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

Институт комплексной безопасности и специального приборостроения

КБ-4 «Интеллектуальные системы информационной безопасности»

Отчет

по лабораторной работе №1

по дисциплине «Анализ защищенности систем искусственного интеллекта»

Выполнил студент:

Группы: ББМО-02-22

ФИО: Исаев А.М.

Проверил:

к.т.н Спирин А.А

Москва 2022

Ход работы:

1) Клонируем репозиторий

```
[2] #cloning repo
    !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 | 15.99 MiB/s, done.
Resolving deltas: 100% (27/27), done.
```

Рисунок 1 – клонируем репо

2) Переходим в директорию, импортируем библиотеки и выставляем значение rand_seed согласно варианту = Исаев А.М. – 43

```
[3] #заходим в директорию
    %cd EEL6812_DeepFool_Project

/content/EEL6812_DeepFool_Project
```

```
#импорт нужных библиотек
import numpy as np
import json, torch
import os
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
```

```
[5] #значение rand_seed согласно варианту = Исаев А.М. - 43
    rand_seed = 43
    np.random.seed(rand_seed)
    torch.manual_seed(rand_seed)

    # в качестсве устройства используем видеокарту
    use_cuda = torch.cuda.is_available()
    device = torch.device('cuda' if use_cuda else 'cpu')
```

Рисунок 2 – cd в директорию

3) Меняем среду исполнения

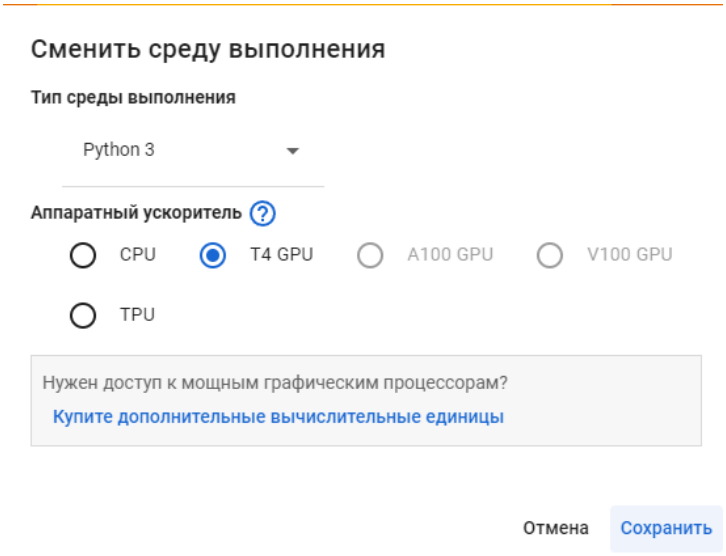


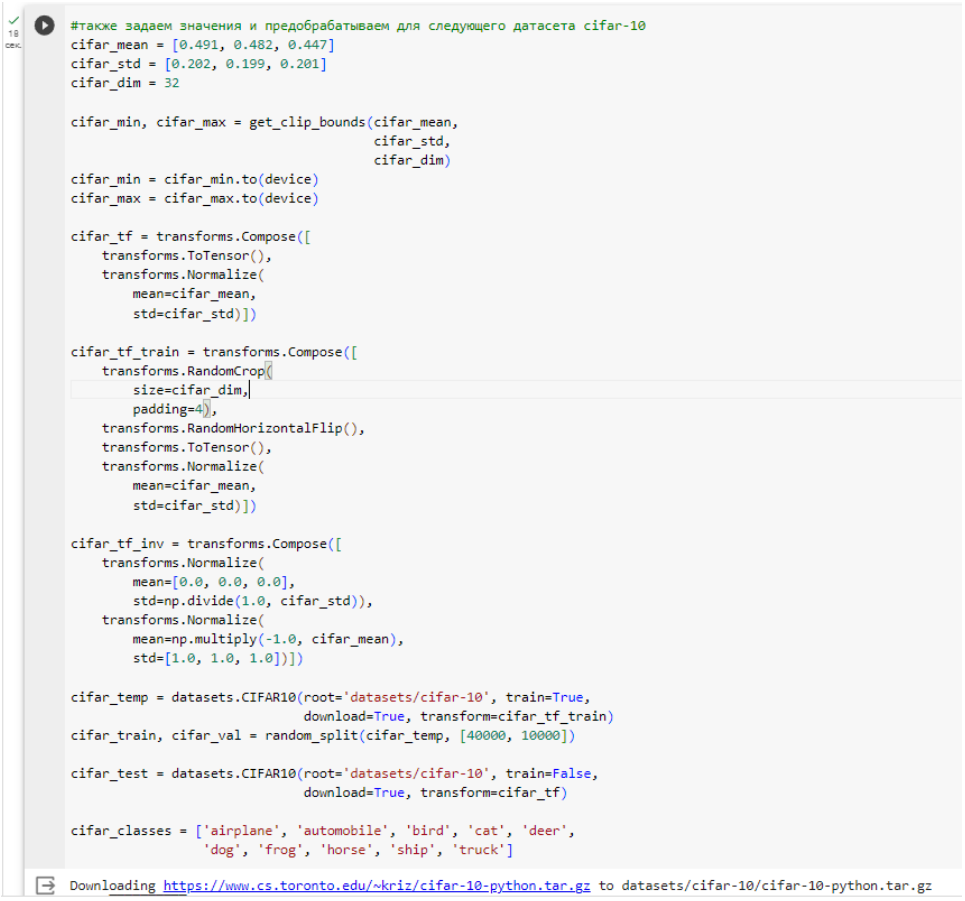
Рисунок 3 – смена среды выполнения на GPU

4) Задаем значения для предобработки датасета MNIST

```
#задаем значения для предобработки датасета 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)
```

Рисунок 4 – загрузка датасета MNIST

5) Задаем значения для предобработки датасета CIFAR-10

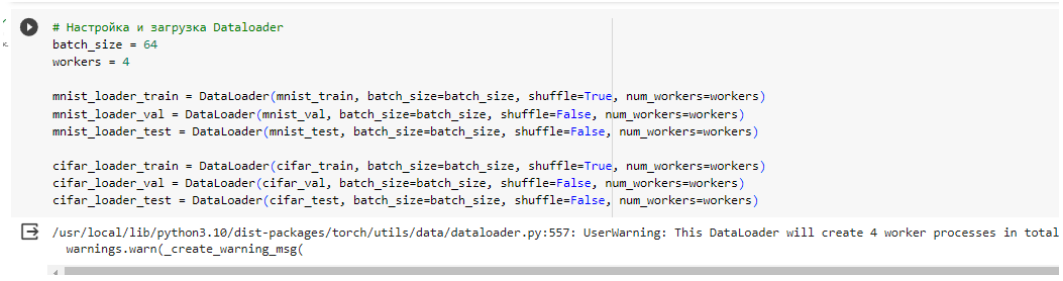


```
#также задаем значения и предобрабатываем для следующего датасета cifar-10
cifar_mean = [0.491, 0.482, 0.447]
cifar_std = [0.202, 0.199, 0.201]
cifar_dim = 32

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']
```

```
Downloading https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz to datasets/cifar-10/cifar-10-python.tar.gz
```

Рисунок 5 – загрузка датасета CIFAR-10

6) Настраиваем и загружаем DataLoader



```
# Настройка и загрузка 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)
```

```
/usr/local/lib/python3.10/dist-packages/torch/utils/data/dataloader.py:557: UserWarning: This DataLoader will create 4 worker processes in total
  warnings.warn(_create_warning_msg(
```

Рисунок 6 – настройка и загрузка dataloader

7) Загружаем и оцениваем стойкость модели LeNet к FGSM и DeepFool атакам

```
# Загружаем и оцениваем стойкость модели LeNet к FGSM и DeepFool атакам
fgsm_eps = 0.6

model = LeNet_MNIST().to(device)
model.load_state_dict(torch.load('weights/clean/mnist_lenet.pth', map_location=torch.device('cpu')))

evaluate_attack('mnist_lenet_fgsm.csv', 'results', device, model, mnist_loader_test, mnist_min, mnist_max, fgsm_eps, is_fgsm=True)

print('')

batch = 64
num_classes = 10
overshoot = 0.02
max_iter = 50
deep_arg = [batch, num_classes, overshoot, max_iter]

evaluate_attack('mnist_lenet_deepfool.csv', 'results', device, model, mnist_loader_test, mnist_min, mnist_max, deep_arg, is_fgsm=False)

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

```
FGSM Test Error : 87.89%
FGSM Robustness : 4.58e-01
FGSM Time (All Images) : 0.29 s
FGSM Time (Per Image) : 28.86 us

