МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
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 дисциплине

«Анализ защищенности систем искусственного интеллекта»

Группа:
ББМО-01-22
Выполнил:
Некрасов Е.А.
Вариант 36

Проверил:
Спирин А.А.

Москва 2023

Цель лабораторной работы

В данной лабораторной работе необходимо выявить закономерность или обнаружить отсутствие влияния параметра fgsm_eps на стойкость моделей к атаке. Закономерности или их отсутствие необходимо выявить для сети FC LeNet на датасете MNIST и для сети NiN LeNEt на датасете CIFAR.

Клонируем репозиторий, сменим директорию на EEL6812_DeepFool_Project и загрузим нужные библиотеки.

```
!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 | 14.58 MiB/s, done.
Resolving deltas: 100% (27/27), done.
```

еним директорию

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

```
/content/EEL6812_DeepFool_Project
```

олним импорт 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 рандомное значение 36 и выбираекм устройство выполнения.

```
rand_seed = 36
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')
```

Загрузим датасет MNIST.

```
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.

```
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']
```

Редактируем гиперпараметры.

```
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
  warnings.warn(_create_warning_msg(
```

Зададим параметры на модель.

```
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
```

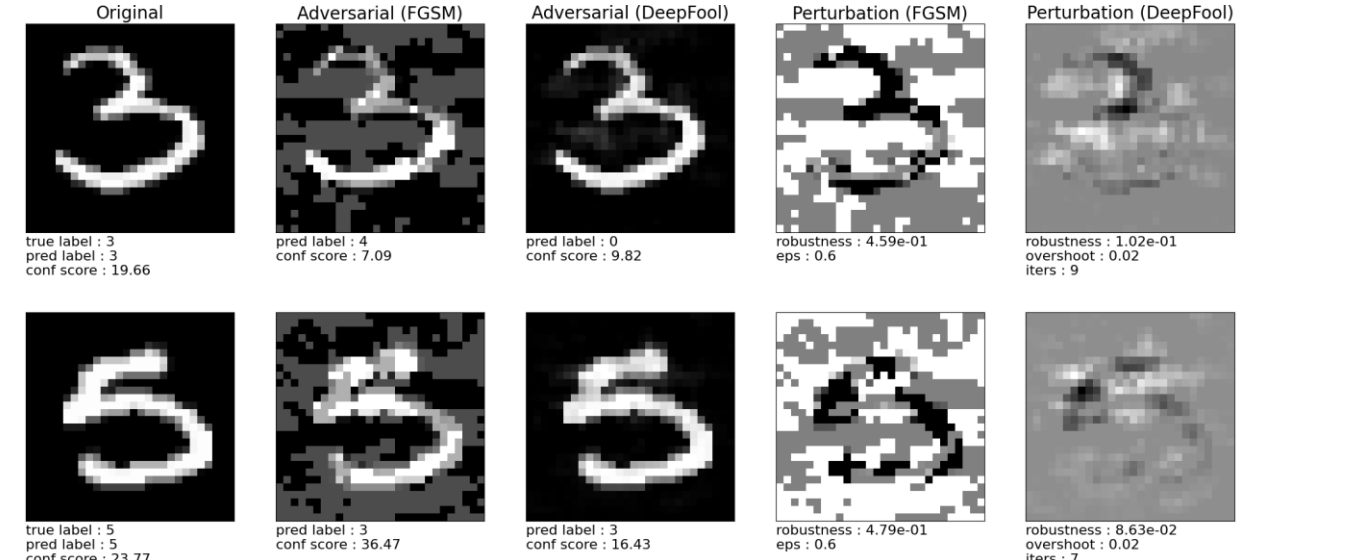
Загрузим и оценим стойкость модели LeNet к FGSM и DeepFool атакам на основе датасета CIFAR-10.

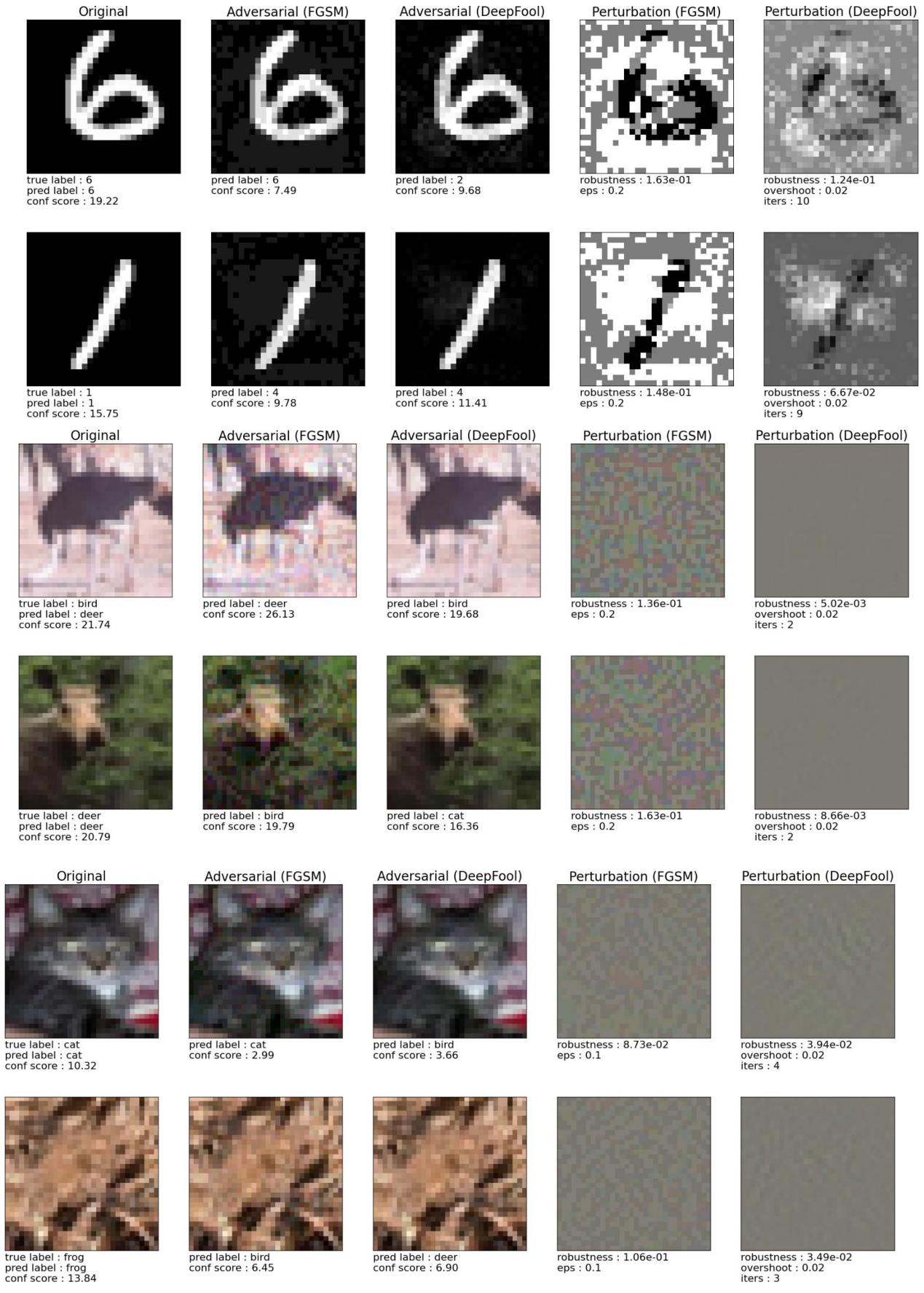
```
fgsm_eps = 0.1
model = LeNet_CIFAR().to(device)
model.load_state_dict(torch.load('weights/clean/cifar_lenet.pth', map_location=torch.device('cpu')))
evaluate_attack('cifar_lenet_fgsm.csv', 'results', device, model, cifar_loader_test, cifar_min, cifar_max, fgsm_eps, is_fgsm=True)
print('')
evaluate_attack('cifar_lenet_deepfool.csv', '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.

