МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«МИРЭА – Российский технологический университет»

Институт кибербезопасности и цифровых технологий

Кафедра КБ-4 «Интеллектуальные системы информационной безопасности»

Отчёт по лабораторной работе № 1

По дисциплине

«Анализ защищенности систем искусственного интеллекта»

Выполнил:

ББМО–02–22

Агишевский П. Д.

Проверил:

Спирин А. А.

Москва, 2024

Каждый фрагмент кода начинается с краткого комментария, описывающего процесс, в целях которого этот фрагмент и написан.

Вывод для каждого фрагмента кода для наглядности будет написан на сером фоне.

```
# Сначала скопируем проект по ссылке в локальную среду Google Colab !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 | 11.54 MiB/s, done. Resolving deltas:
100% (27/27), done.
```

```
# Меняем дирректорию исполнения на папку проекта
%cd /content/EEL6812_DeepFool_Project
```

```
/content/EEL6812_DeepFool_Project
```

```
# Импортируем необходимые библиотеки
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


# Импортируем вспомогательные библиотеки из файлов проекта
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"}, укажем
значение для np.random.seed и torch.manual_seed rand_seed = 22
np.random.seed(rand_seed) torch.manual_seed(rand_seed)
```

```
<torch._C.Generator at 0x7ce54824a550>
```

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

# Загрузим датасет MNIST с параметрами mnist_mean = 0.5, mnist_std = 0.5,
mnist_dim = 28 mnist_mean = 0.5 mnist_std = 0.5
mnist_dim = 28
 mnist_min, mnist_max =
get_clip_bounds(mnist_mean,
mnist_std,
mnist_dim) mnist_min = mnist_min.to(device)
mnist_max = mnist_max.to(device)
```

```
 mnist_tf =
transforms.Compose([
transforms.ToTensor(),
transforms.Normalize(
mean=mnist_mean,
std=mnist_std)])
 mnist_tf_train = transforms.Compose([
transforms.RandomHorizontalFlip(),
transforms.ToTensor(),
transforms.Normalize(
mean=mnist_mean,
std=mnist_std)])

mnist_tf_inv = transforms.Compose([
transforms.Normalize(
mean=0.0,
        std=np.divide(1.0, mnist_std)),
transforms.Normalize(
        mean=np.multiply(-1.0, mnist_std),
std=1.0)])
 mnist_temp   =   datasets.MNIST(root='datasets/mnist',   train=True,
download=True,  transform=mnist_tf_train)  mnist_train,  mnist_val  =
random_split(mnist_temp, [50000, 10000])
 mnist_test = datasets.MNIST(root='datasets/mnist',
train=False,                              download=True,
transform=mnist_tf)

cifar_classes = ['airplane', 'automobile', 'bird', 'cat', 'deer',
                 'dog', 'frog', 'horse', 'ship', 'truck']
```

```
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 da
tasets/mnist/MNIST/raw/train-images-idx3-ubyte.gz
100%|██████████| 9912422/9912422 [00:00<00:00, 122629555.19it/s]
Extracting datasets/mnist/MNIST/raw/train-images-idx3-ubyte.gz to datasets/mn
ist/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 da
tasets/mnist/MNIST/raw/train-labels-idx1-ubyte.gz
100%|██████████| 28881/28881 [00:00<00:00, 39677593.78it/s]
Extracting datasets/mnist/MNIST/raw/train-labels-idx1-ubyte.gz to datasets/mn
ist/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 dat
asets/mnist/MNIST/raw/t10k-images-idx3-ubyte.gz
100%|██████████| 1648877/1648877 [00:00<00:00, 47687909.56it/s]
Extracting datasets/mnist/MNIST/raw/t10k-images-idx3-ubyte.gz to datasets/mni
st/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 dat
asets/mnist/MNIST/raw/t10k-labels-idx1-ubyte.gz
100%|██████████| 4542/4542 [00:00<00:00, 19012503.76it/s]
Extracting datasets/mnist/MNIST/raw/t10k-labels-idx1-ubyte.gz to datasets/mni
st/MNIST/raw
```

```
# Загрузим датасет CIFAR-10 с параметрами cifar_mean = [0.491, 0.482,
```

```
0.447] cifar_std = [0.202, 0.199, 0.201] cifar_dim = 32
cifar_mean = [0.491, 0.482, 0.447] cifar_std = [0.202,
0.199, 0.201] cifar_dim = 32
 cifar_min,                cifar_max                =
get_clip_bounds(cifar_mean,
cifar_std,                                 cifar_dim)
cifar_min   =   cifar_min.to(device)   cifar_max   =
cifar_max.to(device)
 cifar_tf =
transforms.Compose([
transforms.ToTensor(),
transforms.Normalize(
mean=cifar_mean,
std=cifar_std)])

cifar_tf_train = transforms.Compose([
transforms.RandomCrop(
size=cifar_dim,         padding=4),
    transforms.RandomHorizontalFlip(),
transforms.ToTensor(),
transforms.Normalize(
mean=cifar_mean,         std=cifar_std)])

cifar_tf_inv = transforms.Compose([
transforms.Normalize(
mean=[0.0, 0.0, 0.0],
std=np.divide(1.0, cifar_std)),
transforms.Normalize(
        mean=np.multiply(-1.0, cifar_mean),
std=[1.0, 1.0, 1.0])])
 cifar_temp  =  datasets.CIFAR10(root='datasets/cifar-10',  train=True,
download=True,   transform=cifar_tf_train)   cifar_train,   cifar_val   =
random_split(cifar_temp, [40000, 10000])
 cifar_test = datasets.CIFAR10(root='datasets/cifar-10',
train=False,                               download=True,
transform=cifar_tf)
Downloading https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz to datase
ts/cifar-10/cifar-10-python.tar.gz
100%|██████████| 170498071/170498071 [00:03<00:00, 43315076.71it/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
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: U
serWarning: This DataLoader will create 4 worker processes in total. Our sugg
ested max number of worker in current system is 2, which is smaller than what
this DataLoader is going to create. Please be aware that excessive worker cre
```

```
ation might get DataLoader running slow or even freeze, lower the worker numb
er to avoid potential slowness/freeze if necessary.
  warnings.warn(_create_warning_msg(
 import os
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 = 64
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
= 0.6
model = LeNet_MNIST().to(device)
model.load_state_dict(torch.load('weights/clean/mnist_lenet.pth',map_locat
ion=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('')
evaluate_attack('mnist_lenet_deepfool.csv', 'results', device, model,
mnist_loader_test, mnist_min, mnist_max, deep_args, 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
```