DeepFool Test Error : 98.74%
DeepFool Robustness : 9.64e-02
DeepFool Time (All Images) : 193.32 s
DeepFool Time (Per Image) : 19.33 ms
```

Рисунок 7 – загрузка и оценка стойкости LeNet к FGSM и DeepFool атакам

8) Далее, загружаем и оцениваем стойкость модели FC к FGSM и DeepFool атакам

```
# Загружаем и оцениваем стойкость модели FC к FGSM и DeepFool атакам
fgsm_eps = 0.2

model = FC_500_150().to(device)
model.load_state_dict(torch.load('weights/clean/mnist_fc.pth', map_location=torch.device('cpu')))

evaluate_attack('mnist_fc_fgsm.csv', 'results', device, model, mnist_loader_test, mnist_min, mnist_max, fgsm_eps, is_fgsm=True)

print('')

batch = 64
num_classes = 10
overshoot = 0.02
max_iter = 50
deep_arg = [batch, num_classes, overshoot, max_iter]

evaluate_attack('mnist_fc_deepfool.csv', 'results', device, model, mnist_loader_test, mnist_min, mnist_max, deep_arg, is_fgsm=False)

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

```
FGSM Test Error : 87.08%
FGSM Robustness : 1.56e-01
FGSM Time (All Images) : 0.15 s
FGSM Time (Per Image) : 14.99 us

DeepFool Test Error : 97.92%
DeepFool Robustness : 6.78e-02
DeepFool Time (All Images) : 141.81 s
DeepFool Time (Per Image) : 14.18 ms
```

Рисунок 8 – загрузка и оценка стойкости FC к FGSM и DeepFool атакам

9) Необходимо выполнить оценку атакующих примеров для сетей.

```
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_arg, 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
  warnings.warn(_create_warning_msg(
```

Рисунок 9 – загрузка модели LeNet на датасете MNIST

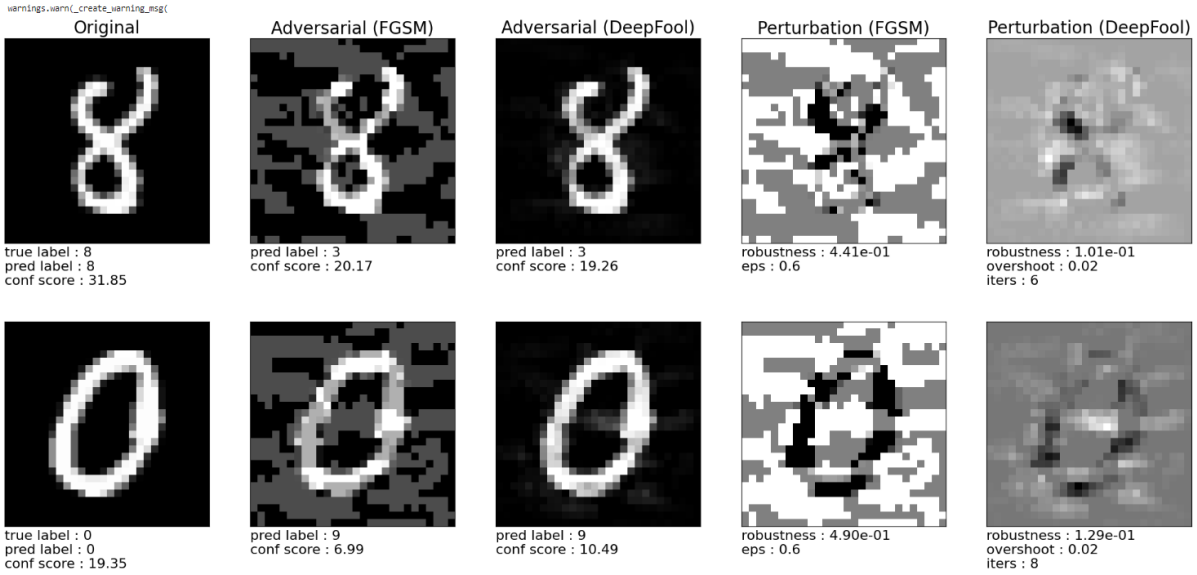


Рисунок 10 – Оценка атакующих примеров для LeNet на MNIST

Далее, загружаем модель FC на датасете MNIST

```
fgsm_eps = 0.2
model = FC_500_150().to(device)
model.load_state_dict(torch.load('weights/clean/mnist_fc.pth', map_location=device))

display_attack(device, model, mnist_test, mnist_tf_inv, mnist_min, mnist_max, fgsm_eps, deep_arg, 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()
```

Рисунок 11 – загрузка модели FC на датасете MNIST

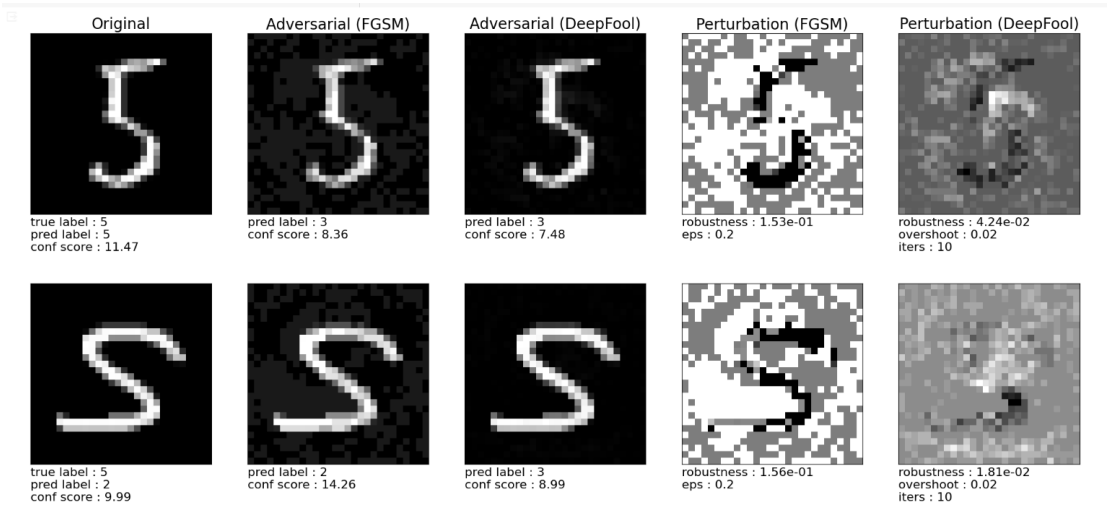


Рисунок 12 – Оценка атакующих примеров для FC на датасете MNIST

```
#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_arg, 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()
```

Рисунок 13 – загрузка модели NiN на датасете CIFAR-10

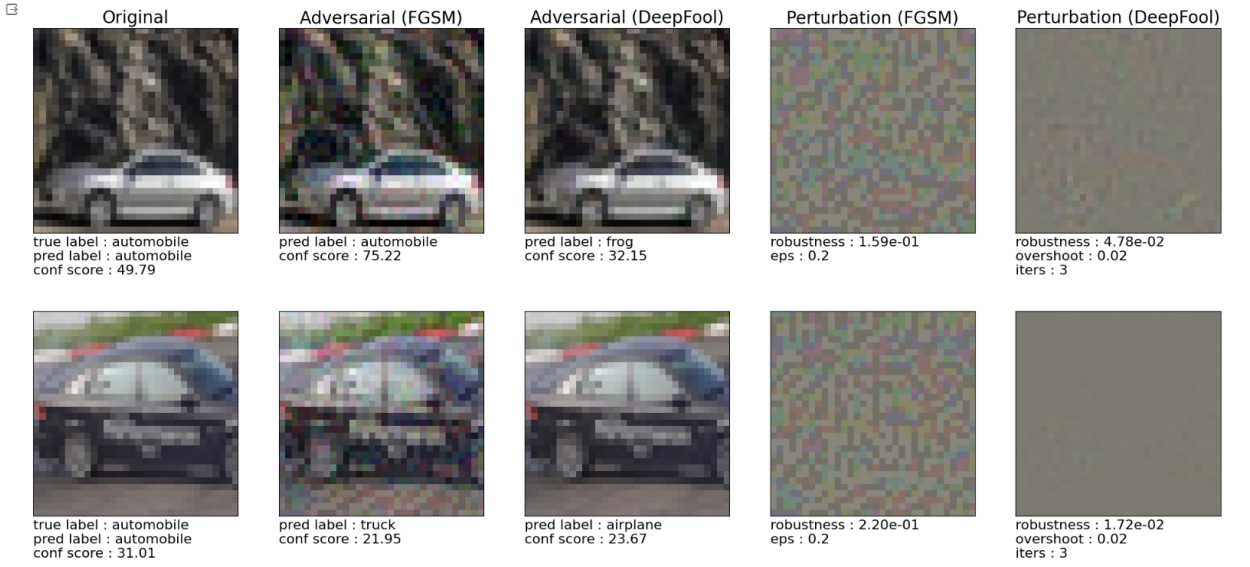


Рисунок 14 – Оценка атакующих примеров для NiN на датасете CIFAR-10

```
[15] #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_arg, 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()
```

