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 №4

по дисциплине: «Анализ защищенности систем искусственного интеллекта»

Выполнил:

Студент группы ББМО-02-22

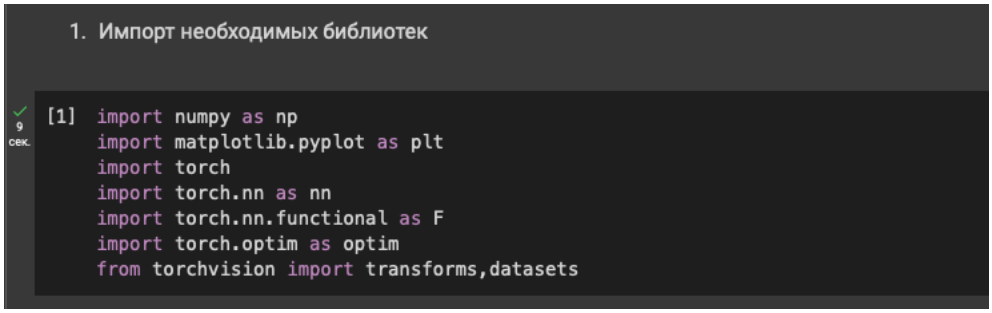
Филиппов Леонид Алексеевич

Проверил:

