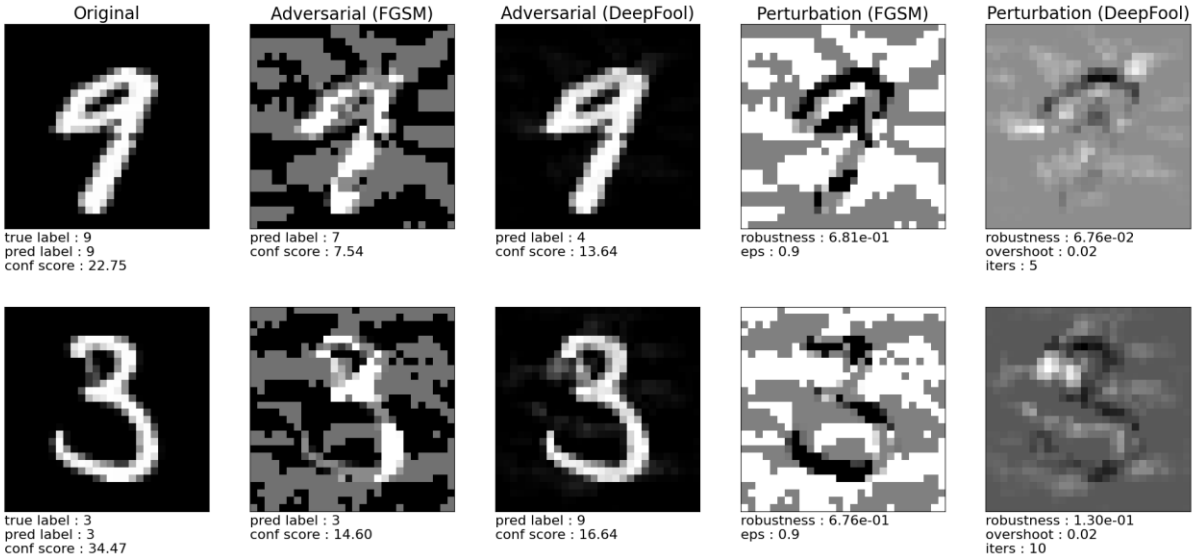
```
[27] fgsm_eps = 0.02
     model = LeNet_MNIST().to(device)
     model.load_state_dict(torch.load('weights/clean/mnist_lenet.pth'))

     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()
```



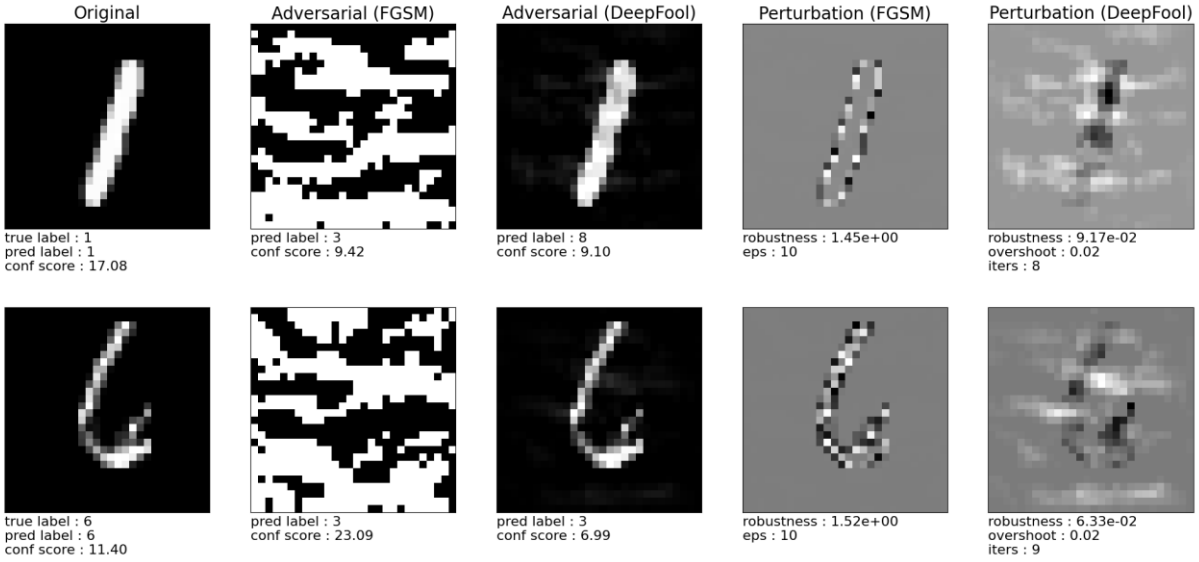

```
[28] fgsm_eps = 0.5
     model = LeNet_MNIST().to(device)
     model.load_state_dict(torch.load('weights/clean/mnist_lenet.pth'))

     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()
```



```
[29] fgsm_eps = 0.9
     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()
```




```
[30] fgsm_eps = 10
     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()
```