Рисунок 15 – загрузка модели LeNet на датасете CIFAR-10

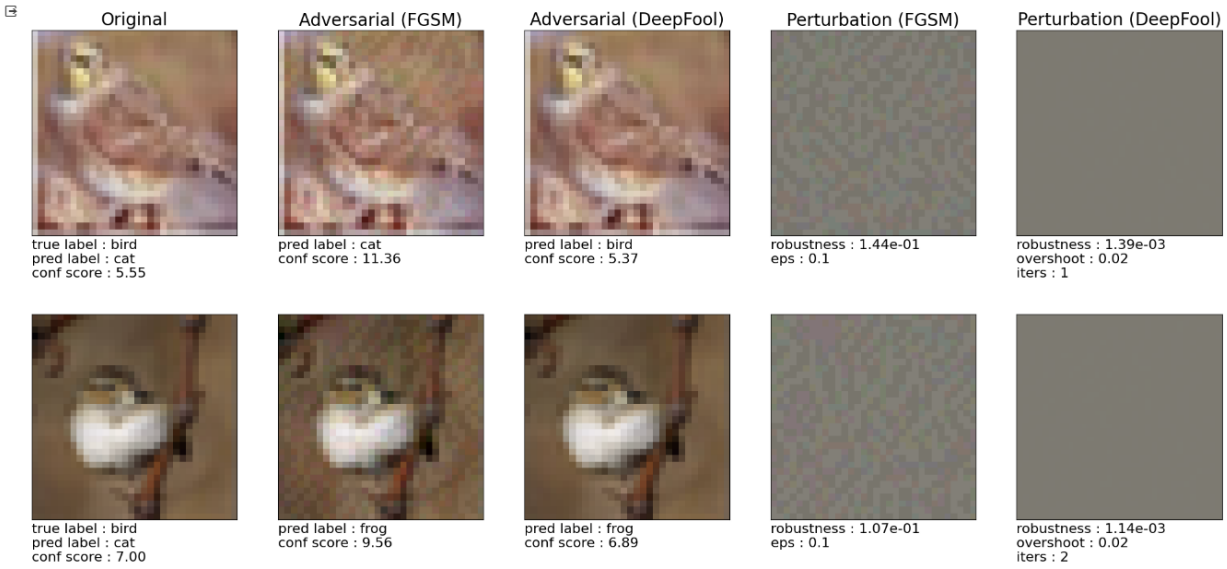


Рисунок 16 – Оценка атакующих примеров для LeNet на CIFAR-10

Далее, необходимо отразить отличия для fgsm_eps=(0,001,0,02,0.5,0.9,10) и выявить закономерность или обнаружить отсутствие влияния eps для сетей FC LeNet на датасете MNIST, NetforkinNetwork LeNet на датасете CIFAR-10

Приступим с оценки FC на MNIST

```
[16] # отличия для fgsm_eps=(0.001, 0.02, 0.5, 0.9, 10) и выявить закономерность/обнаружить отсутсвие влияние параметра eps для сетей FC LeNet на датасете MNIST, NiN LeNEt на датасете CIFAR).

     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_arg, 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()
```

Рисунок 17 –цикл перебора fgsm_eps для FC на MNIST при 0,001

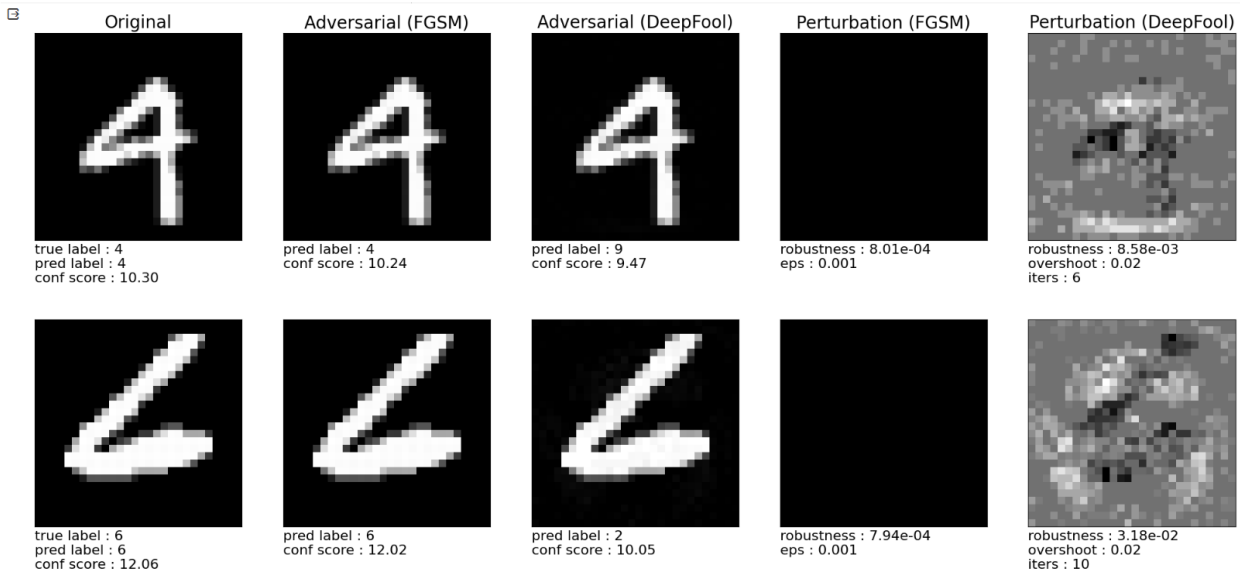


Рисунок 18 – Оценка атакующих примеров для FC на MNIST при 0,001

```
#0.02
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_arg, 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()
```

Рисунок 19 – цикл перебора fgsm_eps для FC на MNIST при 0,02

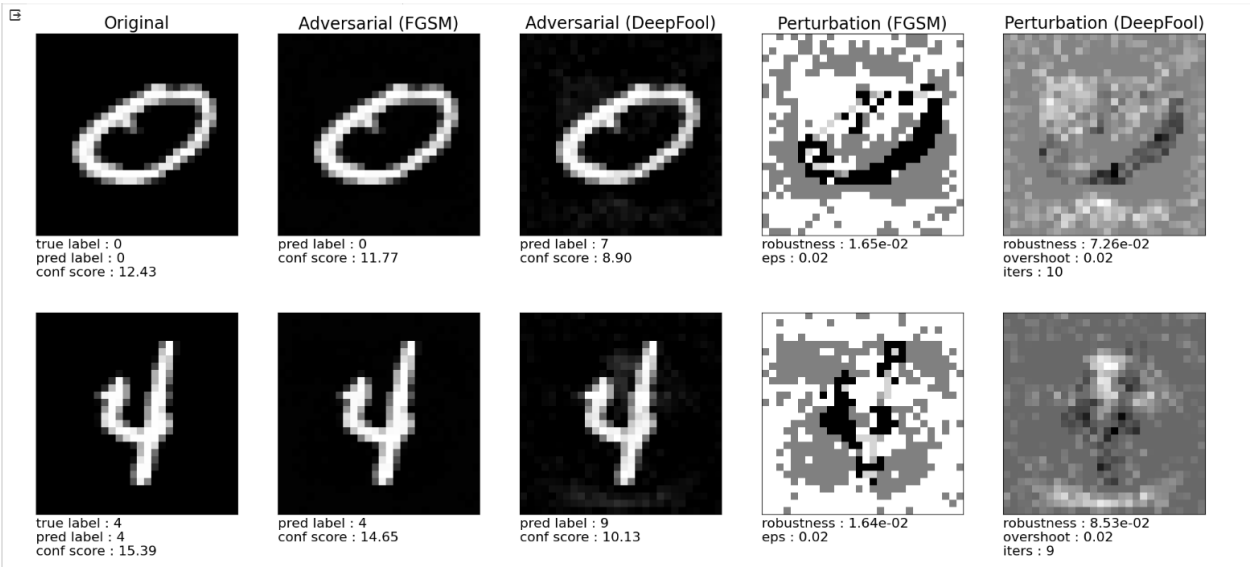


Рисунок 20 – Оценка атакующих примеров FC на MNIST при 0,02

```
#0.5
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_arg, 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()
```

