



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

Институт кибербезопасности и цифровых технологий
КБ-4 «Интеллектуальные системы информационной безопасности»

Отчет по лабораторной работе №1
по дисциплине: «Анализ защищенности систем искусственного
интеллекта»

Выполнил:
Тимофеев И.О
Группа:
ББМО-01-22

Проверил:
К.т.н. Спирин Андрей Андреевич

Москва, 2023

Шаг 1. Скопировать проект по ссылке в локальную среду выполнения Jupyter (Google Colab) .

```
Шаг 1 Скопировать проект по ссылке в локальную среду выполнения Jupyter (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 | 25.39 MiB/s, done.
Resolving deltas: 100% (27/27), done.
```

Рисунок 1 – Загрузка проекта

Шаг 2. Сменить директорию исполнения на вновь созданную папку "EEL6812_DeepFool_Project" проекта.

```
Шаг 2 Сменить директорию исполнения на вновь созданную папку "EEL6812_DeepFool_Project" проекта.

[2] %cd EEL6812_DeepFool_Project/

/content/EEL6812_DeepFool_Project
```

Рисунок 2 – Смена директории

Шаг 3. Выполнить импорт библиотек.

```
Шаг 3 Выполнить 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
```

Рисунок 3 – Импорт библиотек

Шаг 4. Выполним импорт вспомогательных библиотек из локальных файлов проекта.

```
Шаг 4 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
```

Рисунок 4 – Импорт вспомогательных библиотек

Шаг 5. Установим случайное рандомное значение в виде переменной `rand_seed={"Порядковый номер ученика группы в Гугл-таблице"}`.

```
Шаг 5 Установим случайное рандомное значение в виде переменной rand_seed={"Порядковый номер ученика группы в Гугл-таблице"}.

[5] rand_seed = 28
```

Рисунок 5 – Установка значения для генератора случайных чисел

Шаг 6. Установим указанное значение для `np.random.seed` и `torch.manual_seed`.

```
Шаг 6 Установим указанное значение для np.random.seed и torch.manual_seed.

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

<torch._C.Generator at 0x794bc8199b10>
```

Рисунок 6 – Настройка генераторов случайных чисел

Шаг 7. Используем в качестве устройства видеокарту (Среды выполнения--> Сменить среду выполнения --> T4 GPU).

```
Шаг 7 Используем в качестве устройства видеокарту (Среды выполнения--> Сменить среду выполнения --> T4 GPU).

[7] import torch
    print(torch.cuda.is_available())

True

[8] print(torch.cuda.device_count())
    print(torch.cuda.current_device())

1
0

[9] import torch
    device = torch.device("cuda")
```

Рисунок 7 – Выбор среды выполнения

Шаг 8. Загрузим датасет MNIST с параметрами `mnist_mean = 0.5`, `mnist_std = 0.5`, `mnist_dim = 28`.

```
Шаг 8 Загрузим датасет MNIST с параметрами mnist_mean = 0.5, mnist_std = 0.5, mnist_dim = 28.

[10] 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(
mnist_tf_inv = transforms.Compose([transforms.Normalize(mean=0.0, std=np.divide(1.0, mnist_std)), transforms.Normalize(

     mnist_temp = datasets.MNIST(root='datasets/mnist', train=True, download=True, transform=mnist_tf_train)
     mnist_train, mnist_val = random_split(mnist_temp, [50000, 10000])
     mnist_test = datasets.MNIST(root='datasets/mnist', train=False, download=True, transform=mnist_tf)

Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz to datasets/mnist/MNIST/raw/train-images-idx3-ubyte.gz
100%|██████████| 9912422/9912422 [00:00<00:00, 118362318.42it/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, 114063741.83it/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, 36267148.75it/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, 17287231.19it/s]
Extracting datasets/mnist/MNIST/raw/t10k-labels-idx1-ubyte.gz to datasets/mnist/MNIST/raw
```

Рисунок 8 – Загрузка датасета MNIST

Шаг 9. Загрузим датасет CIFAR-10 с параметрами.

```
Шаг 9 Загрузим датасет CIFAR-10 с параметрами.

[11] cifar_mean = [0.491, 0.482, 0.447]
     cifar_std = [0.202, 0.199, 0.201]
     cifar_dim = 32

     cifar_min, cifar_max = get_clip_bounds(cifar_mean, cifar_std, cifar_dim)
     cifar_min = cifar_min.to(device)
     cifar_max = cifar_max.to(device)

     cifar_tf = transforms.Compose([transforms.ToTensor(), transforms.Normalize(mean=cifar_mean, std=cifar_std)])
     cifar_tf_train = transforms.Compose([transforms.RandomCrop(size=cifar_dim, padding=4), transforms.RandomHorizontalFlip()])
     cifar_tf_inv = transforms.Compose([transforms.Normalize(mean=[0.0, 0.0, 0.0], std=np.divide(1.0, cifar_std)), transforms.ToTensor()])

     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, 42752644.07it/s]
Extracting datasets/cifar-10/cifar-10-python.tar.gz to datasets/cifar-10
Files already downloaded and verified
```

Рисунок 9 – Загрузка датасета CIFAR-10

Шаг 10. Выполним настройку и загрузку DataLoader.

```
Шаг 10 Выполним настройку и загрузку DataLoader.

[12] 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
warnings.warn(_create_warning_msg(
```

Рисунок 10 – Настройка и загрузка DataLoader

Шаг 11. Загрузим и оценим стойкость модели Network-In-Network Model к FGSM и DeepFool атакам на основе датасета CIFAR-10.

```
Шаг 11 Загрузим и оценим стойкость модели Network-In-Network Model к FGSM и DeepFool атакам на основе датасета CIFAR-10.

[13] 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,
     print('')
     batch = 64
     num_classes = 10
     overshoot = 0.02
     max_iter = 50
     deep_args = [batch, num_classes, overshoot, max_iter]

     evaluate_attack('cifar_nin_deepfool.csv', 'results', device, model, cifar_loader_test, cifar_min, cifar_max, deep_i
     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
```

Рисунок 11 – Загрузка и оценка стойкости модели к атакам

Шаг 12. Загрузим и оценим стойкость модели LeNet к FGSM и DeepFool атакам на основе датасета CIFAR-10.

