Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования
«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ имени Н.Э.БАУМАНА
(национальный исследовательский университет)»

Факультет: Информатика и системы управления

Кафедра: Теоретическая информатика и компьютерные технологии

Лабораторная работа №5

«Сверточная нейронная сеть»

По дисциплине «Теория искусственных нейронных сетей»

Работу выполнил

студент группы ИУ9-72Б

Владиславов А.Г.

Москва, 2021

Init

```
import torch
import torch.nn as nn
import torchvision
import torchvision.transforms as transforms

batch_size = 64
num_classes = 10
learning_rate = 0.001
num_epochs = 30

device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
```

Dataset CIFAR10

Реформатирование изображение

```
all_transforms = transforms.Compose([transforms.Resize((32,32)),
                                     transforms.ToTensor(),

transforms.Normalize(mean=[0.4914, 0.4822, 0.4465],
                                                          std=[0.2023,
0.1994, 0.2010])
                                     ])
```

Датасеты для обучения и теста

```
train_dataset = torchvision.datasets.CIFAR10(root = './data',
                                             train = True,
                                             transform =
all_transforms,
                                             download = True)

test_dataset = torchvision.datasets.CIFAR10(root = './data',
                                            train = False,
                                            transform =
all_transforms,
                                            download=True)
```

```
Files already downloaded and verified
Files already downloaded and verified
```

Загрузчики датасетов - чтобы не держать весь датасет в ОЗУ

```
train_loader = torch.utils.data.DataLoader(dataset = train_dataset,
                                           batch_size = batch_size,
                                           shuffle = True)
```

```
test_loader = torch.utils.data.DataLoader(dataset = test_dataset,
                                          batch_size = batch_size,
                                          shuffle = True)
```

Класс нейронной сети, наследуется от nn.Module из pytorch Задаём слои forward - порядок, в котором данные проходят через нейронку

Cond2d - свёрточные слои, задаём получаемые каналы (начинаем с 3х т.к. RGB), каналы после свёртки, размер свёрт. ядра MaxPooling - уменьшение размеров изображения путём сложения блоков пикселей