Рисунок 21 – цикл перебора fgsm_eps для FC на MNIST при 0,5

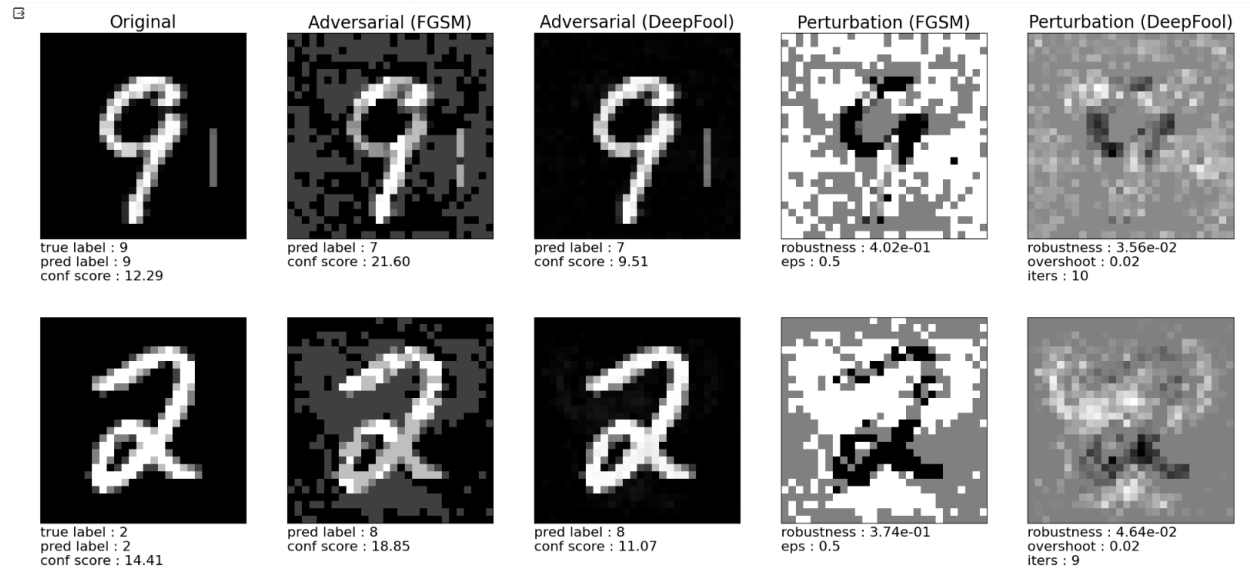


Рисунок 22 – Оценка атакующих примеров для FC на MNIST при 0,5

```
#0.9
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_arg, 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()
```

Рисунок 23 – цикл перебора fgsm_eps для FC на MNIST при 0,9

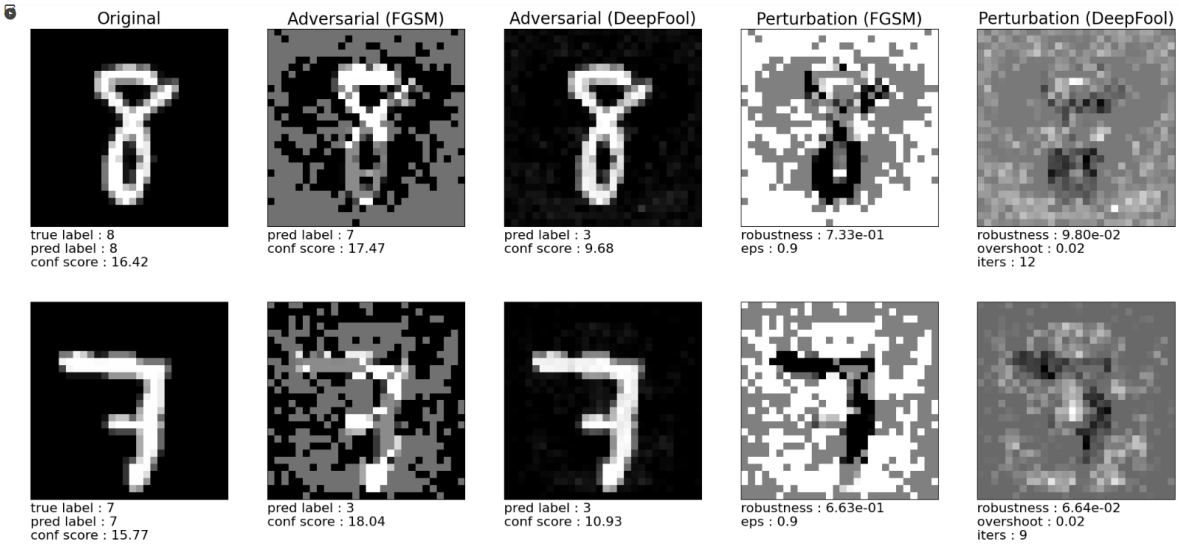


Рисунок 24 – Оценка атакующих примеров FC на MNIST при 0,9

```
#10
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_arg, 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()
```

Рисунок 25 – цикл перебора fgsm_eps для FC на MNIST при 10

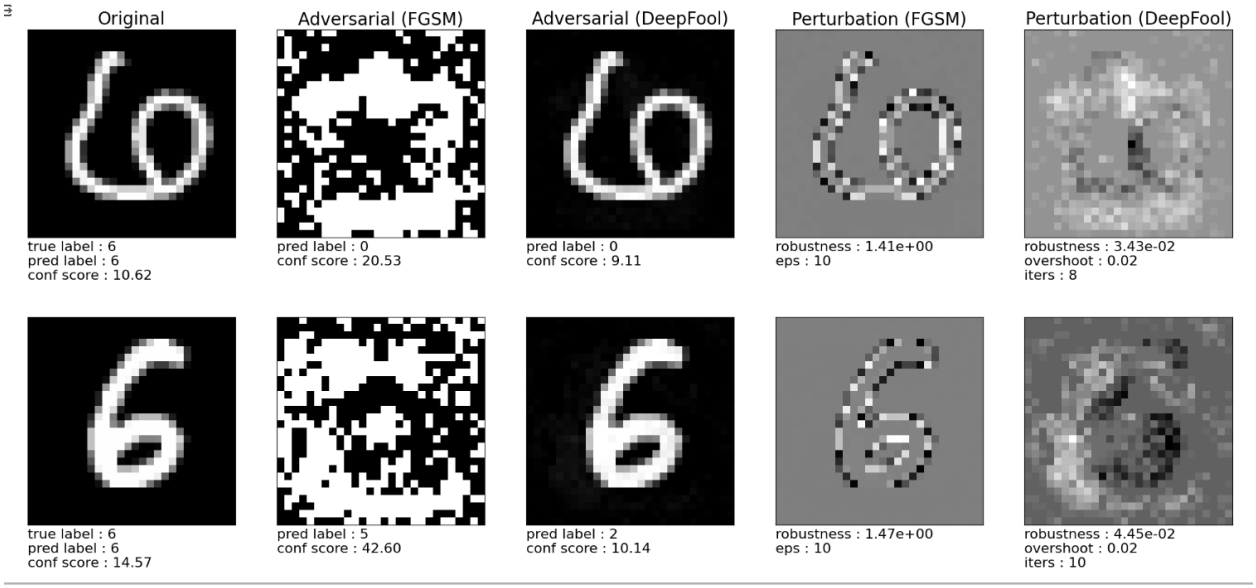


Рисунок 26 – Оценка атакующих примеров FC на MNIST при 10

Продолжим с LeNet на MNIST

оценка LeNet на 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_arg, 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()
```

Рисунок 27 – цикл перебора fgsm_eps для LeNet на MNIST при 0,001

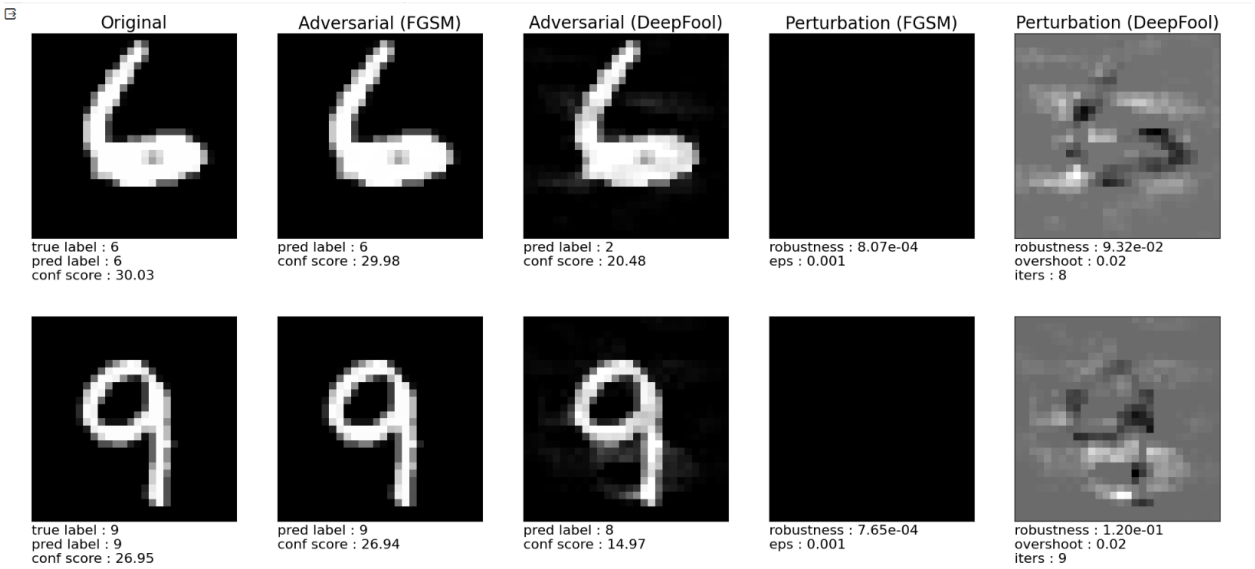


Рисунок 28 – Оценка атакующих примеров для LeNet на MNIST при 0,001