Шаг 12 Загрузим и оценим стойкость модели LeNet к FGSM и DeepFool атакам на основе датасета CIFAR-10.

```
[14] 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 12 – Загрузка и оценка стойкости модели к атакам

Шаг 13. Выполним оценку атакующих примеров для сетей.

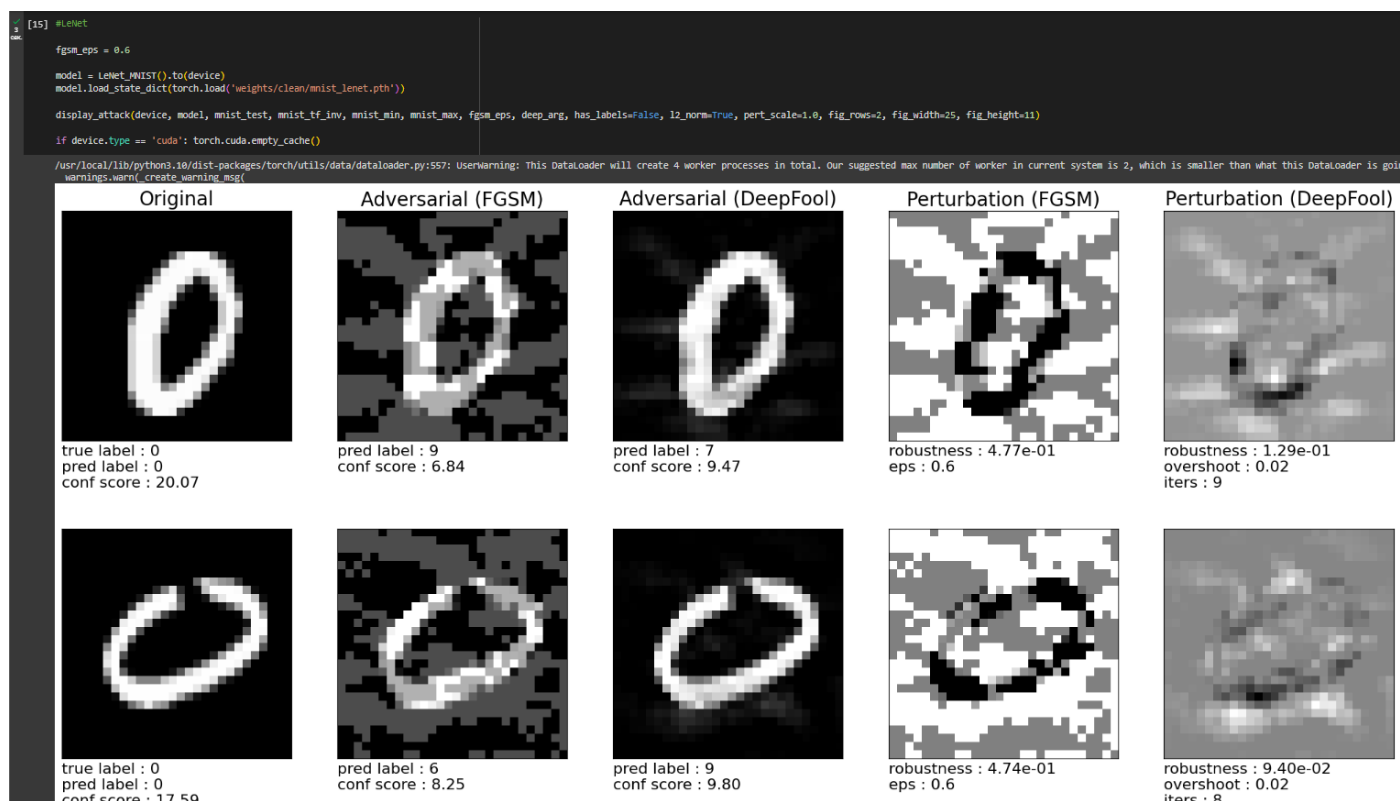


Рисунок 13 – Загрузка модели LeNet и оценка атакующих примеров(MNIST).

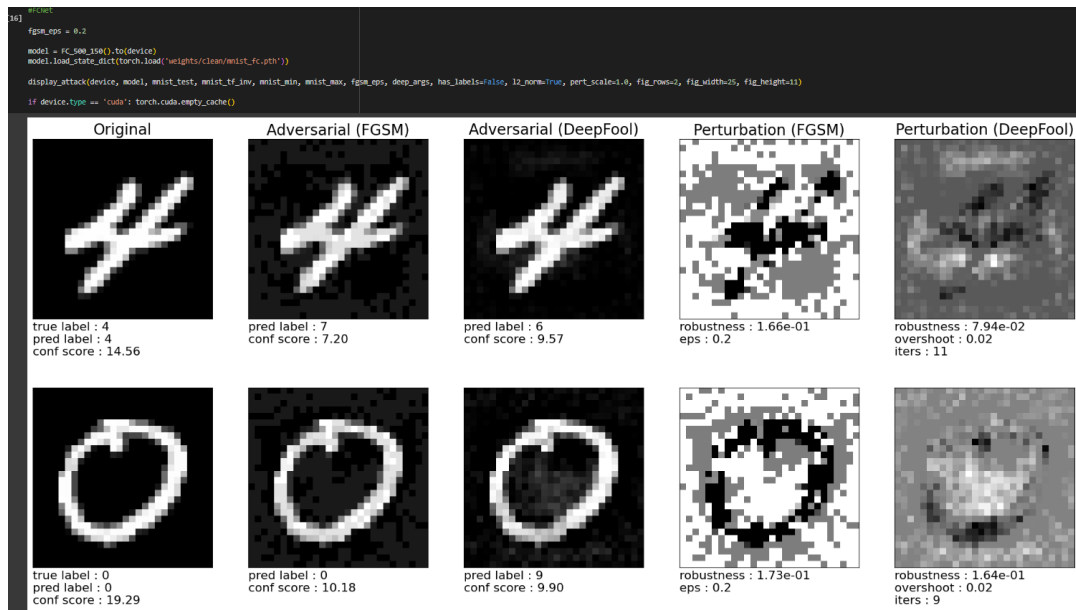


Рисунок 14 – Загрузка модели FC и оценка атакующих примеров(MNIST).

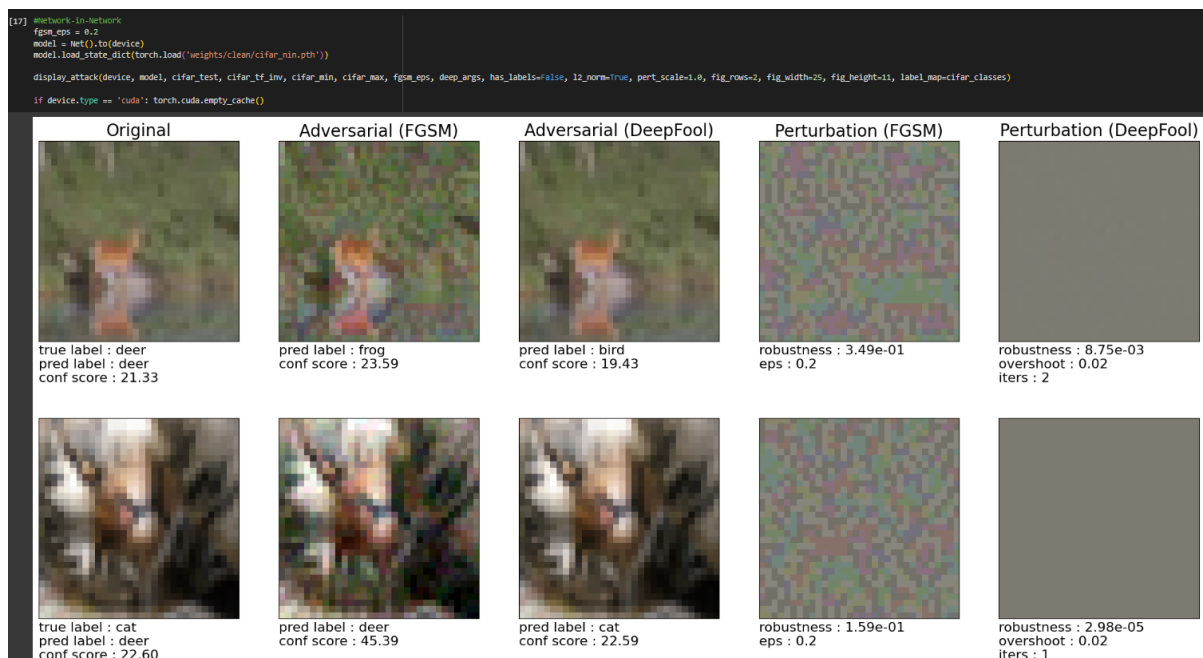