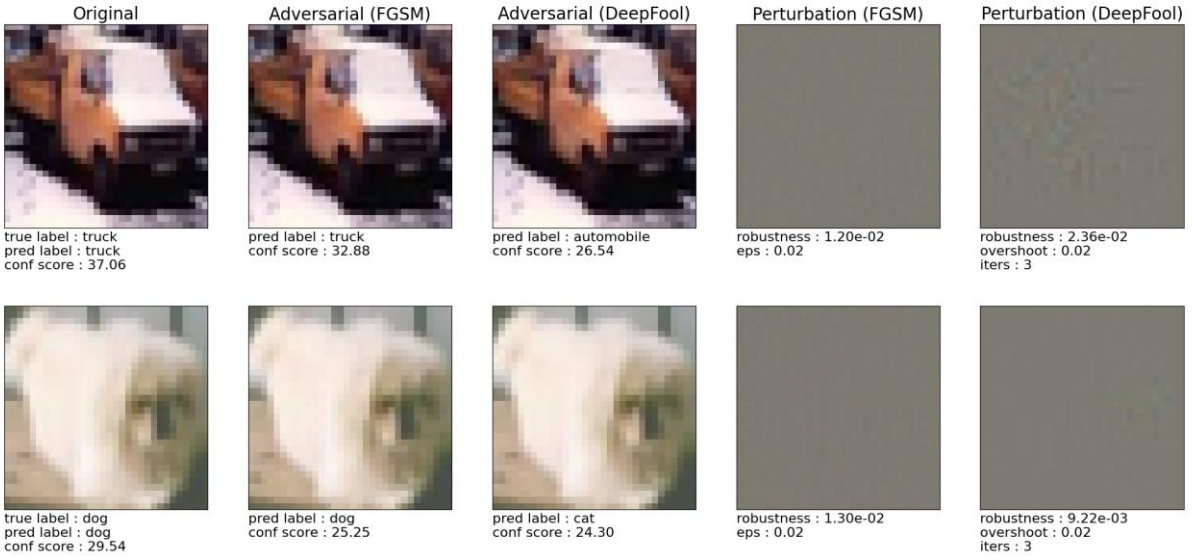


Проверим влияние параметра для NiN на датасете CIFAR.

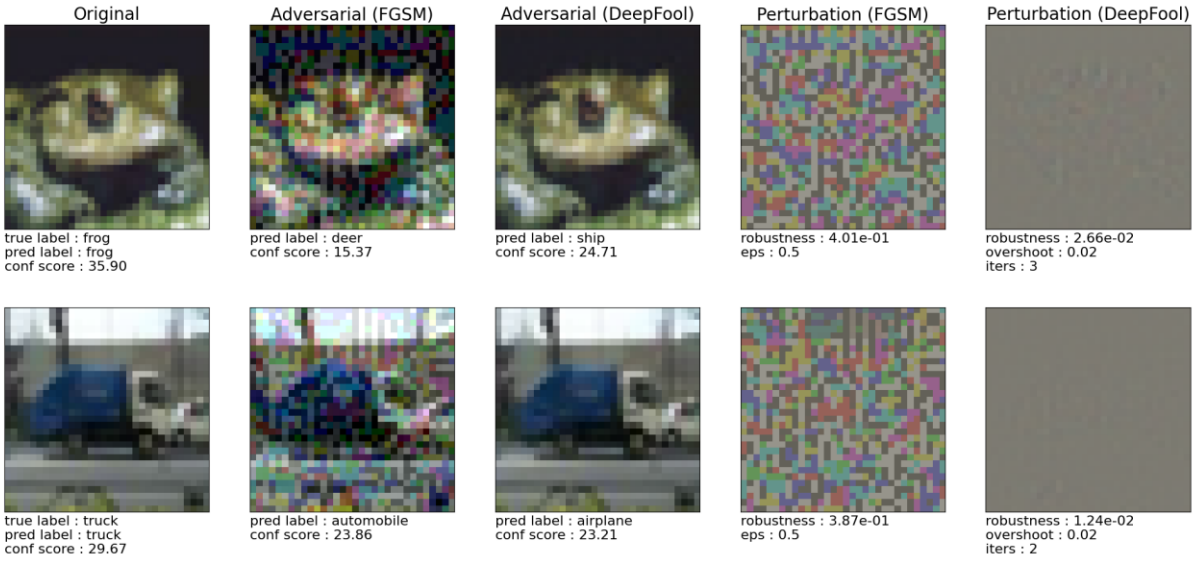
```
[35] fgsm_eps = 0.001
     model = Net().to(device)
     model.load_state_dict(torch.load('weights/clean/cifar_nin.pth'))

     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()
```

```
/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 worker in current system is 2, which is smaller than what this DataLoader is going
  warnings.warn(_create_warning_msg(
```