```
[22] 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_arg, 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()
```

Рисунок 29 – загрузка модели для LeNet на MNIST при 0,02

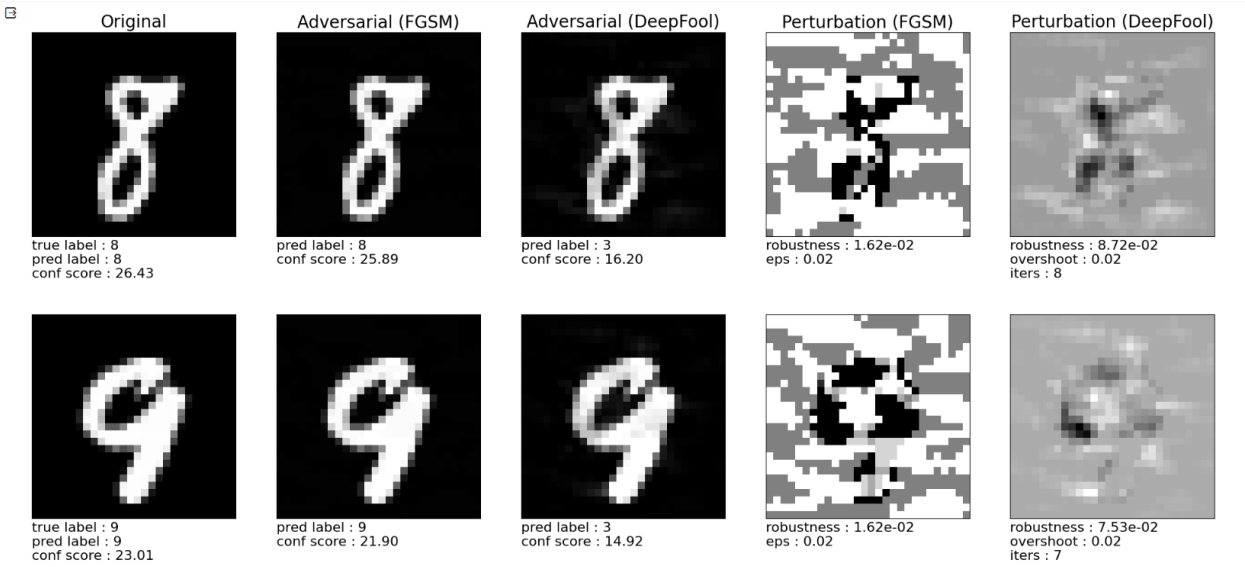


Рисунок 30 – Оценка атакующих примеров для LeNet на MNIST при 0,02

```
[23] 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_arg, 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()
```

Рисунок 31 – загрузка модели для LeNet на MNIST при 0,5

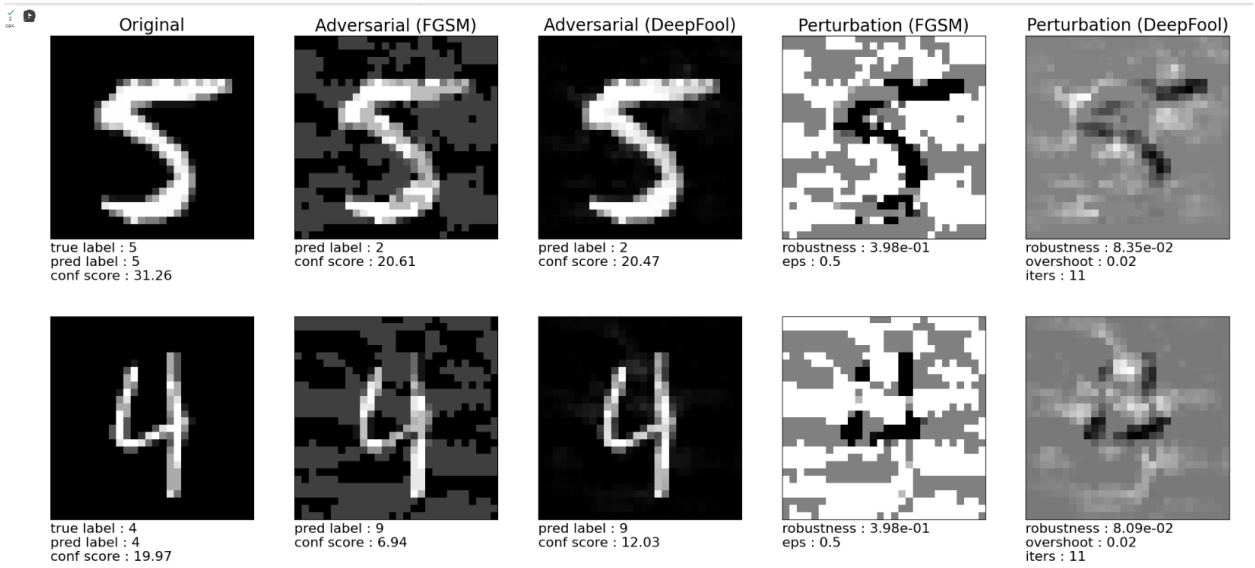


Рисунок 32 – Оценка атакующих примеров для LeNet на MNIST при 0,5

```
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_arg, 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()
```

Рисунок 33 – загрузка модели для LeNet на MNIST при 0,9

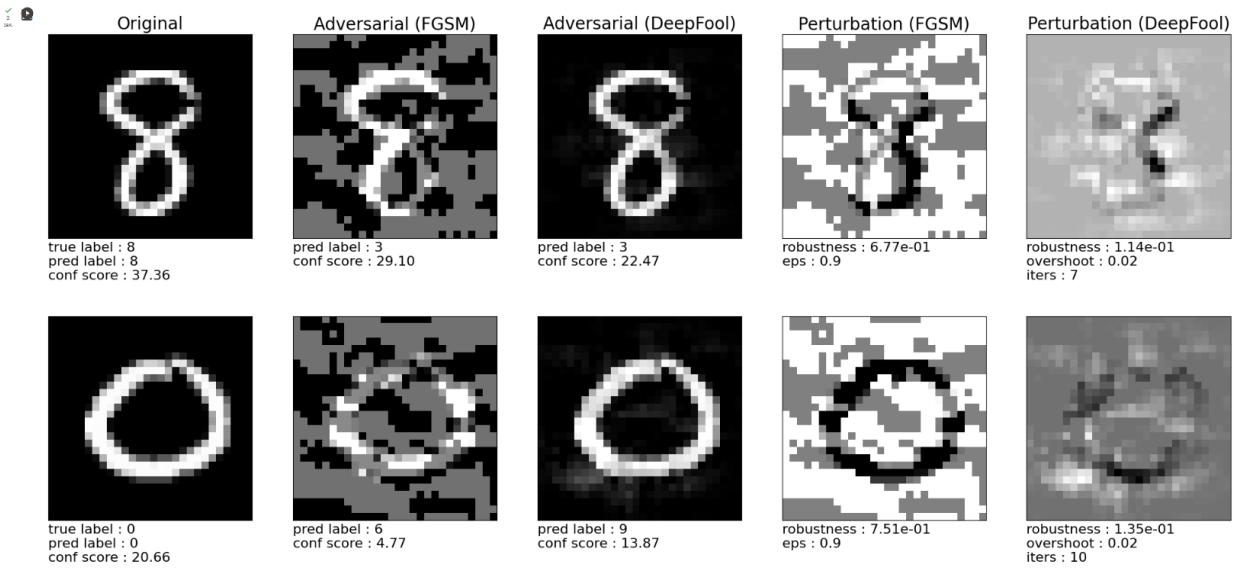


Рисунок 34 – Оценка атакующих примеров для LeNet на MNIST при 0,9

