



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

ИКБ направление «Киберразведка и противодействие угрозам
с применением технологий искусственного интеллекта» 10.04.01

Кафедра КБ-4 «Интеллектуальные системы информационной
безопасности»

**Лабораторная работа №4 + Практическая
работа №6**

по дисциплине

«Анализ защищённости систем искусственного
интеллекта»

Группа:
ББМО-02-22
Выполнил:
Запруднов
М.С.

Проверил:
Спирин А.А.

Загрузка и разбивка данных

```
transform = transforms.Compose([transforms.ToTensor(),
                                transforms.Normalize((0.0,), (1.0,))])
dataset = datasets.MNIST(root = './data', train=True, transform = transform, download=True)
train_set, val_set = torch.utils.data.random_split(dataset, [50000, 10000])
test_set = datasets.MNIST(root = './data', train=False, transform = transform, download=True)
train_loader = torch.utils.data.DataLoader(train_set, batch_size=1, shuffle=True)
val_loader = torch.utils.data.DataLoader(val_set, batch_size=1, shuffle=True)
test_loader = torch.utils.data.DataLoader(test_set, batch_size=1, shuffle=True)
print("Training data:", len(train_loader), "Validation data:", len(val_loader), "Test data:", len(test_loader))
```

Подключение графического ускорителя

```
: use_cuda=True
device = torch.device("cuda" if (use_cuda and torch.cuda.is_available()) else "cpu")
```

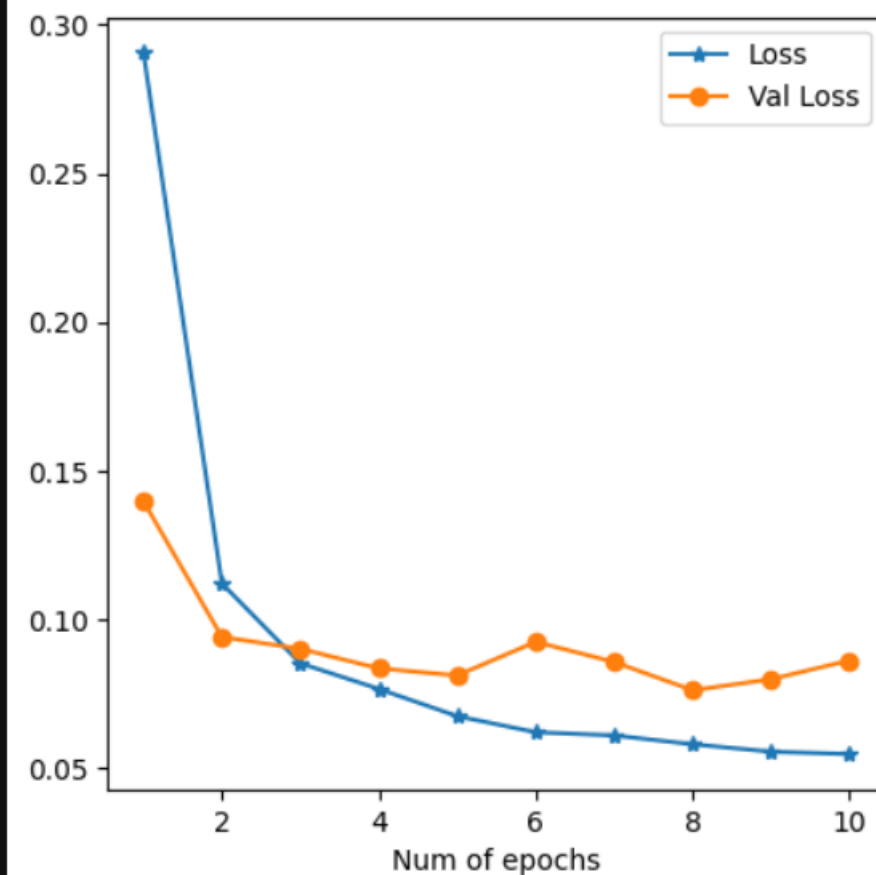
Создание класса НС на основе фреймворка torch.