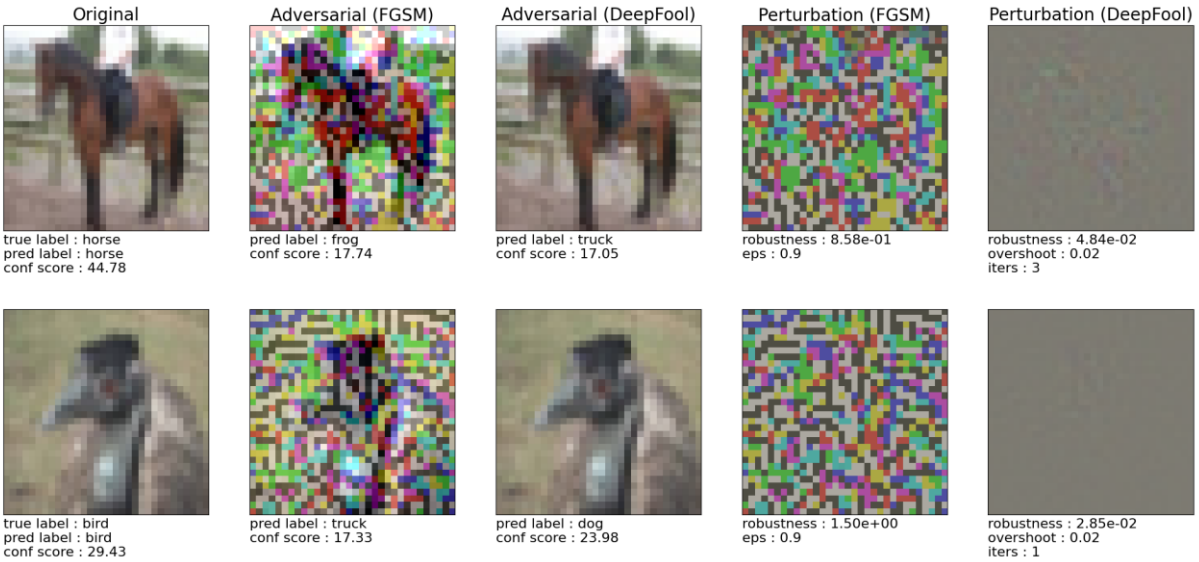

```
[36] fgsm_eps = 0.02
     model = Net().to(device)
     model.load_state_dict(torch.load('weights/clean/cifar_nin.pth'))

     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()
```