Рисунок 15 – Загрузка модели NiN оценка атакующих примеров (CIFAR-10).

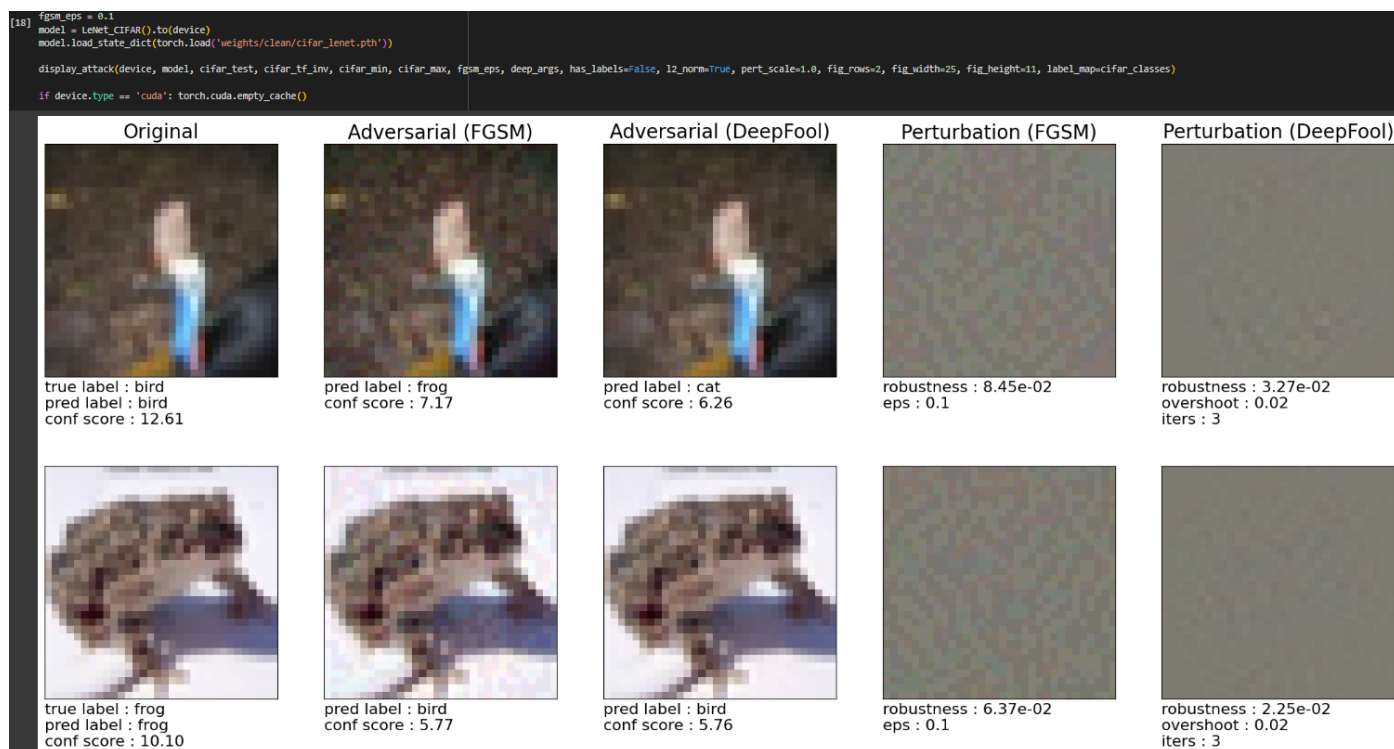


Рисунок 16 – Загрузка модели LeNet оценка атакующих примеров (CIFAR-10)

Шаг 14. Попробуем отразить отличия для $fgsm_eps = (0.001, 0.02, 0.5, 0.9, 10)$ и выявить закономерность/обнаружить отсутствие влияние параметра eps для сетей FC LeNet на датасете MNIST, NiN LeNet на датасете CIFAR).

Начнём с оценки FC на MNIST.

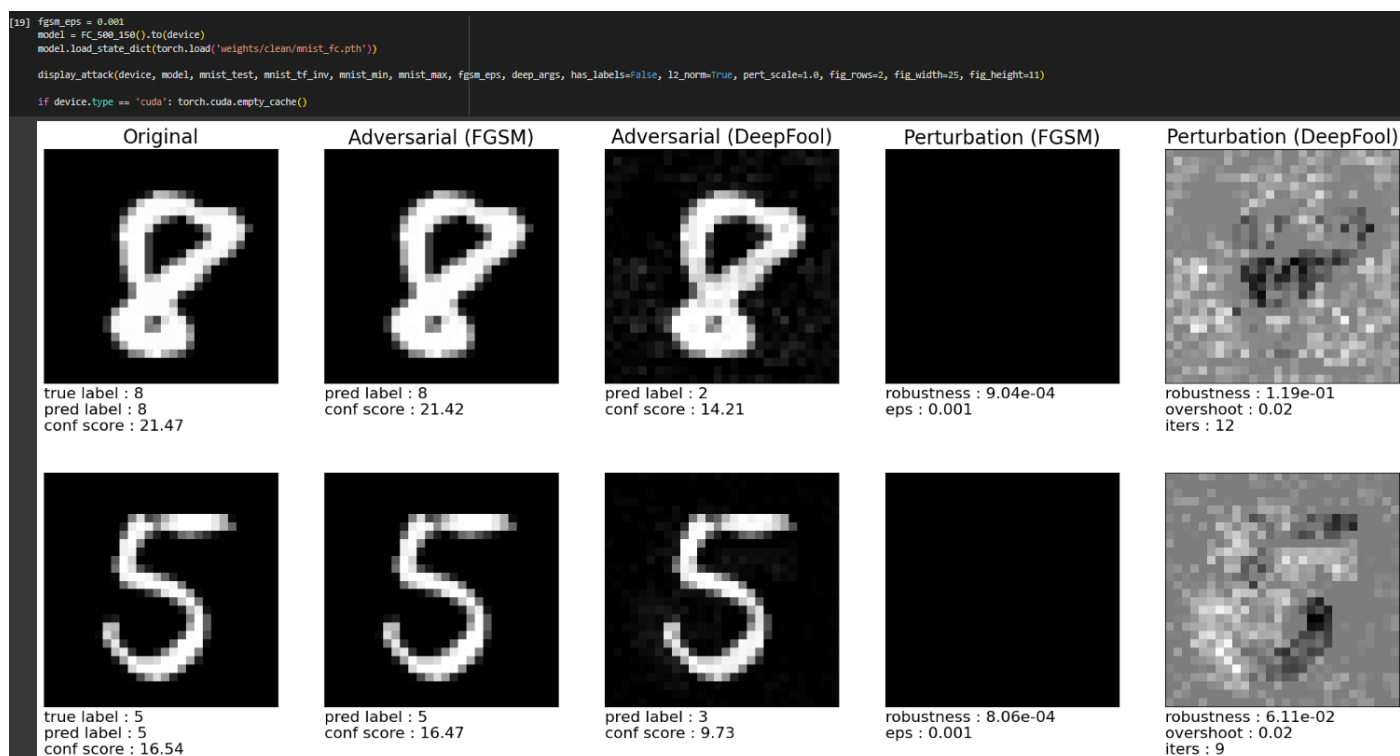


Рисунок 17 – Оценка атакующих примеров для FC на MNIST при eps равном 0.001