```
# Загрузим и оценим стойкость модели LeNet к FGSM и DeepFool атакам на
основе датасета CIFAR-10 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('')  evaluate_attack('mnist_fc_deepfool.csv', 'results',
device, model, mnist_loader_test, mnist_min, mnist_max, deep_args,
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
```

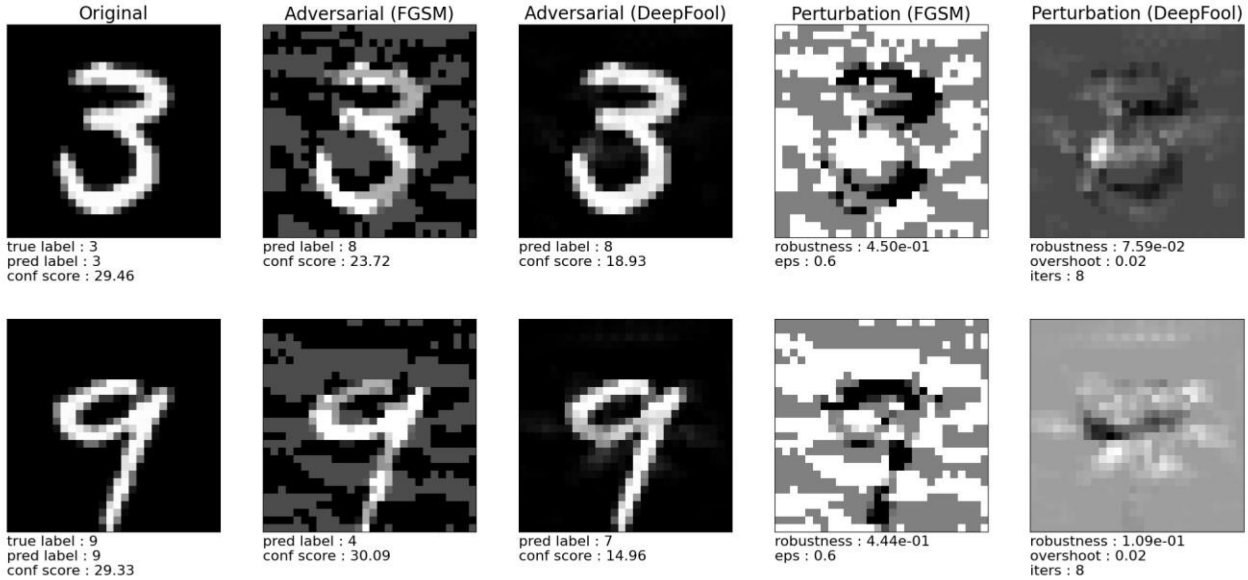
```
# Загрузим и оценим стойкость модели LeNet к FGSM и DeepFool атакам на
основе датасета CIFAR-10 fgsm_eps = 0.1
model = LeNet_CIFAR().to(device)
model.load_state_dict(torch.load('weights/clean/cifar_lenet.pth',
map_location=torch.device('cpu')))
 evaluate_attack('cifar_lenet_fgsm.csv', 'results', 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
```

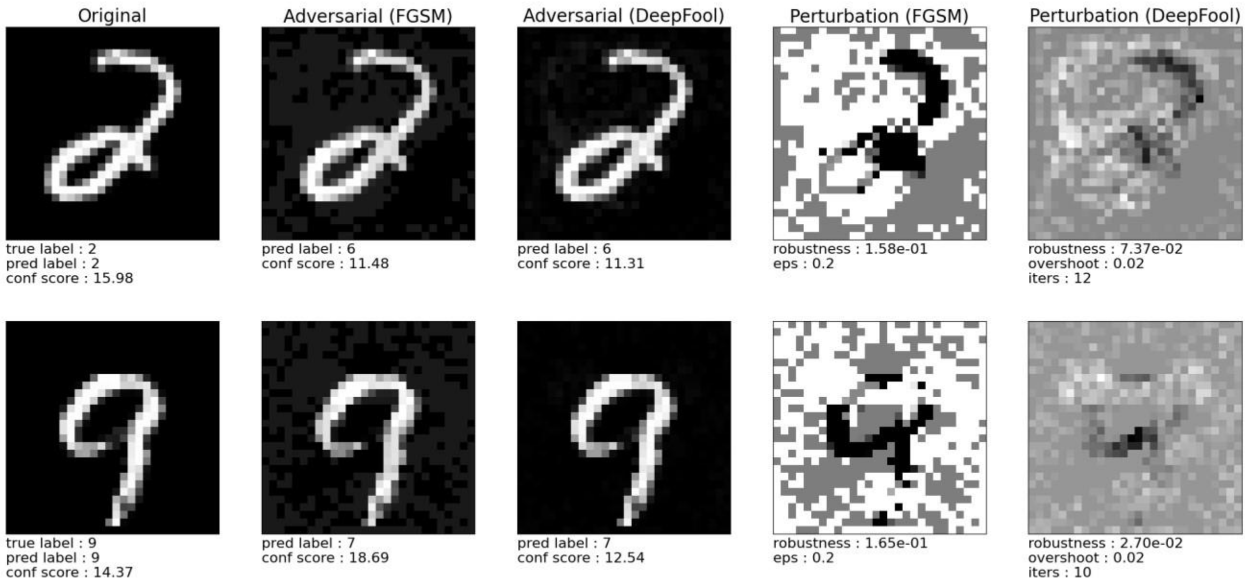
Выполним оценку атакующих примеров для сетей

```
# LeNet на датасете MNIST 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_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: U
serWarning: This DataLoader will create 4 worker processes in total. Our sugg
ested max number of worker in current system is 2, which is smaller than what
this DataLoader is going to create. Please be aware that excessive worker cre
ation might get DataLoader running slow or even freeze, lower the worker numb
er to avoid potential slowness/freeze if necessary.
  warnings.warn(_create_warning_msg(
```



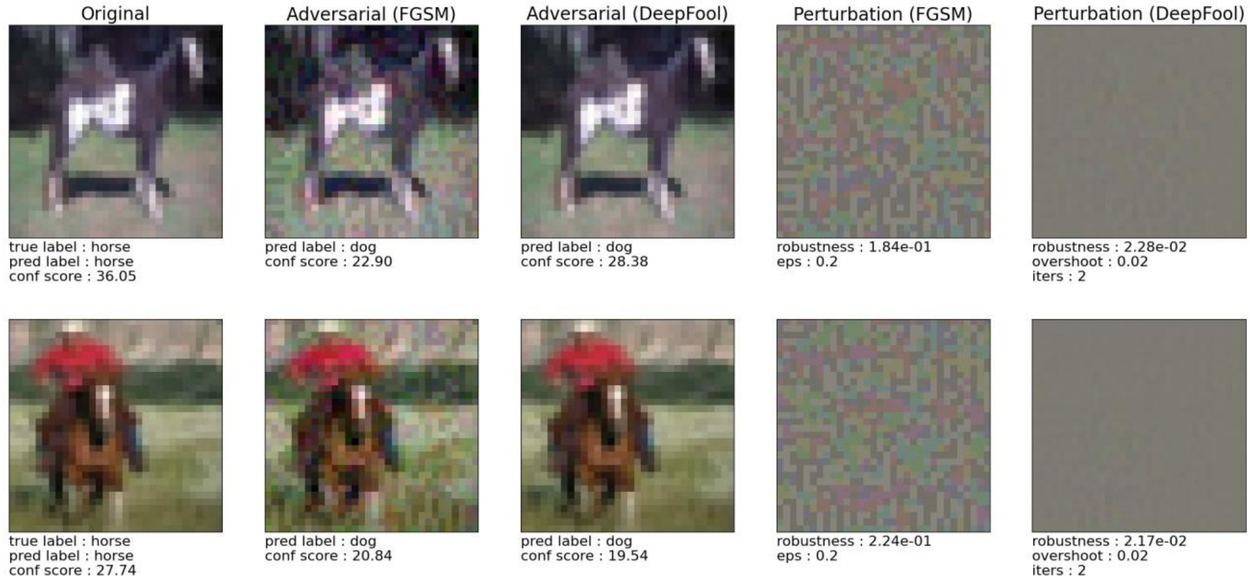

