Лабораторная работа **3. MNIST (**Мигранов Денис Игоревич, группа **20225).**

In [ ]:

```
import torch
import torchvision
from torchvision import transforms

import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [ ]:

```
batch_size = 4

transform = transforms.Compose(
    [transforms.ToTensor()])

trainset = torchvision.datasets.MNIST(root='./data', train=True,
                                      download=True, transform=transform)
trainloader = torch.utils.data.DataLoader(trainset, batch_size=batch_size,
                                          shuffle=True, num_workers=2)

testset = torchvision.datasets.MNIST(root='./data', train=False,
                                     download=True, transform=transform)
testloader = torch.utils.data.DataLoader(testset, batch_size=batch_size,
                                         shuffle=False, num_workers=2)

classes = tuple(str(i) for i in range(10))
```

```
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/r
aw/train-images-idx3-ubyte.gz


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/r
aw/train-labels-idx1-ubyte.gz


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/ra
w/t10k-images-idx3-ubyte.gz


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/ra
w/t10k-labels-idx1-ubyte.gz


Extracting ./data/MNIST/raw/t10k-labels-idx1-ubyte.gz to ./data/MNIST/raw
```

In [ ]:

```
import torch.nn as nn
import torch.nn.functional as F  # Functional
```

In [ ]:

In [ ]:

```
from tqdm import tqdm_notebook
```

In [ ]:

```
import math
```

In [ ]:

```
# класс наследуется от nn.Module
class SimpleConvNet1(nn.Module):
    def __init__(self, channels1, channels2, kernel_size1, kernel_size2, is_max_pool = T
rue):
        # вызов конструктора предка
        super(SimpleConvNet1, self).__init__()
        # необходмо заранее знать, сколько каналов у картинки (сейчас = 1),
        # которую будем подавать в сеть, больше ничего
        # про входящие картинки знать не нужно
        self.conv1 = nn.Conv2d(in_channels=1, out_channels=channels1, kernel_size=5)
        new_size = 28 - kernel_size1 + 1
        if is_max_pool:
          self.pool = nn.MaxPool2d(kernel_size=2, stride=2)
        else:
          self.pool = nn.AvgPool2d(kernel_size=2, stride=2)
        new_size = new_size // 2
        self.conv2 = nn.Conv2d(in_channels=channels1, out_channels=channels2, kernel_siz
e=5)
        new_size = new_size - kernel_size2 + 1
        new_size = new_size // 2

        self.fc1_size = new_size * new_size * channels2

        self.fc1 = nn.Linear(new_size * new_size * channels2, 120)  # !!!
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        #print(x.shape)
        x = self.pool(F.relu(self.conv2(x)))
        #print(x.shape)
        x = x.view(-1, self.fc1_size)  # !!!
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x

    def train(self, learning_rate = 1e-4, num_epochs = 3):
        loss_fn = torch.nn.CrossEntropyLoss()
        optimizer = torch.optim.Adam(self.parameters(), lr=learning_rate)
        # итерируемся
        for epoch in tqdm_notebook(range(num_epochs)):
            running_loss = 0.0
            for i, batch in enumerate(tqdm_notebook(trainloader)):
                # так получаем текущий батч
                X_batch, y_batch = batch

                # обнуляем веса
                optimizer.zero_grad()

                # forward + backward + optimize
                y_pred = self(X_batch)
                loss = loss_fn(y_pred, y_batch)
                loss.backward()
                optimizer.step()

                # выведем текущий loss
                running_loss += loss.item()
                # выведем качество каждые 2000 батчей
                if i % 2000 == 1999:
                    print('[%d, %5d] loss: %.3f' %
```

```
                        (epoch + 1, i + 1, running_loss / 2000))
                running_loss = 0.0

        print('Обучение закончено')
```

In [ ]:

```
net = SimpleConvNet1(6, 16, 5, 5) #как в примере
net.train()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:43: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:45: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
[1,  2000] loss: 1.071
[1,  4000] loss: 0.427
[1,  6000] loss: 0.324
[1,  8000] loss: 0.281
[1, 10000] loss: 0.224
[1, 12000] loss: 0.196
[1, 14000] loss: 0.175

[2,  2000] loss: 0.157
[2,  4000] loss: 0.130
[2,  6000] loss: 0.121
[2,  8000] loss: 0.122
[2, 10000] loss: 0.114
[2, 12000] loss: 0.115
[2, 14000] loss: 0.099

[3,  2000] loss: 0.088
[3,  4000] loss: 0.097
[3,  6000] loss: 0.082
[3,  8000] loss: 0.084
[3, 10000] loss: 0.083
[3, 12000] loss: 0.081
[3, 14000] loss: 0.084
Обучение закончено
```

In [ ]:

```
def check_network(net):
  class_correct = list(0. for i in range(10))
  class_total = list(0. for i in range(10))

  with torch.no_grad():
      for data in testloader:
          images, labels = data
          y_pred = net(images)
          _, predicted = torch.max(y_pred, 1)
          c = (predicted == labels).squeeze()
          for i in range(4):
              label = labels[i]
              class_correct[label] += c[i].item()
              class_total[label] += 1

  for i in range(10):
      print('Accuracy of %2s : %2d %%' % (
          classes[i], 100 * class_correct[i] / class_total[i]))

  class_correct_t = sum(class_correct)
  class_total_t = sum(class_total)

  print('\nTotal accuracy:', (100. * class_correct_t / class_total_t))
```

In [ ]:

In [ ]:

```
check_network(net)
```

```
Accuracy of  0 : 99 %
Accuracy of  1 : 99 %
Accuracy of  2 : 99 %
Accuracy of  3 : 98 %
Accuracy of  4 : 98 %
Accuracy of  5 : 96 %
Accuracy of  6 : 98 %
Accuracy of  7 : 96 %
Accuracy of  8 : 97 %
Accuracy of  9 : 97 %

Total accuracy: 98.17
```

Попробуем **Average Pooling** с теми же параметрами**:**

In [ ]:

```
net_avg1 = SimpleConvNet1(6, 16, 5, 5, False) #как в примере, Но с Avg пулингом
net_avg1.train()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:43: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:45: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
[1,  2000] loss: 1.060
[1,  4000] loss: 0.473
[1,  6000] loss: 0.392
[1,  8000] loss: 0.342
[1, 10000] loss: 0.312
[1, 12000] loss: 0.278
[1, 14000] loss: 0.242