Спирин Андрей Андреевич

Москва 2024

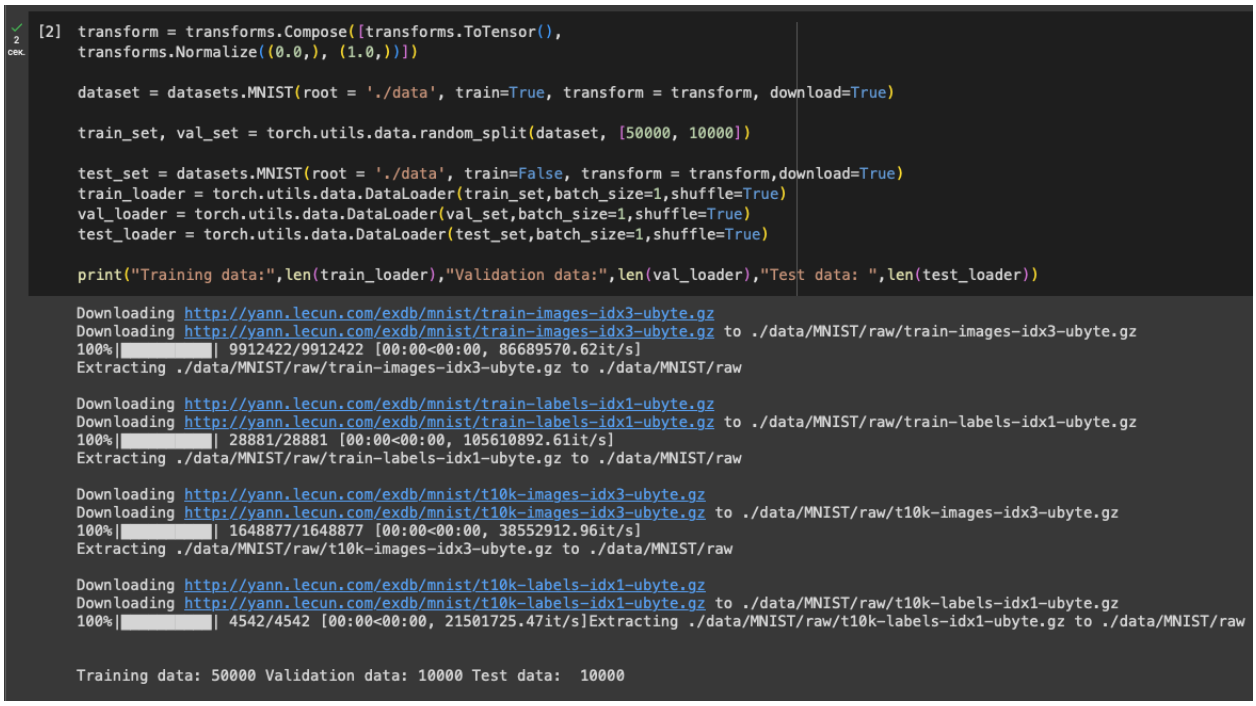
1. Импорт необходимых библиотек



1. Импорт необходимых библиотек

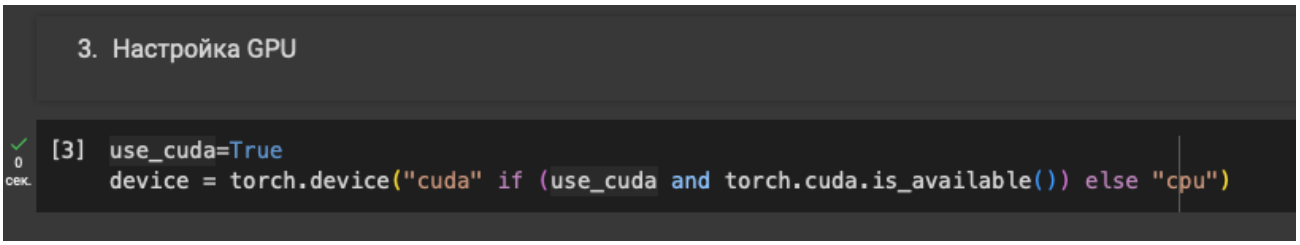
```
[1]  import numpy as np
     import matplotlib.pyplot as plt
     import torch
     import torch.nn as nn
     import torch.nn.functional as F
     import torch.optim as optim
     from torchvision import transforms,datasets
```

2. Задание нормализующих преобразований и загрузка набора данных (MNIST), разбиение данных на подвыборки



```
[2]  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))

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 ./data/MNIST/raw/train-images-idx3-ubyte.gz
100%|| 9912422/9912422 [00:00<00:00, 86689570.62it/s]
Extracting ./data/MNIST/raw/train-images-idx3-ubyte.gz to ./data/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 ./data/MNIST/raw/train-labels-idx1-ubyte.gz
100%|| 28881/28881 [00:00<00:00, 105610892.61it/s]
Extracting ./data/MNIST/raw/train-labels-idx1-ubyte.gz to ./data/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 ./data/MNIST/raw/t10k-images-idx3-ubyte.gz
100%|| 1648877/1648877 [00:00<00:00, 38552912.96it/s]
Extracting ./data/MNIST/raw/t10k-images-idx3-ubyte.gz to ./data/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 ./data/MNIST/raw/t10k-labels-idx1-ubyte.gz
100%|| 4542/4542 [00:00<00:00, 21501725.47it/s]Extracting ./data/MNIST/raw/t10k-labels-idx1-ubyte.gz to ./data/MNIST/raw


Training data: 50000 Validation data: 10000 Test data:  10000
```

3. Настройка GPU



3. Настройка GPU

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

4. Создание класса НС на основе фреймворка torch

4. Создание класса НС на основе фреймворка torch

```
[4]  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.Dropout(0.25)
         self.dropout2 = nn.Dropout(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
```