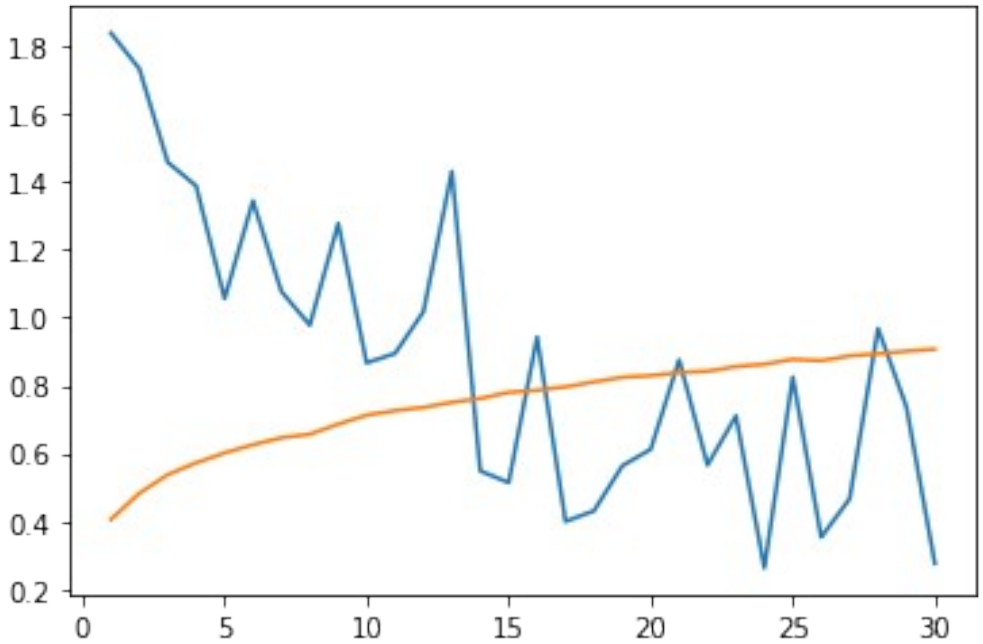
```
# Creating a CNN class
class ConvNeuralNet(nn.Module):
      #  Determine what layers and their order in CNN object
    def __init__(self, num_classes):
        super(ConvNeuralNet, self).__init__()
        self.conv_layer1 = nn.Conv2d(in_channels=3, out_channels=32,
kernel_size=3)
        self.conv_layer2 = nn.Conv2d(in_channels=32, out_channels=32,
kernel_size=3)
        self.max_pool1 = nn.MaxPool2d(kernel_size = 2, stride = 2)

        self.conv_layer3 = nn.Conv2d(in_channels=32, out_channels=64,
kernel_size=3)
        self.conv_layer4 = nn.Conv2d(in_channels=64, out_channels=64,
kernel_size=3)
        self.max_pool2 = nn.MaxPool2d(kernel_size = 2, stride = 2)

        self.fc1 = nn.Linear(1600, 128)
        self.relu1 = nn.ReLU()
        self.fc2 = nn.Linear(128, num_classes)

    # Progresses data across layers
    def forward(self, x):
        out = self.conv_layer1(x)
        out = self.conv_layer2(out)
        out = self.max_pool1(out)

        out = self.conv_layer3(out)
        out = self.conv_layer4(out)
        out = self.max_pool2(out)

        out = out.reshape(out.size(0), -1)

        out = self.fc1(out)
        out = self.relu1(out)
        out = self.fc2(out)
        return out
```

```
model = ConvNeuralNet(num_classes)

# Set Loss function with criterion
criterion = nn.CrossEntropyLoss()

# Set optimizer with optimizer
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate,
weight_decay = 0.005, momentum = 0.9)

total_step = len(train_loader)

# We use the pre-defined number of epochs to determine how many
iterations to train the network on
import time
import matplotlib.pyplot as plt

start = time.time()
losses = []
scores = []
for epoch in range(num_epochs):
     #Load in the data in batches using the train_loader object
    for i, (images, labels) in enumerate(train_loader):
        # Move tensors to the configured device
        images = images.to(device)
        labels = labels.to(device)

        # Forward pass
        outputs = model(images)
        loss = criterion(outputs, labels)

        # Backward and optimize
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

    with torch.no_grad():
        correct = 0
        total = 0
        for images, labels in train_loader:
            images = images.to(device)
            labels = labels.to(device)
            outputs = model(images)
            _, predicted = torch.max(outputs.data, 1)
            total += labels.size(0)
            correct += (predicted == labels).sum().item()
        print('Accuracy of the network on the {} train images: {}
%'.format(50000, 100 * correct / total))
        scores.append(correct / total)

    print('Epoch [{}/{}], Loss: {:.4f}'.format(epoch+1, num_epochs,
```

```
loss.item()))
    losses.append(loss.item())
print(time.time() - start)

xs = list(range(1, num_epochs + 1))
plt.plot(xs, losses, label = 'loses')
plt.show()
plt.plot(xs, scores, label = 'scores')
plt.show()
```

```
Accuracy of the network on the 50000 train images: 40.498 %
Epoch [1/30], Loss: 1.8398
Accuracy of the network on the 50000 train images: 48.214 %
Epoch [2/30], Loss: 1.7340
Accuracy of the network on the 50000 train images: 53.706 %
Epoch [3/30], Loss: 1.4585
Accuracy of the network on the 50000 train images: 57.204 %
Epoch [4/30], Loss: 1.3886
Accuracy of the network on the 50000 train images: 60.108 %
Epoch [5/30], Loss: 1.0563
Accuracy of the network on the 50000 train images: 62.526 %
Epoch [6/30], Loss: 1.3447
Accuracy of the network on the 50000 train images: 64.584 %
Epoch [7/30], Loss: 1.0776
Accuracy of the network on the 50000 train images: 65.632 %
Epoch [8/30], Loss: 0.9782
Accuracy of the network on the 50000 train images: 68.596 %
Epoch [9/30], Loss: 1.2776
Accuracy of the network on the 50000 train images: 71.264 %
Epoch [10/30], Loss: 0.8668
Accuracy of the network on the 50000 train images: 72.558 %
Epoch [11/30], Loss: 0.8946
Accuracy of the network on the 50000 train images: 73.532 %
Epoch [12/30], Loss: 1.0180
Accuracy of the network on the 50000 train images: 75.03 %
Epoch [13/30], Loss: 1.4313
Accuracy of the network on the 50000 train images: 76.186 %
Epoch [14/30], Loss: 0.5481
Accuracy of the network on the 50000 train images: 77.946 %
Epoch [15/30], Loss: 0.5137
Accuracy of the network on the 50000 train images: 78.616 %
Epoch [16/30], Loss: 0.9429
Accuracy of the network on the 50000 train images: 79.61 %
Epoch [17/30], Loss: 0.3991
Accuracy of the network on the 50000 train images: 81.048 %
Epoch [18/30], Loss: 0.4304
Accuracy of the network on the 50000 train images: 82.422 %
Epoch [19/30], Loss: 0.5632
Accuracy of the network on the 50000 train images: 82.912 %
Epoch [20/30], Loss: 0.6120
```

```
Accuracy of the network on the 50000 train images: 83.822 %
Epoch [21/30], Loss: 0.8756
Accuracy of the network on the 50000 train images: 84.21 %
Epoch [22/30], Loss: 0.5655
Accuracy of the network on the 50000 train images: 85.576 %
Epoch [23/30], Loss: 0.7105
Accuracy of the network on the 50000 train images: 86.282 %
Epoch [24/30], Loss: 0.2625
Accuracy of the network on the 50000 train images: 87.702 %
Epoch [25/30], Loss: 0.8249
Accuracy of the network on the 50000 train images: 87.328 %
Epoch [26/30], Loss: 0.3528
Accuracy of the network on the 50000 train images: 88.718 %
Epoch [27/30], Loss: 0.4657
Accuracy of the network on the 50000 train images: 89.352 %
Epoch [28/30], Loss: 0.9681
Accuracy of the network on the 50000 train images: 90.112 %
Epoch [29/30], Loss: 0.7372
Accuracy of the network on the 50000 train images: 90.706 %
Epoch [30/30], Loss: 0.2754
4000.9419581890106
```