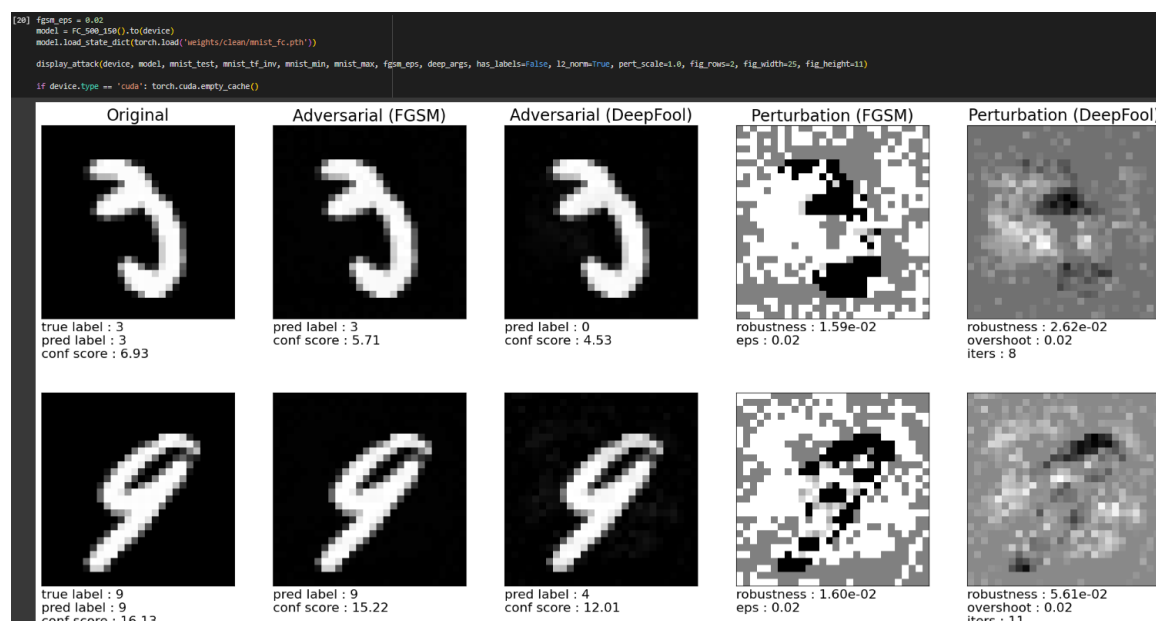


Рисунок 18 – Оценка атакующих примеров для FC на MNIST при eps равном 0.02

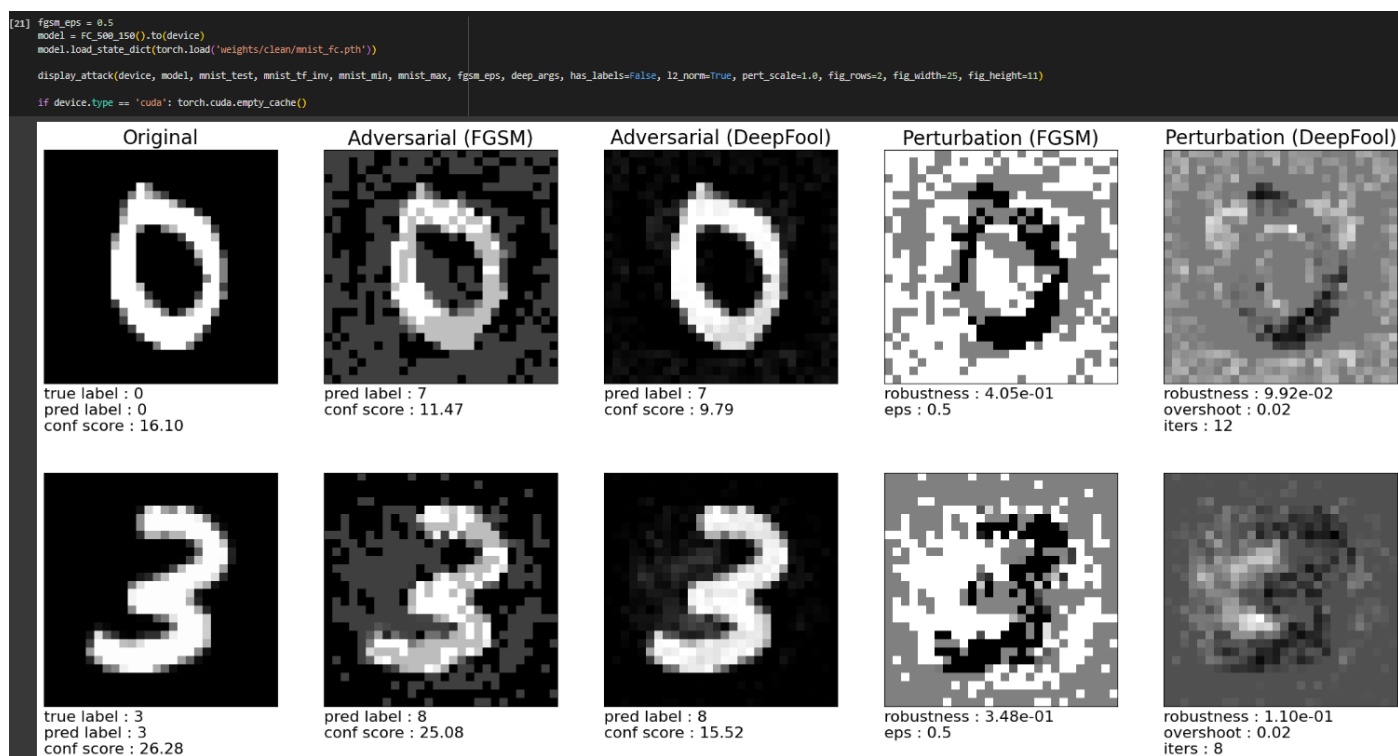


Рисунок 19 – Оценка атакующих примеров для FC на MNIST при eps равном 0.5



Рисунок 20 – Оценка атакующих примеров для FC на MNIST при eps равном 0.9

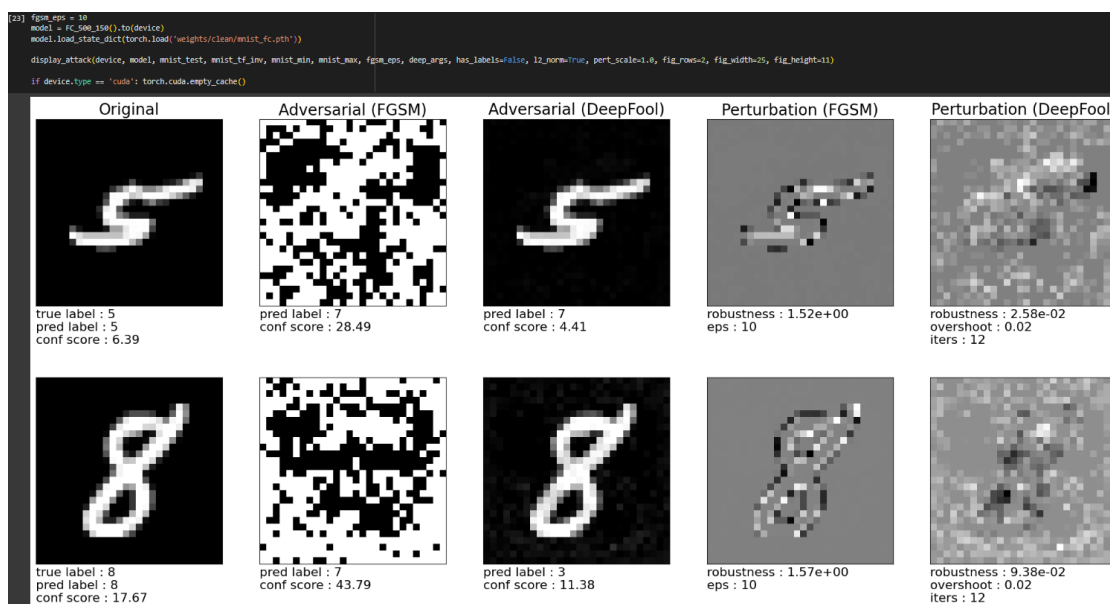


Рисунок 21 – Оценка атакующих примеров для FC на MNIST при eps равном 10

Продолжим с оценки LeNet на MNIST.



Рисунок 22 – Оценка атакующих примеров для LeNet на MNIST при eps равном 0.001

