

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«МИРЭА – Российский технологический университет» РТУ МИРЭА

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

Отчёт по лабораторной работе № 1

По дисциплине

«Анализ защищенности систем искусственного интеллекта»

Студент Чадов Виктор Тимофеевич Группа ББМО-01-22

> Работу принял Спирин А.А.

Скопируем проект по ссылке в локальную среду выполнения Jupyter (Google Colab)

```
!git clone https://github.com/ewatson2/EEL6812_DeepFool_Project.git
Cloning into 'EEL6812_DeepFool_Project'...
remote: Enumerating objects: 96, done.ote: Counting objects: 100%
(3/3), done.ote: Compressing objects: 100% (2/2), done.ote: Total 96
(delta 2), reused 1 (delta 1), pack-reused 93
```

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

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

Выполним импорт необходимых библиотек:

```
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
```

Выполним импорт вспомогательных библиотек из локальных 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 = 8
np.random.seed(rand_seed)
torch.manual_seed(rand_seed)
<torch._C.Generator at 0x78e3193f40f0>
```

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)
```

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

```
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']

Files already downloaded and verified
Files already downloaded and verified
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:560: 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(
```

Инициализируем deep_args

```
batch_size = 10
num_classes = 10
overshoot = 0.02
```

```
max_iters = 50
deep_args = [batch_size, num_classes, overshoot, max_iters]
```

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
```

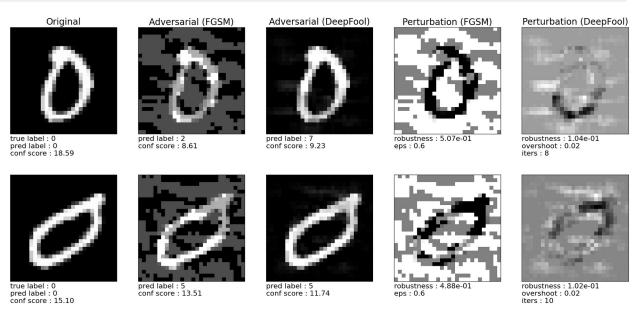
DeepFool Test Error : 87.81% DeepFool Robustness : 1.78e-02

DeepFool Time (All Images) : 73.27 s DeepFool Time (Per Image) : 7.33 ms

Выполним оценку атакующих примеров для сетей

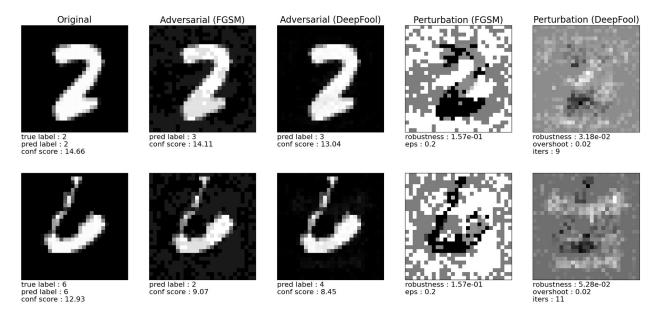
LeNet на датасете MNIST

```
fqsm 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, 12 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:560: 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 msq(
```



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()
```



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()
```



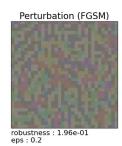
true label : bird pred label : deer conf score : 16.71



pred label : deer conf score : 27.08



pred label : bird conf score : 16.41





robustness : 1.38e-03 overshoot : 0.02 iters : 1



true label : frog pred label : frog conf score : 22.12



pred label: ship conf score: 18.22



pred label : ship conf score : 18.26



robustness: 2.48e-01 eps: 0.2



robustness: 1.29e-02 overshoot: 0.02 iters: 2

LeNet на датасете CIFAR

fgsm eps = 0.1modell = 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, 12 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()



true label : airplane pred label : airplane conf score : 6.62



pred label : deer conf score : 6.69



pred label : ship conf score : 5.80





robustness: 1.14e-02 overshoot: 0.02 iters: 2





pred label : bird conf score : 7.79



pred label : dog conf score : 5.54



robustness: 8.74e-02 eps: 0.1



robustness : 4.18e-04 overshoot : 0.02 iters : 1

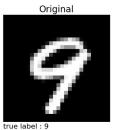
Отразим отличия для fgsm_eps=(0.001, 0.02, 0.5, 0.9, 10) и выявим закономерность/обнаружим отсутсвие влияние параметра eps для сетей FC LeNet на датасете MNIST, NiN LeNEt на датасете CIFAR

```
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 fqsm eps {fqsm 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