```
[25] 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_arg, 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()
```

Рисунок 35 – загрузка модели для LeNet на MNIST при 10

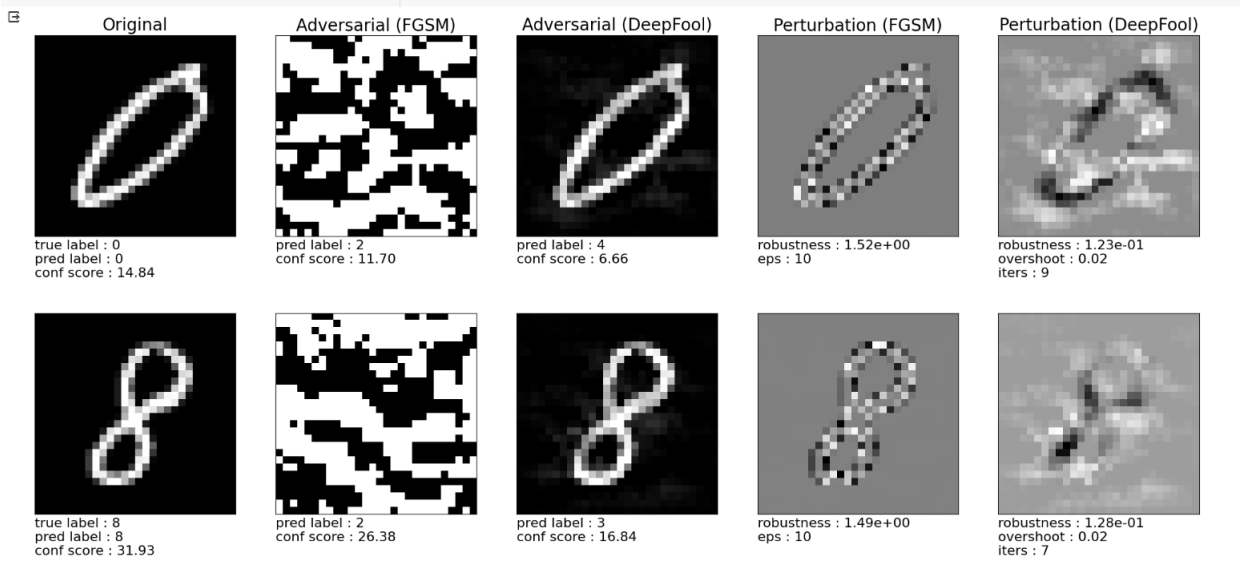


Рисунок 36 – Оценка атакующих примеров для LeNet на MNIST при 10

Оценка NetworkinNetwork на CIFAR-10

```
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_arg, 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()
```

Рисунок 37 – загрузка модели для NiN на CIFAR-10 при 0,001

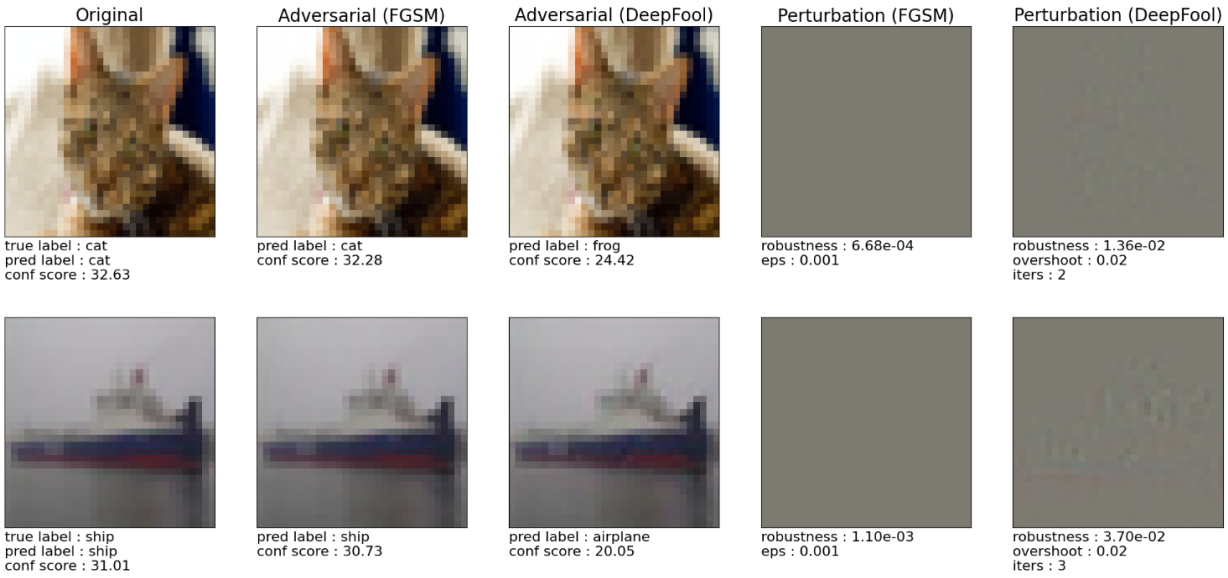


Рисунок 38 – Оценка атакующих примеров для NiN на CIFAR-10 при 0,001

```
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_arg, 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()
```

Рисунок 39 – загрузка модели для NiN на CIFAR-10 при 0,02

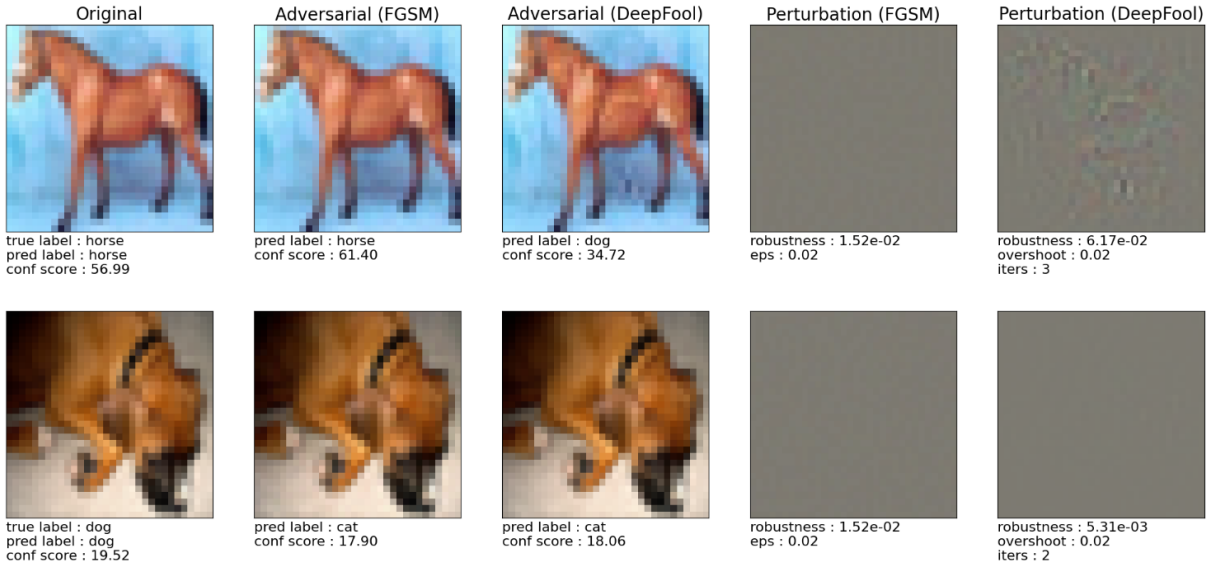


Рисунок 40 – Оценка атакующих примеров для NiN на CIFAR-10 при 0,02