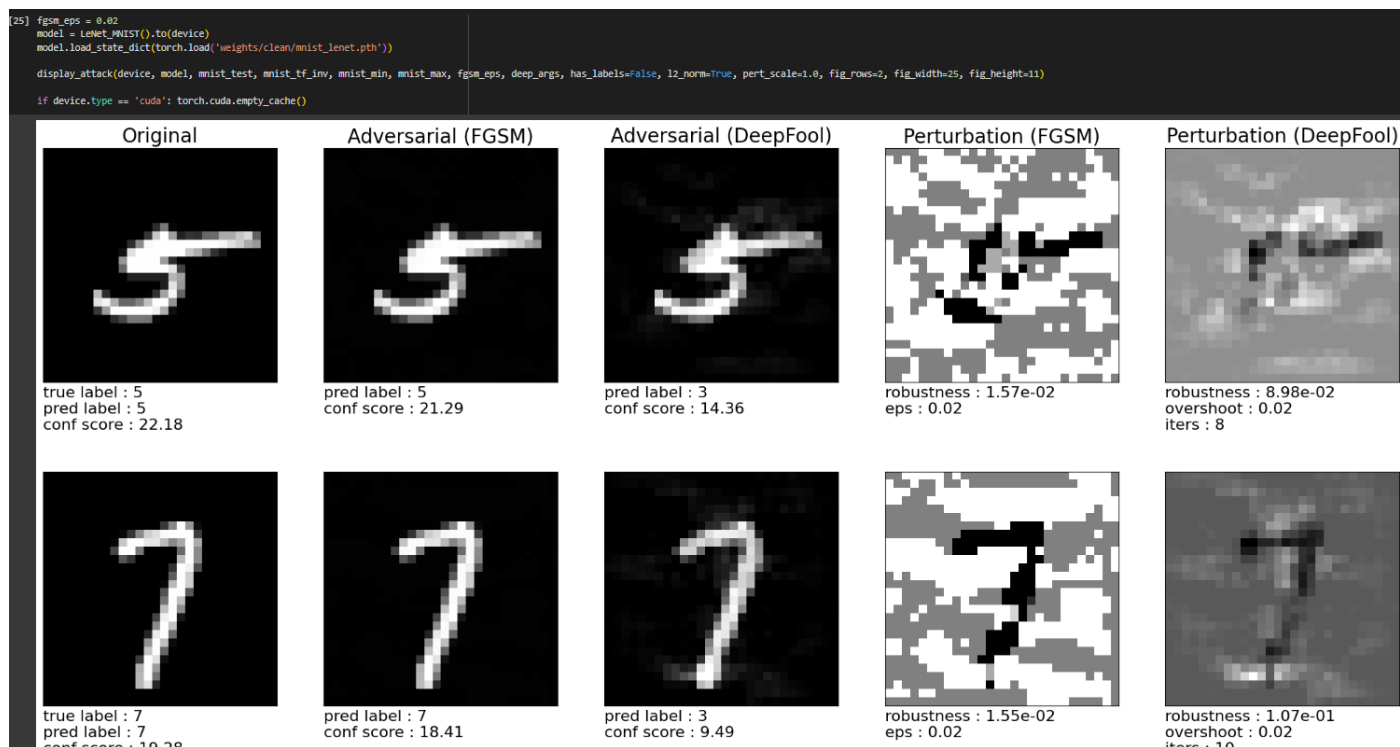


Рисунок 23 – Оценка атакующих примеров для LeNet на MNIST при eps равном 0.02

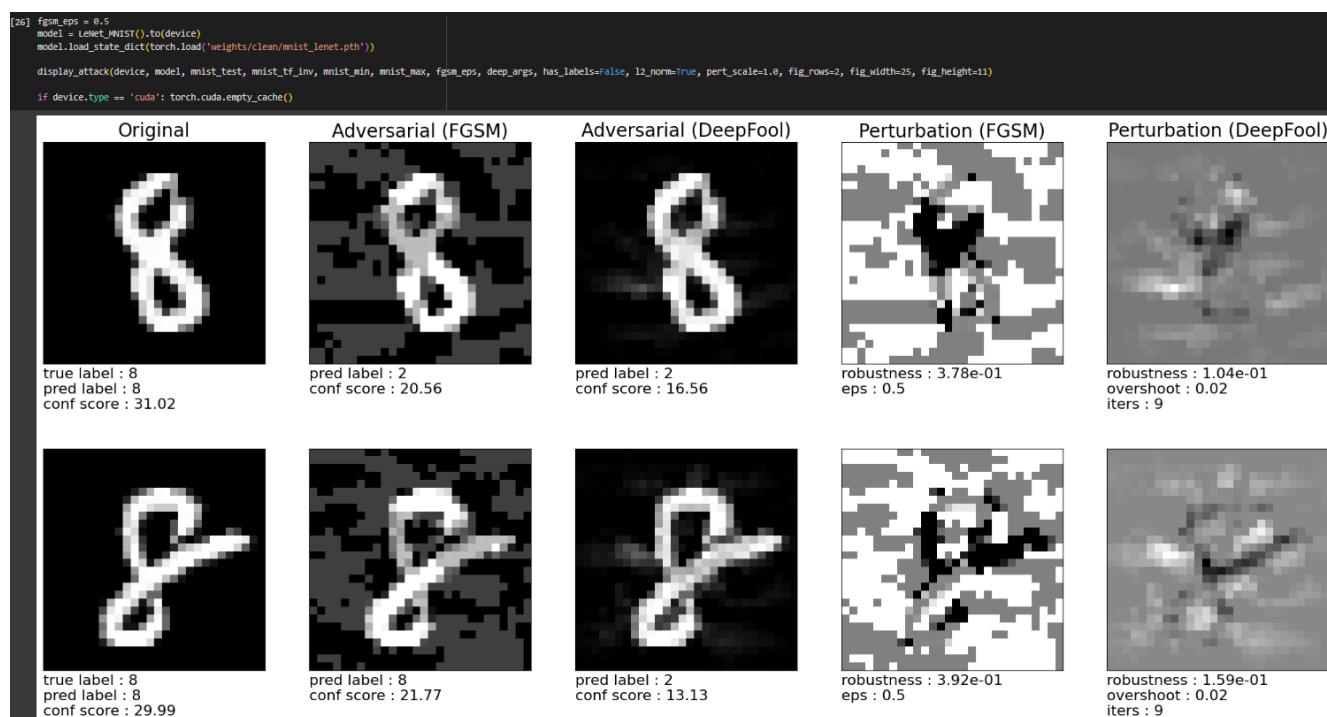


Рисунок 24 – Оценка атакующих примеров для LeNet на MNIST при eps равном 0.5



Рисунок 25 – Оценка атакующих примеров для LeNet на MNIST при eps равном 0.9

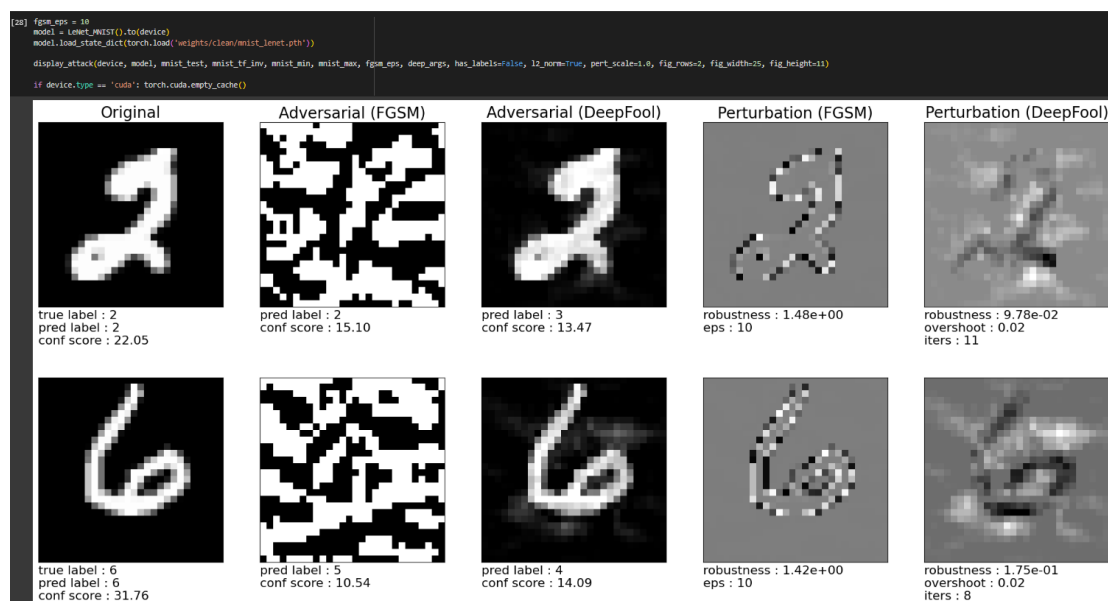


Рисунок 26 – Оценка атакующих примеров для LeNet на MNIST при eps равном 10

Продолжим с оценки NiN на CIFAR-10.

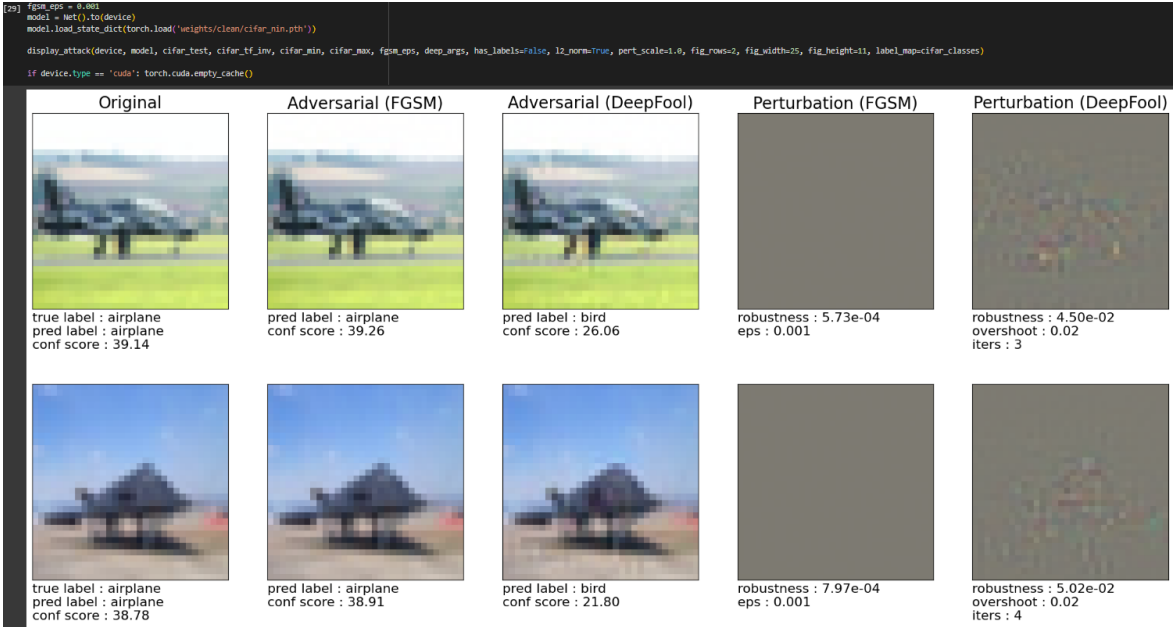