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(1, 32, 3, 1)
        self.conv2 = nn.Conv2d(32, 64, 3, 1)
        self.dropout1 = nn.Dropout2d(0.25)
        self.dropout2 = nn.Dropout2d(0.5)
        self.fc1 = nn.Linear(9216, 128)
        self.fc2 = nn.Linear(128, 10)

    def forward(self, x):
        x = self.conv1(x)
        x = F.relu(x)
        x = self.conv2(x)
        x = F.relu(x)
        x = F.max_pool2d(x, 2)
        x = self.dropout1(x)
        x = torch.flatten(x, 1)
        x = self.fc1(x)
        x = F.relu(x)
        x = self.dropout2(x)
        x = self.fc2(x)
        output = F.log_softmax(x, dim=1)
        return output
```

```
model = Net().to(device)
```

Графики потерь



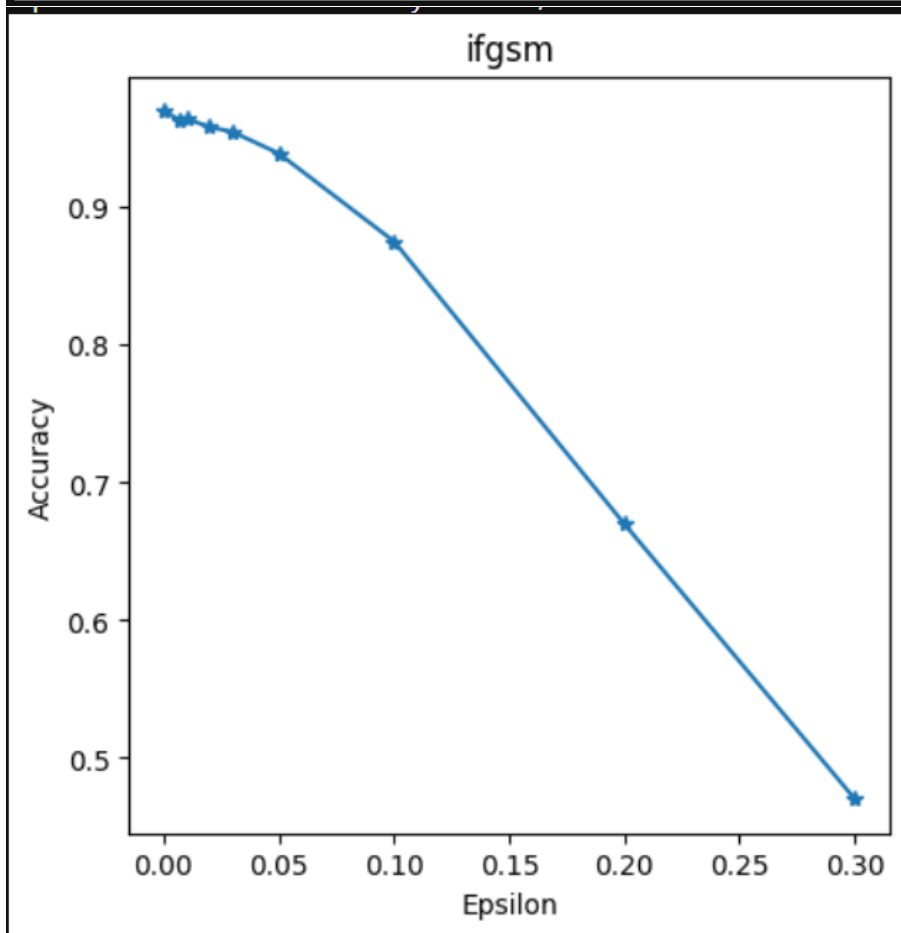
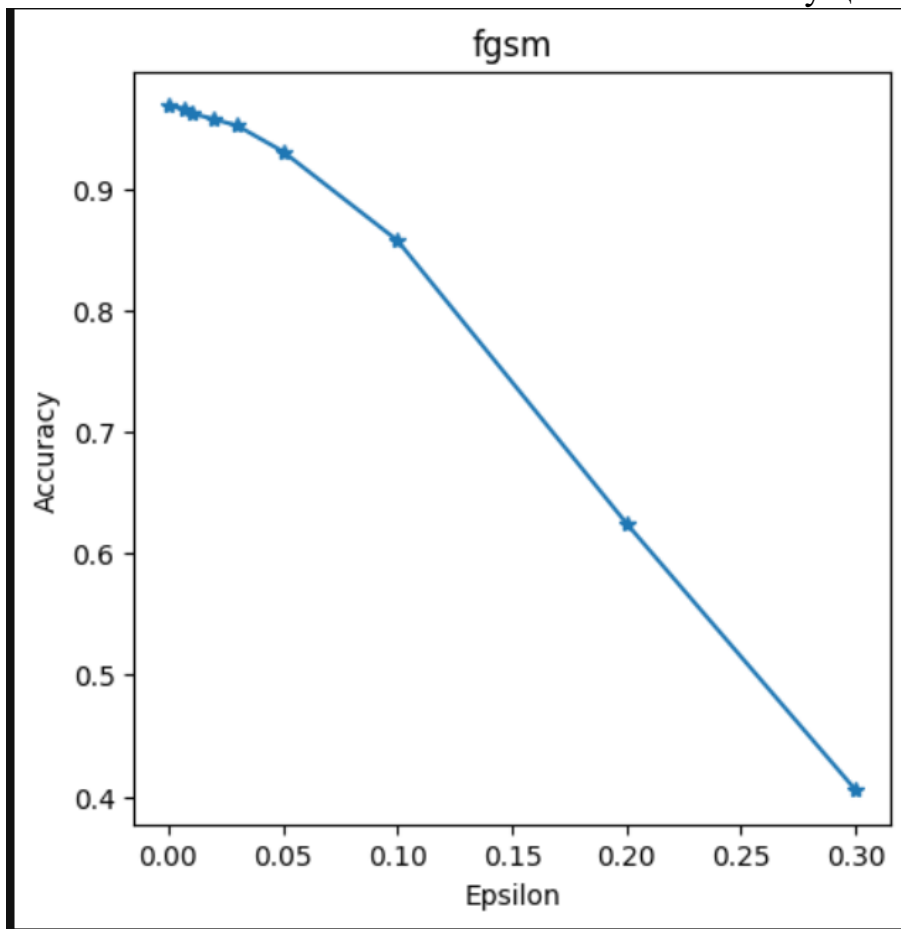
Функции атак FGSM, I-FGSM, MI-FGSM.

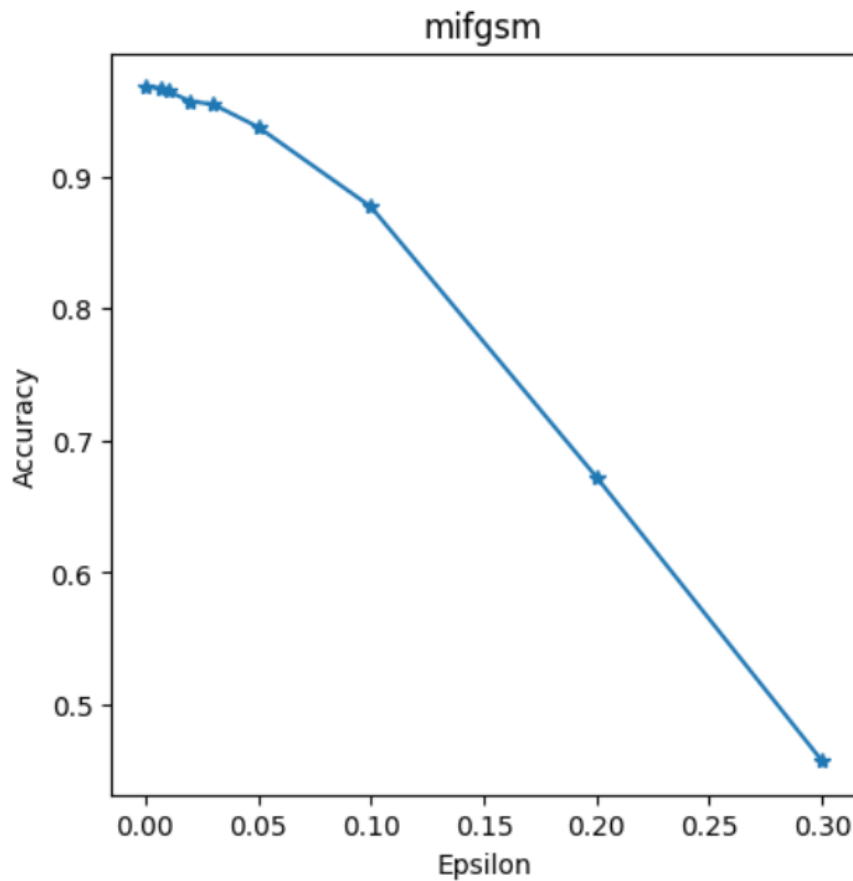
```
def fgsm_attack(input,epsilon,data_grad):
    pert_out = input + epsilon*data_grad.sign()
    pert_out = torch.clamp(pert_out, 0, 1)
    return pert_out
```