[2,  2000] loss: 0.211
[2,  4000] loss: 0.194
[2,  6000] loss: 0.175
[2,  8000] loss: 0.161
[2, 10000] loss: 0.147
[2, 12000] loss: 0.152
[2, 14000] loss: 0.139

[3,  2000] loss: 0.120
[3,  4000] loss: 0.106
[3,  6000] loss: 0.106
[3,  8000] loss: 0.106
[3, 10000] loss: 0.101
[3, 12000] loss: 0.097
[3, 14000] loss: 0.102
Обучение закончено
```

In [ ]:

```
check_network(net_avg1)
```

```
Accuracy of  0 : 99 %
Accuracy of  1 : 99 %
Accuracy of  2 : 98 %
Accuracy of  3 : 96 %
Accuracy of  4 : 96 %
Accuracy of  5 : 97 %
Accuracy of  6 : 97 %
Accuracy of  7 : 97 %
Accuracy of  8 : 97 %
```

```
Accuracy of  9 : 96 %

Total accuracy: 97.67
```

Max с остальными такими же параметрами был лучше**...**

Добавим в конструктор возможность регулировать размер **fully connected** слоёв**:**

In [ ]:

```
import math

# класс наследуется от nn.Module
class SimpleConvNet2(nn.Module):
    def __init__(self, channels1, channels2, kernel_size1, kernel_size2, fc1, fc2, is_ma
x_pool = True):
        # вызов конструктора предка
        super(SimpleConvNet2, self).__init__()
        # необходмо заранее знать, сколько каналов у картинки (сейчас = 1),
        # которую будем подавать в сеть, больше ничего
        # про входящие картинки знать не нужно
        self.conv1 = nn.Conv2d(in_channels=1, out_channels=channels1, kernel_size=kernel
_size1)
        new_size = 28 - kernel_size1 + 1
        if is_max_pool:
          self.pool = nn.MaxPool2d(kernel_size=2, stride=2)
        else:
          self.pool = nn.AvgPool2d(kernel_size=2, stride=2)
        new_size = new_size // 2
        self.conv2 = nn.Conv2d(in_channels=channels1, out_channels=channels2, kernel_siz
e=kernel_size2)
        new_size = new_size - kernel_size2 + 1
        new_size = new_size // 2

        self.fc1_size = new_size * new_size * channels2

        self.fc1 = nn.Linear(new_size * new_size * channels2, fc1)  # !!!
        self.fc2 = nn.Linear(fc1, fc2)
        self.fc3 = nn.Linear(fc2, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        #print(x.shape)
        x = self.pool(F.relu(self.conv2(x)))
        #print(x.shape)
        x = x.view(-1, self.fc1_size)  # !!!
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x

    def train(self, train_loader, learning_rate = 1e-4, num_epochs = 3):
        loss_fn = torch.nn.CrossEntropyLoss()
        optimizer = torch.optim.Adam(self.parameters(), lr=learning_rate)
        # итерируемся
        for epoch in tqdm_notebook(range(num_epochs)):
            running_loss = 0.0
            for i, batch in enumerate(tqdm_notebook(train_loader)):
                # так получаем текущий батч
                X_batch, y_batch = batch

                # обнуляем веса
                optimizer.zero_grad()

                # forward + backward + optimize
                y_pred = self(X_batch)
                loss = loss_fn(y_pred, y_batch)
                loss.backward()
                optimizer.step()

                # выведем текущий loss
```

```
                running_loss += loss.item()
                # выведем качество каждые 2000 батчей
                if i % 2000 == 1999:
                    print('[%d, %5d] loss: %.3f' %
                          (epoch + 1, i + 1, running_loss / 2000))
                    running_loss = 0.0

        print('Обучение закончено')

    def predict(self, test_loader):
        class_correct = list(0. for i in range(10))
        class_total = list(0. for i in range(10))

        with torch.no_grad():
            for data in test_loader:
                images, labels = data
                y_pred = self(images)
                _, predicted = torch.max(y_pred, 1)
                c = (predicted == labels).squeeze()
                for i in range(4):
                    label = labels[i]
                    class_correct[label] += c[i].item()
                    class_total[label] += 1

        for i in range(10):
            print('Accuracy of %2s : %2d %%' % (
                classes[i], 100 * class_correct[i] / class_total[i]))

        class_correct_t = sum(class_correct)
        class_total_t = sum(class_total)

        print('\nTotal accuracy:', (100. * class_correct_t / class_total_t))
```

In [ ]:

```
net_2 = SimpleConvNet2(6, 16, 5, 5, 200, 100, False) #с Avg пулингом
net_2.train(trainloader, num_epochs=4)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:43: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:45: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
[1,  2000] loss: 0.985
[1,  4000] loss: 0.449
[1,  6000] loss: 0.383
[1,  8000] loss: 0.339
[1, 10000] loss: 0.268
[1, 12000] loss: 0.252
[1, 14000] loss: 0.233

[2,  2000] loss: 0.196
[2,  4000] loss: 0.181
[2,  6000] loss: 0.163
[2,  8000] loss: 0.154
[2, 10000] loss: 0.143
[2, 12000] loss: 0.139
[2, 14000] loss: 0.127

[3,  2000] loss: 0.114
[3,  4000] loss: 0.116
[3,  6000] loss: 0.114
[3,  8000] loss: 0.096
[3, 10000] loss: 0.101
[3, 12000] loss: 0.093
[3, 14000] loss: 0.089
```

```
[4,  2000] loss: 0.084
[4,  4000] loss: 0.075
[4,  6000] loss: 0.075
[4,  8000] loss: 0.075
[4, 10000] loss: 0.072
[4, 12000] loss: 0.077
[4, 14000] loss: 0.075
Обучение закончено
```

In [ ]:

```
check_network(net_2)
```

```
Accuracy of  0 : 99 %
Accuracy of  1 : 99 %
Accuracy of  2 : 98 %
Accuracy of  3 : 98 %
Accuracy of  4 : 98 %
Accuracy of  5 : 98 %
Accuracy of  6 : 97 %
Accuracy of  7 : 96 %
Accuracy of  8 : 95 %
Accuracy of  9 : 97 %

Total accuracy: 97.86
```

In [ ]:

```
net_3 = SimpleConvNet2(6, 16, 5, 5, 200, 100) #с Max пулингом
net_3.train(trainloader, num_epochs=4)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:43: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:45: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
[1,  2000] loss: 0.907
[1,  4000] loss: 0.374
[1,  6000] loss: 0.287
[1,  8000] loss: 0.232
[1, 10000] loss: 0.206
[1, 12000] loss: 0.174
[1, 14000] loss: 0.145

[2,  2000] loss: 0.145
[2,  4000] loss: 0.121
[2,  6000] loss: 0.114
[2,  8000] loss: 0.108
[2, 10000] loss: 0.103
[2, 12000] loss: 0.091
[2, 14000] loss: 0.084

[3,  2000] loss: 0.078
[3,  4000] loss: 0.076
[3,  6000] loss: 0.082
[3,  8000] loss: 0.071
[3, 10000] loss: 0.072
[3, 12000] loss: 0.074
[3, 14000] loss: 0.063

[4,  2000] loss: 0.056
[4,  4000] loss: 0.064
[4,  6000] loss: 0.054
[4,  8000] loss: 0.062
[4, 10000] loss: 0.050
[4, 12000] loss: 0.054
```

```
[4, 14000] loss: 0.056
Обучение закончено
```

In [ ]:

```
net_3.predict(testloader)
```

```
Accuracy of  0 : 99 %
Accuracy of  1 : 99 %
Accuracy of  2 : 98 %
Accuracy of  3 : 98 %
Accuracy of  4 : 99 %
Accuracy of  5 : 99 %
Accuracy of  6 : 98 %
Accuracy of  7 : 97 %
Accuracy of  8 : 99 %
Accuracy of  9 : 97 %

Total accuracy: 98.7
```

Попробуем добавить трансформы картинкам...

In [ ]:

```
batch_size = 4

transform = transforms.Compose(
    [transforms.ToTensor(), transforms.RandomRotation(30)])

trainset_t = torchvision.datasets.MNIST(root='./data', train=True,
                                        download=True, transform=transform)
trainset_inc = torch.utils.data.ConcatDataset([trainset,trainset_t])
trainloader_inc = torch.utils.data.DataLoader(trainset_inc, batch_size=batch_size,
                                              shuffle=True, num_workers=2)
```

In [ ]:

```
net_4 = SimpleConvNet2(6, 16, 5, 5, 200, 100) #с Max пулингом
net_4.train(trainloader_inc, num_epochs=2)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:43: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:45: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
[1,  2000] loss: 1.077
[1,  4000] loss: 0.446
[1,  6000] loss: 0.342
[1,  8000] loss: 0.312
[1, 10000] loss: 0.266
[1, 12000] loss: 0.231
[1, 14000] loss: 0.214
[1, 16000] loss: 0.199
[1, 18000] loss: 0.173
[1, 20000] loss: 0.177
[1, 22000] loss: 0.170
[1, 24000] loss: 0.170
[1, 26000] loss: 0.147
[1, 28000] loss: 0.143
[1, 30000] loss: 0.128

[2,  2000] loss: 0.131
[2,  4000] loss: 0.120
[2,  6000] loss: 0.119
[2,  8000] loss: 0.120
[2, 10000] loss: 0.124
[2, 12000] loss: 0.112
```

```
[2, 14000] loss: 0.098
[2, 16000] loss: 0.106
[2, 18000] loss: 0.100
[2, 20000] loss: 0.091
[2, 22000] loss: 0.095
[2, 24000] loss: 0.086
[2, 26000] loss: 0.089
[2, 28000] loss: 0.091
[2, 30000] loss: 0.087
Обучение закончено
```

In [ ]:

```
net_4.predict(testloader)
```

```
Accuracy of  0 : 98 %
Accuracy of  1 : 99 %
Accuracy of  2 : 98 %
Accuracy of  3 : 99 %
Accuracy of  4 : 98 %
Accuracy of  5 : 97 %
Accuracy of  6 : 98 %
Accuracy of  7 : 97 %
Accuracy of  8 : 95 %
Accuracy of  9 : 97 %

Total accuracy: 98.21
```

In [ ]:

```
net_4.train(trainloader_inc, num_epochs=1)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:43: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:45: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
[1,  2000] loss: 0.080
[1,  4000] loss: 0.086
[1,  6000] loss: 0.086
[1,  8000] loss: 0.073
[1, 10000] loss: 0.073
[1, 12000] loss: 0.070
[1, 14000] loss: 0.081
[1, 16000] loss: 0.075
[1, 18000] loss: 0.075
[1, 20000] loss: 0.069
[1, 22000] loss: 0.065
[1, 24000] loss: 0.074
[1, 26000] loss: 0.071
[1, 28000] loss: 0.068
[1, 30000] loss: 0.060
Обучение закончено
```

In [ ]:

```
net_4.predict(testloader)
```

```
Accuracy of  0 : 99 %
Accuracy of  1 : 99 %
Accuracy of  2 : 98 %
Accuracy of  3 : 98 %
Accuracy of  4 : 98 %
Accuracy of  5 : 97 %
Accuracy of  6 : 98 %
Accuracy of  7 : 98 %
Accuracy of  8 : 98 %
Accuracy of  9 : 97 %
```

```
Total accuracy: 98.53
```

In [ ]:

```
net_4.train(trainloader_inc, num_epochs=1)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:43: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:45: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
[1,  2000] loss: 0.059
[1,  4000] loss: 0.063
[1,  6000] loss: 0.058
[1,  8000] loss: 0.066
[1, 10000] loss: 0.056
[1, 12000] loss: 0.059
[1, 14000] loss: 0.060
[1, 16000] loss: 0.056
[1, 18000] loss: 0.056
[1, 20000] loss: 0.059
[1, 22000] loss: 0.057
[1, 24000] loss: 0.056
[1, 26000] loss: 0.054
[1, 28000] loss: 0.059
[1, 30000] loss: 0.055
Обучение закончено
```

In [ ]:

```
net_4.predict(testloader)
```

```
Accuracy of  0 : 99 %
Accuracy of  1 : 99 %
Accuracy of  2 : 99 %
Accuracy of  3 : 98 %
Accuracy of  4 : 97 %
Accuracy of  5 : 98 %
Accuracy of  6 : 99 %
Accuracy of  7 : 99 %
Accuracy of  8 : 97 %
Accuracy of  9 : 98 %

Total accuracy: 98.87
```

In [ ]:

```
net_4.train(trainloader_inc, num_epochs=1)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:43: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:45: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
[1,  2000] loss: 0.048
[1,  4000] loss: 0.055
[1,  6000] loss: 0.054
[1,  8000] loss: 0.045
[1, 10000] loss: 0.055
[1, 12000] loss: 0.049
[1, 14000] loss: 0.051
[1, 16000] loss: 0.050
[1, 18000] loss: 0.037
```

```
[1, 20000] loss: 0.044
[1, 22000] loss: 0.040
[1, 24000] loss: 0.051
[1, 26000] loss: 0.047
[1, 28000] loss: 0.045
[1, 30000] loss: 0.044
Обучение закончено
```

In [ ]:

```
net_4.predict(testloader) #итого 5 эпох
```

```
Accuracy of  0 : 99 %
Accuracy of  1 : 99 %
Accuracy of  2 : 99 %
Accuracy of  3 : 99 %
Accuracy of  4 : 97 %
Accuracy of  5 : 97 %
Accuracy of  6 : 99 %
Accuracy of  7 : 99 %
Accuracy of  8 : 97 %
Accuracy of  9 : 98 %

Total accuracy: 98.78
```

Попробуем другие размеры ядра, а потом увеличим число сверточных слоев

In [ ]:

```
net_5 = SimpleConvNet2(6, 16, 7, 3, 160, 80) #с Max пулингом
net_5.train(trainloader_inc, num_epochs=3)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:43: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:45: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
[1,  2000] loss: 1.078
[1,  4000] loss: 0.487
[1,  6000] loss: 0.418
[1,  8000] loss: 0.367
[1, 10000] loss: 0.306
[1, 12000] loss: 0.278
[1, 14000] loss: 0.270
[1, 16000] loss: 0.229
[1, 18000] loss: 0.211
[1, 20000] loss: 0.194
[1, 22000] loss: 0.186
[1, 24000] loss: 0.168
[1, 26000] loss: 0.168
[1, 28000] loss: 0.155
[1, 30000] loss: 0.135

[2,  2000] loss: 0.147
[2,  4000] loss: 0.129
[2,  6000] loss: 0.130
[2,  8000] loss: 0.137
[2, 10000] loss: 0.125
[2, 12000] loss: 0.119
[2, 14000] loss: 0.114
[2, 16000] loss: 0.113
[2, 18000] loss: 0.110
[2, 20000] loss: 0.113
[2, 22000] loss: 0.109
[2, 24000] loss: 0.102
[2, 26000] loss: 0.110
[2, 28000] loss: 0.101
```

```
[2, 30000] loss: 0.101

[3,  2000] loss: 0.083
[3,  4000] loss: 0.087
[3,  6000] loss: 0.088
[3,  8000] loss: 0.086
[3, 10000] loss: 0.083
[3, 12000] loss: 0.087
[3, 14000] loss: 0.088
[3, 16000] loss: 0.085
[3, 18000] loss: 0.083
[3, 20000] loss: 0.085
[3, 22000] loss: 0.076
[3, 24000] loss: 0.075
[3, 26000] loss: 0.078
[3, 28000] loss: 0.075
[3, 30000] loss: 0.077
Обучение закончено
```

In [ ]:

```
net_5.predict(testloader)
```

```
Accuracy of  0 : 99 %
Accuracy of  1 : 99 %
Accuracy of  2 : 98 %
Accuracy of  3 : 97 %
Accuracy of  4 : 98 %
Accuracy of  5 : 96 %
Accuracy of  6 : 98 %
Accuracy of  7 : 98 %
Accuracy of  8 : 97 %
Accuracy of  9 : 96 %

Total accuracy: 98.35
```

Добавим ещё сверточный слой

In [ ]:

```
# класс наследуется от nn.Module
class SimpleConvNet3C(nn.Module):
    def __init__(self, channels1, channels2, channels3, kernel_size1, kernel_size2, kern
el_size3, fc1, fc2, is_max_pool = True):
        # вызов конструктора предка
        super(SimpleConvNet3C, self).__init__()

        self.conv1 = nn.Conv2d(in_channels=1, out_channels=channels1, kernel_size=kernel
_size1)
        new_size = 28 - kernel_size1 + 1

        if is_max_pool:
          self.pool = nn.MaxPool2d(kernel_size=2, stride=2)
        else:
          self.pool = nn.AvgPool2d(kernel_size=2, stride=2)

        new_size = new_size // 2
        self.conv2 = nn.Conv2d(in_channels=channels1, out_channels=channels2, kernel_siz
e=kernel_size2)
        new_size = new_size - kernel_size2 + 1
        new_size = new_size // 2

        self.conv3 = nn.Conv2d(in_channels=channels2, out_channels=channels3, kernel_siz
e=kernel_size3)
        new_size = new_size - kernel_size3 + 1
        #new_size = new_size // 2
        #print(new_size)
        self.fc1_size = new_size * new_size * channels3

        self.fc1 = nn.Linear(new_size * new_size * channels3, fc1)   # !!!
```

```
        self.fc2 = nn.Linear(fc1, fc2)
        self.fc3 = nn.Linear(fc2, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        #print(x.shape)
        x = self.pool(F.relu(self.conv2(x)))
        #print(x.shape)
        x = F.relu(self.conv3(x))         #x = self.pool(F.relu(self.conv3(x)))
        #print(x.shape)
        x = x.view(-1, self.fc1_size)  # !!!
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x

    def train(self, train_loader, learning_rate = 1e-4, num_epochs = 3):
        loss_fn = torch.nn.CrossEntropyLoss()
        optimizer = torch.optim.Adam(self.parameters(), lr=learning_rate)
        # итерируемся
        for epoch in tqdm_notebook(range(num_epochs)):
            running_loss = 0.0
            for i, batch in enumerate(tqdm_notebook(train_loader)):
                # так получаем текущий батч
                X_batch, y_batch = batch

                # обнуляем веса
                optimizer.zero_grad()

                # forward + backward + optimize
                y_pred = self(X_batch)
                loss = loss_fn(y_pred, y_batch)
                loss.backward()
                optimizer.step()