```
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_arg, 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()
```

Рисунок 41 – загрузка модели для NiN на CIFAR-10 при 0,5

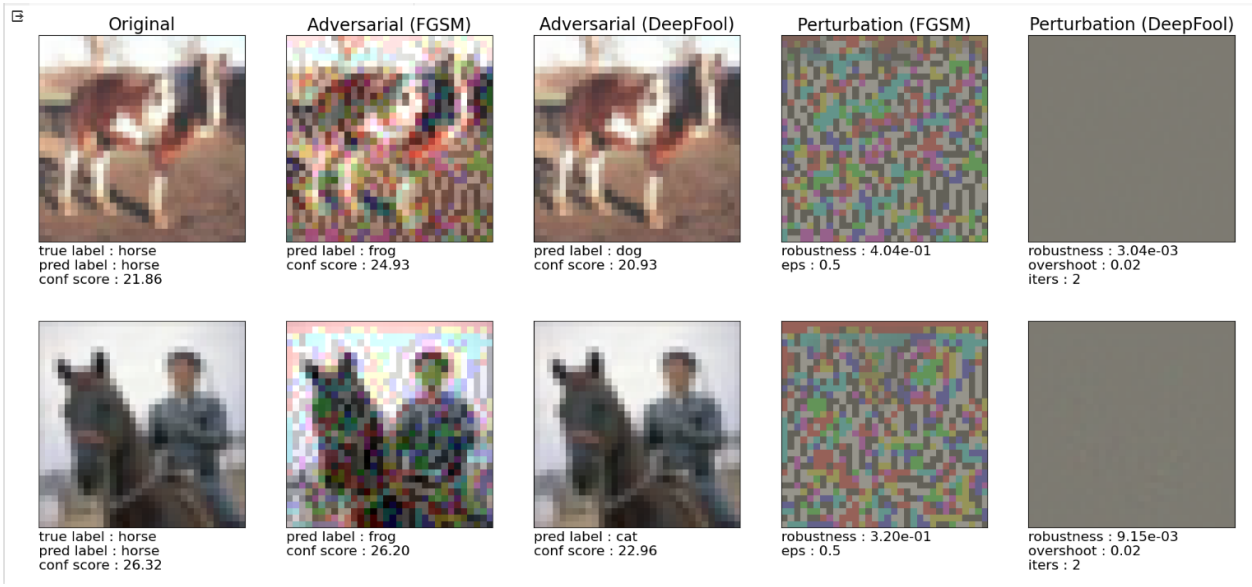


Рисунок 42 – Оценка атакующих примеров для для NiN на CIFAR-10 при 0,5

Рисунок 43 – загрузка модели для NiN на CIFAR-10 при 0,7

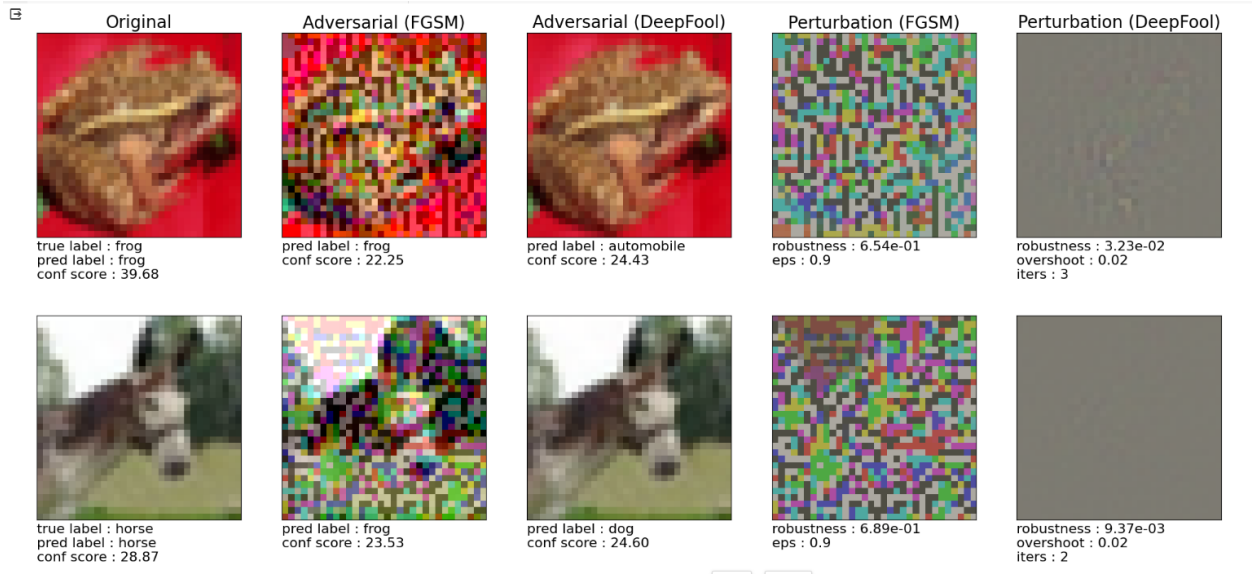


Рисунок 44 – Оценка атакующих примеров для LeNet на MNIST при 0,7

```
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_arg, 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()
```

Рисунок 45 – загрузка модели для LeNet на MNIST при 10

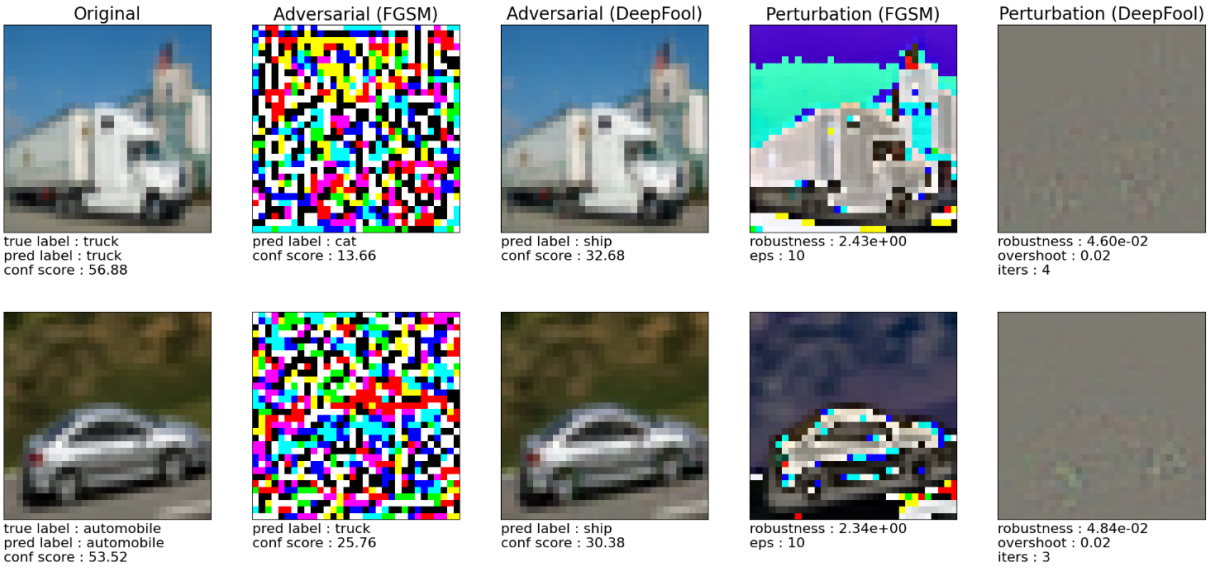


Рисунок 46 – Оценка атакующих примеров для LeNet на MNIST при 10

Оценка LeNet на датасет 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_arg, 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()
```

Рисунок 47 – загрузка модели для LeNet на CIFAR-10 при 0,001

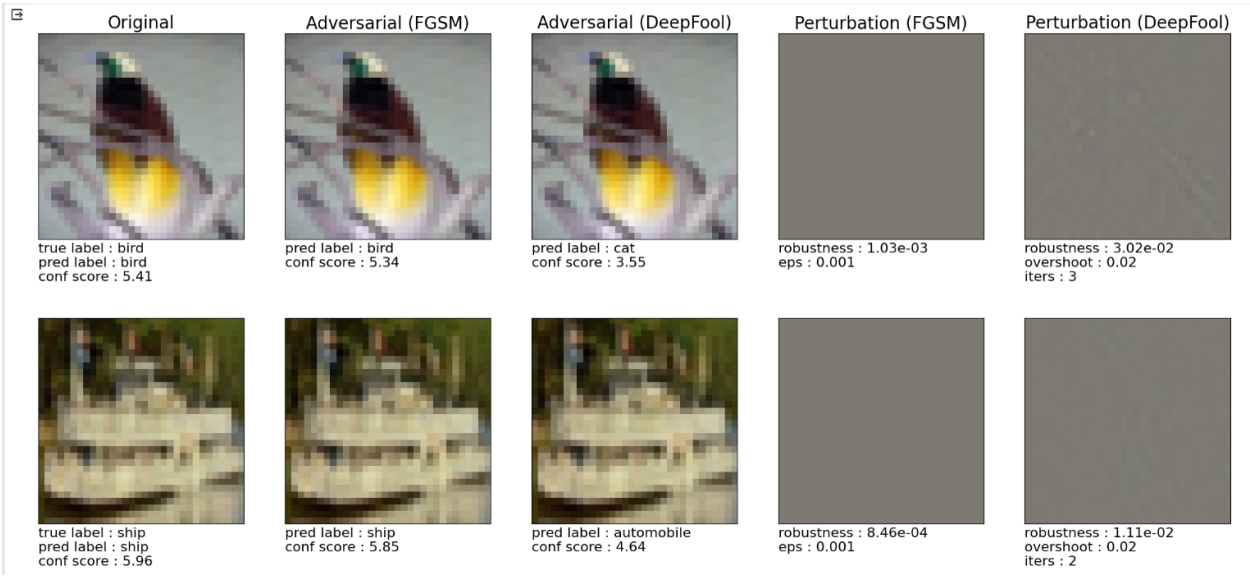


Рисунок 48 – Оценка атакующих примеров для LeNet на CIFAR-10 при 0,001

```
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_arg, 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()
```