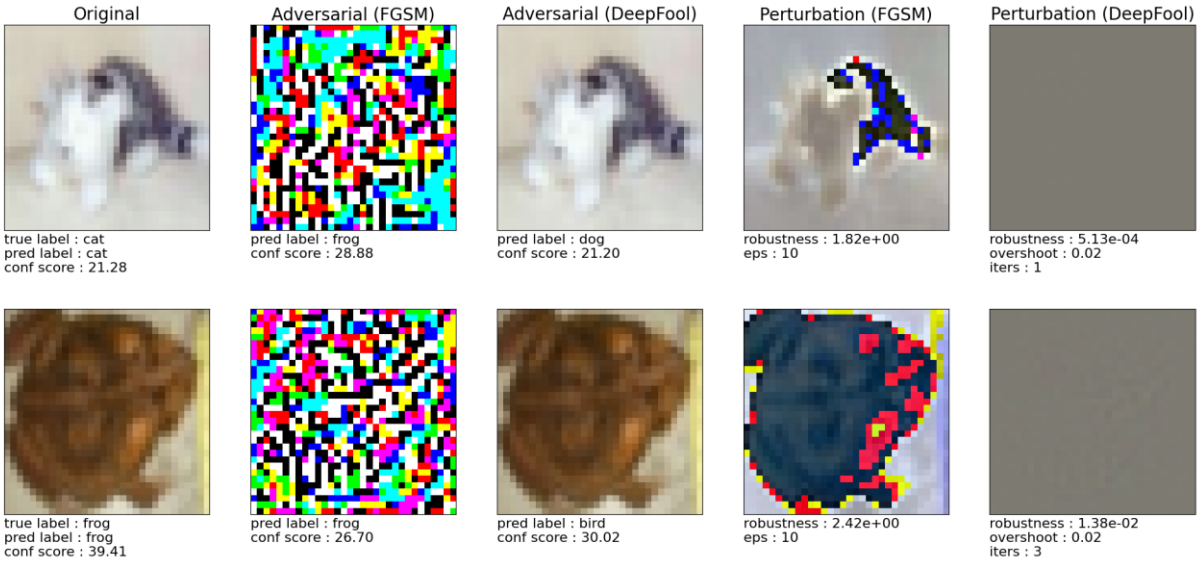
```
[37] fgsm_eps = 0.5
     model = Net().to(device)
     model.load_state_dict(torch.load('weights/clean/cifar_nin.pth'))

     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()
```




```
[38] fgsm_eps = 0.9
     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()
```

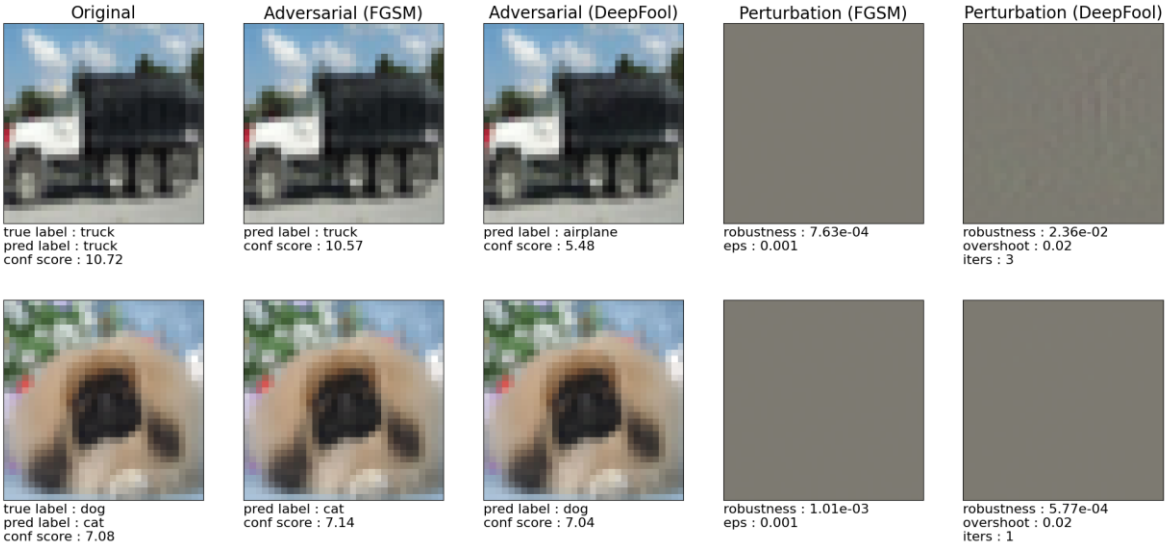


```
[39] fgsm_eps = 10
     model = Net().to(device)
     model.load_state_dict(torch.load('weights/clean/cifar_nin.pth'))

     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()
```