                # выведем текущий loss
                running_loss += loss.item()
                # выведем качество каждые 2000 батчей
                if i % 2000 == 1999:
                    print('[%d, %5d] loss: %.3f' %
                          (epoch + 1, i + 1, running_loss / 2000))
                    running_loss = 0.0

        print('Обучение закончено')

    def predict(self, test_loader):
        class_correct = list(0. for i in range(10))
        class_total = list(0. for i in range(10))

        with torch.no_grad():
            for data in test_loader:
                images, labels = data
                y_pred = self(images)
                _, predicted = torch.max(y_pred, 1)
                c = (predicted == labels).squeeze()
                for i in range(4):
                    label = labels[i]
                    class_correct[label] += c[i].item()
                    class_total[label] += 1

        for i in range(10):
            print('Accuracy of %2s : %2d %%' % (
                classes[i], 100 * class_correct[i] / class_total[i]))

        class_correct_t = sum(class_correct)
        class_total_t = sum(class_total)

        print('\nTotal accuracy:', (100. * class_correct_t / class_total_t))
```

In [ ]:

```
net_6 = SimpleConvNet3C(6, 12, 30, 3, 3, 5, 100, 60) #с Max пулингом
```

```
net_6.train(trainloader_inc, num_epochs=3, learning_rate = 1e-3)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:47: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:49: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
[1,  2000] loss: 0.734
[1,  4000] loss: 0.295
[1,  6000] loss: 0.241
[1,  8000] loss: 0.201
[1, 10000] loss: 0.192
[1, 12000] loss: 0.172
[1, 14000] loss: 0.161
[1, 16000] loss: 0.156
[1, 18000] loss: 0.143
[1, 20000] loss: 0.148
[1, 22000] loss: 0.138
[1, 24000] loss: 0.129
[1, 26000] loss: 0.135
[1, 28000] loss: 0.124
[1, 30000] loss: 0.122

[2,  2000] loss: 0.114
[2,  4000] loss: 0.104
[2,  6000] loss: 0.113
[2,  8000] loss: 0.110
[2, 10000] loss: 0.109
[2, 12000] loss: 0.100
[2, 14000] loss: 0.107
[2, 16000] loss: 0.103
[2, 18000] loss: 0.095
[2, 20000] loss: 0.096
[2, 22000] loss: 0.106
[2, 24000] loss: 0.099
[2, 26000] loss: 0.091
[2, 28000] loss: 0.098
[2, 30000] loss: 0.112

[3,  2000] loss: 0.089
[3,  4000] loss: 0.086
[3,  6000] loss: 0.087
[3,  8000] loss: 0.092
[3, 10000] loss: 0.090
[3, 12000] loss: 0.091
[3, 14000] loss: 0.086
[3, 16000] loss: 0.099
[3, 18000] loss: 0.077
[3, 20000] loss: 0.095
[3, 22000] loss: 0.087
[3, 24000] loss: 0.083
[3, 26000] loss: 0.091
[3, 28000] loss: 0.089
[3, 30000] loss: 0.094
Обучение закончено
```

In [ ]:

```
net_6.predict(testloader)
```

```
Accuracy of  0 : 99 %
Accuracy of  1 : 98 %
Accuracy of  2 : 98 %
Accuracy of  3 : 98 %
Accuracy of  4 : 99 %
Accuracy of  5 : 98 %
Accuracy of  6 : 98 %
```

```
Accuracy of  7 : 98 %
Accuracy of  8 : 97 %
Accuracy of  9 : 96 %

Total accuracy: 98.51
```

Улучшения нет... Вернемся к варианту с двумя слоями и добавим дропаут

In [ ]:

```
from torch.nn import Dropout
```

In [ ]:

```
import math

# класс наследуется от nn.Module
class SimpleConvNet2D(nn.Module):
    def __init__(self, channels1, channels2, kernel_size1, kernel_size2, fc1, dropout, i
s_max_pool = True):
        # вызов конструктора предка
        super(SimpleConvNet2D, self).__init__()
        # необходмо заранее знать, сколько каналов у картинки (сейчас = 1),
        # которую будем подавать в сеть, больше ничего
        # про входящие картинки знать не нужно
        self.conv1 = nn.Conv2d(in_channels=1, out_channels=channels1, kernel_size=kernel
_size1)
        new_size = 28 - kernel_size1 + 1
        if is_max_pool:
          self.pool = nn.MaxPool2d(kernel_size=2, stride=2)
        else:
          self.pool = nn.AvgPool2d(kernel_size=2, stride=2)
        new_size = new_size // 2
        self.conv2 = nn.Conv2d(in_channels=channels1, out_channels=channels2, kernel_siz
e=kernel_size2)
        new_size = new_size - kernel_size2 + 1
        new_size = new_size // 2

        self.fc1_size = new_size * new_size * channels2

        self.fc1 = nn.Linear(new_size * new_size * channels2, fc1)  # !!!
        #self.fc2 = nn.Linear(fc1, fc2)
        self.fc3 = nn.Linear(fc1, 10)

        self.dropout1 = Dropout(dropout)
        self.dropout2 = Dropout(dropout)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        #print(x.shape)
        x = self.pool(F.relu(self.conv2(x)))
        #print(x.shape)
        x = x.view(-1, self.fc1_size)  # !!!
        x = self.dropout1(F.relu(self.fc1(x)))
        #x = self.dropout2(F.relu(self.fc2(x)))
        x = self.fc3(x)
        return x

    def train_network(self, train_loader, learning_rate = 1e-4, num_epochs = 3):
        loss_fn = torch.nn.CrossEntropyLoss()
        optimizer = torch.optim.Adam(self.parameters(), lr=learning_rate)
        # итерируемся
        for epoch in tqdm_notebook(range(num_epochs)):
            running_loss = 0.0
            for i, batch in enumerate(tqdm_notebook(train_loader)):
                # так получаем текущий батч
                X_batch, y_batch = batch

                # обнуляем веса
                optimizer.zero_grad()
```

```
                # forward + backward + optimize
                y_pred = self(X_batch)
                loss = loss_fn(y_pred, y_batch)
                loss.backward()
                optimizer.step()

                # выведем текущий loss
                running_loss += loss.item()
                # выведем качество каждые 2000 батчей
                if i % 2000 == 1999:
                    print('[%d, %5d] loss: %.3f' %
                          (epoch + 1, i + 1, running_loss / 2000))
                    running_loss = 0.0

        print('Обучение закончено')

    def predict(self, test_loader):
        class_correct = list(0. for i in range(10))
        class_total = list(0. for i in range(10))

        self.eval()

        with torch.no_grad():
            for data in test_loader:
                images, labels = data
                y_pred = self(images)
                _, predicted = torch.max(y_pred, 1)
                c = (predicted == labels).squeeze()
                for i in range(4):
                    label = labels[i]
                    class_correct[label] += c[i].item()
                    class_total[label] += 1

        for i in range(10):
            print('Accuracy of %2s : %2d %%' % (
                classes[i], 100 * class_correct[i] / class_total[i]))

        class_correct_t = sum(class_correct)
        class_total_t = sum(class_total)

        print('\nTotal accuracy:', (100. * class_correct_t / class_total_t))

        self.train()
```

In [ ]:

```
net_2D1 = SimpleConvNet2D(6, 16, 5, 5, 200, 0.5) #с Max пулингом
net_2D1.train_network(trainloader_inc, num_epochs=2, learning_rate=1e-3)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:46: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:48: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
[1,  2000] loss: 0.590
[1,  4000] loss: 0.249
[1,  6000] loss: 0.215
[1,  8000] loss: 0.189
[1, 10000] loss: 0.181
[1, 12000] loss: 0.165
[1, 14000] loss: 0.155
[1, 16000] loss: 0.141
[1, 18000] loss: 0.135
[1, 20000] loss: 0.127
[1, 22000] loss: 0.137
[1, 24000] loss: 0.110
[1, 26000] loss: 0.124
[1, 28000] loss: 0.125
```

```
[1, 30000] loss: 0.116

[2,  2000] loss: 0.121
[2,  4000] loss: 0.103
[2,  6000] loss: 0.107
[2,  8000] loss: 0.103
[2, 10000] loss: 0.103
[2, 12000] loss: 0.103
[2, 14000] loss: 0.098
[2, 16000] loss: 0.106
[2, 18000] loss: 0.094
[2, 20000] loss: 0.100
[2, 22000] loss: 0.098
[2, 24000] loss: 0.099
[2, 26000] loss: 0.091
[2, 28000] loss: 0.100
[2, 30000] loss: 0.103
Обучение закончено
```

In [ ]:

```
net_2D1.predict(testloader)
```

```
Accuracy of  0 : 99 %
Accuracy of  1 : 99 %
Accuracy of  2 : 98 %
Accuracy of  3 : 98 %
Accuracy of  4 : 99 %
Accuracy of  5 : 98 %
Accuracy of  6 : 99 %
Accuracy of  7 : 98 %
Accuracy of  8 : 98 %
Accuracy of  9 : 97 %

Total accuracy: 98.75
```

In [ ]:

```
net_2D1.train_network(trainloader_inc, num_epochs=1, learning_rate=1e-3)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:46: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:48: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
[1,  2000] loss: 0.102
[1,  4000] loss: 0.101
[1,  6000] loss: 0.091
[1,  8000] loss: 0.098
[1, 10000] loss: 0.100
[1, 12000] loss: 0.085
[1, 14000] loss: 0.095
[1, 16000] loss: 0.090
[1, 18000] loss: 0.092
[1, 20000] loss: 0.096
[1, 22000] loss: 0.097
[1, 24000] loss: 0.092
[1, 26000] loss: 0.094
[1, 28000] loss: 0.089
[1, 30000] loss: 0.101
Обучение закончено
```

In [ ]:

```
net_2D1.predict(testloader)
```

```
Accuracy of  0 : 99 %
Accuracy of  1 : 99 %
```

```
Accuracy of  2 : 98 %
Accuracy of  3 : 98 %
Accuracy of  4 : 98 %
Accuracy of  5 : 97 %
Accuracy of  6 : 99 %
Accuracy of  7 : 98 %
Accuracy of  8 : 98 %
Accuracy of  9 : 98 %

Total accuracy: 98.67
```

In [ ]:

```
net_2D1.train_network(trainloader_inc, num_epochs=1, learning_rate=5e-4)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:46: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:48: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
[1,  2000] loss: 0.127
[1,  4000] loss: 0.119
[1,  6000] loss: 0.121
[1,  8000] loss: 0.107
[1, 10000] loss: 0.115
[1, 12000] loss: 0.098
[1, 14000] loss: 0.095
[1, 16000] loss: 0.109
[1, 18000] loss: 0.095
[1, 20000] loss: 0.097
[1, 22000] loss: 0.098
[1, 24000] loss: 0.097
[1, 26000] loss: 0.095
[1, 28000] loss: 0.086
[1, 30000] loss: 0.094
Обучение закончено
```

In [ ]:

```
net_2D1.predict(testloader)
```

```
Accuracy of  0 : 99 %
Accuracy of  1 : 99 %
Accuracy of  2 : 98 %
Accuracy of  3 : 99 %
Accuracy of  4 : 97 %
Accuracy of  5 : 97 %
Accuracy of  6 : 98 %
Accuracy of  7 : 98 %
Accuracy of  8 : 97 %
Accuracy of  9 : 96 %

Total accuracy: 98.28
```

In [ ]:

```
net_2D1.train_network(trainloader_inc, num_epochs=2, learning_rate=5e-4)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:46: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:48: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
[1,  2000] loss: 0.093
```

```
[1,  4000] loss: 0.093
[1,  6000] loss: 0.084
[1,  8000] loss: 0.087
[1, 10000] loss: 0.092
[1, 12000] loss: 0.089
[1, 14000] loss: 0.083
[1, 16000] loss: 0.096
[1, 18000] loss: 0.082
[1, 20000] loss: 0.085
[1, 22000] loss: 0.098
[1, 24000] loss: 0.089
[1, 26000] loss: 0.089
[1, 28000] loss: 0.096
[1, 30000] loss: 0.080

[2,  2000] loss: 0.082
[2,  4000] loss: 0.077
[2,  6000] loss: 0.084
[2,  8000] loss: 0.074
[2, 10000] loss: 0.082
[2, 12000] loss: 0.098
[2, 14000] loss: 0.080
[2, 16000] loss: 0.085
[2, 18000] loss: 0.085
[2, 20000] loss: 0.085
[2, 22000] loss: 0.074
[2, 24000] loss: 0.092
[2, 26000] loss: 0.088
[2, 28000] loss: 0.080
[2, 30000] loss: 0.080
Обучение закончено
```

In [ ]:

```
net_2D1.predict(testloader)
```

```
Accuracy of  0 : 99 %
Accuracy of  1 : 99 %
Accuracy of  2 : 98 %
Accuracy of  3 : 98 %
Accuracy of  4 : 97 %
Accuracy of  5 : 98 %
Accuracy of  6 : 98 %
Accuracy of  7 : 99 %
Accuracy of  8 : 99 %
Accuracy of  9 : 97 %