```
# FCNet на датасете MNIST fgsm_eps
= 0.2
model = FC_500_150().to(device)
model.load_state_dict(torch.load('weights/clean/mnist_fc.pth'))
 display_attack(device, model, mnist_test, mnist_tf_inv,
mnist_min, mnist_max, fgsm_eps, deep_args, has_labels=False,
l2_norm=True, pert_scale=1.0, fig_rows=2, fig_width=25,
fig_height=11)

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



In [19]:

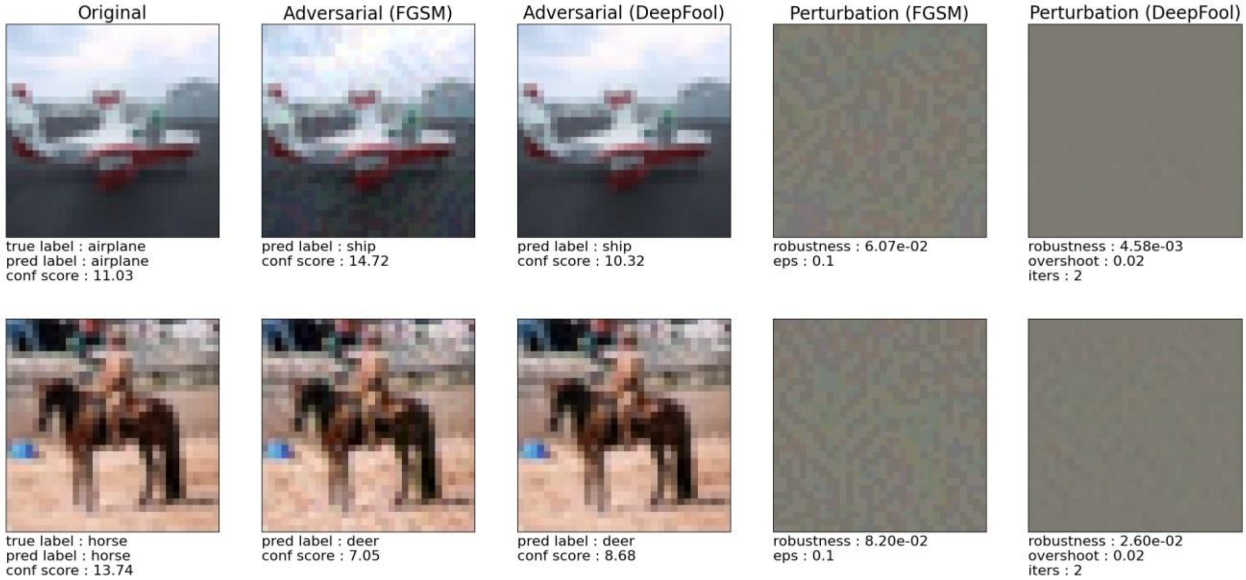
```
# Network-in-Network на датасете CIFAR
fgsm_eps = 0.2 model =
Net().to(device)
model.load_state_dict(torch.load('weights/clean/cifar_nin.pth'))
 display_attack(device, model, cifar_test, cifar_tf_inv,
cifar_min, cifar_max, fgsm_eps, deep_args, has_labels=False,
l2_norm=True, pert_scale=1.0, fig_rows=2, fig_width=25,
fig_height=11, label_map=cifar_classes)

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

```
/usr/local/lib/python3.10/dist-packages/torch/utils/data/dataloader.py:557: U
serWarning: This DataLoader will create 4 worker processes in total. Our sugg
ested max number of worker in current system is 2, which is smaller than what
this DataLoader is going to create. Please be aware that excessive worker cre
ation might get DataLoader running slow or even freeze, lower the worker numb
er to avoid potential slowness/freeze if necessary.
  warnings.warn(_create_warning_msg(
```