Рисунок 27 – Оценка атакующих примеров для NiN на CIFAR-10 при eps равном 0.001

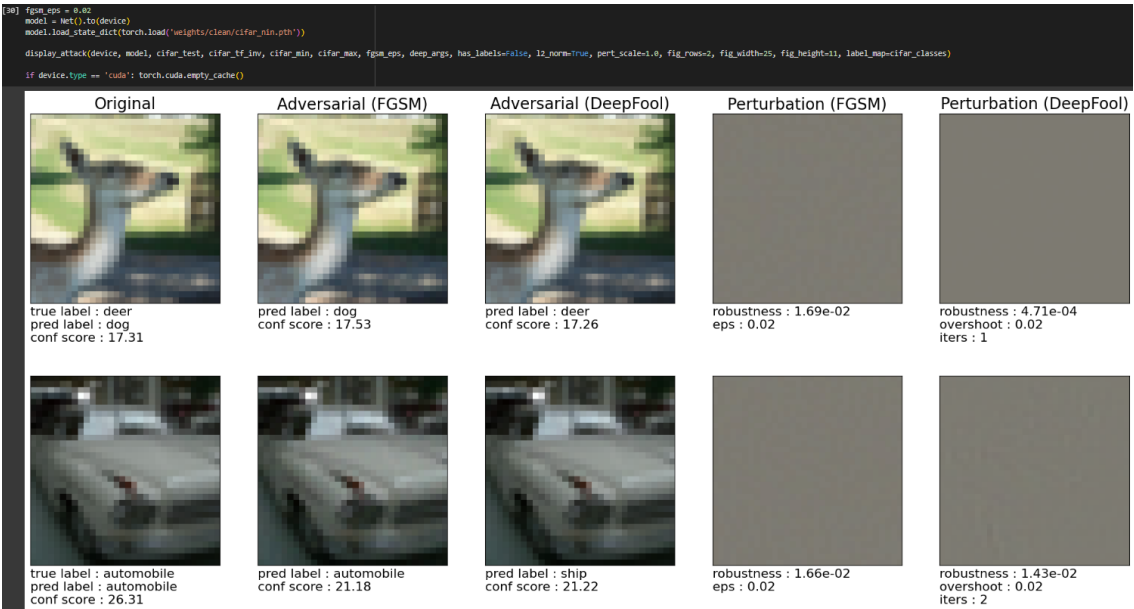


Рисунок 28 – Оценка атакующих примеров для NiN на CIFAR-10 при eps равном 0.02

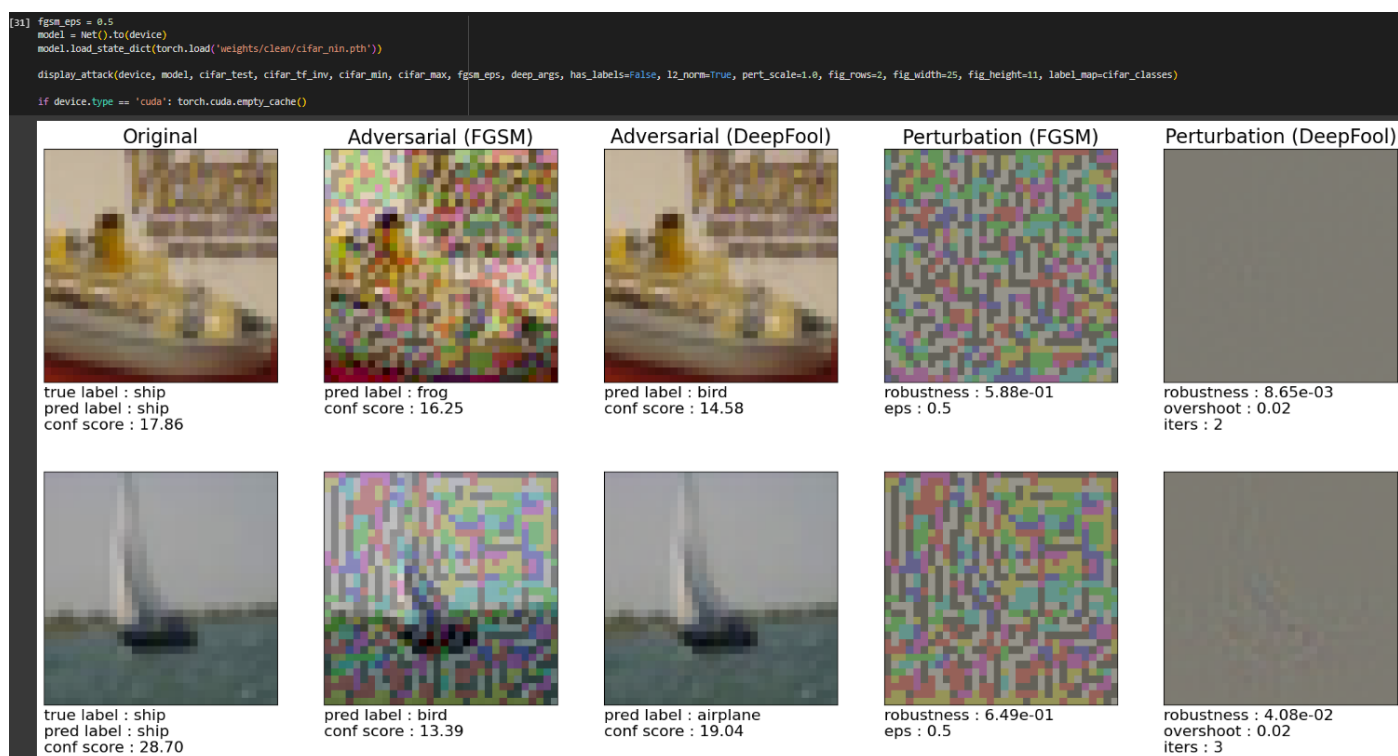


Рисунок 29 – Оценка атакующих примеров для NiN на CIFAR-10 при eps равном 0.5

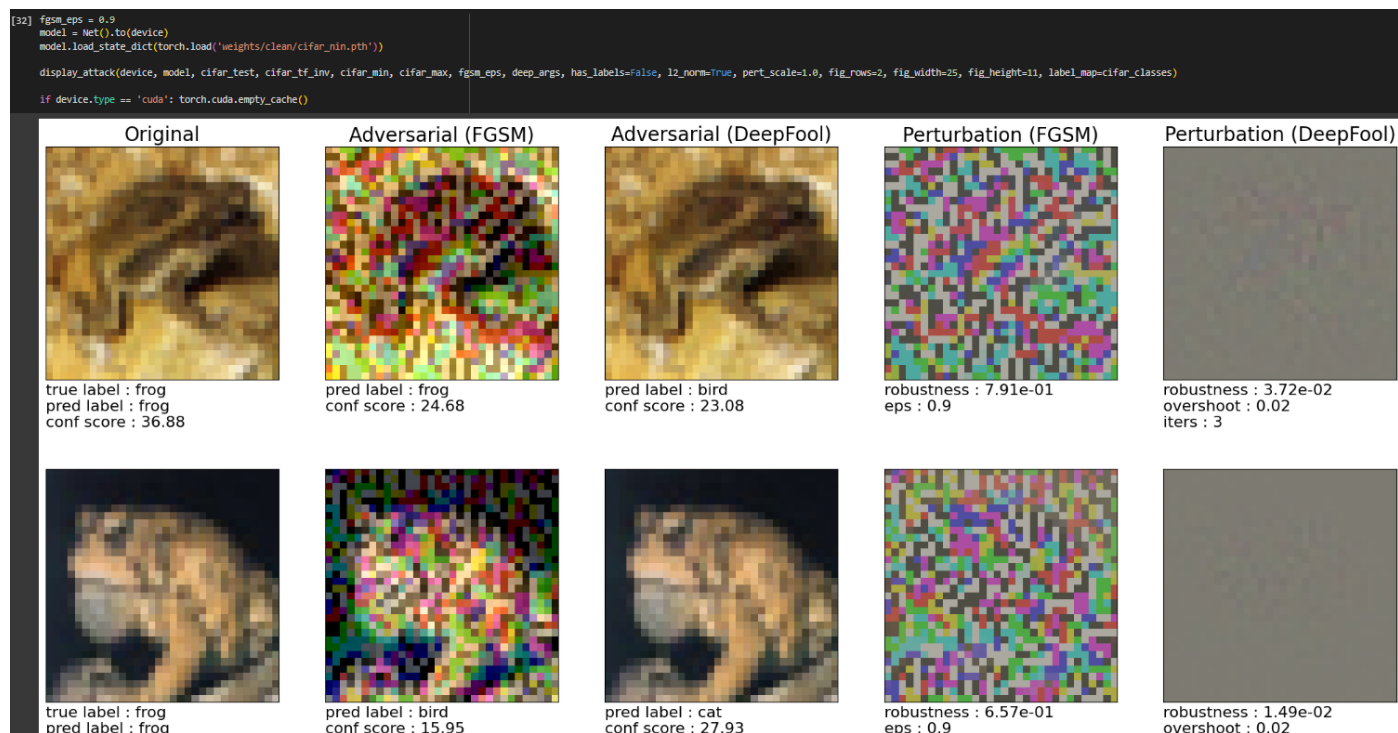


Рисунок 30 – Оценка атакующих примеров для NiN на CIFAR-10 при eps равном 0.9

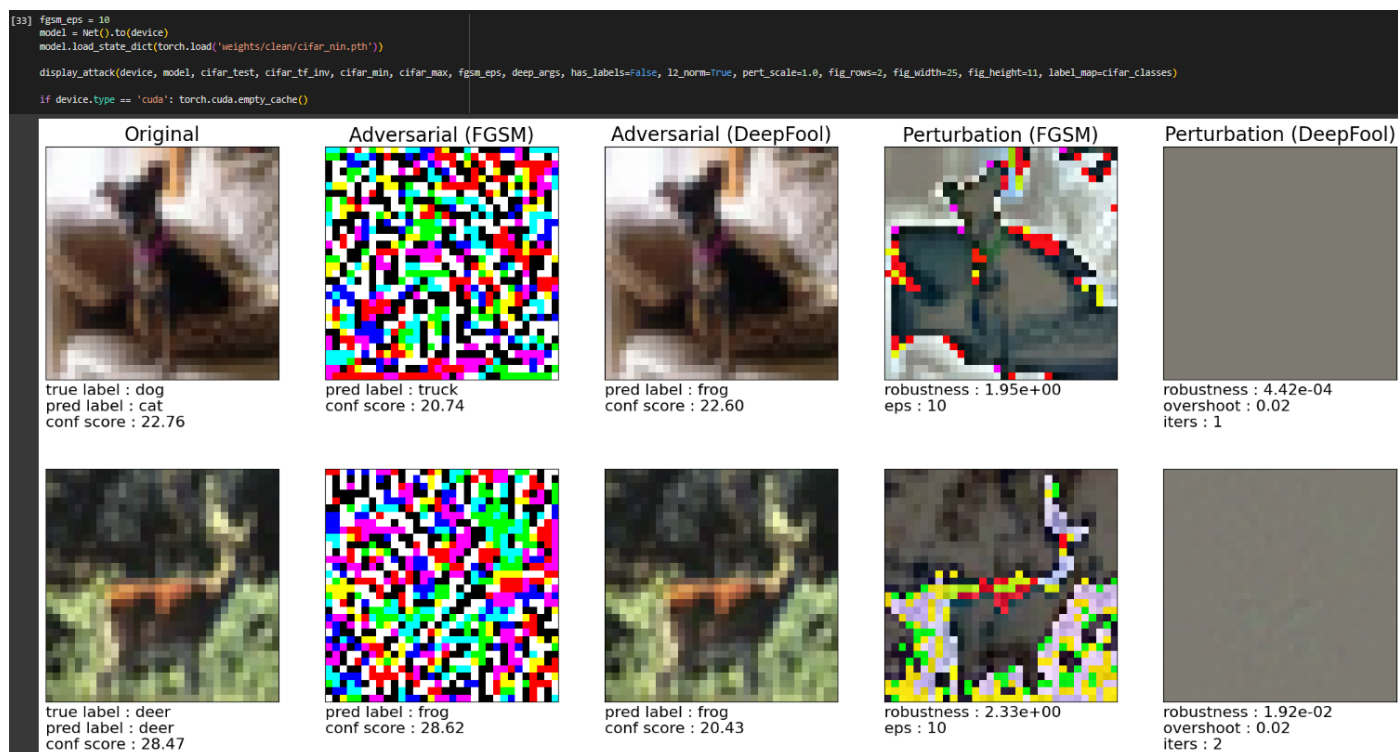


Рисунок 31 – Оценка атакующих примеров для NiN на CIFAR-10 при eps равном 10