5. Проверка работоспособности созданного класса НС

5. Проверка работоспособности созданного класса НС

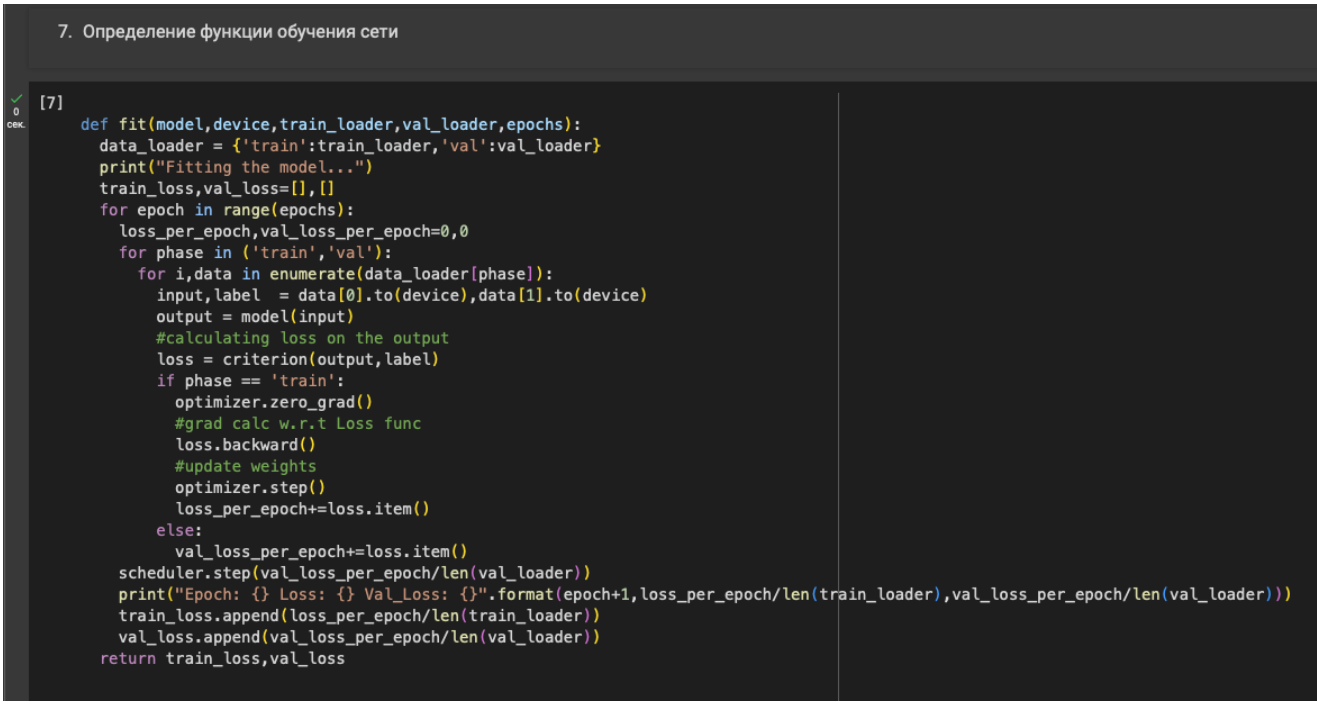
```
[5]  model = Net().to(device)
```

6. Создание оптимизатора, функции потерь и трейнер сети

6. Создание оптимизатора, функции потерь и трейнер сети

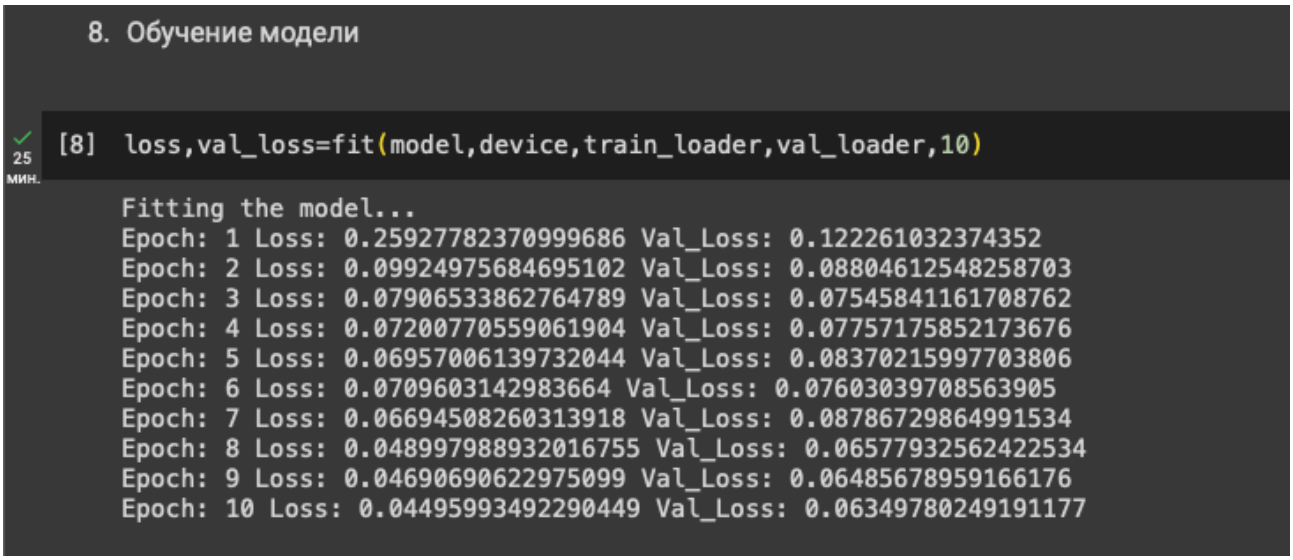
```
[6]  optimizer = optim.Adam(model.parameters(),lr=0.0001, betas=(0.9, 0.999))
     criterion = nn.NLLLoss()
     scheduler = optim.lr_scheduler.ReduceLROnPlateau(optimizer, mode='min', factor=0.1, patience=3)
```