```
# LeNet на датасете CIFAR 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()
```

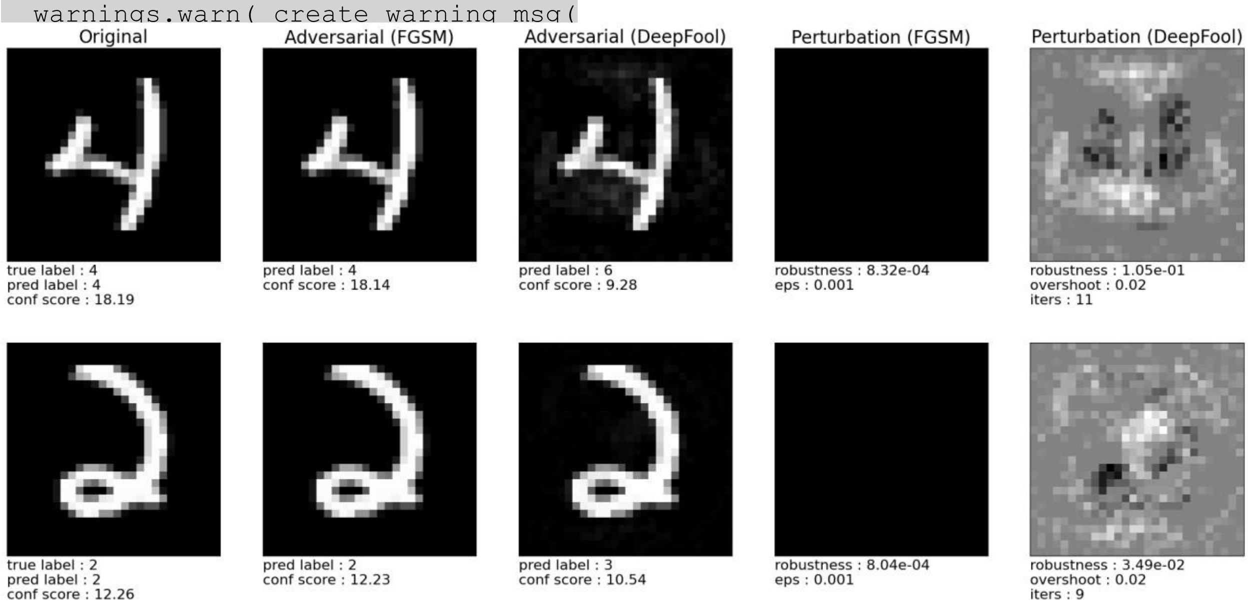
```
/usr/local/lib/python3.10/dist-packages/torch/utils/data/dataloader.py:557: UserWarning: This DataLoader will create 4 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller than what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader running slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  warnings.warn(_create_warning_msg(
```



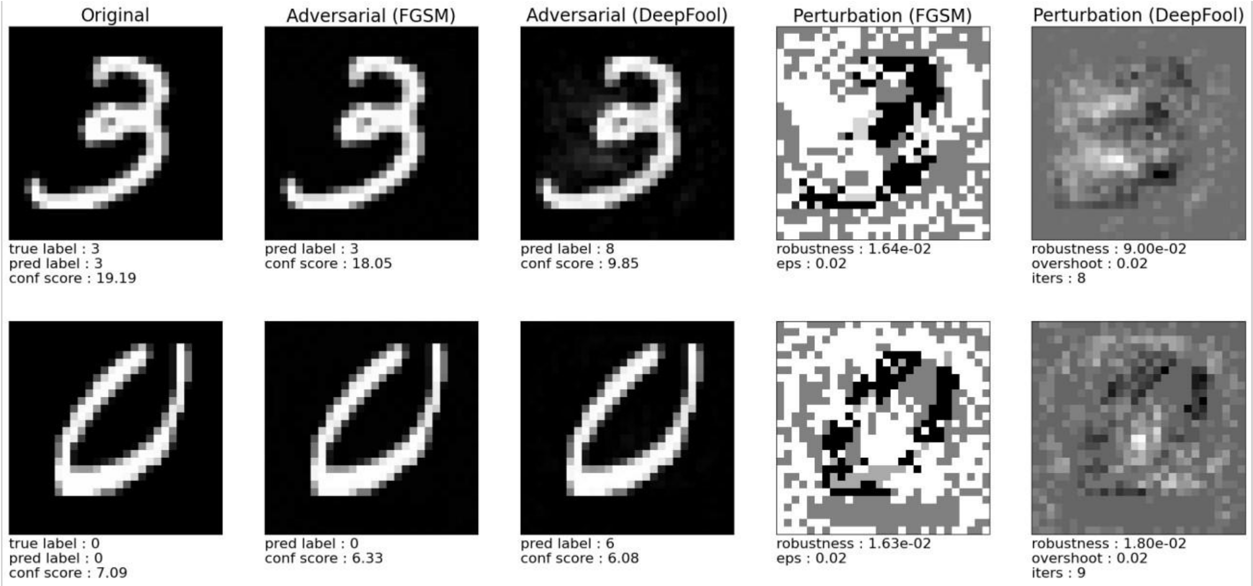

```
# Отразим отличия для fgsm_eps=(0.001, 0.02, 0.5, 0.9, 10) и выявим
закономерность/обнаружим отсутсвие влияние параметра eps для сетей FC
LeNet на датасете MNIST, NiN LeNEt на датасете CIFAR fgsm_epss
= [0.001, 0.02, 0.5, 0.9, 10]
 for fgsm_eps in
fgsm_epss:

    print(f"Используется fgsm_eps {fgsm_eps}")
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()
 for fgsm_eps in
fgsm_epss:
     print(f"Используется fgsm_eps
{fgsm_eps}")      model =
FC_500_150().to(device)
    model.load_state_dict(torch.load('weights/clean/mnist_fc.pth'))
evaluate_attack(f'mnist_fc_fgsm_eps{fgsm_eps}.csv', 'results', device,
model, mnist_loader_test, mnist_min, mnist_max, fgsm_eps, is_fgsm=True)

    if device.type == 'cuda': torch.cuda.empty_cache()
 for fgsm_eps in
fgsm_epss:

    print(f"Используется fgsm_eps {fgsm_eps}")
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()
 for fgsm_eps in
fgsm_epss:

    print(f"Используется fgsm_eps {fgsm_eps}")
model = Net().to(device)
    model.load_state_dict(torch.load('weights/clean/cifar_nin.pth'))
     evaluate_attack(f'cifar_nin_fgsm_eps{fgsm_eps}.csv',
'results', device, model, cifar_loader_test, cifar_min, cifar_max,
fgsm_eps, is_fgsm=True)

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

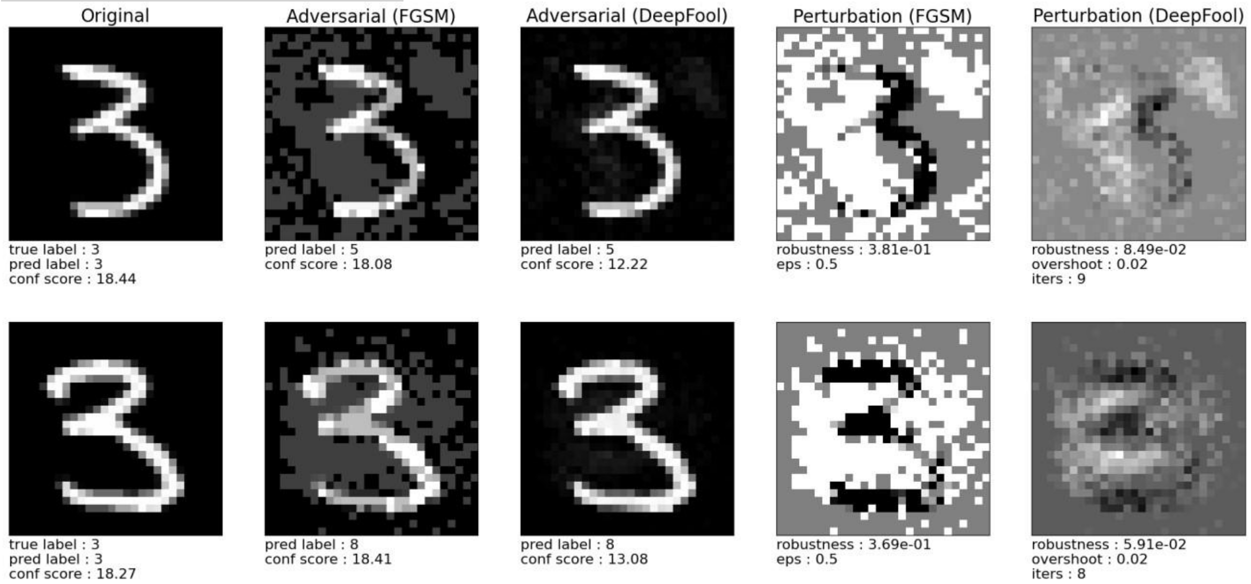
```
Используется fgsm eps 0.001
/usr/local/lib/python3.10/dist-packages/torch/utils/data/dataloader.py:557: U
serWarning: This DataLoader will create 4 worker processes in total. Our sugg
ested max number of worker in current system is 2, which is smaller than what
this DataLoader is going to create. Please be aware that excessive worker cre
ation might get DataLoader running slow or even freeze, lower the worker numb
er to avoid potential slowness/freeze if necessary.
  warnings.warn(_create_warning_msg(
```