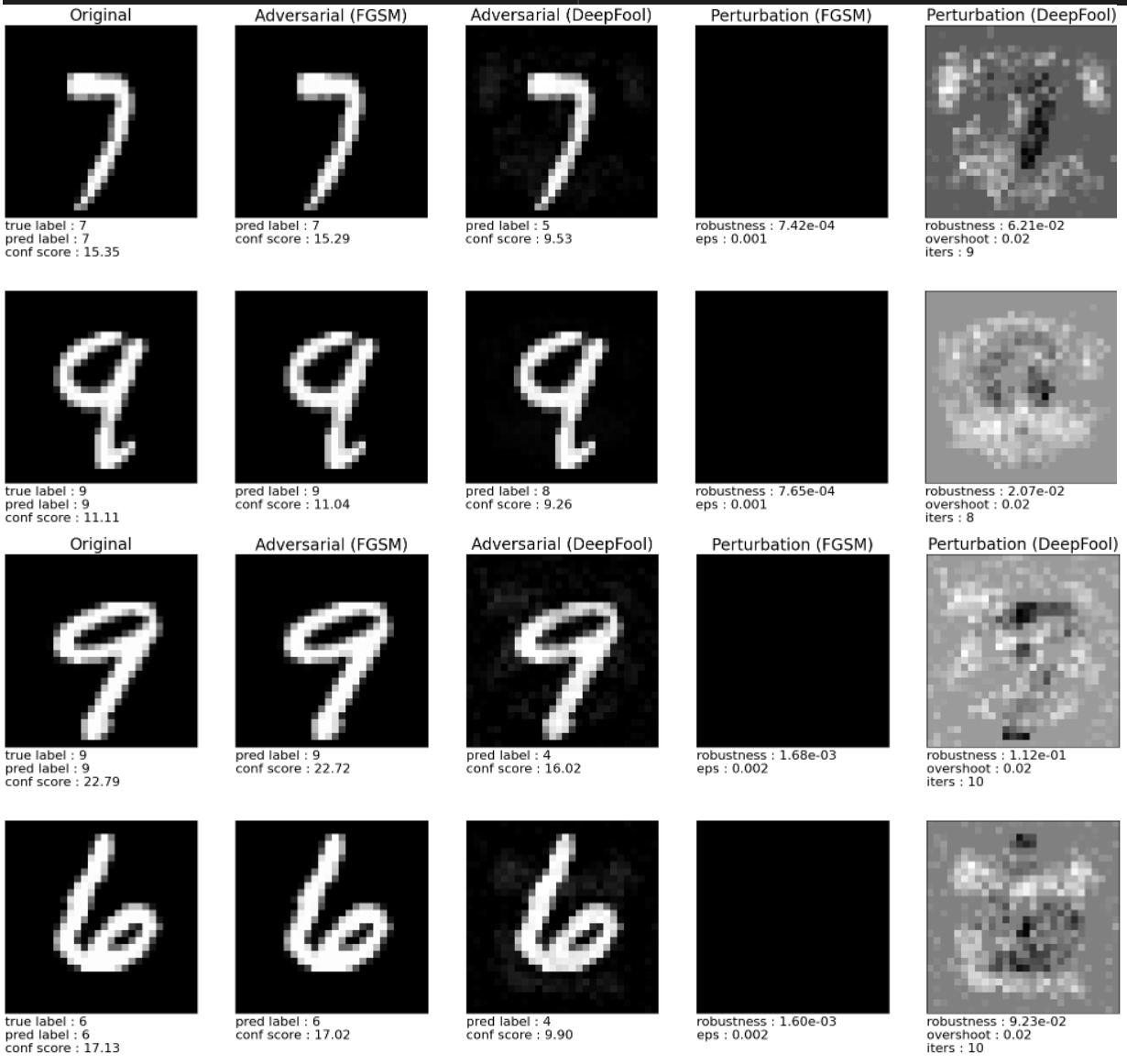
```
#LeNet
fgsm_eps = 0.6
model = LeNet_MNIST().to(device)
model.load_state_dict(torch.load('weights/clean/mnist_lenet.pth'))
display_attack(device, model, mnist_test, mnist_tf_inv, mnist_min, mnist_max, fgsm_eps, deep_args, has_labels=False, l2_norm=True, pert_scale=1.0, fig_rows=2, fig_width=25, fig_height=11)
if device.type == 'cuda': torch.cuda.empty_cache()

#FCNet
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()

#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()

#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()
```