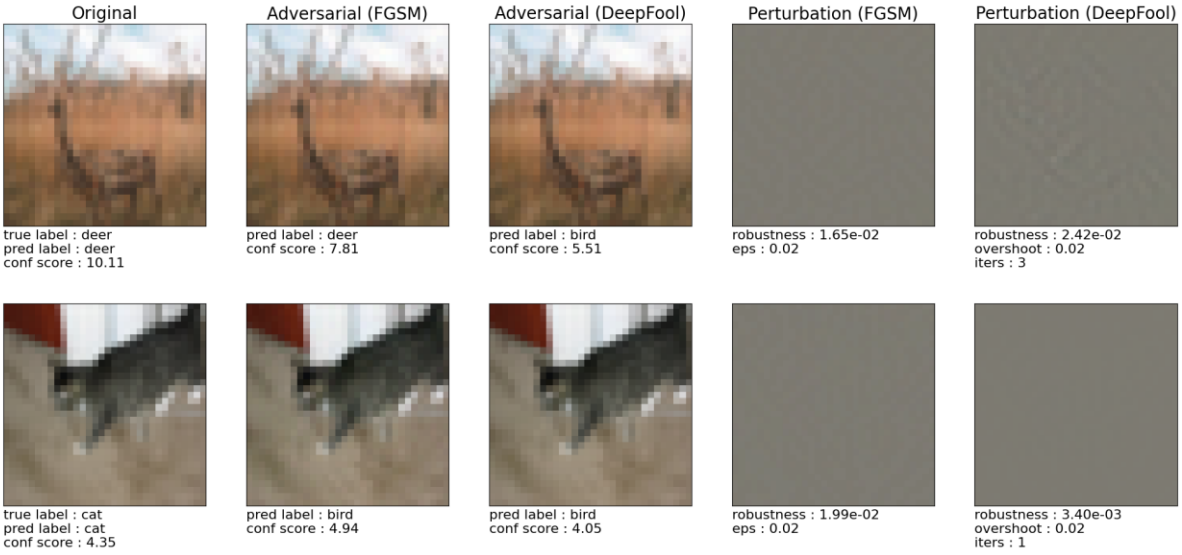


Проверим влияние параметра для LeNet на датасете CIFAR-10.

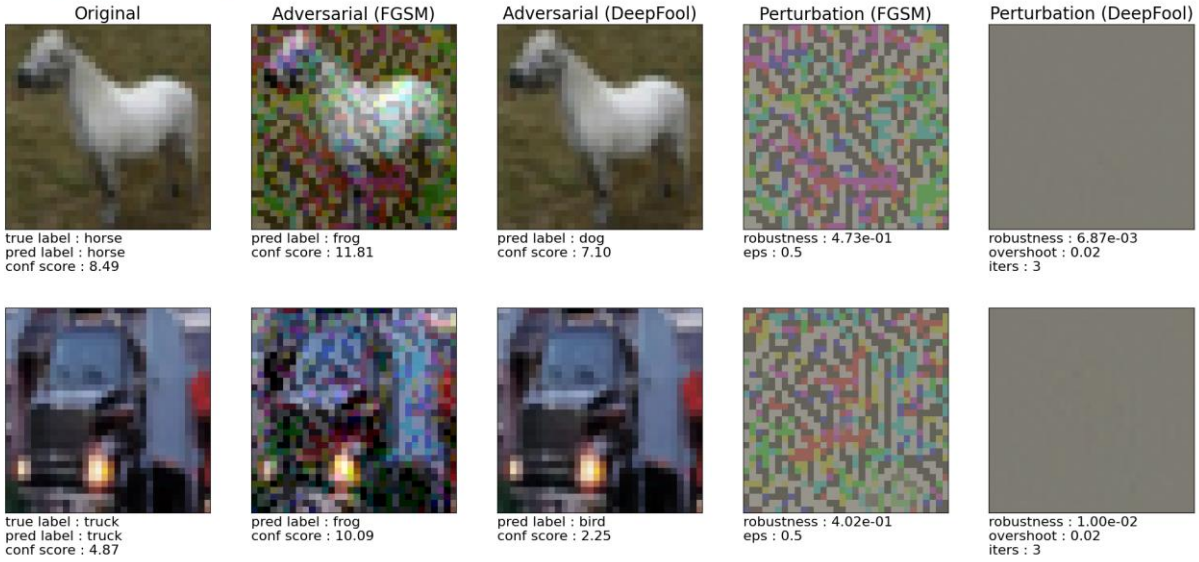
```
[40] fgsm_eps = 0.001
     model = LeNet_CIFAR().to(device)
     model.load_state_dict(torch.load('weights/clean/cifar_lenet.pth'))

     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()
```