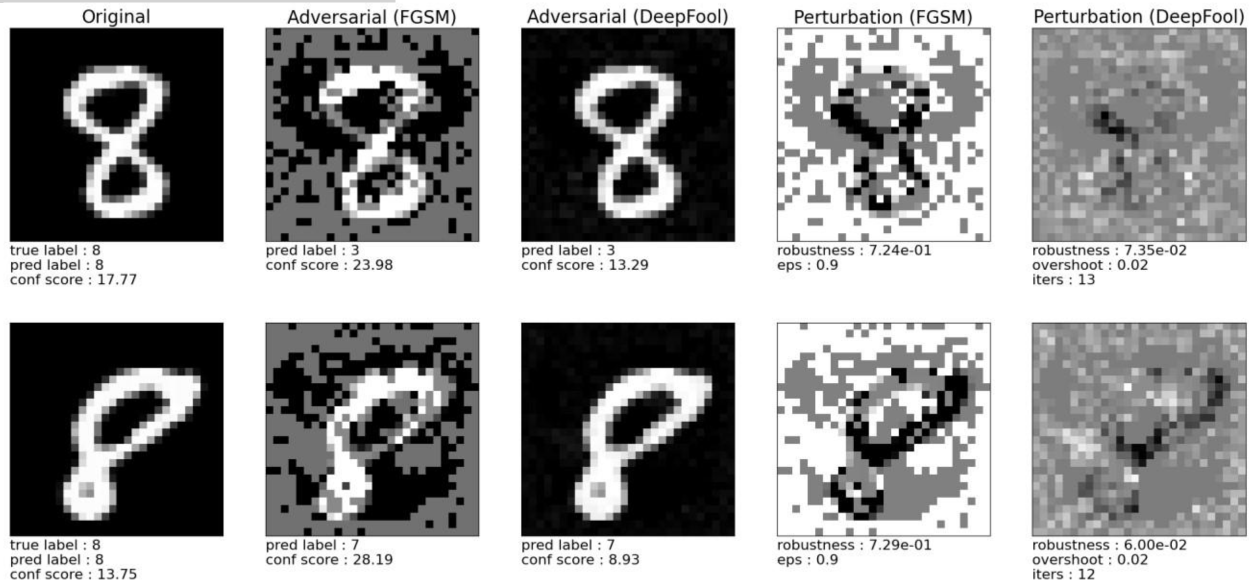

```
Используется fgsm_eps 0.02
```



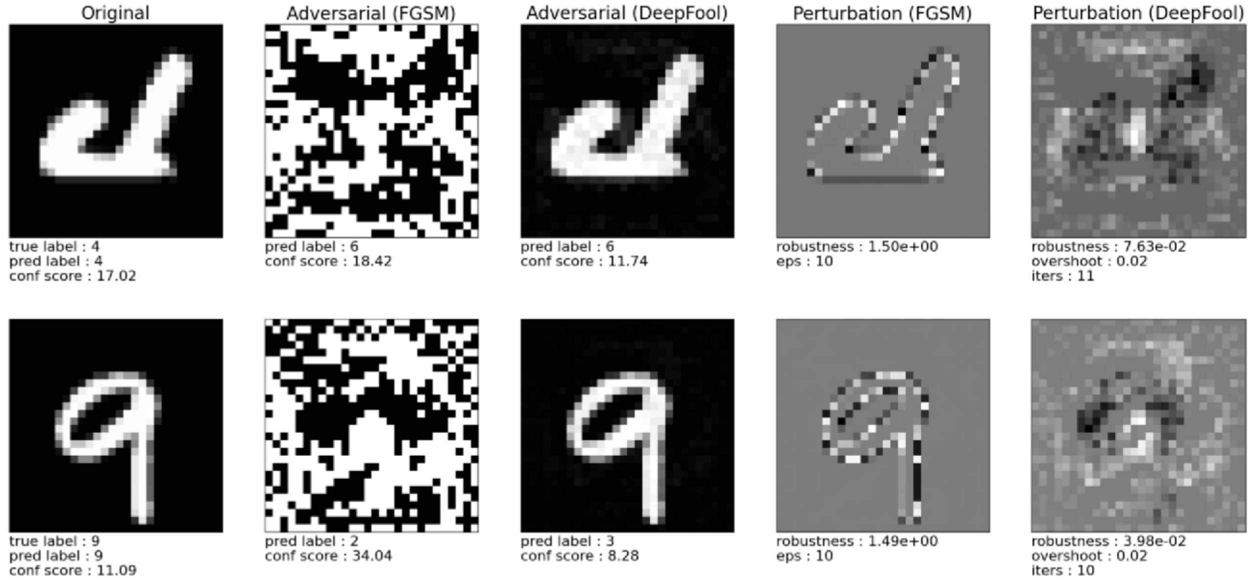
Используется fgsm_eps 0.5



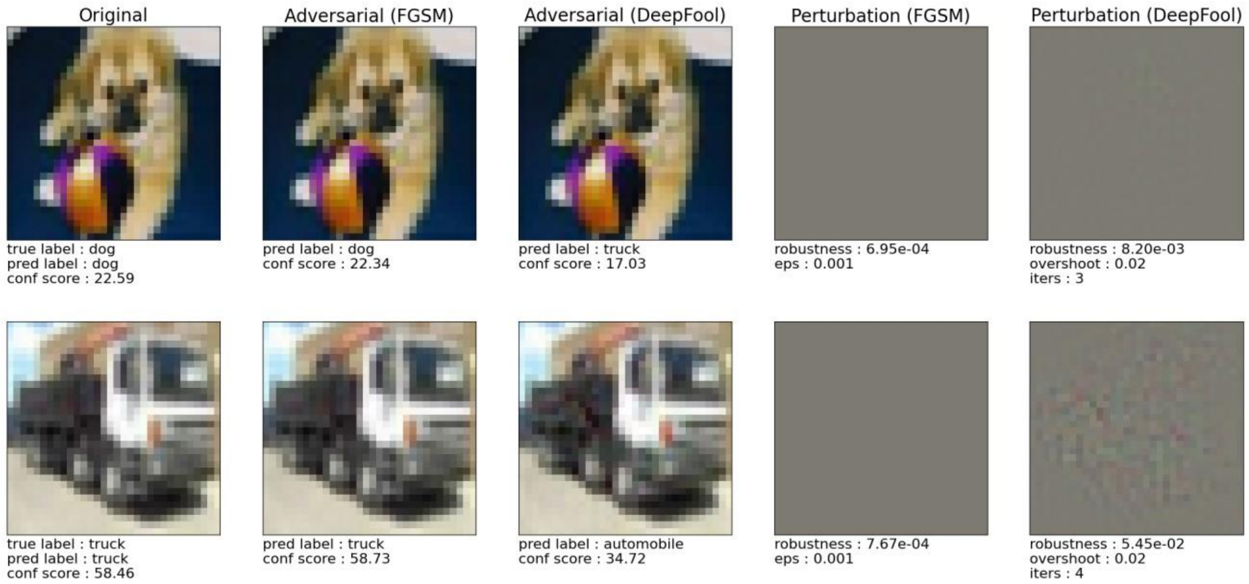
Используется fgsm_eps 0.9



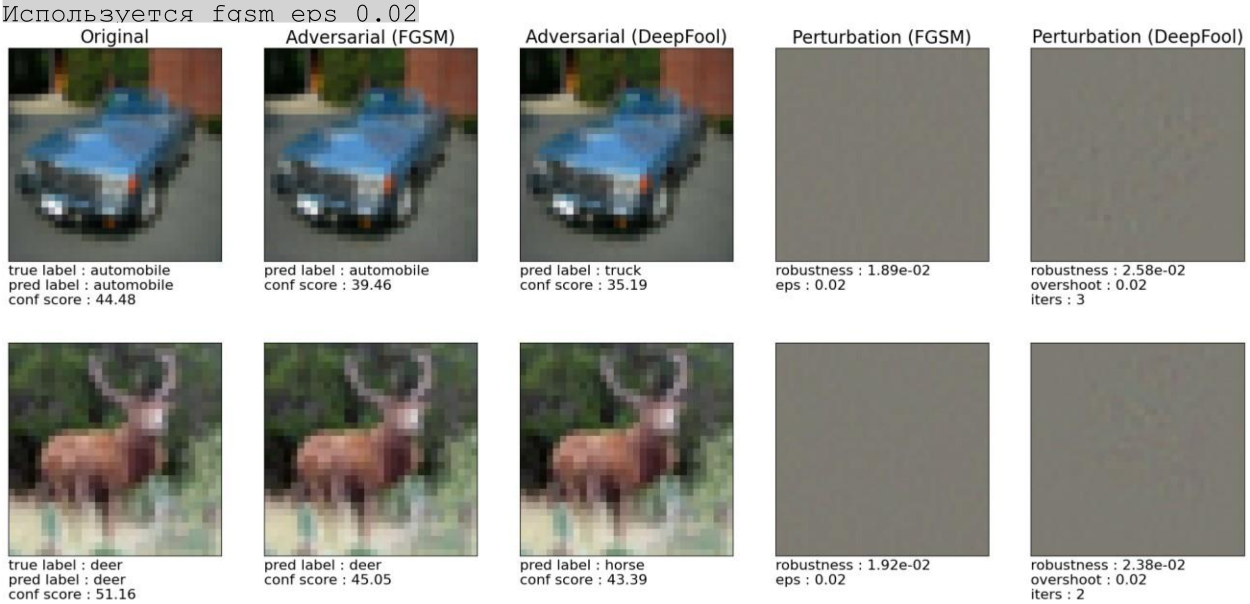
Используется fgsm_eps 10



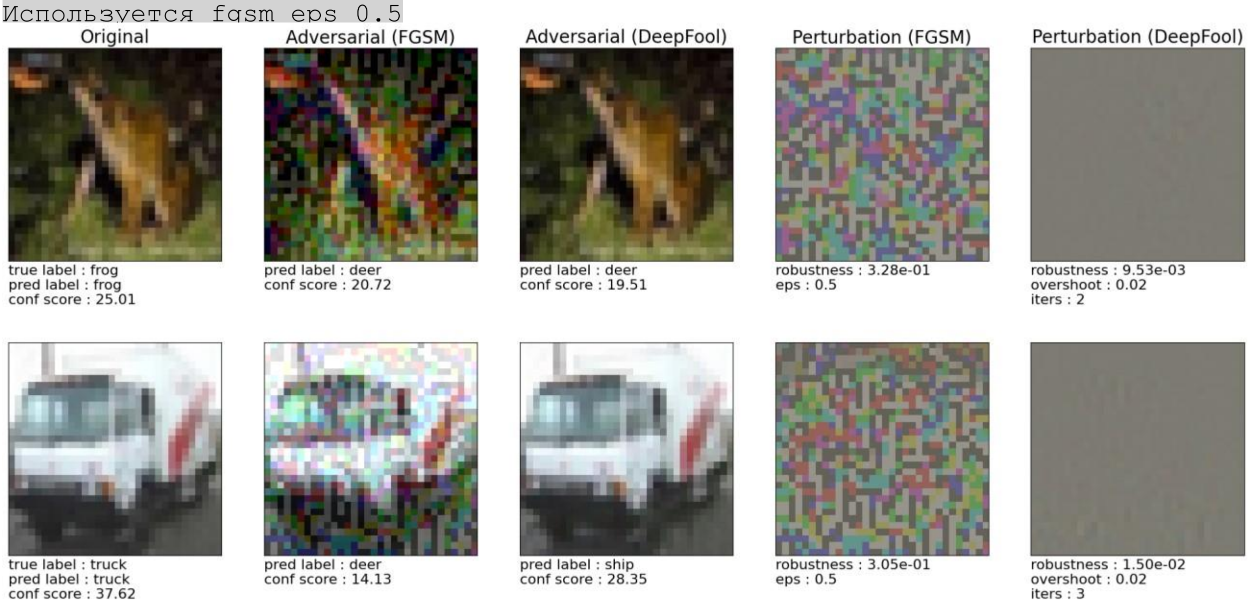

```
Используется fgsm_eps 0.001
FGSM Test Error : 3.07%
FGSM Robustness : 8.08e-04
FGSM Time (All Images) : 0.72 s
FGSM Time (Per Image) : 72.00 us
Используется fgsm_eps 0.02
FGSM Test Error : 5.54%
FGSM Robustness : 1.60e-02
FGSM Time (All Images) : 0.54 s
FGSM Time (Per Image) : 53.68 us
Используется fgsm_eps 0.5
FGSM Test Error : 99.21%
FGSM Robustness : 3.86e-01
FGSM Time (All Images) : 0.56 s
FGSM Time (Per Image) : 56.40 us
Используется fgsm_eps 0.9
FGSM Test Error : 99.87%
FGSM Robustness : 6.86e-01
FGSM Time (All Images) : 0.60 s
FGSM Time (Per Image) : 60.29 us
Используется fgsm_eps 10
FGSM Test Error : 99.87%
FGSM Robustness : 1.47e+00
FGSM Time (All Images) : 0.55 s
FGSM Time (Per Image) : 55.18 us
Используется fgsm_eps 0.001
/usr/local/lib/python3.10/dist-packages/torch/utils/data/dataloader.py:557: U
serWarning: This DataLoader will create 4 worker processes in total. Our sugg
ested max number of worker in current system is 2, which is smaller than what
this DataLoader is going to create. Please be aware that excessive worker cre
ation might get DataLoader running slow or even freeze, lower the worker numb
er to avoid potential slowness/freeze if necessary.
  warnings.warn(_create_warning_msg(
```