Total accuracy: 98.88
```

In [ ]:

```
net_2D1.train_network(trainloader_inc, num_epochs=1, learning_rate=1e-4)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:46: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:48: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
[1,  2000] loss: 0.072
[1,  4000] loss: 0.063
[1,  6000] loss: 0.055
[1,  8000] loss: 0.061
[1, 10000] loss: 0.069
[1, 12000] loss: 0.064
[1, 14000] loss: 0.065
[1, 16000] loss: 0.063
[1, 18000] loss: 0.064
```

```
[1, 20000] loss: 0.058
[1, 22000] loss: 0.059
[1, 24000] loss: 0.067
[1, 26000] loss: 0.066
[1, 28000] loss: 0.065
[1, 30000] loss: 0.061
Обучение закончено
```

In [ ]:

```
net_2D1.predict(testloader)
```

```
Accuracy of  0 : 99 %
Accuracy of  1 : 99 %
Accuracy of  2 : 99 %
Accuracy of  3 : 99 %
Accuracy of  4 : 98 %
Accuracy of  5 : 98 %
Accuracy of  6 : 98 %
Accuracy of  7 : 98 %
Accuracy of  8 : 99 %
Accuracy of  9 : 98 %

Total accuracy: 98.91
```

In [ ]:

```
net_2D1.train_network(trainloader_inc, num_epochs=1, learning_rate=1e-4)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:46: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:48: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
[1,  2000] loss: 0.061
[1,  4000] loss: 0.063
[1,  6000] loss: 0.064
[1,  8000] loss: 0.063
[1, 10000] loss: 0.057
[1, 12000] loss: 0.068
[1, 14000] loss: 0.067
[1, 16000] loss: 0.057
[1, 18000] loss: 0.055
[1, 20000] loss: 0.048
[1, 22000] loss: 0.063
[1, 24000] loss: 0.053
[1, 26000] loss: 0.055
[1, 28000] loss: 0.068
[1, 30000] loss: 0.062
Обучение закончено
```

In [ ]:

```
net_2D1.predict(testloader)
```

```
Accuracy of  0 : 99 %
Accuracy of  1 : 99 %
Accuracy of  2 : 98 %
Accuracy of  3 : 99 %
Accuracy of  4 : 98 %
Accuracy of  5 : 98 %
Accuracy of  6 : 98 %
Accuracy of  7 : 99 %
Accuracy of  8 : 98 %
Accuracy of  9 : 98 %

Total accuracy: 98.97
```

In [ ]:

```
net_2D1.train_network(trainloader_inc, num_epochs=1, learning_rate=1e-4)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:46: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:48: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
[1,  2000] loss: 0.050
[1,  4000] loss: 0.072
[1,  6000] loss: 0.060
[1,  8000] loss: 0.054
[1, 10000] loss: 0.054
[1, 12000] loss: 0.064
[1, 14000] loss: 0.058
[1, 16000] loss: 0.056
[1, 18000] loss: 0.056
[1, 20000] loss: 0.054
[1, 22000] loss: 0.060
[1, 24000] loss: 0.061
[1, 26000] loss: 0.061
[1, 28000] loss: 0.050
[1, 30000] loss: 0.066
Обучение закончено
```

In [ ]:

```
net_2D1.predict(testloader)
```

```
Accuracy of  0 : 99 %
Accuracy of  1 : 99 %
Accuracy of  2 : 98 %
Accuracy of  3 : 99 %
Accuracy of  4 : 98 %
Accuracy of  5 : 98 %
Accuracy of  6 : 98 %
Accuracy of  7 : 99 %
Accuracy of  8 : 99 %
Accuracy of  9 : 98 %

Total accuracy: 99.07
```

Итак, добились точности **99.07%.** Экспериментировали с разным количеством сверточных слоёв и разным количеством полносвязных. Итого - два сверточных слоя и два полносвязных, между полносвязными - дропаут.

Ради интереса попробуем другую функцию активации

In [ ]:

```
import math
# класс наследуется от nn.Module
class SimpleConvNet2DLeaky(nn.Module):
    def __init__(self, channels1, channels2, kernel_size1, kernel_size2, fc1, dropout, i
s_max_pool = True):
        # вызов конструктора предка
        super(SimpleConvNet2DLeaky, self).__init__()
        # необходмо заранее знать, сколько каналов у картинки (сейчас = 1),
        # которую будем подавать в сеть, больше ничего
        # про входящие картинки знать не нужно
        self.conv1 = nn.Conv2d(in_channels=1, out_channels=channels1, kernel_size=kernel
_size1)
        new_size = 28 - kernel_size1 + 1
        if is_max_pool:
          self.pool = nn.MaxPool2d(kernel_size=2, stride=2)
        else:
```

```
            self.pool = nn.AvgPool2d(kernel_size=2, stride=2)
        new_size = new_size // 2
        self.conv2 = nn.Conv2d(in_channels=channels1, out_channels=channels2, kernel_siz
e=kernel_size2)
        new_size = new_size - kernel_size2 + 1
        new_size = new_size // 2

        self.fc1_size = new_size * new_size * channels2

        self.fc1 = nn.Linear(new_size * new_size * channels2, fc1)  # !!!
        #self.fc2 = nn.Linear(fc1, fc2)
        self.fc3 = nn.Linear(fc1, 10)

        self.dropout1 = Dropout(dropout)
        self.dropout2 = Dropout(dropout)

    def forward(self, x):
        x = self.pool(F.leaky_relu(self.conv1(x)))
        #print(x.shape)
        x = self.pool(F.leaky_relu(self.conv2(x)))
        #print(x.shape)
        x = x.view(-1, self.fc1_size)  # !!!
        x = self.dropout1(F.leaky_relu(self.fc1(x)))
        #x = self.dropout2(F.leaky_relu(self.fc2(x)))
        x = self.fc3(x)
        print(x.shape)
        return x

    def train_network(self, train_loader, learning_rate = 1e-4, num_epochs = 3):
        loss_fn = torch.nn.CrossEntropyLoss()
        optimizer = torch.optim.Adam(self.parameters(), lr=learning_rate)
        # итерируемся
        for epoch in tqdm_notebook(range(num_epochs)):
            running_loss = 0.0
            for i, batch in enumerate(tqdm_notebook(train_loader)):
                # так получаем текущий батч
                X_batch, y_batch = batch