Продолжим с оценки LeNet на CIFAR-10.

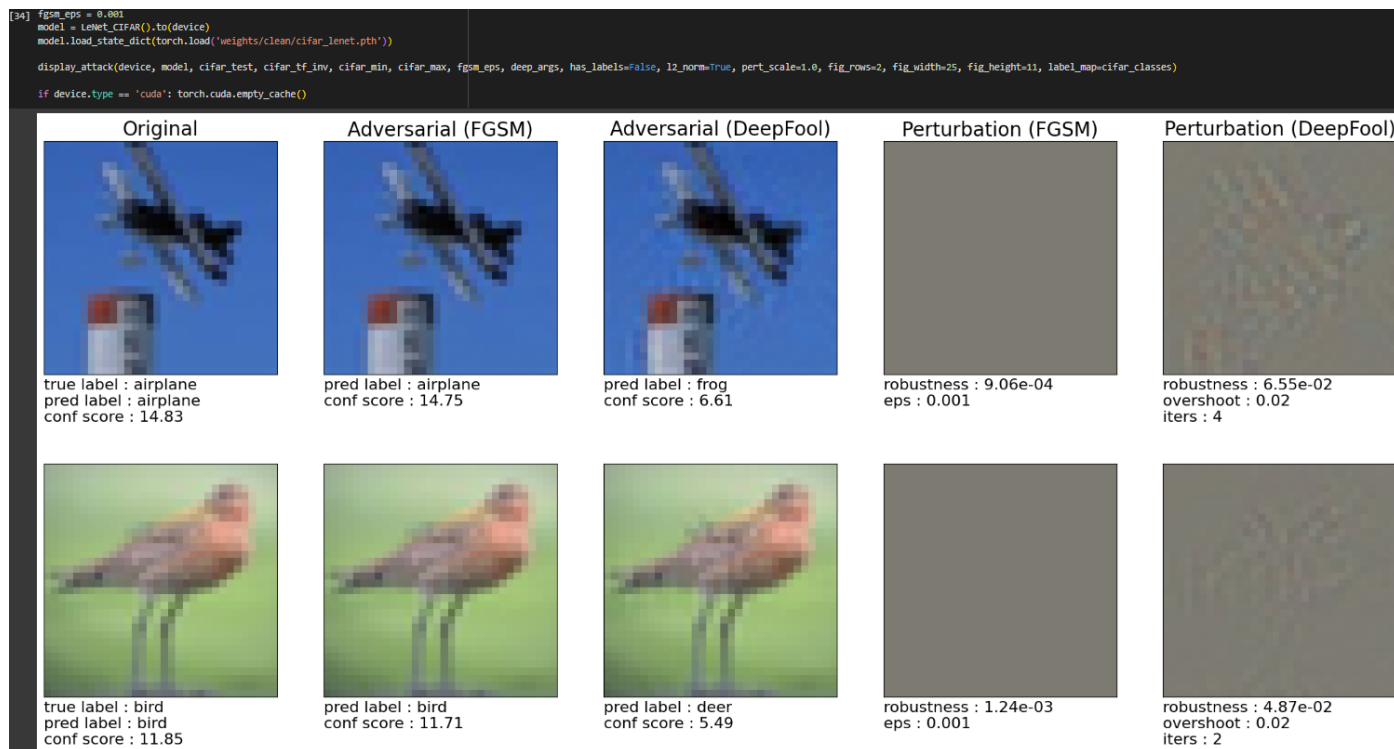


Рисунок 32 – Оценка атакующих примеров для LeNet на CIFAR-10 при ϵ ps равном 0.001

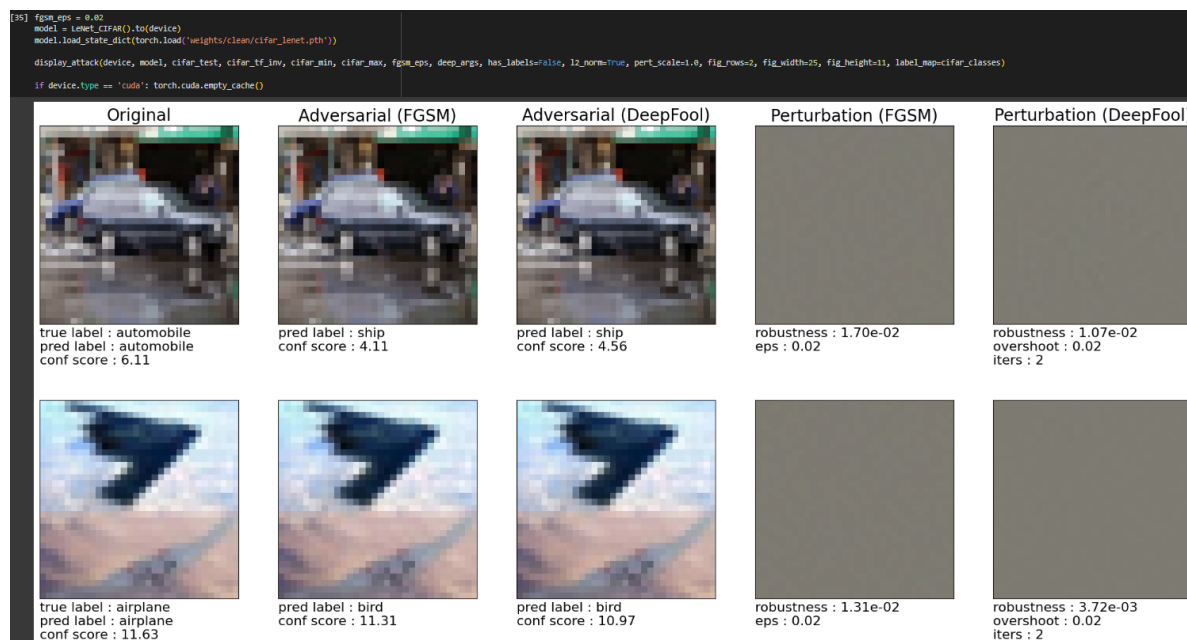


Рисунок 33 – Оценка атакующих примеров для LeNet на CIFAR-10 при ϵ ps равном 0.02

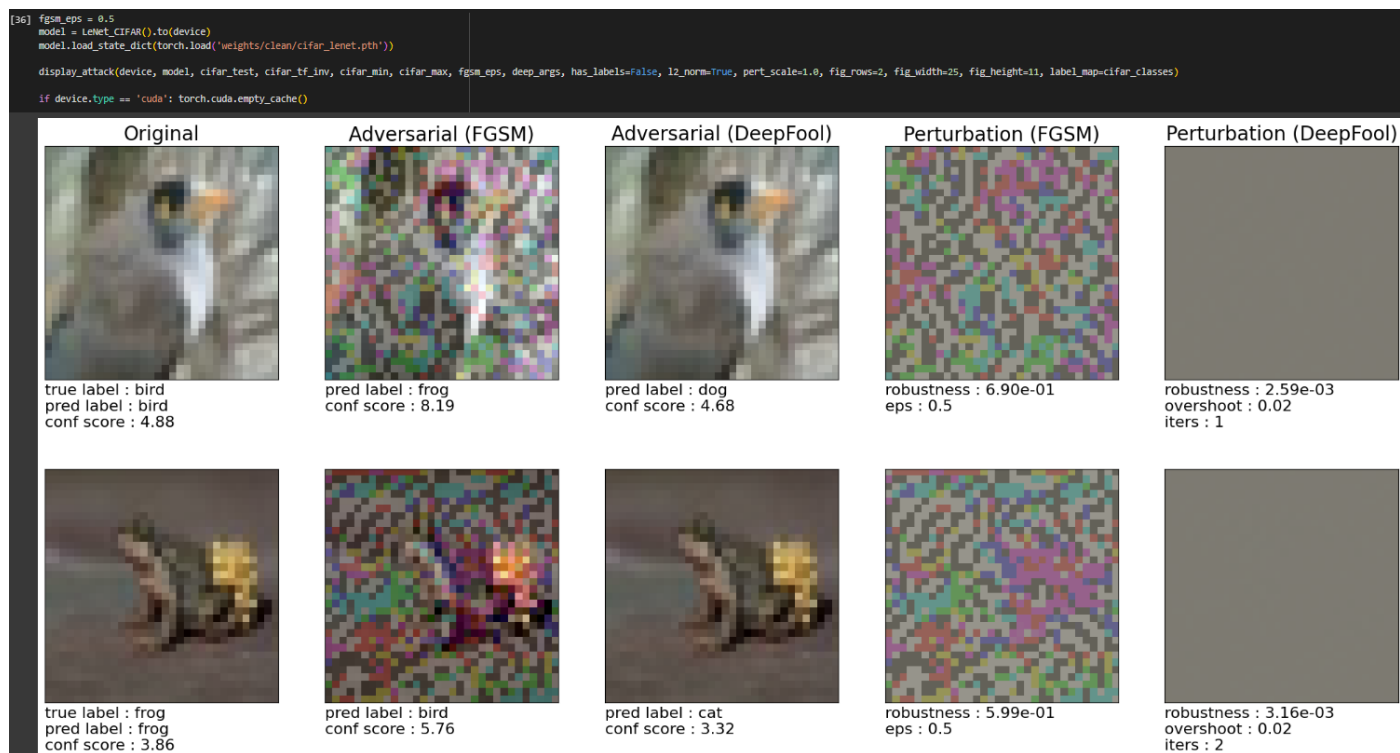


Рисунок 34 – Оценка атакующих примеров для LeNet на CIFAR-10 при ϵ равном 0.5

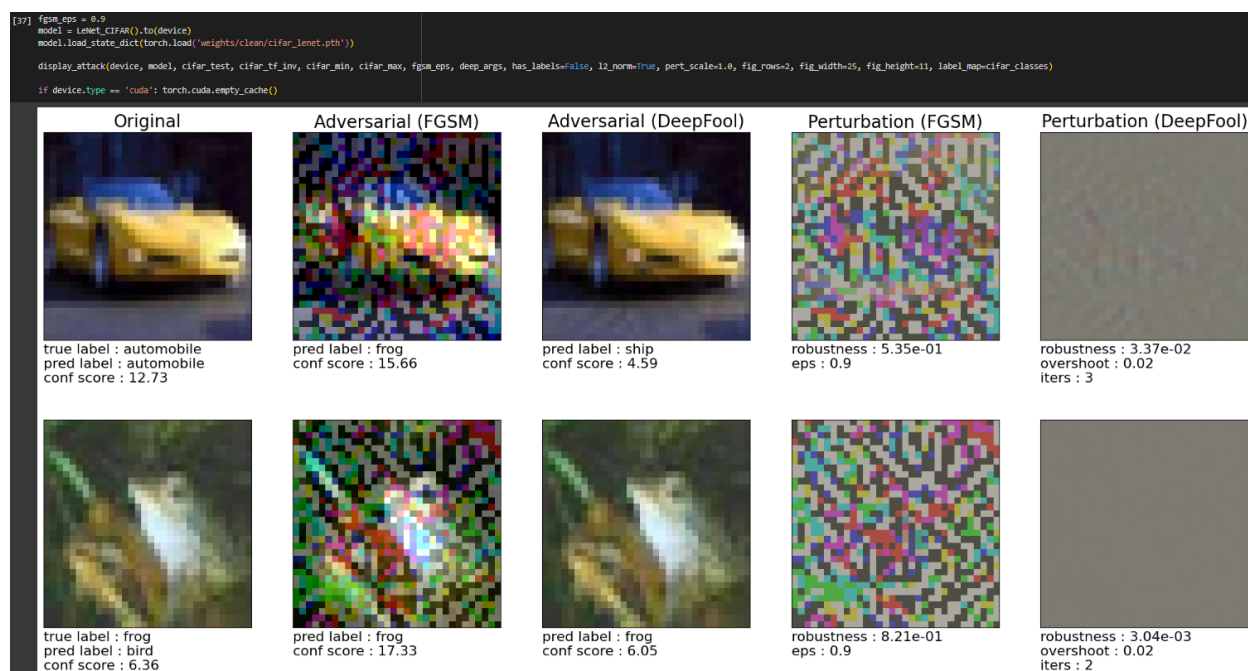


Рисунок 35 – Оценка атакующих примеров для LeNet на CIFAR-10 при ϵ равном 0.9