7. Определение функции обучения сети



```
7. Определение функции обучения сети

[7]
0 сек.
    def fit(model,device,train_loader,val_loader,epochs):
      data_loader = {'train':train_loader,'val':val_loader}
      print("Fitting the model...")
      train_loss,val_loss=[],[]
      for epoch in range(epochs):
        loss_per_epoch,val_loss_per_epoch=0,0
        for phase in ('train','val'):
          for i,data in enumerate(data_loader[phase]):
            input,label  = data[0].to(device),data[1].to(device)
            output = model(input)
            #calculating loss on the output
            loss = criterion(output,label)
            if phase == 'train':
              optimizer.zero_grad()
              #grad calc w.r.t Loss func
              loss.backward()
              #update weights
              optimizer.step()
              loss_per_epoch+=loss.item()
            else:
              val_loss_per_epoch+=loss.item()
        scheduler.step(val_loss_per_epoch/len(val_loader))
        print("Epoch: {} Loss: {} Val_Loss: {}".format(epoch+1,loss_per_epoch/len(train_loader),val_loss_per_epoch/len(val_loader)))
        train_loss.append(loss_per_epoch/len(train_loader))
        val_loss.append(val_loss_per_epoch/len(val_loader))
      return train_loss,val_loss
```

8. Обучение модели



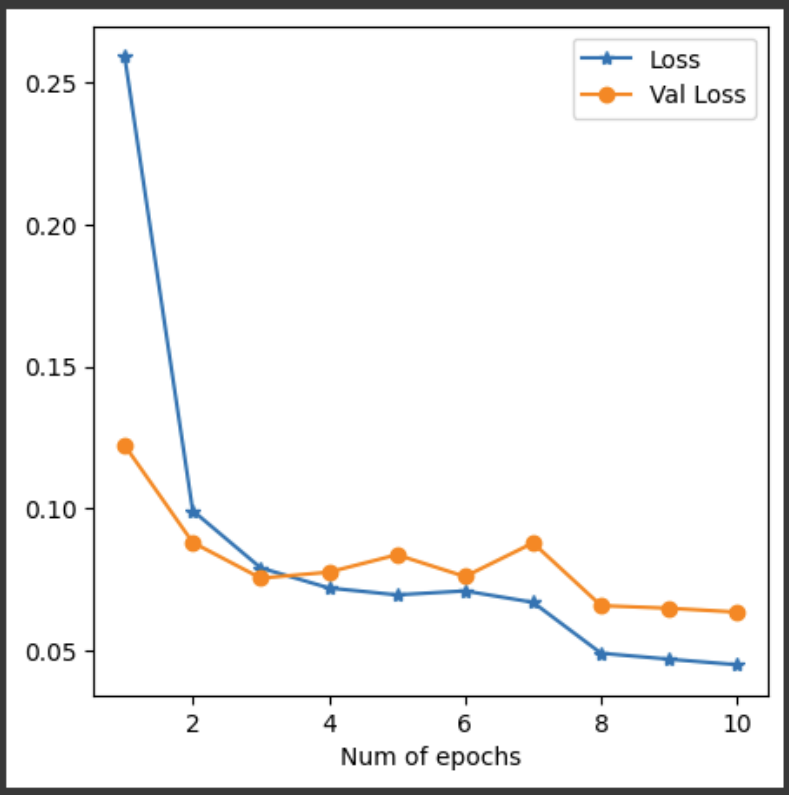
```
8. Обучение модели

[8]  loss,val_loss=fit(model,device,train_loader,val_loader,10)
25 мин.

Fitting the model...
Epoch: 1 Loss: 0.25927782370999686 Val_Loss: 0.122261032374352
Epoch: 2 Loss: 0.09924975684695102 Val_Loss: 0.08804612548258703
Epoch: 3 Loss: 0.07906533862764789 Val_Loss: 0.07545841161708762
Epoch: 4 Loss: 0.07200770559061904 Val_Loss: 0.07757175852173676
Epoch: 5 Loss: 0.06957006139732044 Val_Loss: 0.08370215997703806
Epoch: 6 Loss: 0.0709603142983664 Val_Loss: 0.07603039708563905
Epoch: 7 Loss: 0.06694508260313918 Val_Loss: 0.08786729864991534
Epoch: 8 Loss: 0.048997988932016755 Val_Loss: 0.06577932562422534
Epoch: 9 Loss: 0.04690690622975099 Val_Loss: 0.06485678959166176
Epoch: 10 Loss: 0.04495993492290449 Val_Loss: 0.06349780249191177
```