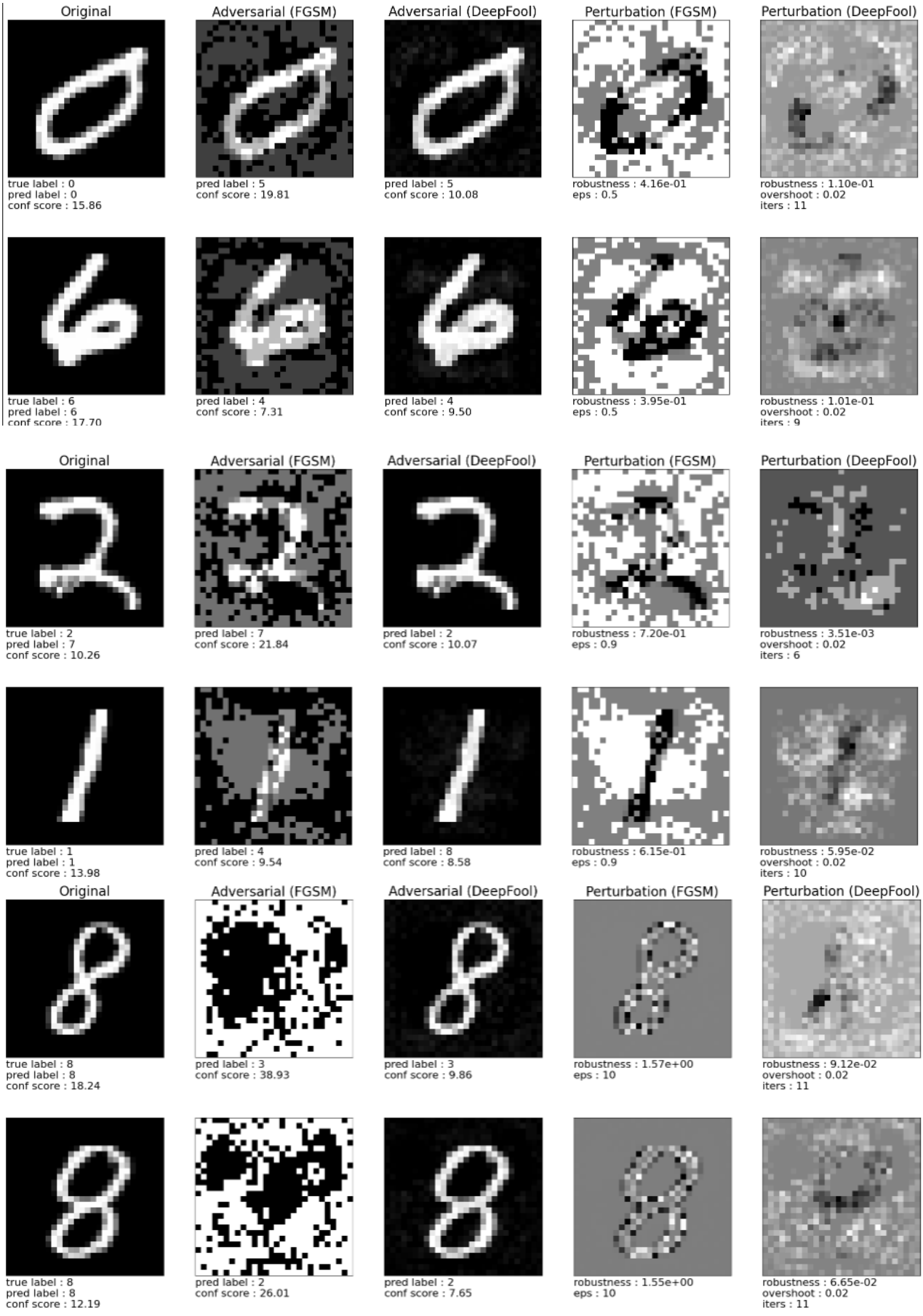



Original
Adversarial (FGSM)
Adversarial (DeepFool)
Perturbation (FGSM)
Perturbation (DeepFool)
true label : 6
pred label : 6
conf score : 19.22
pred label : 6
conf score : 7.49
pred label : 2
conf score : 9.68
robustness : 1.63e-01
eps : 0.2
robustness : 1.24e-01
overshoot : 0.02
iters : 10
true label : 1
pred label : 1
conf score : 15.75
pred label : 4
conf score : 9.78
pred label : 4
conf score : 11.41
robustness : 1.48e-01
eps : 0.2
robustness : 6.67e-02
overshoot : 0.02
iters : 9
Original
Adversarial (FGSM)
Adversarial (DeepFool)
Perturbation (FGSM)
Perturbation (DeepFool)
true label : bird
pred label : deer
conf score : 21.74
pred label : deer
conf score : 26.13
pred label : bird
conf score : 19.68
robustness : 1.36e-01
eps : 0.2
robustness : 5.02e-03
overshoot : 0.02
iters : 2
true label : deer
pred label : deer
conf score : 20.79
pred label : bird
conf score : 19.79
pred label : cat
conf score : 16.36
robustness : 1.63e-01
eps : 0.2
robustness : 8.66e-03
overshoot : 0.02
iters : 2
Original
Adversarial (FGSM)
Adversarial (DeepFool)
Perturbation (FGSM)
Perturbation (DeepFool)
true label : cat
pred label : cat
conf score : 10.32
pred label : cat
conf score : 2.99
pred label : bird
conf score : 3.66
robustness : 8.73e-02
eps : 0.1
robustness : 3.94e-02
overshoot : 0.02
iters : 4
true label : frog
pred label : frog
conf score : 13.84
pred label : bird
conf score : 6.45
pred label : deer
conf score : 6.90
robustness : 1.06e-01
eps : 0.1
robustness : 3.49e-02
overshoot : 0.02
iters : 3