MNIST

```
device2 = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

train_dataset2 = torchvision.datasets.MNIST(root = './data',
                                            train = True,
                                            transform =
```

```
all_transforms,
                                          download = True)

test_dataset2 = torchvision.datasets.MNIST(root = './data',
                                           train = False,
                                           transform =
all_transforms,
                                           download=True)
```

```
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
```

{"version_major":2,"version_minor":0,"model_id":"500abd8300b846669f16fc905eb49492"}

```
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
```

{"version_major":2,"version_minor":0,"model_id":"4e57bd8ace164cbc92960ca2da295d18"}

```
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
```

{"version_major":2,"version_minor":0,"model_id":"318c9197add8458a8ebfad2dedb5b6f6"}

```
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
```

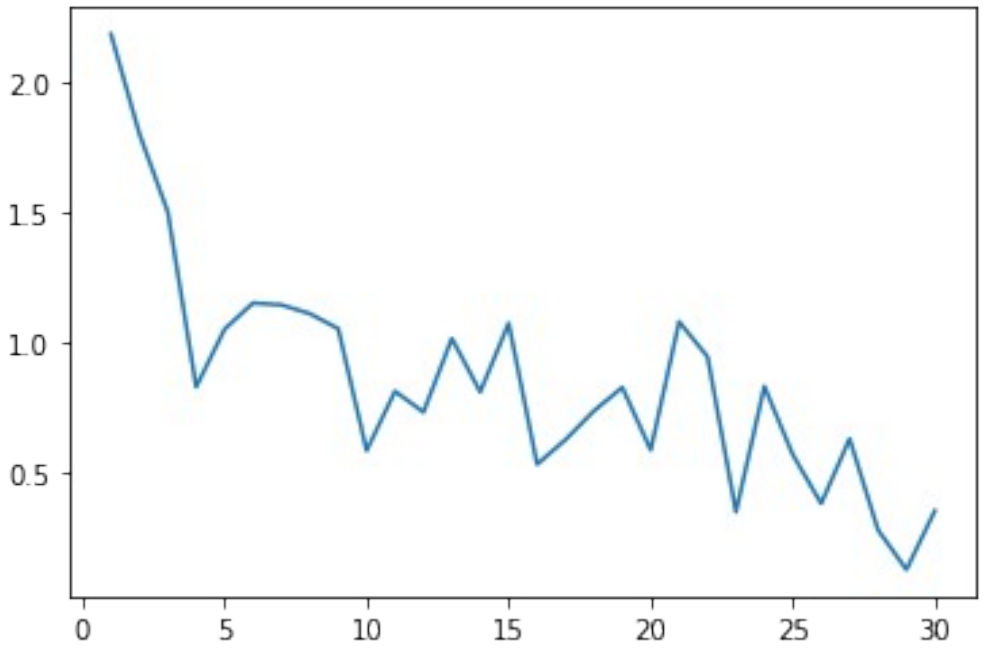
{"version_major":2,"version_minor":0,"model_id":"7e70573321fb45009a6b24127d79f6ee"}