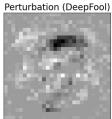


true label : 9 pred label : 9 conf score : 20.57







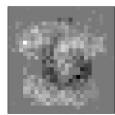




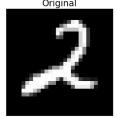








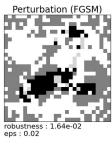
Используется fgsm_eps 0.02

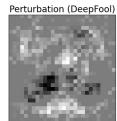


true label : 2 pred label : 2 conf score : 18.67









robustness: 1.04e-01 overshoot: 0.02 iters: 10

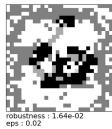


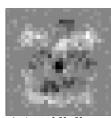


pred label : 6 conf score : 16.58



pred label : 4 conf score : 9.50



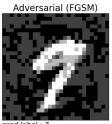


robustness: 1.01e-01 overshoot: 0.02 iters: 9

Используется fgsm eps 0.5

Original true label : 9 pred label : 9 conf score : 12.71

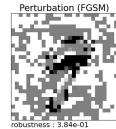
true label : 5 pred label : 5 conf score : 18.22



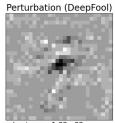
pred label : 7 conf score : 22.93



pred label : 7 conf score : 12.08



eps : 0.5



robustness: 1.02e-02 overshoot: 0.02 iters: 8



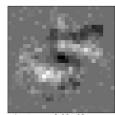


pred label : 0 conf score : 12.54



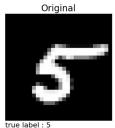
pred label : 0 conf score : 9.09



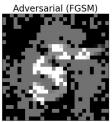


robustness: 8.66e-02 overshoot: 0.02 iters: 8

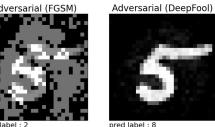
Используется fgsm_eps 0.9

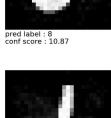


true label : 5 pred label : 5 conf score : 20.42

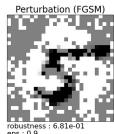


pred label : 2 conf score : 9.46

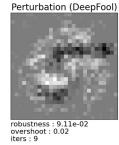


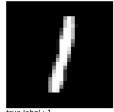


pred label : 4 conf score : 11.29

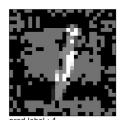


robustness: 6.81e-01 eps: 0.9



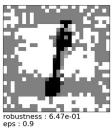


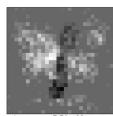
true label : 1 pred label : 1 conf score : 16.62



pred label : 4 conf score : 9.80

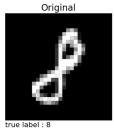






robustness : 7.70e-02 overshoot : 0.02 iters : 9

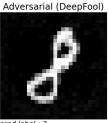
Используется fgsm_eps 10



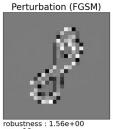
true label : 8 pred label : 8 conf score : 14.35



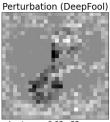
pred label : 7 conf score : 42.97

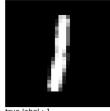


pred label : 3 conf score : 8.70



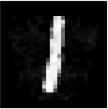
robustness: 1.56e+00 eps: 10



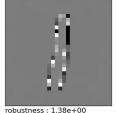




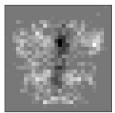
pred label: 9 conf score: 12.87



pred label: 8 conf score: 10.25



robustness: 1.38e+00 eps: 10



robustness : 6.27e-02 overshoot : 0.02 iters : 10

Используется fgsm eps 0.001 FGSM Test Error: 3.07%

FGSM Robustness: 8.08e-04

FGSM Time (All Images): 0.63 s FGSM Time (Per Image) : 62.81 us

Используется fgsm eps 0.02 FGSM Test Error: 5.54% FGSM Robustness : 1.60e-02

FGSM Time (All Images): 0.77 s FGSM Time (Per Image) : 77.36 us

Используется fgsm eps 0.5 FGSM Test Error: 99.21% FGSM Robustness : 3.86e-01

FGSM Time (All Images) : 0.51 s FGSM Time (Per Image) : 50.81 us

Используется fgsm eps 0.9 FGSM Test Error: 99.87% FGSM Robustness: 6.86e-01

FGSM Time (All Images) : 0.59 s FGSM Time (Per Image) : 58.95 us

Используется fgsm eps 10 FGSM Test Error: 99.87% FGSM Robustness: 1.47e+00 FGSM Time (All Images) : 0.57 s FGSM Time (Per Image) : 56.79 us

Используется fgsm eps 0.001

/usr/local/lib/python3.10/dist-packages/torch/utils/data/ dataloader.py:560: 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(