9. Построение графиков потерь при обучении и валидации в зависимости от эпохи

9. Построение графиков потерь при обучении и валидации в зависимости от эпохи

```
[9]  fig = plt.figure(figsize=(5,5))
     plt.plot(np.arange(1,11), loss, "*-",label="Loss")
     plt.plot(np.arange(1,11), val_loss,"o-",label="Val Loss")
     plt.xlabel("Num of epochs")
     plt.legend()
     plt.show()
```

0 сек.



10.Создание функций атак FGSM, I-FGSM, MI-FGSM

10. Создание функций атак FGSM, I-FGSM, MI-FGSM

```
[10] 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
```

11.Создание функций проверки

11. Создание функций проверки

```
[11] def test(model,device,test_loader,epsilon,attack):
       correct = 0
       adv_examples = []
       for data, target in test_loader:
         data, target = data.to(device), target.to(device)
         data.requires_grad = True
         output = model(data)
         init_pred = output.max(1, keepdim=True)[1]
         if init_pred.item() != target.item():
           continue
         loss = F.nll_loss(output, target)
         model.zero_grad()
         loss.backward()
         data_grad = data.grad.data
         if attack == "fgsm":
           perturbed_data = fgsm_attack(data,epsilon,data_grad)
         elif attack == "ifgsm":
           perturbed_data = ifgsm_attack(data,epsilon,data_grad)
         elif attack == "mifgsm":
           perturbed_data = mifgsm_attack(data,epsilon,data_grad)
         output = model(perturbed_data)
         final_pred = output.max(1, keepdim=True)[1]
         if final_pred.item() == target.item():
           correct += 1
         if (epsilon == 0) and (len(adv_examples) < 5):
           adv_ex = perturbed_data.squeeze().detach().cpu().numpy()
           adv_examples.append( (init_pred.item(), final_pred.item(), adv_ex))
         else:
           if len(adv_examples) < 5:
             adv_ex = perturbed_data.squeeze().detach().cpu().numpy()
             adv_examples.append( (init_pred.item(), final_pred.item(), adv_ex))
       final_acc = correct/float(len(test_loader))
       print("Epsilon: {}\tTest Accuracy = {} / {} = {}".format(epsilon, correct,  len(test_loader), final_acc))
       return final_acc, adv_examples
```