```
def ifgsm_attack(input,epsilon,data_grad):
    iter = 10
    alpha = epsilon/iter
    pert_out = input
    for i in range(iter-1):
        pert_out = pert_out + alpha*data_grad.sign()
        pert_out = torch.clamp(pert_out, 0, 1)
        if torch.norm((pert_out-input),p=float('inf')) > epsilon:
            break
    return pert_out
```

```
def mifgsm_attack(input,epsilon,data_grad):
    iter=10
    decay_factor=1.0
    pert_out = input
    alpha = epsilon/iter
    g=0
    for i in range(iter-1):
        g = decay_factor*g + data_grad/torch.norm(data_grad,p=1)
        pert_out = pert_out + alpha*torch.sign(g)
        pert_out = torch.clamp(pert_out, 0, 1)
        if torch.norm((pert_out-input),p=float('inf')) > epsilon:
            break
    return pert_out
```

Построим графики успешности атак(Ассурасу/эпсилон) и примеры выполненных атак в зависимости от степени возмущения epsilon:



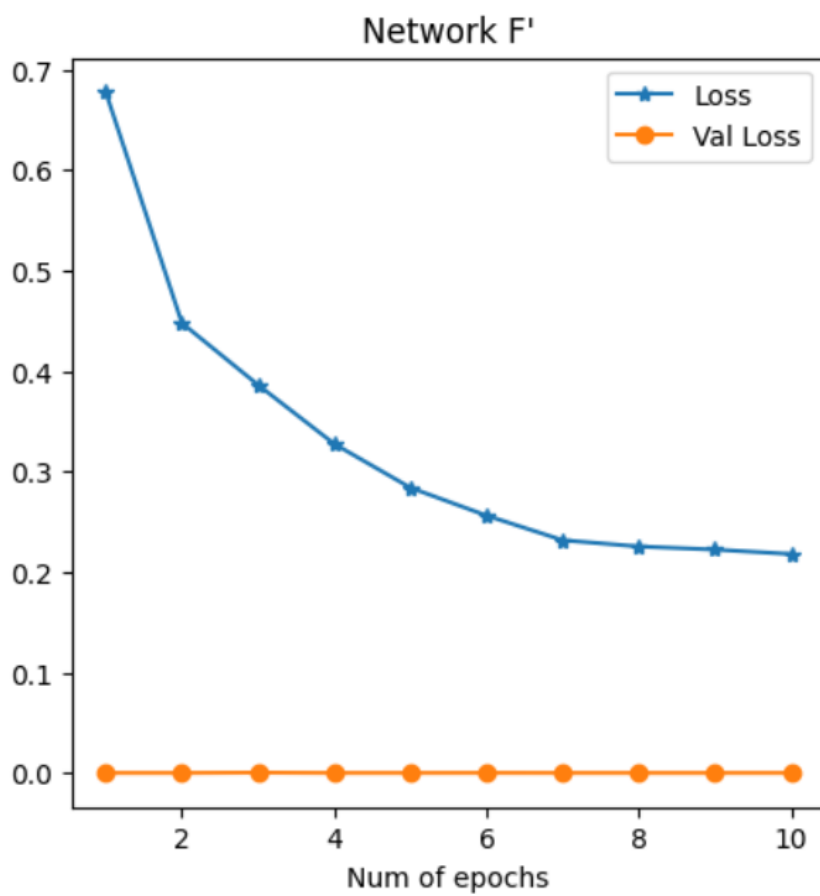
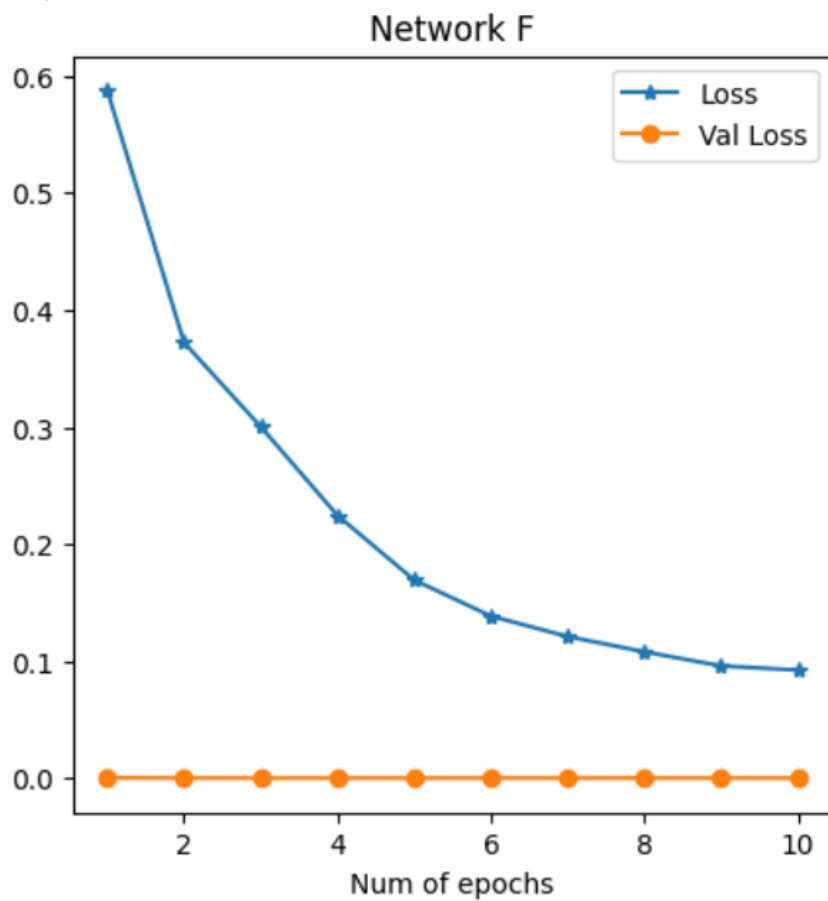


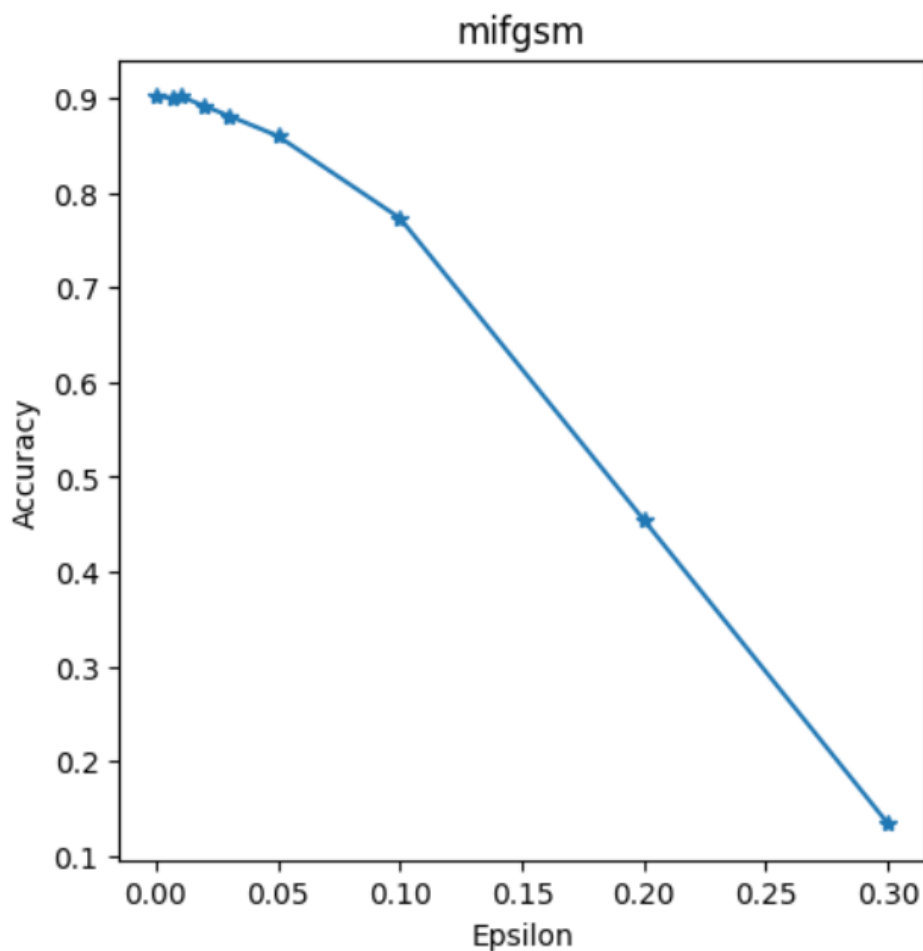
Создадим 2 класса НС и переопределим функцию обучения и тестирования.
Создадим функцию защиты методом дистилляции.