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

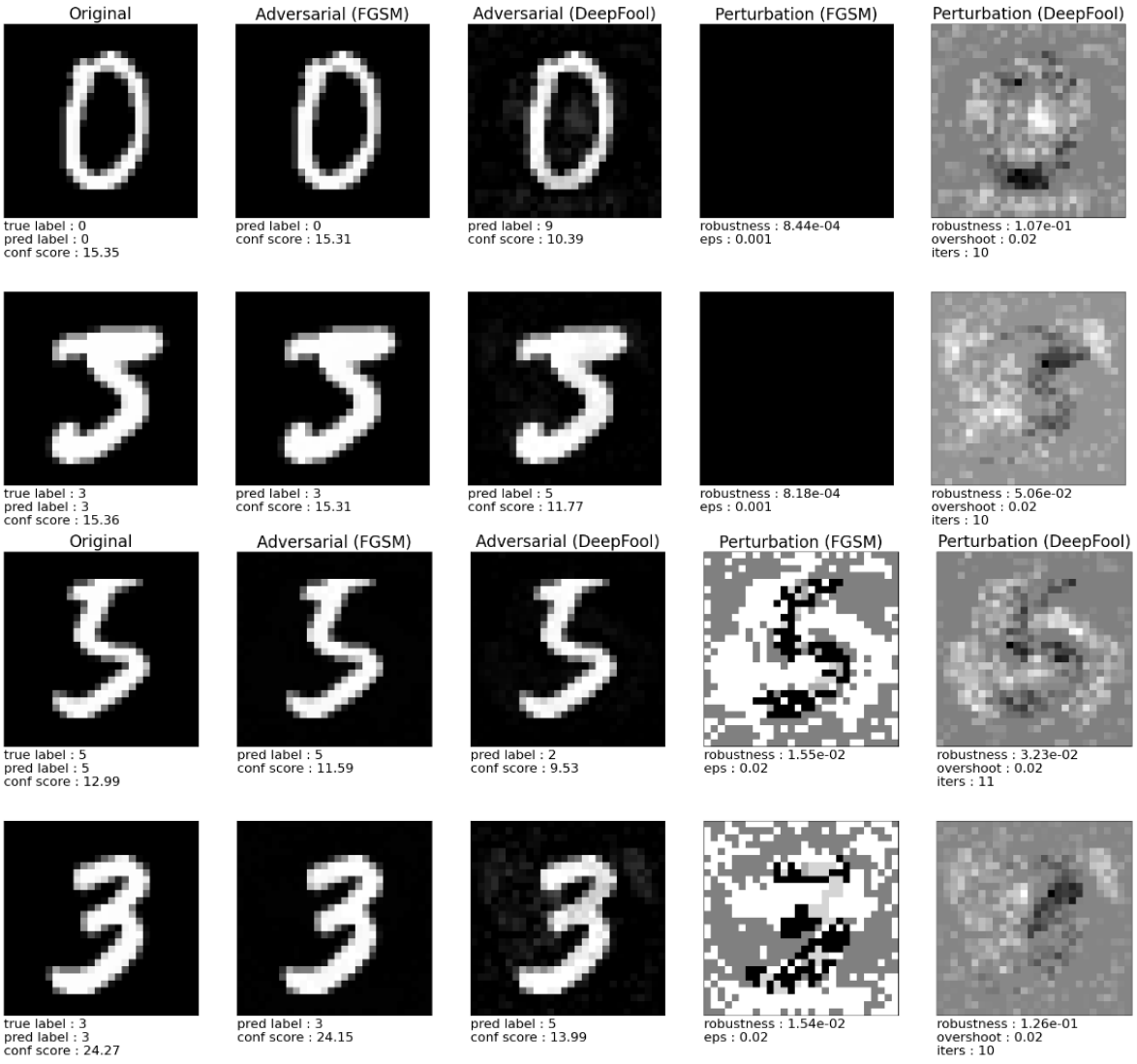
```
fgsm_eps = 0.001
model = FC_500_150().to(device)
model.load_state_dict(torch.load('weights/clean/mnist_fc.pth'))
display_attack(device, model, mnist_test, mnist_tf_inv, mnist_min, mnist_max, fgsm_eps, deep_args, has_labels=False, l2_norm=True, pert_scale=1.0, fig_rows=2, fig_width=25, fig
if device.type == 'cuda': torch.cuda.empty_cache()

fgsm_eps = 0.002
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
if device.type == 'cuda': torch.cuda.empty_cache()

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

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

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

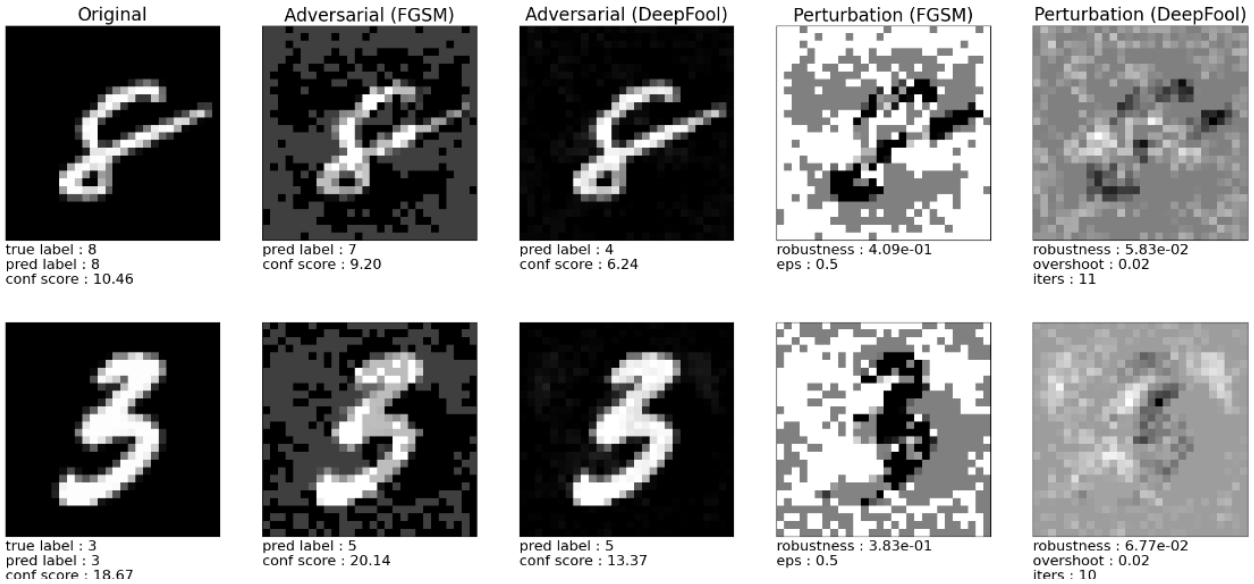


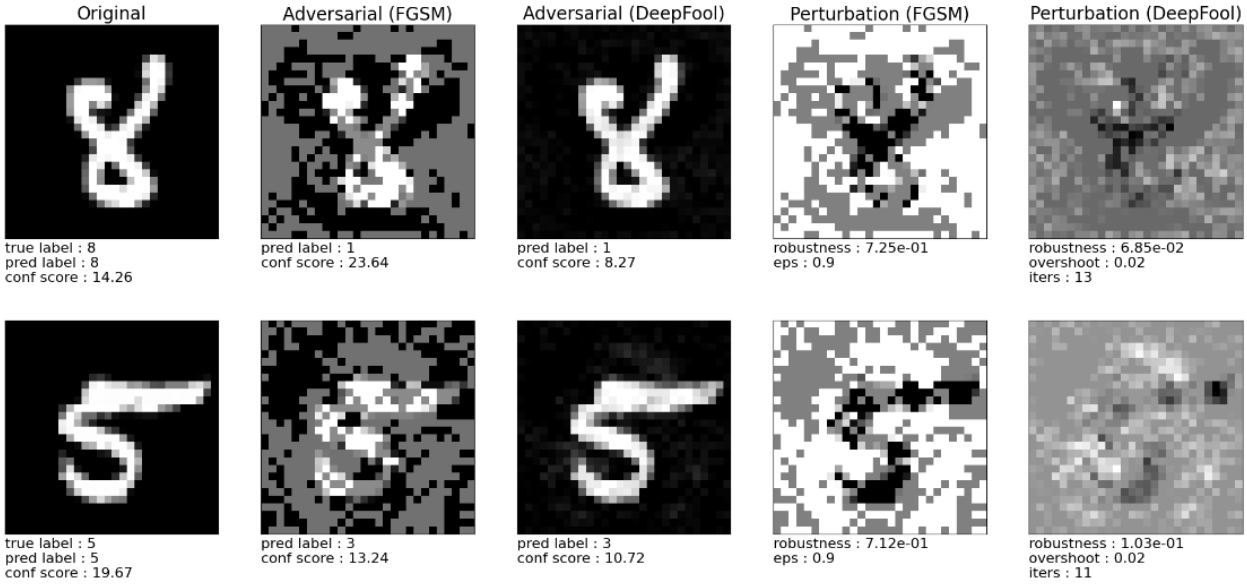

Original
Adversarial (FGSM)
Adversarial (DeepFool)
Perturbation (FGSM)
Perturbation (DeepFool)
true label : 0
pred label : 0
conf score : 15.86
pred label : 5
conf score : 19.81
pred label : 5
conf score : 10.08
robustness : 4.16e-01
eps : 0.5
robustness : 1.10e-01
overshoot : 0.02
iters : 11
true label : 6
pred label : 6
conf score : 17.70
pred label : 4
conf score : 7.31