                # обнуляем веса
                optimizer.zero_grad()

                # forward + backward + optimize
                y_pred = self(X_batch)
                loss = loss_fn(y_pred, y_batch)
                loss.backward()
                optimizer.step()

                # выведем текущий loss
                running_loss += loss.item()
                # выведем качество каждые 2000 батчей
                if i % 2000 == 1999:
                    print('[%d, %5d] loss: %.3f' %
                          (epoch + 1, i + 1, running_loss / 2000))
                    running_loss = 0.0

        print('Обучение закончено')

    def predict(self, test_loader):
        class_correct = list(0. for i in range(10))
        class_total = list(0. for i in range(10))

        self.eval()

        with torch.no_grad():
            for data in test_loader:
                images, labels = data
                y_pred = self(images)
                _, predicted = torch.max(y_pred, 1)
                c = (predicted == labels).squeeze()
                for i in range(4):
                    label = labels[i]
                    class_correct[label] += c[i].item()
```

```
                    class_total[label] += 1

        for i in range(10):
            print('Accuracy of %2s : %2d %%' % (
                classes[i], 100 * class_correct[i] / class_total[i]))

        class_correct_t = sum(class_correct)
        class_total_t = sum(class_total)

        print('\nTotal accuracy:', (100. * class_correct_t / class_total_t))

        self.train()
```

In [ ]:

```
net_2D1L = SimpleConvNet2DLeaky(6, 16, 5, 5, 200, 0.5) #с Max пулингом
net_2D1L.train_network(trainloader_inc, num_epochs=4, learning_rate=1e-3)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:45: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:47: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
[1,  2000] loss: 0.564
[1,  4000] loss: 0.239
[1,  6000] loss: 0.177
[1,  8000] loss: 0.155
[1, 10000] loss: 0.145
[1, 12000] loss: 0.131
[1, 14000] loss: 0.127
[1, 16000] loss: 0.113
[1, 18000] loss: 0.111
[1, 20000] loss: 0.106
[1, 22000] loss: 0.122
[1, 24000] loss: 0.115
[1, 26000] loss: 0.106
[1, 28000] loss: 0.104
[1, 30000] loss: 0.096

[2,  2000] loss: 0.090
[2,  4000] loss: 0.097
[2,  6000] loss: 0.096
[2,  8000] loss: 0.110
[2, 10000] loss: 0.083
[2, 12000] loss: 0.091
[2, 14000] loss: 0.095
[2, 16000] loss: 0.087
[2, 18000] loss: 0.077
[2, 20000] loss: 0.088
[2, 22000] loss: 0.088
[2, 24000] loss: 0.097
[2, 26000] loss: 0.094
[2, 28000] loss: 0.089
[2, 30000] loss: 0.087

[3,  2000] loss: 0.091
[3,  4000] loss: 0.081
[3,  6000] loss: 0.079
[3,  8000] loss: 0.071
[3, 10000] loss: 0.087
[3, 12000] loss: 0.092
[3, 14000] loss: 0.087
[3, 16000] loss: 0.075
[3, 18000] loss: 0.077
[3, 20000] loss: 0.090
[3, 22000] loss: 0.083
[3, 24000] loss: 0.082
```

```
[3, 26000] loss: 0.083
[3, 28000] loss: 0.093
[3, 30000] loss: 0.085

[4,  2000] loss: 0.067
[4,  4000] loss: 0.078
[4,  6000] loss: 0.082
[4,  8000] loss: 0.086
[4, 10000] loss: 0.083
[4, 12000] loss: 0.083
[4, 14000] loss: 0.081
[4, 16000] loss: 0.071
[4, 18000] loss: 0.090
[4, 20000] loss: 0.078
[4, 22000] loss: 0.078
[4, 24000] loss: 0.077
[4, 26000] loss: 0.096
[4, 28000] loss: 0.070
[4, 30000] loss: 0.073
Обучение закончено
```

In [ ]:

```
net_2D1L.predict(testloader)
```

```
Accuracy of  0 : 99 %
Accuracy of  1 : 99 %
Accuracy of  2 : 99 %
Accuracy of  3 : 99 %
Accuracy of  4 : 98 %
Accuracy of  5 : 97 %
Accuracy of  6 : 98 %
Accuracy of  7 : 99 %
Accuracy of  8 : 98 %
Accuracy of  9 : 97 %

Total accuracy: 98.89
```

In [ ]:

```
net_2D1L.train_network(trainloader_inc, num_epochs=1, learning_rate=1e-4)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:45: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:47: TqdmDeprecationWarning:
This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
[1,  2000] loss: 0.073
[1,  4000] loss: 0.058
[1,  6000] loss: 0.052
[1,  8000] loss: 0.056
[1, 10000] loss: 0.050
[1, 12000] loss: 0.058
[1, 14000] loss: 0.051
[1, 16000] loss: 0.057
[1, 18000] loss: 0.049
[1, 20000] loss: 0.058
[1, 22000] loss: 0.049
[1, 24000] loss: 0.053
[1, 26000] loss: 0.051
[1, 28000] loss: 0.039
[1, 30000] loss: 0.046
Обучение закончено
```

In [ ]:

```
net_2D1L.predict(testloader)
```

```
Accuracy of  0 : 99 %
Accuracy of  1 : 99 %
Accuracy of  2 : 99 %
Accuracy of  3 : 99 %
Accuracy of  4 : 98 %
Accuracy of  5 : 98 %
Accuracy of  6 : 98 %
Accuracy of  7 : 99 %
Accuracy of  8 : 99 %
Accuracy of  9 : 97 %

Total accuracy: 99.1
```

Улучшили ещё! Причём за меньшее число эпох, чем в случае с **ReLU** - всего **5**. Если продолжим дообучать, можем улучшить ещё.

Итак, лучший результат - **99.1%**. Экспериментировал с разным количеством сверточных слоёв и разным количеством полносвязных. Итого - два сверточных слоя и два полносвязных, между полносвязными - дропаут. Увеличение количества свёрточных слоёв не помогло улучшить результат (к тому же там возникают проблемы с тем, что размера картинки не хватает - но можно исправить пэддингом и убрав пулинг). Экспериментировал также с разными функциями активациями - вместо **ReLU** брал **LReLU**, для которой и был получен лучший результат (за меньшее в сравнение с **ReLU** число эпох). **Average pooling** работал хуже, чем **Max**.