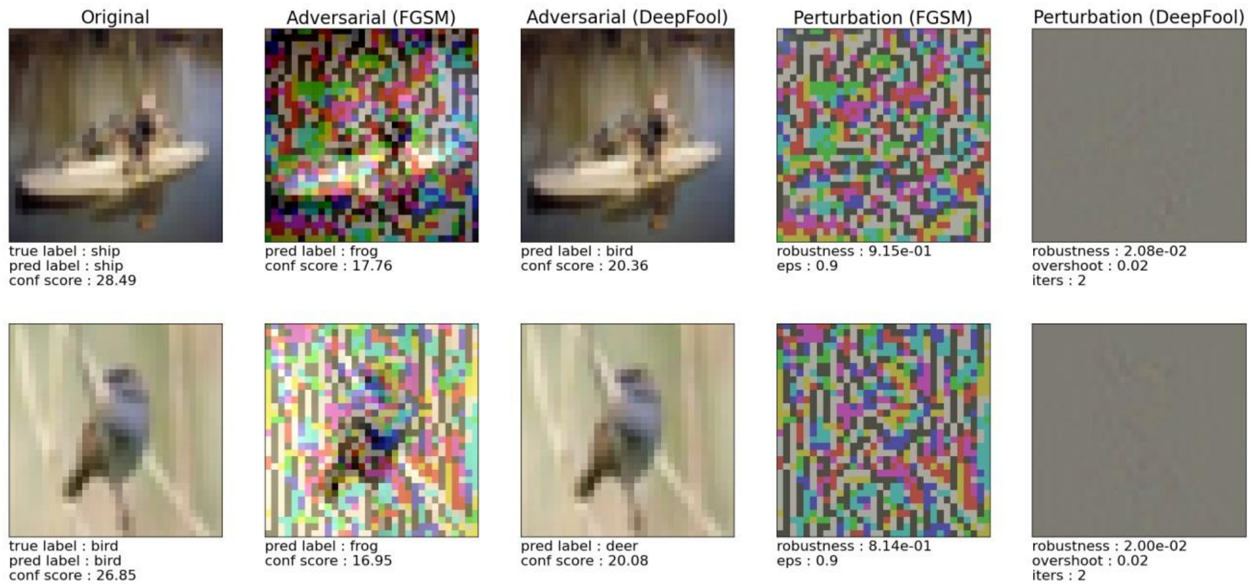
Используется fgsm_eps 0.02



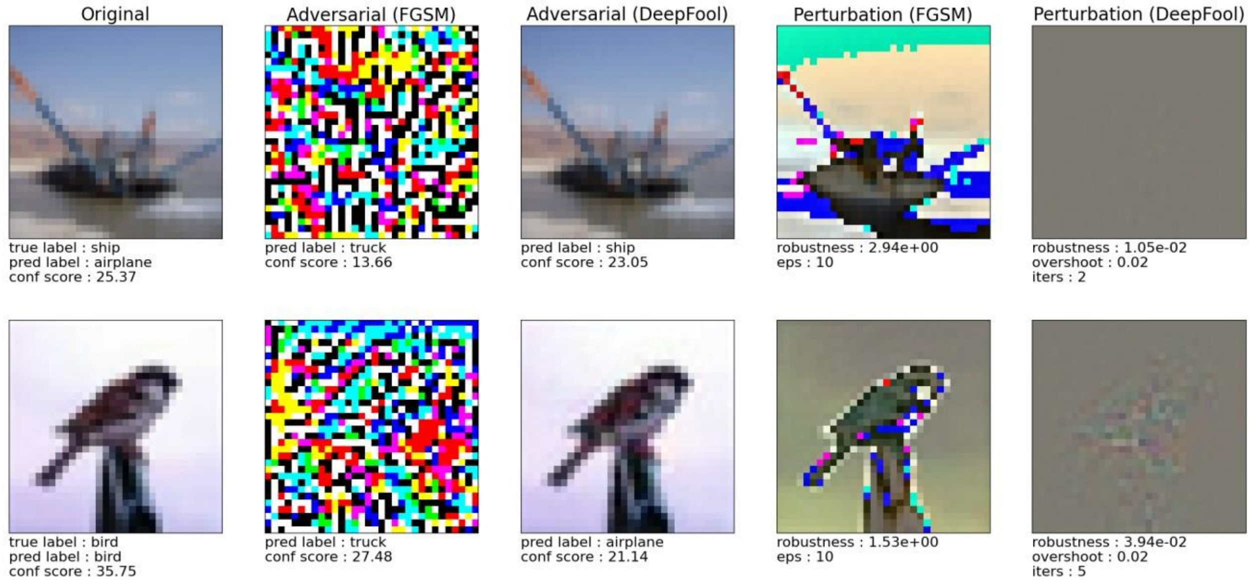
Используется fgsm_eps 0.5



Используется fgsm_eps 0.9



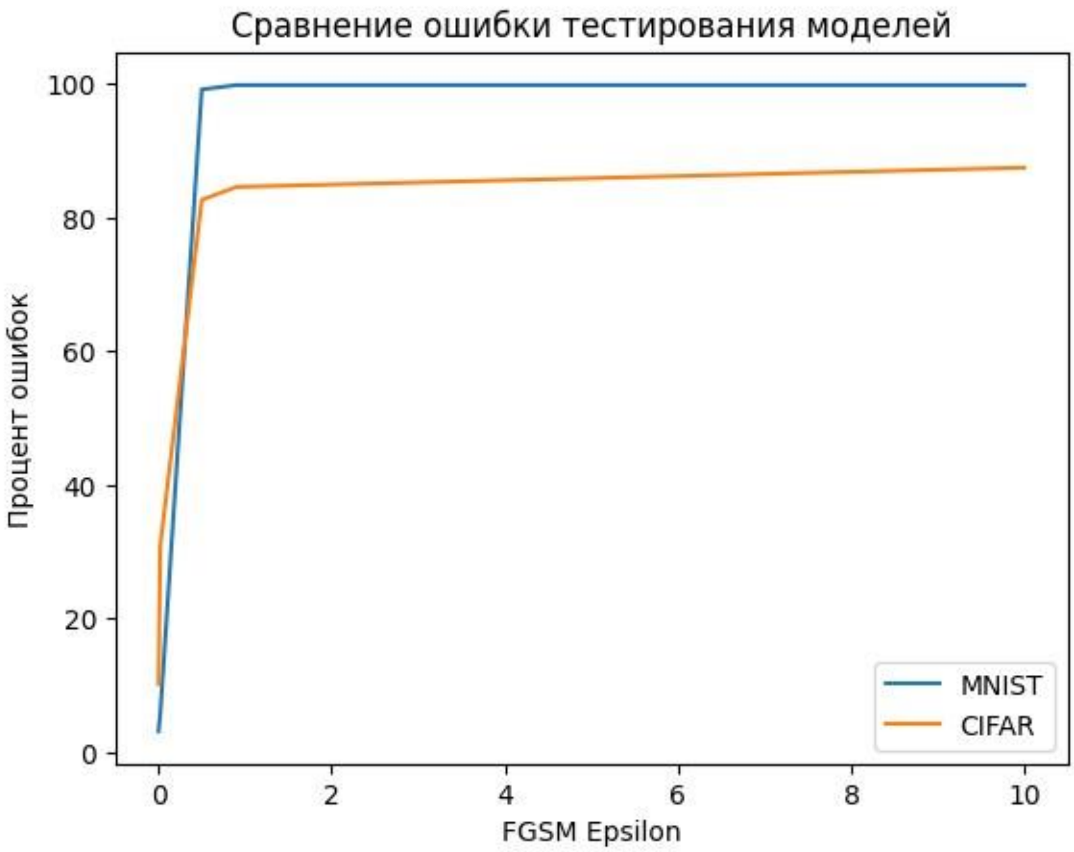

```
Используется fgsm eps 10
```




```
Используется fgsm eps 0.001
FGSM Test Error : 10.12%
FGSM Robustness : 8.92e-04
FGSM Time (All Images) : 1.09 s
FGSM Time (Per Image) : 108.82 us
Используется fgsm eps 0.02
FGSM Test Error : 30.76%
FGSM Robustness : 1.78e-02
FGSM Time (All Images) : 1.31 s
FGSM Time (Per Image) : 131.25 us
Используется fgsm eps 0.5
FGSM Test Error : 82.65%
FGSM Robustness : 4.40e-01
FGSM Time (All Images) : 1.12 s
FGSM Time (Per Image) : 112.16 us
Используется fgsm eps 0.9
FGSM Test Error : 84.60%
FGSM Robustness : 7.79e-01
FGSM Time (All Images) : 1.16 s
FGSM Time (Per Image) : 116.42 us
Используется fgsm eps 10
FGSM Test Error : 87.53%
FGSM Robustness : 2.46e+00
FGSM Time (All Images) : 1.15 s
```