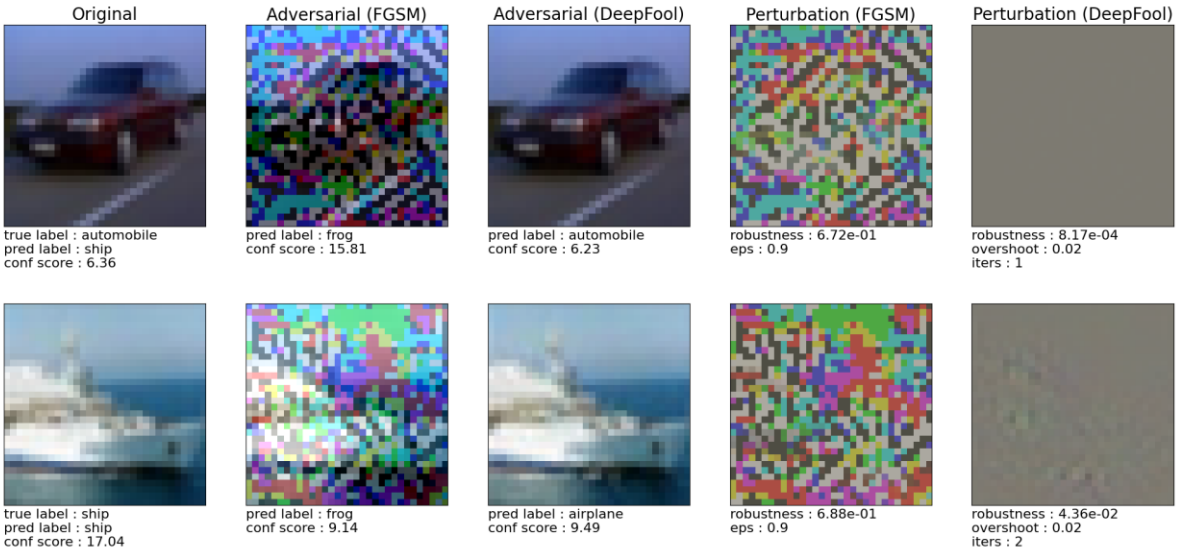
```
[41] fgsm_eps = 0.02
     model = LeNet_CIFAR().to(device)
     model.load_state_dict(torch.load('weights/clean/cifar_lenet.pth'))

     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()
```



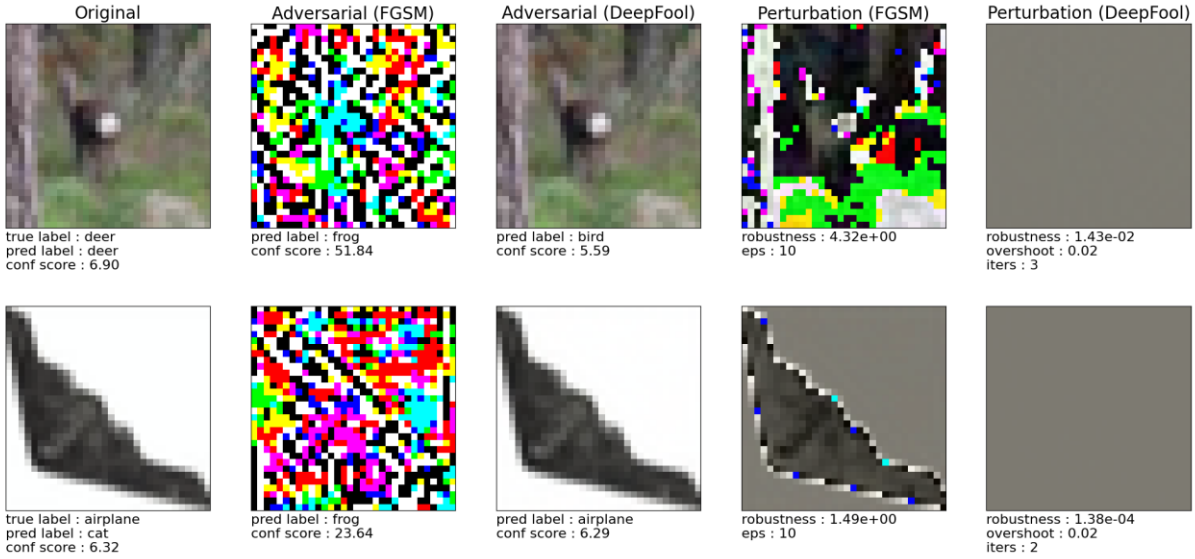

```
[42] fgsm_eps = 0.5
     model = LeNet_CIFAR().to(device)
     model.load_state_dict(torch.load('weights/clean/cifar_lenet.pth'))

     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()
```



```
[43] fgsm_eps = 0.9
     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()
```




```
[44] fgsm_eps = 10
     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()
```



Вывод: значение параметра fgsm_esp оказывает влияние на устойчивость сети. Чем больше значение параметра, тем больше ошибка классификации и сети более уязвимы к атакам. Чем меньше значение параметра, тем сети менее уязвимы к атакам.