pred label : 4
conf score : 9.50
robustness : 3.95e-01
eps : 0.5
robustness : 1.01e-01
overshoot : 0.02
iters : 9
Original
Adversarial (FGSM)
Adversarial (DeepFool)
Perturbation (FGSM)
Perturbation (DeepFool)
true label : 2
pred label : 7
conf score : 10.26
pred label : 7
conf score : 21.84
pred label : 2
conf score : 10.07
robustness : 7.20e-01
eps : 0.9
robustness : 3.51e-03
overshoot : 0.02
iters : 6
true label : 1
pred label : 1
conf score : 13.98
pred label : 4
conf score : 9.54
pred label : 8
conf score : 8.58
robustness : 6.15e-01
eps : 0.9
robustness : 5.95e-02
overshoot : 0.02
iters : 10
Original
Adversarial (FGSM)
Adversarial (DeepFool)
Perturbation (FGSM)
Perturbation (DeepFool)
true label : 8
pred label : 8
conf score : 18.24
pred label : 3
conf score : 38.93
pred label : 3
conf score : 9.86
robustness : 1.57e+00
eps : 10
robustness : 9.12e-02
overshoot : 0.02
iters : 11
true label : 8
pred label : 8
conf score : 12.19
pred label : 2
conf score : 26.01
pred label : 2
conf score : 7.65
robustness : 1.55e+00
eps : 10
robustness : 6.65e-02
overshoot : 0.02
iters : 11

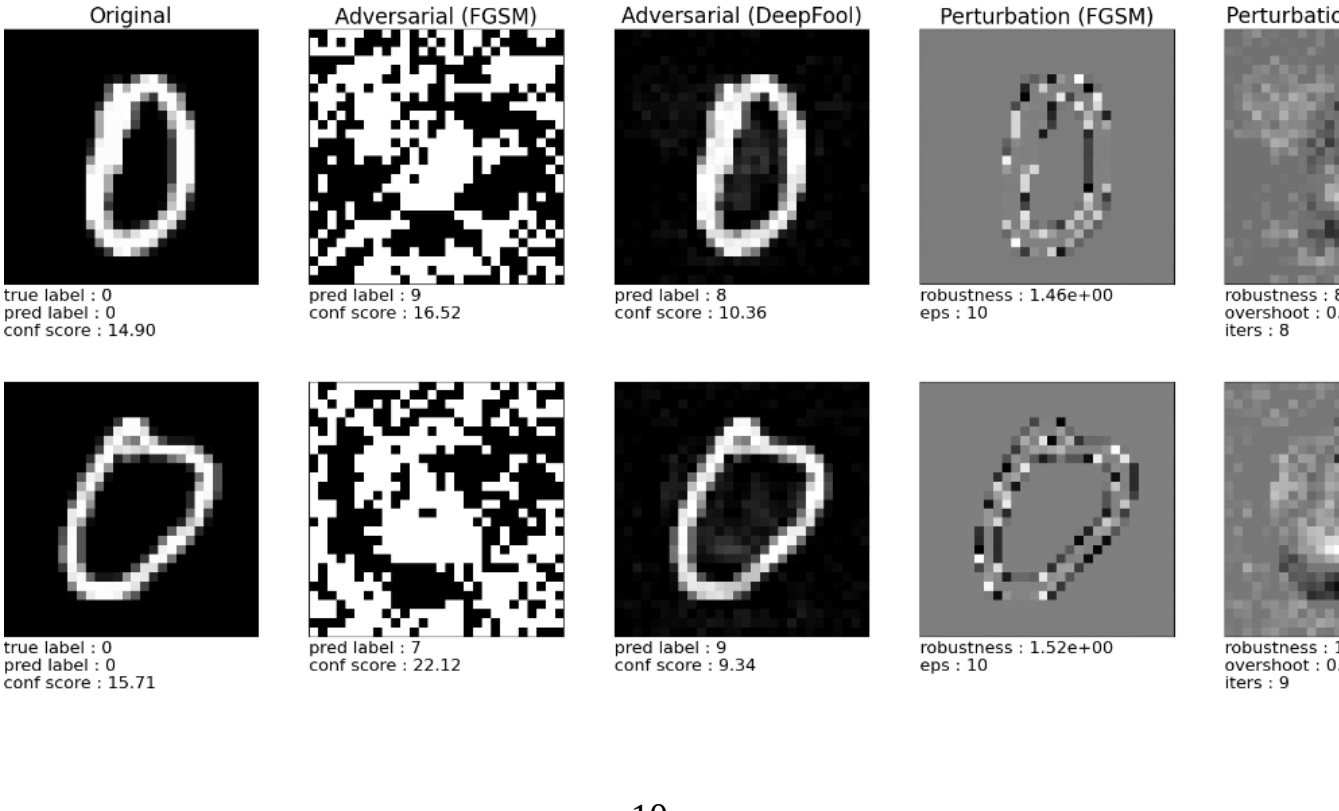
Проверим влияние параметра fgsm_eps для FC на датасетеMNIST.