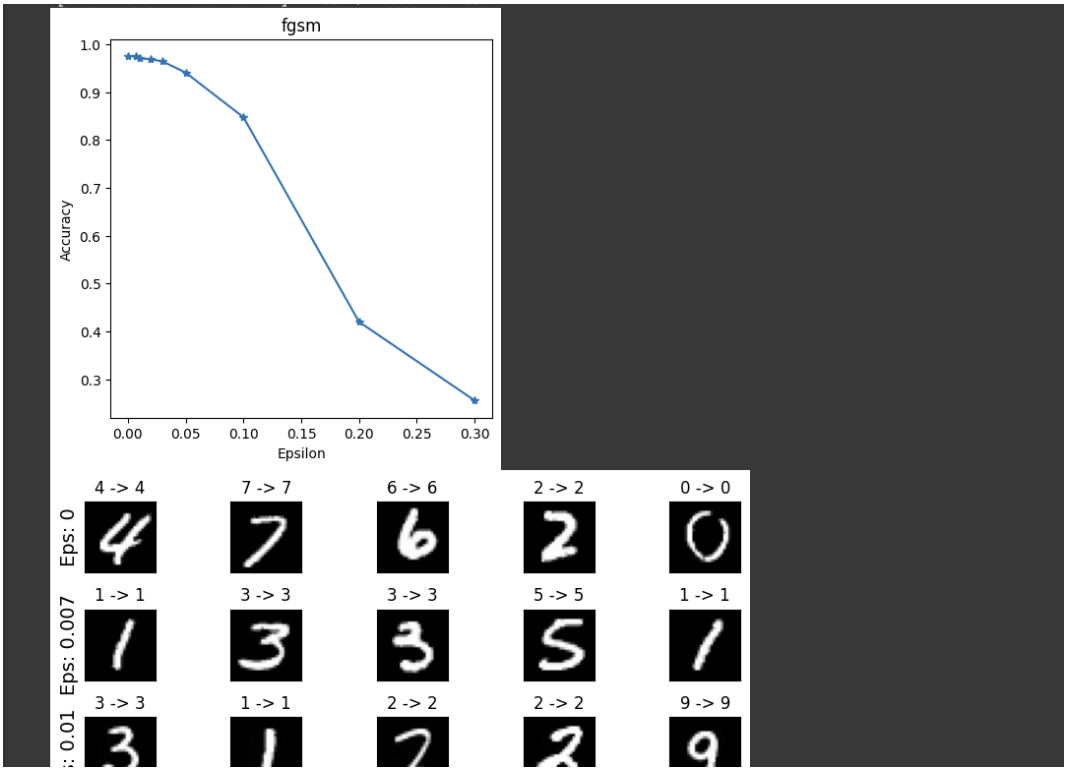
12. Построение графиков успешности атак(Accuracy/эпсилон) и примеры выполненных атак в зависимости от степени возмущения epsilon(на моменте остановил расчет, так как гугл начал меня выбивать по OOM-киллеру. Остальная часть в jupyter-ноутбуке):

12. Построение графиков успешности атак(Accuracy/эпсилон) и примеры выполненных атак в зависимости от степени возмущения epsilon:

+ Код + Текст

```
epsilons = [0,0.007,0.01,0.02,0.03,0.05,0.1,0.2,0.3]
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()
```

```
Epsilon: 0      Test Accuracy = 9750 / 10000 = 0.975
Epsilon: 0.007  Test Accuracy = 9747 / 10000 = 0.9747
Epsilon: 0.01   Test Accuracy = 9708 / 10000 = 0.9708
Epsilon: 0.02   Test Accuracy = 9688 / 10000 = 0.9688
Epsilon: 0.03   Test Accuracy = 9643 / 10000 = 0.9643
Epsilon: 0.05   Test Accuracy = 9409 / 10000 = 0.9409
Epsilon: 0.1    Test Accuracy = 8476 / 10000 = 0.8476
Epsilon: 0.2    Test Accuracy = 4199 / 10000 = 0.4199
Epsilon: 0.3    Test Accuracy = 2556 / 10000 = 0.2556
```