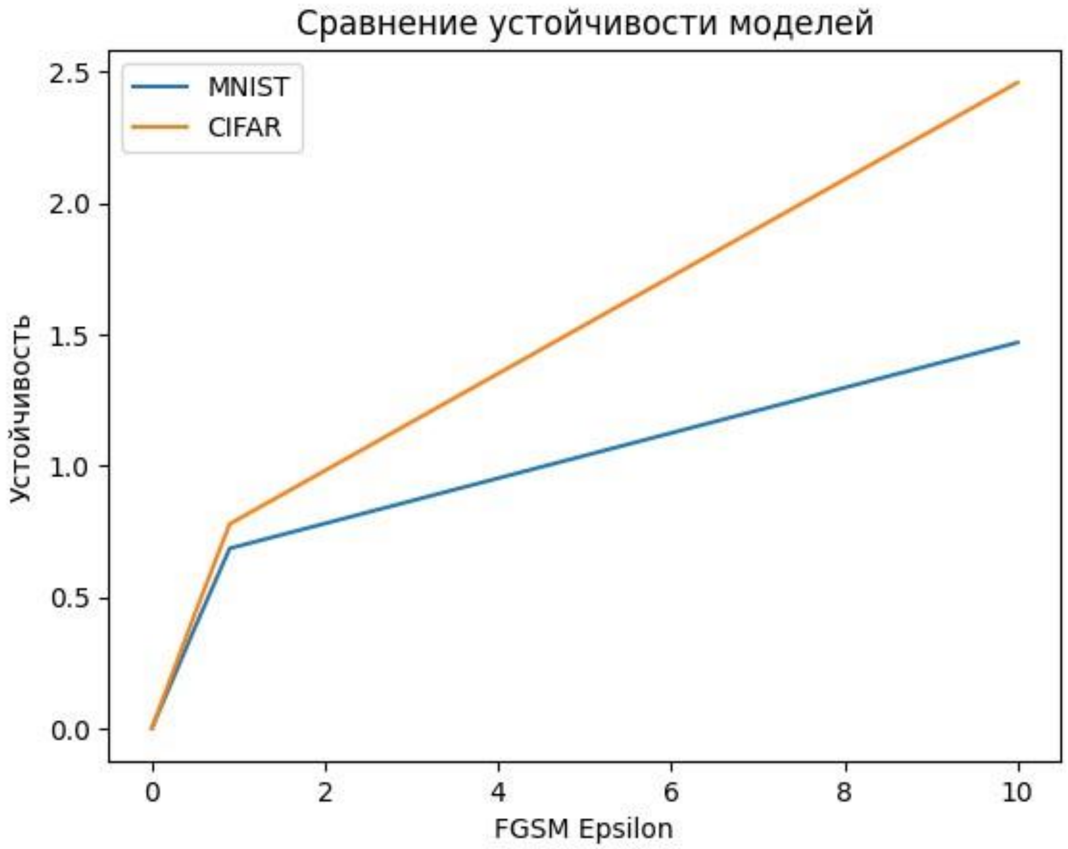
```
FGSM Time (Per Image) : 115.36 us
```

```
import matplotlib.pyplot as plt

fgsm_eps = [0.001, 0.02, 0.5, 0.9, 10]
fgsm_test_error_MNIST = [3.07, 5.54, 99.21, 99.87, 99.87]
fgsm_robustness_MNIST = [8.08e-04, 1.60e-02, 3.86e-01, 6.86e-01, 1.47e+00]
fgsm_test_error_CIFAR = [10.12, 30.76, 82.67, 84.62, 87.50]
fgsm_robustness_CIFAR = [8.92e-04, 1.78e-02, 4.40e-01, 7.79e-01, 2.46e+00]
 plt.plot(fgsm_eps, fgsm_test_error_MNIST,
label='MNIST') plt.plot(fgsm_eps, fgsm_test_error_CIFAR,
label='CIFAR')
 plt.xlabel('FGSM Epsilon')
plt.ylabel('Процент ошибок')
plt.title('Сравнение ошибки тестирования моделей')
plt.legend()

plt.show()
```



In [24]:

```
plt.plot(fgsm_eps, fgsm_robustness_MNIST, label='MNIST')
plt.plot(fgsm_eps, fgsm_robustness_CIFAR, label='CIFAR')
 plt.xlabel('FGSM
Epsilon')
plt.ylabel('Устойчивость')
plt.title('Сравнение устойчивости моделей') plt.legend()
 plt.show()
```



По картинкам понятно что от параметра fgsm_eps зависит степень шума. Параметр fgsm_eps имеет значительное влияние на производительность модели и ее устойчивость к атакам.

При увеличении значения fgsm_eps увеличивается ошибка тестирования модели, что указывает на снижение ее производительности.

В то же время, устойчивость модели также увеличивается с увеличением fgsm_eps, что указывает на то, что модель становится менее устойчивой к атакам. Это означает, что модель более подвержена ошибкам при обработке входных данных, которые были зашумлены.

Время вычислений не показывает последовательной тенденции с увеличением fgsm_eps. Это говорит о том, что величина возмущений не оказывает существенного влияния на время обработки изображений.