```
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()

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()

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()

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()

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()
```



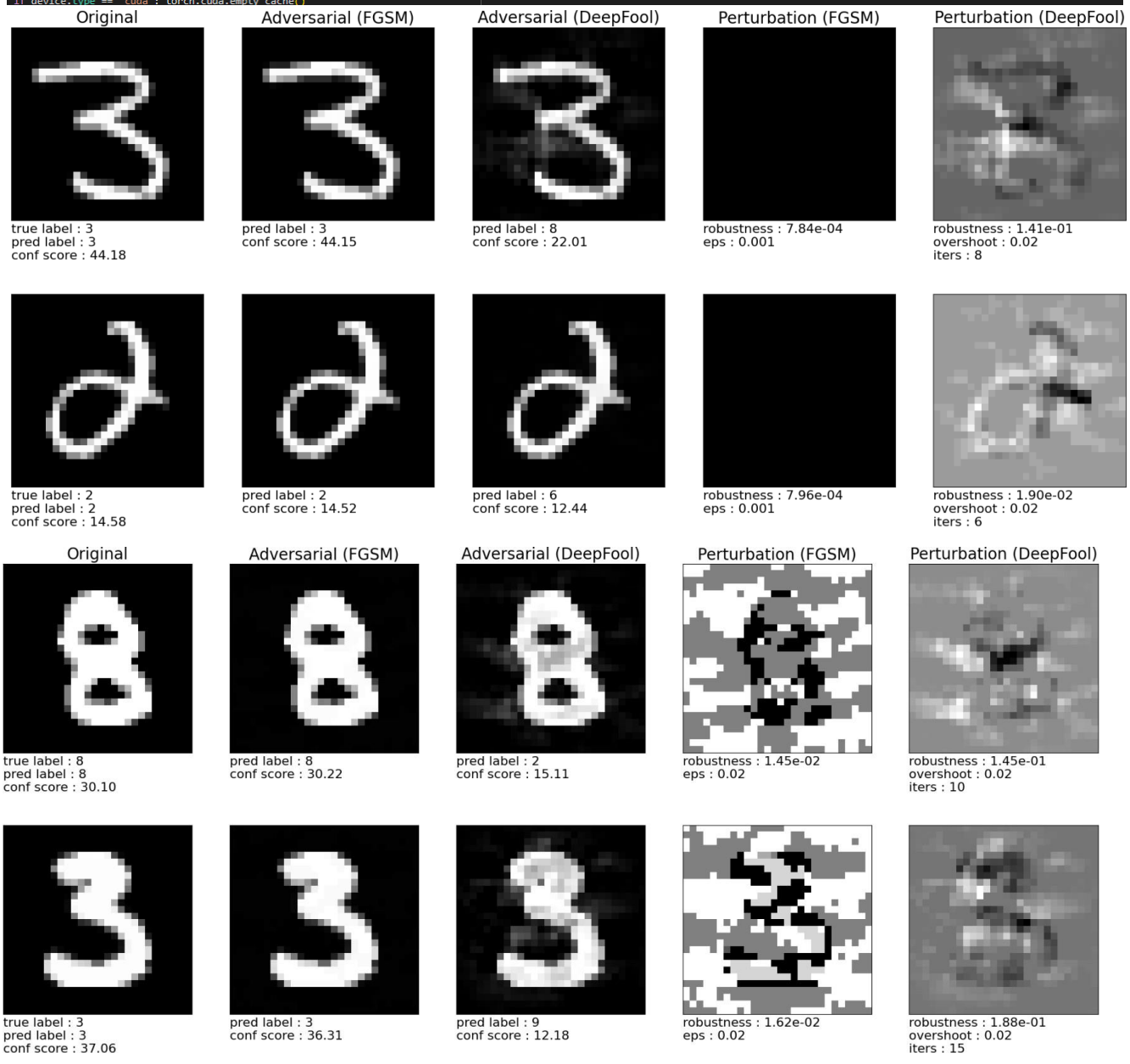

Original
Adversarial (FGSM)
Adversarial (DeepFool)
Perturbation (FGSM)
Perturbation (DeepFool)
true label : 8
pred label : 8
conf score : 10.46
pred label : 7
conf score : 9.20
pred label : 4
conf score : 6.24
robustness : 4.09e-01
eps : 0.5
robustness : 5.83e-02
overshoot : 0.02
iters : 11
true label : 3
pred label : 3
conf score : 18.67
pred label : 5
conf score : 20.14
pred label : 5
conf score : 13.37
robustness : 3.83e-01
eps : 0.5
robustness : 6.77e-02
overshoot : 0.02
iters : 10

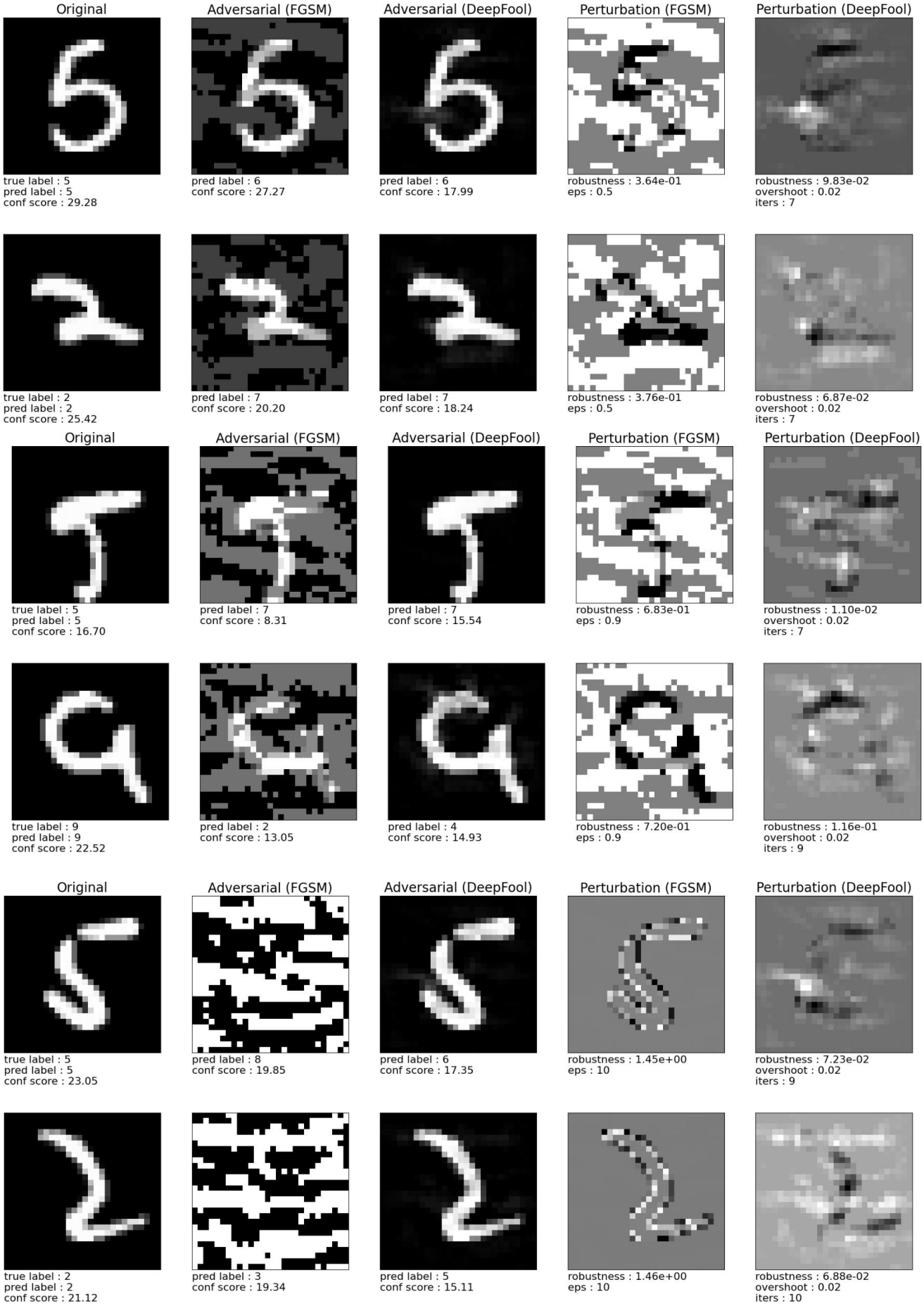

Original
Adversarial (FGSM)
Adversarial (DeepFool)
Perturbation (FGSM)
Perturbation (DeepFool)
true label : 8
pred label : 8
conf score : 14.26
pred label : 1
conf score : 23.64
pred label : 1
conf score : 8.27
robustness : 7.25e-01
eps : 0.9
robustness : 6.85e-02
overshoot : 0.02
iters : 13
true label : 5
pred label : 5
conf score : 19.67
pred label : 3
conf score : 13.24
pred label : 3
conf score : 10.72
robustness : 7.12e-01
eps : 0.9
robustness : 1.03e-01
overshoot : 0.02
iters : 11


Original
Adversarial (FGSM)
Adversarial (DeepFool)
Perturbation (FGSM)
Perturbatio
true label : 0
pred label : 0
conf score : 14.90
pred label : 9
conf score : 16.52
pred label : 8
conf score : 10.36
robustness : 1.46e+00
eps : 10
robustness : 8
overshoot : 0.
iters : 8
true label : 0
pred label : 0
conf score : 15.71
pred label : 7
conf score : 22.12
pred label : 9
conf score : 9.34
robustness : 1.52e+00
eps : 10
robustness : 1
overshoot : 0.
iters : 9

13. Проверим влияние параметра fgsm_esp для LeNet на датасете MNIST.

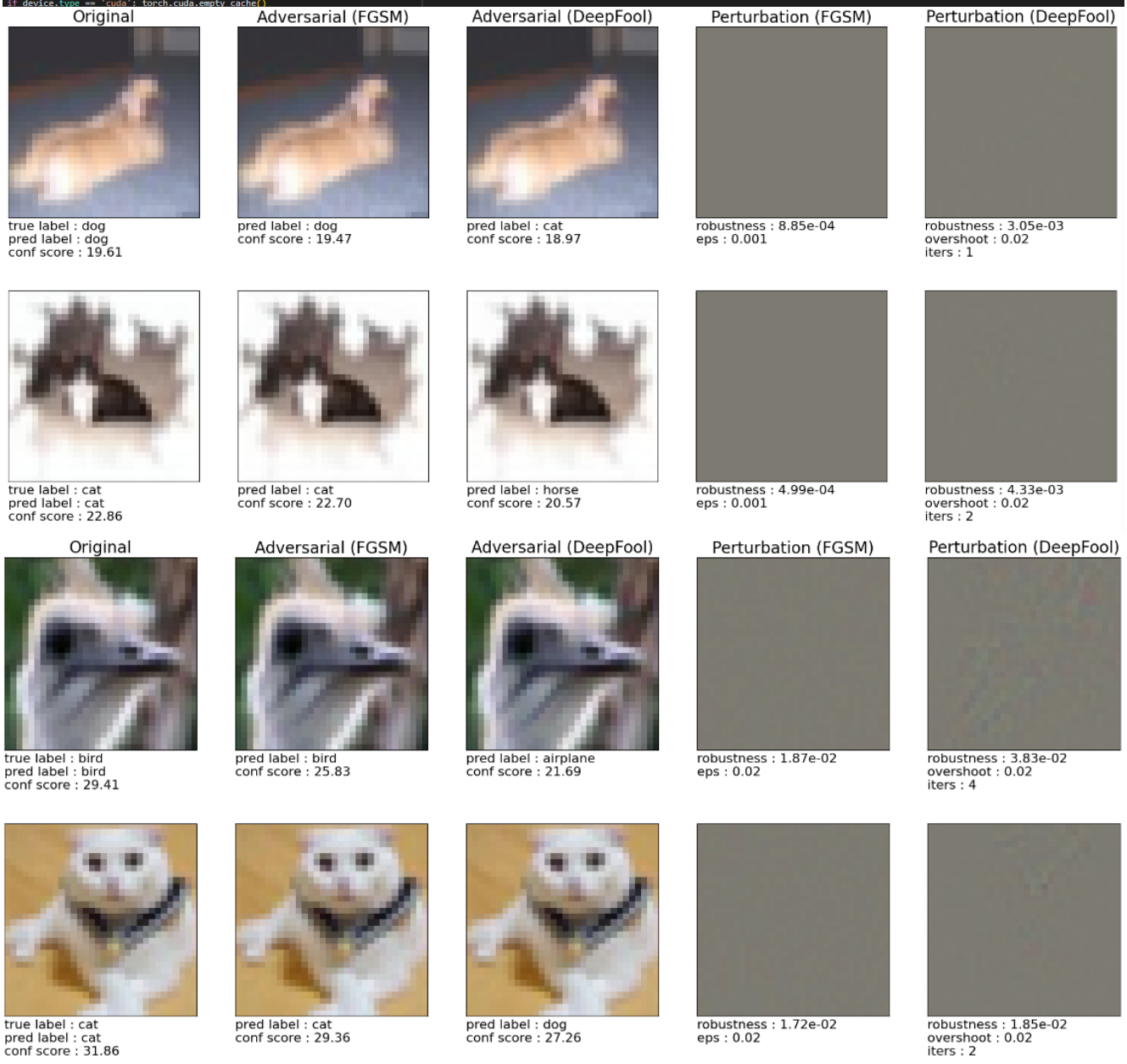
```
fgsm_eps = 0.001
model = LeNet_MNIST().to(device)
model.load_state_dict(torch.load('weights/clean/mnist_lenet.pth'))
display_attack(device, model, mnist_test, mnist_tf_inv, mnist_min, mnist_max, fgsm_eps, deep_args, has_labels=False, l2_norm=True, pert_scale=1.0, fig_rows=2, fig_width=25, fig_height=11)
if device.type == 'cuda': torch.cuda.empty_cache()