13. Создание двух классов НС

13. Создание двух классов НС

```
[13] class NetF(nn.Module):
       def __init__(self):
         super(NetF, 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)
         return x

     class NetF1(nn.Module):
       def __init__(self):
         super(NetF1, self).__init__()
         self.conv1 = nn.Conv2d(1, 16, 3, 1)
         self.conv2 = nn.Conv2d(16, 32, 3, 1)
         self.dropout1 = nn.Dropout2d(0.25)
         self.dropout2 = nn.Dropout2d(0.5)
         self.fc1 = nn.Linear(4608, 64)
         self.fc2 = nn.Linear(64, 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)
         return x
```

14.Переопределение функций обучения и тестирования

14. Переопределение функций обучения и тестирования

0 сек.

```
def fit(model,device,optimizer,scheduler,criterion,train_loader,val_loader,Temp,epochs):
  data_loader = {'train':train_loader,'val':val_loader}
  print("Fitting the model...")
  train_loss,val_loss=[],[]
  for epoch in range(epochs):
    loss_per_epoch,val_loss_per_epoch=0,0
    for phase in ('train','val'):
      for i,data in enumerate(data_loader[phase]):
        input,label = data[0].to(device),data[1].to(device)
        output = model(input)
        output = F.log_softmax(output/Temp,dim=1)
        #calculating loss on the output
        loss = criterion(output,label)
        if phase == 'train':
          optimizer.zero_grad()
          #grad calc w.r.t Loss func
          loss.backward()
          #update weights
          optimizer.step()
          loss_per_epoch+=loss.item()
      else:
        val_loss_per_epoch+=loss.item()
    scheduler.step(val_loss_per_epoch/len(val_loader))
    print("Epoch: {} Loss: {} Val_Loss: {}".format(epoch+1,loss_per_epoch/len(train_loader),val_loss_per_epoch/len(val_loader)))
    train_loss.append(loss_per_epoch/len(train_loader))
    val_loss.append(val_loss_per_epoch/len(val_loader))
  return train_loss,val_loss
def test(model,device,test_loader,epsilon,Temp,attack):
  correct=0
  adv_examples = []
  for data, target in test_loader:
    data, target = data.to(device), target.to(device)
    data.requires_grad = True
    output = model(data)
    output = F.log_softmax(output/Temp,dim=1)
    init_pred = output.max(1, keepdim=True)[1]
    if init_pred.item() != target.item():
      continue
    loss = F.nll_loss(output, target)
    model.zero_grad()
    loss.backward()
    data_grad = data.grad.data
    if attack == "fgsm":
      perturbed_data = fgsm_attack(data,epsilon,data_grad)
    elif attack == "ifgsm":
      perturbed_data = ifgsm_attack(data,epsilon,data_grad)
    elif attack == "mifgsm":
      perturbed_data = mifgsm_attack(data,epsilon,data_grad)
    output = model(perturbed_data)
    final_pred = output.max(1, keepdim=True)[1]
    if final_pred.item() == target.item():
      correct += 1
    if (epsilon == 0) and (len(adv_examples) < 5):
      adv_ex = perturbed_data.squeeze().detach().cpu().numpy()
      adv_examples.append( (init_pred.item(), final_pred.item(), adv_ex) )
    else:
      if len(adv_examples) < 5:
        adv_ex = perturbed_data.squeeze().detach().cpu().numpy()
        adv_examples.append( (init_pred.item(), final_pred.item(), adv_ex) )
  final_acc = correct/float(len(test_loader))
  print("Epsilon: {}\tTest Accuracy = {} / {} = {}".format(epsilon, correct, len(test_loader), final_acc))
  return final_acc,adv_examples
```

15.Создание функции защиты методом дистилляции

15. Создание функции защиты методом дистилляции

0 сек.