Original

true label : automobile pred label : automobile conf score : 49.96



pred label : automobile conf score : 49.73





robustness: 7.19e-04 eps: 0.001



robustness : 3.02e-02 overshoot : 0.02 iters : 4



true label : truck pred label : truck conf score : 36.59







robustness: 8.31e-04 eps: 0.001



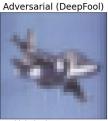
robustness : 2.47e-02 overshoot : 0.02 iters : 1

Используется fgsm_eps 0.02

Original

true label : airplane pred label : airplane conf score : 27.79













pred label : dog conf score : 32.71



pred label : dog conf score : 32.64



robustness: 1.31e-02 eps: 0.02



robustness: 4.49e-03 overshoot: 0.02 iters: 2

Используется fgsm eps 0.5

Original

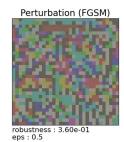
true label : cat pred label : cat conf score : 24.01

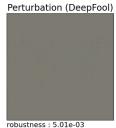


pred label : frog conf score : 21.40



pred label : dog conf score : 23.20





robustness : 5.01e-03 overshoot : 0.02 iters : 1



true label : frog pred label : frog conf score : 28.54



pred label : bird conf score : 30.73



pred label : bird conf score : 25.11



robustness: 5.32e-01 eps: 0.5



robustness: 1.26e-02 overshoot: 0.02 iters: 2

Используется fgsm_eps 0.9



true label : airplane pred label : airplane conf score : 24.56



pred label : frog conf score : 17.16



pred label : ship conf score : 22.31

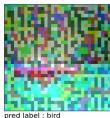




robustness: 1.61e-02 overshoot: 0.02 iters: 2



true label : ship pred label : ship conf score : 15.25



pred label : bird conf score : 11.00



pred label : airplane conf score : 14.96



robustness: 1.04e+00 eps: 0.9



robustness: 1.83e-03 overshoot: 0.02 iters: 1

Используется fgsm_eps 10



true label : automobile pred label : automobile conf score : 40.54



pred label : truck conf score : 25.04



pred label : ship conf score : 25.23



robustness: 1.98e+00 eps: 10



robustness: 1.52e-02 overshoot: 0.02 iters: 3



true label : ship pred label : ship conf score : 33.97



pred label : frog conf score : 21.90



pred label : frog conf score : 22.58



robustness: 2.63e+00 eps: 10



robustness: 3.97e-02 overshoot: 0.02

Используется fgsm_eps 0.001 FGSM Test Error : 10.12% FGSM Robustness : 8.92e-04

FGSM Time (All Images) : 1.41 s FGSM Time (Per Image) : 141.38 us

Используется fgsm_eps 0.02 FGSM Test Error : 30.76% FGSM Robustness : 1.78e-02

FGSM Time (All Images) : 1.23 s FGSM Time (Per Image) : 123.48 us

Используется fgsm_eps 0.5 FGSM Test Error : 82.67% FGSM Robustness : 4.40e-01

FGSM Time (All Images) : 1.37 s FGSM Time (Per Image) : 136.68 us

Используется fgsm_eps 0.9 FGSM Test Error : 84.62% FGSM Robustness : 7.79e-01

FGSM Time (All Images) : 1.19 s FGSM Time (Per Image) : 118.96 us

Используется fgsm_eps 10 FGSM Test Error : 87.50% FGSM Robustness : 2.46e+00 FGSM Time (All Images) : 1.36 s FGSM Time (Per Image) : 136.04 us

import matplotlib.pyplot as plt

```
\begin{array}{lll} 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] \end{array}
```

```
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()

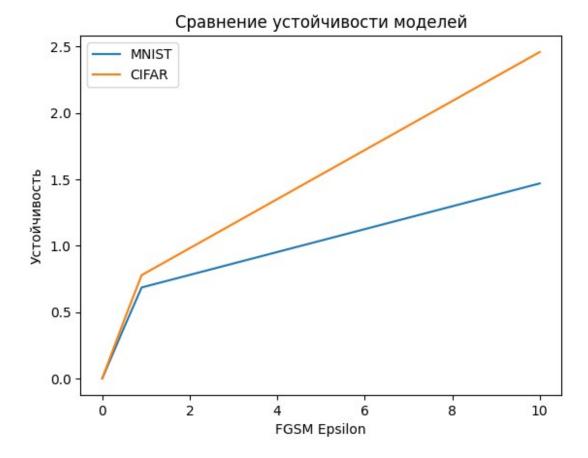
plt.show()
```

Сравнение ошибки тестирования моделей Процент ошибок MNIST CIFAR FGSM Epsilon

```
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.