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()

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()

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()

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()
```

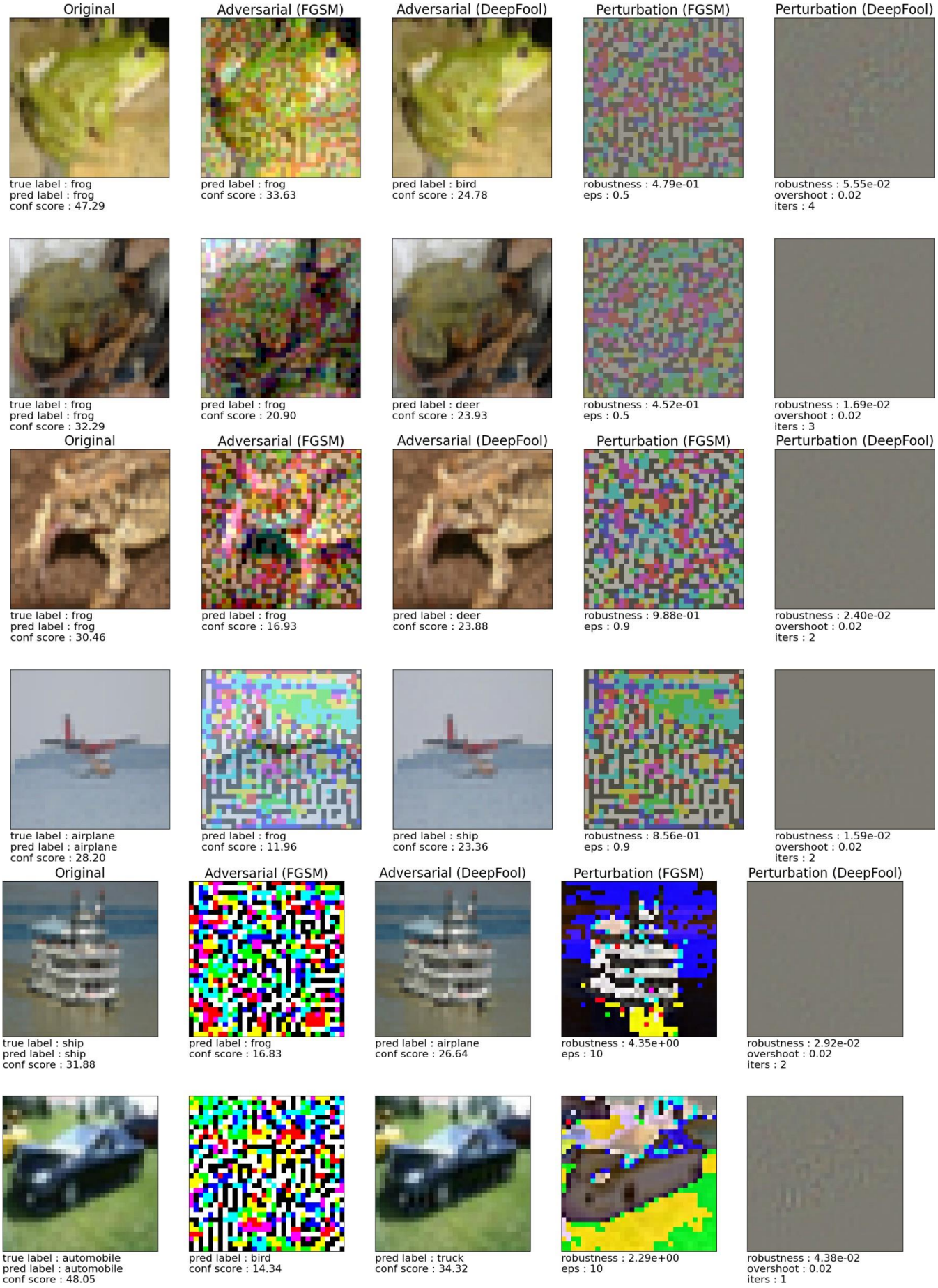




14. Проверим влияние параметра fgsm_esp для NiN на датасете Cifar-

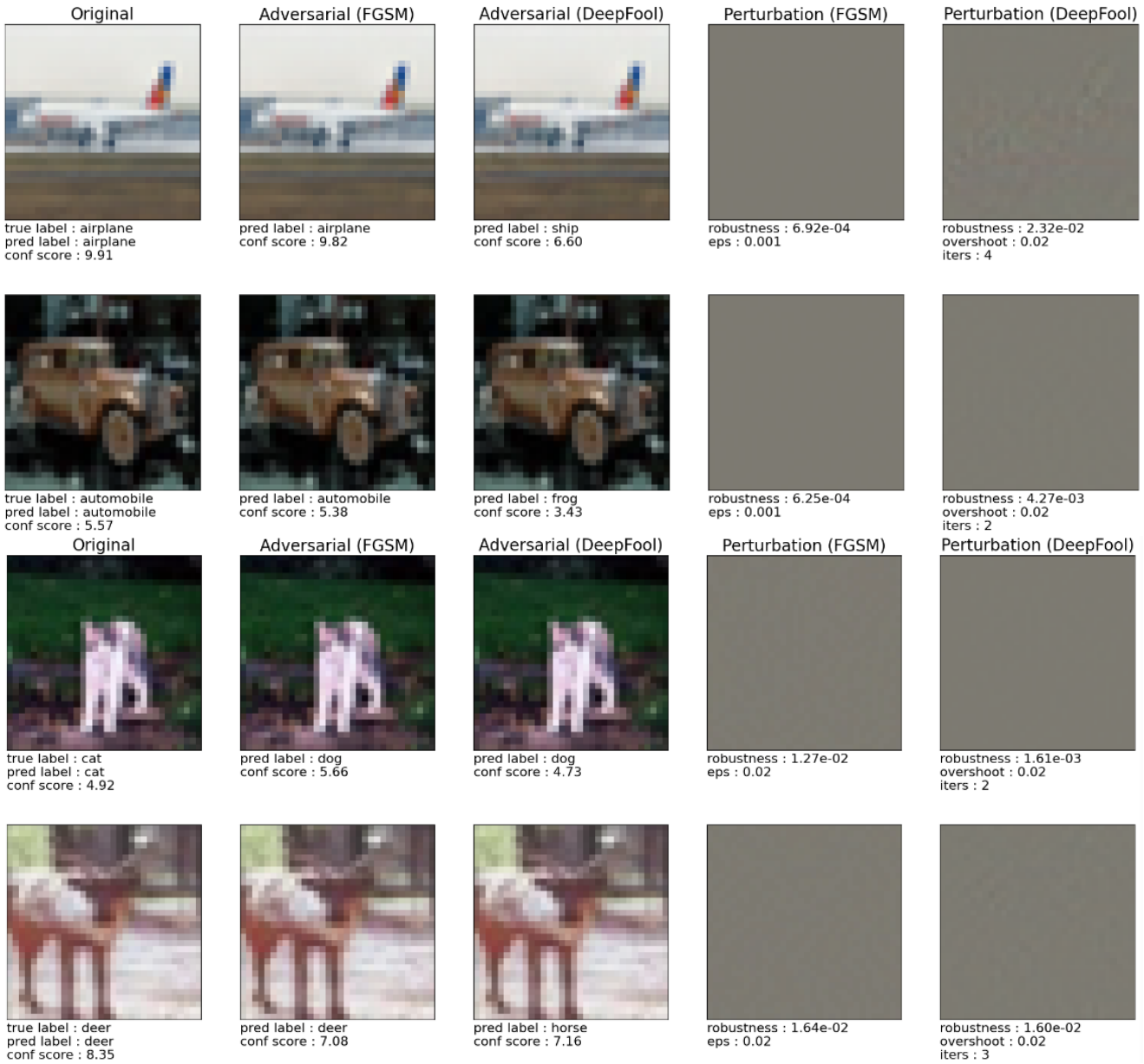
```
fgsm_eps = 0.001
model = Net().to(device)
model.load_state_dict(torch.load('weights/clean/cifar_nin.pth'))
display_attack(device, model, cifar_test, cifar_tf_inv, cifar_min, cifar_max, fgsm_eps, deep_args, has_labels=False, l2_norm=True, pert_scale=1.0, fig_rows=2, fig_width=25, fig_height=11, label_map=cifar_classes)
if device.type == 'cuda': torch.cuda.empty_cache()