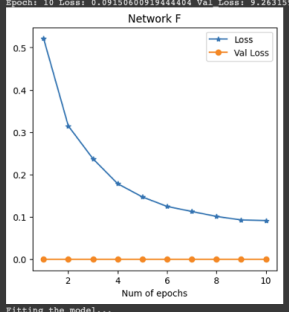
```
def defense(device,train_loader,val_loader,test_loader,epochs,Temp,epsilons):
  modelF = NetF().to(device)
  optimizerF = optim.Adam(modelF.parameters(),lr=0.0001, betas=(0.9, 0.999))
  schedulerF = optim.lr_scheduler.ReduceLROnPlateau(optimizerF, mode='min', factor=0.1, patience=3)
  modelF1 = NetF1().to(device)
  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,1,"fgsm")
      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()
```

16. Получение результатов оценки защищенных сетей(остальная часть в jupyter-notebook)

16. Получение результатов оценки защищенных сетей

```
[ ] Temp=100
    epochs=10
    epsilons=[0,0.007,0.01,0.02,0.03,0.05,0.1,0.2,0.3]
    defense(device,train_loader,val_loader,test_loader,epochs,Temp,epsilons)
```

```
Fitting the model...
/usr/local/lib/python3.10/dist-packages/torch/nn/functional.py:1345: UserWarning: dropout2d: Received a 2-D input to dropout2d, which is deprecated and will result in an error in a future release. To retain the behavior and
  warnings.warn(warn_msg)
Epoch: 1 Loss: 0.5211129647692813 Val_Loss: 6.07259515672922216e-06
Epoch: 2 Loss: 0.31479454748224694 Val_Loss: 2.613734174519777 3e-06
Epoch: 3 Loss: 0.23722507955904096 Val_Loss: 5.431063724972773e-06
Epoch: 4 Loss: 0.17836627665879037 Val_Loss: 2.5205282960087062e-06
Epoch: 5 Loss: 0.14717514184860905 Val_Loss: 9.877499542199076e-07
Epoch: 6 Loss: 0.12481184843863251 Val_Loss: 5.179079668596387e-06
Epoch: 7 Loss: 0.11286765317240482 Val_Loss: 1.7522821872262283e-08
Epoch: 8 Loss: 0.10121211530553217 Val_Loss: 1.0159914963878691e-07
Epoch: 9 Loss: 0.09293931677027 39 Val_Loss: 5.102030610260044 5e-09
Epoch: 10 Loss: 0.09150600919444404 Val_Loss: 9.263159823603928e-07
```




```
Fitting the model...
Epoch: 1 Loss: 0.6893007072611375 Val_Loss: 0.00022473558187484 74
Epoch: 2 Loss: 0.4602636080048564 Val_Loss: 1.090160533785 82e-05
Epoch: 3 Loss: 0.3995274279810201 Val_Loss: 6.612633843906224e-06
Epoch: 4 Loss: 0.34366831902056744 Val_Loss: 0.00034273226994555646
Epoch: 5 Loss: 0.2936364406243109 Val_Loss: 0.00071938800811767 58
Epoch: 6 Loss: 0.24870185530673872 Val_Loss: 1.9565903116017 58e-06
Epoch: 7 Loss: 0.22210440625613329 Val_Loss: 5.164160281419754e-05
Epoch: 8 Loss: 0.20063823280582 38 Val_Loss: 4.078779071569442 6e-05
Epoch: 9 Loss: 0.18425366333763396 Val_Loss: 2.265745773904370 6e-07
Epoch: 10 Loss: 0.17008018550440476 Val_Loss: 1.871431805193424 4e-05
```

Вывод

Метод защитной дистиляции (Protective Distillation) представляет собой стратегию обеспечения безопасности данных и информации от несанкционированного доступа или утечек. Этот метод основан на принципе разделения информации на несколько компонентов или элементов, где каждый из них не предоставляет полной или достаточно информативной картины без наличия остальных частей.

Основная идея защитной дистиляции заключается в том, что доступ к информации возможен только при наличии всех ее компонентов или элементов. Таким образом, информация остается безопасной даже в случае, если одна или несколько ее частей попадают в руки неавторизованного лица или подвергаются компрометации. Таким образом, данные становятся полезными только при условии, что все их элементы находятся в безопасности и доступны только авторизованным пользователям.

Оценка стойкости модели выглядит следующим образом:

- Атака fgsm снизила точность обычной модели до -21%, защищенной - до -90%;
- Атака ifgsm снизила точность не защищенных данных до -29%, защищенных - до -90%;
- Атака mifgsm снизила точность не защищенных данных до -29%, защищенных - до -91%.