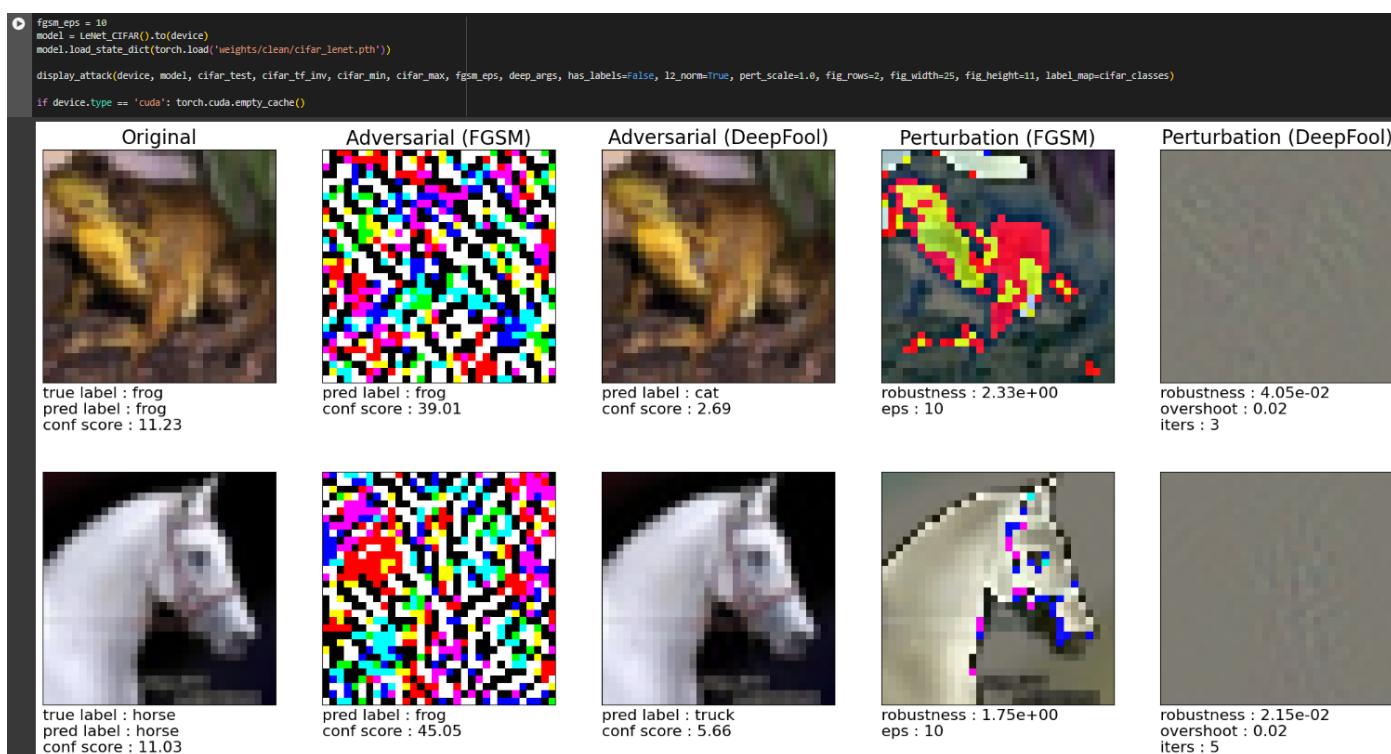


Рисунок 36 – Оценка атакующих примеров для LeNet на CIFAR-10 при ϵ равном 10

Заключение

В ходе лабораторной работы мы исследовали влияние различных значений ϵ на модели при атаках. Было обнаружено, что увеличение ϵ в FGSM приводит к искажению изображения, которое становится заметным для человека. Однако, слишком низкое значение ϵ может не вызвать ошибку в классификации модели. В случае с DeepFool, изображение практически не изменяется при любом значении ϵ , но модель всегда ошибается.