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()

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()

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()

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()
```

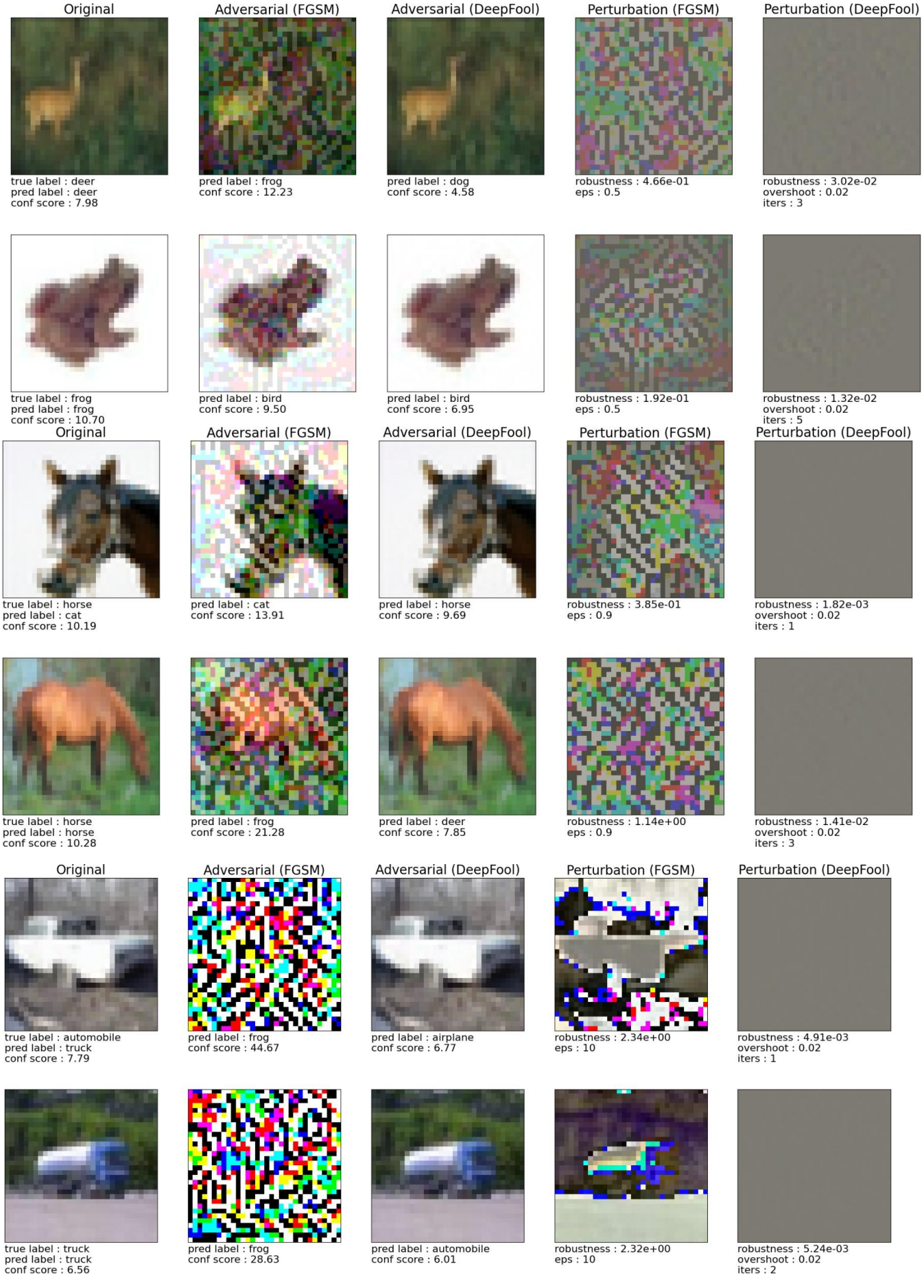




15. Проверим влияние параметра fgsm_esp для LeNet на датасете Cifar-10.

```
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()

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()

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()

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()

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()
```




Original
Adversarial (FGSM)
Adversarial (DeepFool)
Perturbation (FGSM)
Perturbation (DeepFool)
true label : deer
pred label : deer
conf score : 7.98
pred label : frog
conf score : 12.23
pred label : dog
conf score : 4.58
robustness : 4.66e-01
eps : 0.5
robustness : 3.02e-02
overshoot : 0.02
iters : 3
true label : frog
pred label : frog
conf score : 10.70
pred label : bird
conf score : 9.50
pred label : bird
conf score : 6.95
robustness : 1.92e-01
eps : 0.5
robustness : 1.32e-02
overshoot : 0.02
iters : 5
Original
Adversarial (FGSM)
Adversarial (DeepFool)
Perturbation (FGSM)
Perturbation (DeepFool)
true label : horse
pred label : cat
conf score : 10.19
pred label : cat
conf score : 13.91
pred label : horse
conf score : 9.69
robustness : 3.85e-01
eps : 0.9
robustness : 1.82e-03
overshoot : 0.02
iters : 1
true label : horse
pred label : horse
conf score : 10.28
pred label : frog
conf score : 21.28
pred label : deer
conf score : 7.85
robustness : 1.14e+00
eps : 0.9
robustness : 1.41e-02
overshoot : 0.02
iters : 3
Original
Adversarial (FGSM)
Adversarial (DeepFool)
Perturbation (FGSM)
Perturbation (DeepFool)
true label : automobile
pred label : truck
conf score : 7.79
pred label : frog
conf score : 44.67
pred label : airplane
conf score : 6.77
robustness : 2.34e+00
eps : 10
robustness : 4.91e-03
overshoot : 0.02
iters : 1
true label : truck
pred label : truck
conf score : 6.56
pred label : frog
conf score : 28.63
pred label : automobile
conf score : 6.01
robustness : 2.32e+00
eps : 10
robustness : 5.24e-03
overshoot : 0.02
iters : 2

Вывод: параметр fgsm_esp влияет на устойчивость сети. Маленькие значения fgsm_eps сохраняют стойкость сетей к атакам, и ошибки классификации остаются низкими. При увеличении fgsm_eps сети становятся более уязвимыми к атакам и допускают больше ошибок классификации. Для сети FC LeNet на датасете MNIST и для сети NiN LeNEt на датасете CIFAR не наблюдается отсутствие влияния параметра fgsm_eps. Наоборот, параметр fgsm_eps оказывает существенное влияние на стойкость сетей к атакам.