Рисунок 49 – загрузка модели для LeNet на CIFAR-10 при 0,02

Рисунок 50 – Оценка атакующих примеров для LeNet на CIFAR-10 при 0,02

```
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_arg, 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()
```

Рисунок 51 – загрузка модели для LeNet на CIFAR-10 при 0,5

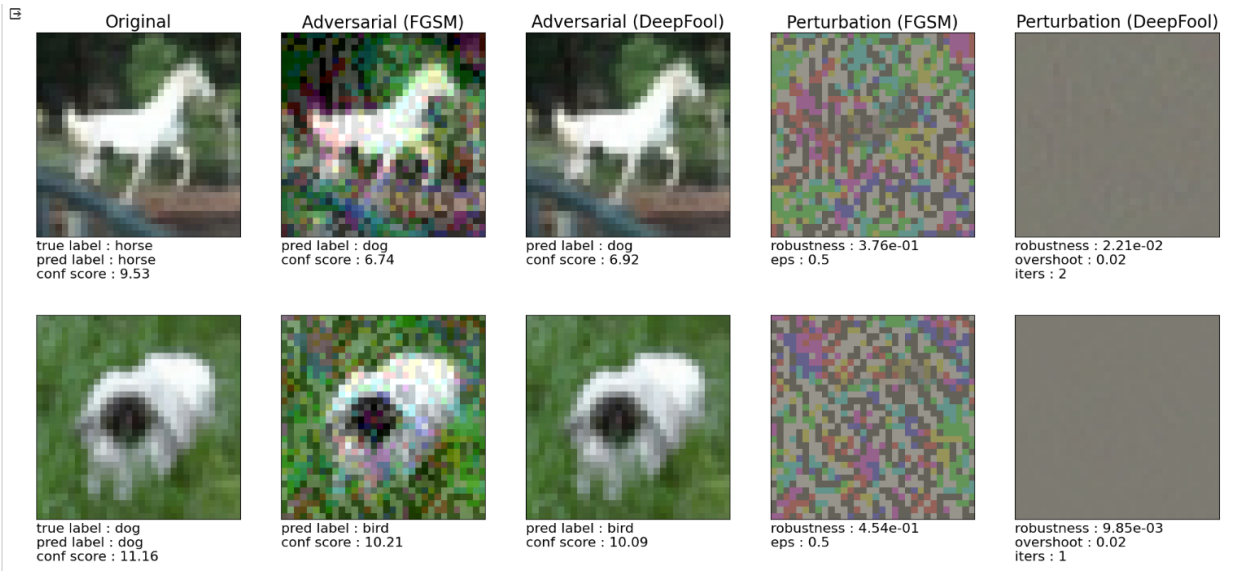


Рисунок 52 – Оценка атакующих примеров для LeNet на CIFAR-10 при 0,5

```
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_arg, 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()
```

Рисунок 53 – загрузка модели для LeNet на CIFAR-10 при 0,9

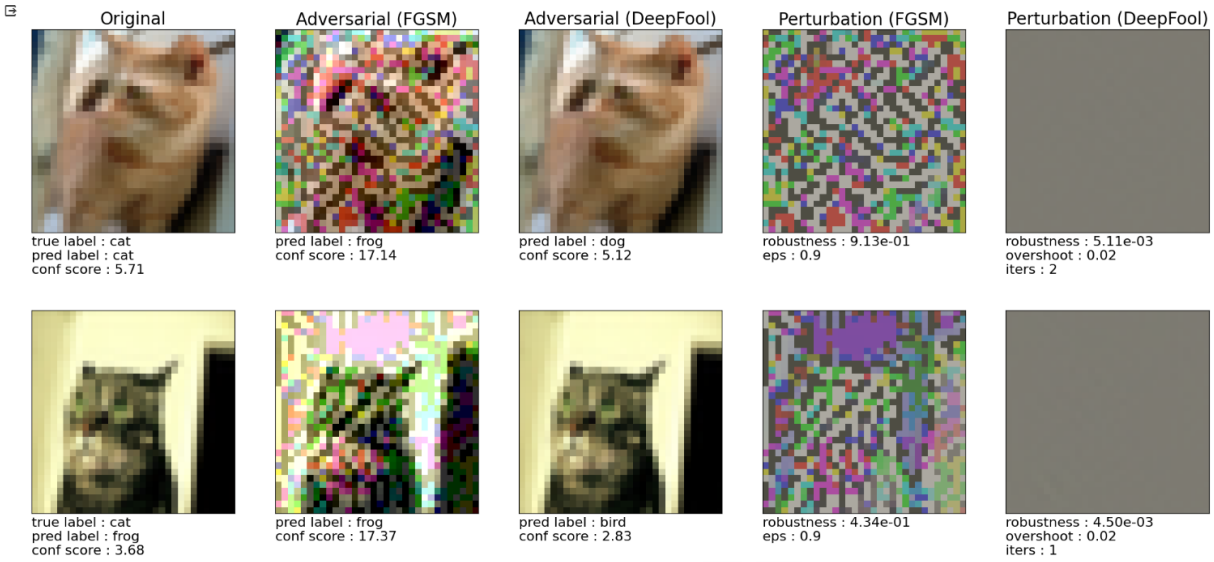


Рисунок 54 – Оценка атакующих примеров для LeNet на CIFAR-10 при 0,9

```
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_arg, 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()
```

Рисунок 55 – загрузка модели для LeNet на CIFAR-10 при 10

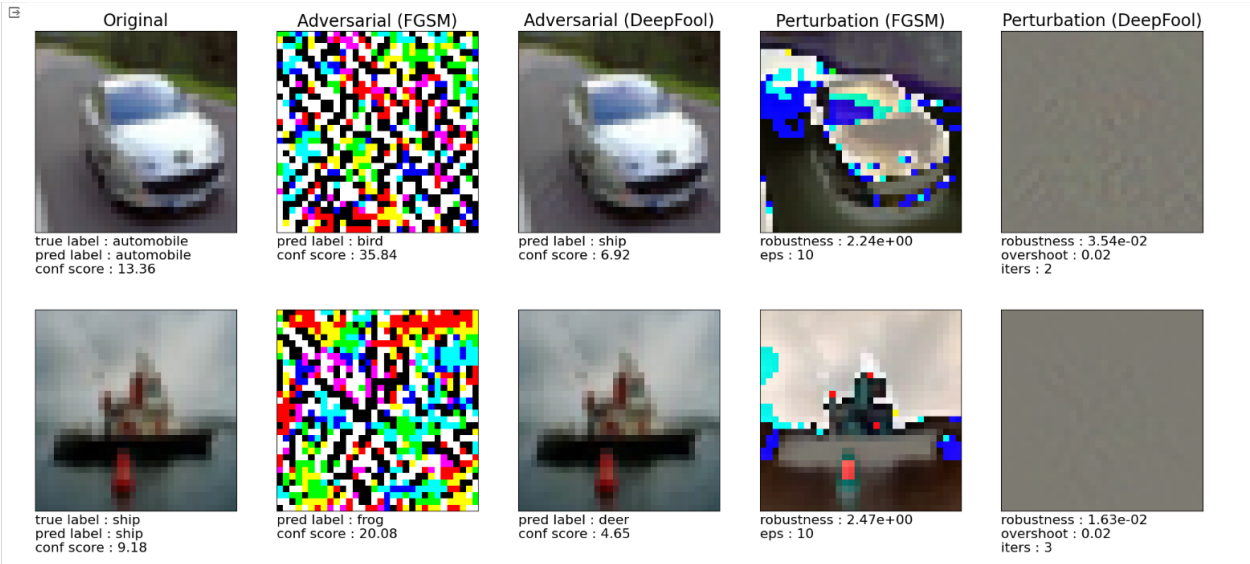


Рисунок 56 – Оценка атакующих примеров для LeNet на CIFAR-10 при 10

Вывод: По результатам эксперимента в ходе лабораторной работы по перебору значения eps была выявлена закономерность – увеличение значения eps способствует искажению изображения. Но исходя из нашего массива значений eps[0,001;0,2;0,5;0,9;10] можно сказать что значения 0.001 и 0.2 могут оказаться недостаточным для совершения ошибки.