```
def defense(device, train_loader, val_loader, test_loader, epochs, Temp, epsilons):
    modelF = NetF().to(device)
    optimizerF = optim.Adam(modelF.parameters(), lr=0.0001, betas=(0.9, 0.999))
    schedulerF = optim.lr_scheduler.ReduceLROnPlateau(optimizerF, mode='min', factor=0.1, patience=3)
    modelF1 = NetF1().to(device)
    optimizerF1 = optim.Adam(modelF1.parameters(), lr=0.0001, betas=(0.9, 0.999))
    schedulerF1 = optim.lr_scheduler.ReduceLROnPlateau(optimizerF1, mode='min', factor=0.1, patience=3)
    criterion = nn.NLLLoss()
    lossF, val_lossF = fit(modelF, device, optimizerF, schedulerF, criterion, train_loader, val_loader, Temp, epochs)
    fig = plt.figure(figsize=(5,5))
    plt.plot(np.arange(1, epochs+1), lossF, "--", label="Loss")
    plt.plot(np.arange(1, epochs+1), val_lossF, "o-", label="Val Loss")
    plt.title("Network F")
    plt.xlabel("Num of epochs")
    plt.legend()
    plt.show()
    #converting target Labels to soft Labels
    for data in train_loader:
        input, label = data[0].to(device), data[1].to(device)
        softlabel = F.log_softmax(modelF(input), dim=1)
        data[1] = softlabel
    lossF1, val_lossF1 = fit(modelF1, device, optimizerF1, schedulerF1, criterion, train_loader, val_loader, Temp, epochs)
    fig = plt.figure(figsize=(5,5))
    plt.plot(np.arange(1, epochs+1), lossF1, "--", label="Loss")
    plt.plot(np.arange(1, epochs+1), val_lossF1, "o-", label="Val Loss")
    plt.title("Network F'")
    plt.xlabel("Num of epochs")
    plt.legend()
    plt.show()
    model = NetF1().to(device)
    model.load_state_dict(modelF1.state_dict())
    for attack in ("fgsm", "ifgsm", "mifgsm"):
        accuracies = []
        examples = []
        for eps in epsilons:
            acc, ex = test(model, device, test_loader, eps, attack)
            accuracies.append(acc)
            examples.append(ex)
        plt.figure(figsize=(5,5))
        plt.plot(epsilons, accuracies, "--")
        plt.title(attack)
        plt.xlabel("Epsilon")
        plt.ylabel("Accuracy")
        plt.show()

    cnt = 0
    plt.figure(figsize=(8,10))
    for i in range(len(epsilons)):
        for j in range(len(examples[i])):
            cnt += 1
            plt.subplot(len(epsilons), len(examples[0]), cnt)
            plt.xticks([], [])
            plt.yticks([], [])
            if j == 0:
                plt.ylabel("Eps: {}".format(epsilons[i]), fontsize=14)
            orig, adv, ex = examples[i][j]
            plt.title("{} -> {}".format(orig, adv))
            plt.imshow(ex, cmap="gray")
    plt.tight_layout()
    plt.show()
```

Результаты оценки защищенных сетей.





Выводы:

После применения алгоритма по защите модели результат работы был улучшен. Несмотря на это, в данном случае полностью восстановить работоспособность и защитить модель полностью не удалось. Вероятно, это зависит от показателей и сложности модели, а также от объема данных обучения и тестирования выборок. Результат проделанной работы демонстрирует улучшение в работе модели, подвергшейся атаке.