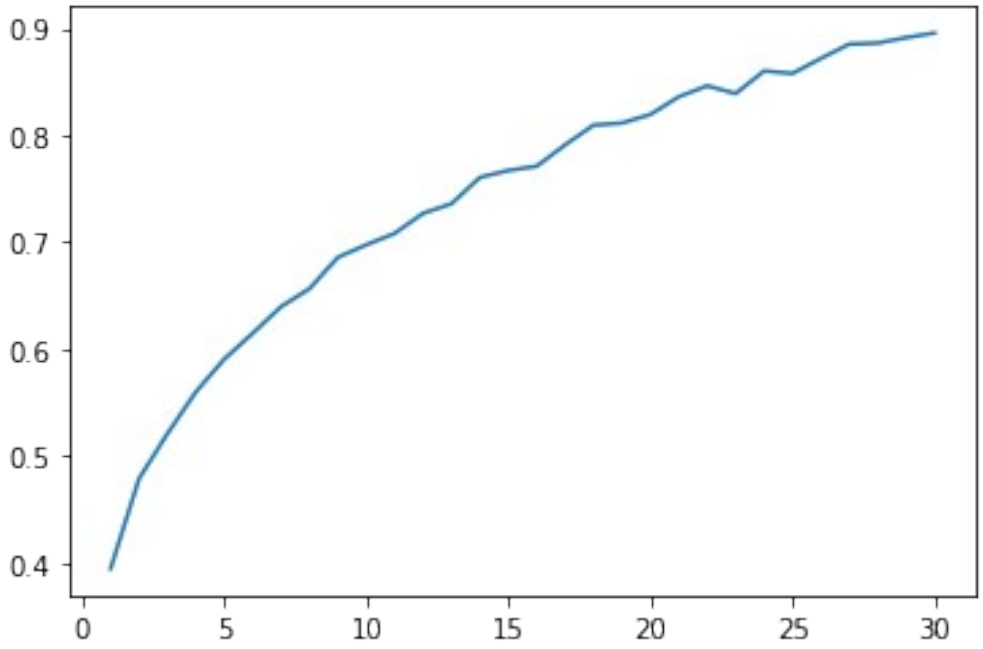
```
Extracting ./data/MNIST/raw/t10k-labels-idx1-ubyte.gz to
./data/MNIST/raw
```

```
train_loader2 = torch.utils.data.DataLoader(dataset = train_dataset,
                                            batch_size = batch_size,
                                            shuffle = True)


test_loader2 = torch.utils.data.DataLoader(dataset = test_dataset,
                                            batch_size = batch_size,
                                            shuffle = True)

model2 = ConvNeuralNet(num_classes)

# Set optimizer with optimizer
optimizer2 = torch.optim.SGD(model2.parameters(), lr=learning_rate,
weight_decay = 0.005, momentum = 0.9)

total_step2 = len(train_loader2)

start = time.time()
losses = []
scores = []
for epoch in range(num_epochs):
     #Load in the data in batches using the train_loader object
    for i, (images, labels) in enumerate(train_loader2):
        # Move tensors to the configured device
        images = images.to(device2)
        labels = labels.to(device2)

        # Forward pass
        outputs = model2(images)
        loss = criterion(outputs, labels)

        # Backward and optimize
        optimizer2.zero_grad()
        loss.backward()
        optimizer2.step()

    with torch.no_grad():
        correct = 0
        total = 0
        for images, labels in train_loader:
            images = images.to(device2)
            labels = labels.to(device2)
            outputs = model2(images)
            _, predicted = torch.max(outputs.data, 1)
            total += labels.size(0)
            correct += (predicted == labels).sum().item()
        print('Accuracy of the network on the {} train images: {}
%'.format(50000, 100 * correct / total))
        scores.append(correct / total)
```

```
    print('Epoch [{}/{}], Loss: {:.4f}'.format(epoch+1, num_epochs,
loss.item()))
    losses.append(loss.item())
print(time.time() - start)

xs = list(range(1, num_epochs + 1))
plt.plot(xs, losses, label = 'loses')
plt.show()
plt.plot(xs, scores, label = 'scores')
plt.show()

Accuracy of the network on the 50000 train images: 39.478 %
Epoch [1/30], Loss: 2.1818
Accuracy of the network on the 50000 train images: 47.886 %
Epoch [2/30], Loss: 1.7992
Accuracy of the network on the 50000 train images: 52.098 %
Epoch [3/30], Loss: 1.4986
Accuracy of the network on the 50000 train images: 56.004 %
Epoch [4/30], Loss: 0.8252
Accuracy of the network on the 50000 train images: 59.07 %
Epoch [5/30], Loss: 1.0469
Accuracy of the network on the 50000 train images: 61.48 %
Epoch [6/30], Loss: 1.1478
Accuracy of the network on the 50000 train images: 63.98 %
Epoch [7/30], Loss: 1.1405
Accuracy of the network on the 50000 train images: 65.658 %
Epoch [8/30], Loss: 1.1069
Accuracy of the network on the 50000 train images: 68.586 %
Epoch [9/30], Loss: 1.0493
Accuracy of the network on the 50000 train images: 69.76 %
Epoch [10/30], Loss: 0.5806
Accuracy of the network on the 50000 train images: 70.828 %
Epoch [11/30], Loss: 0.8088
Accuracy of the network on the 50000 train images: 72.706 %
Epoch [12/30], Loss: 0.7282
Accuracy of the network on the 50000 train images: 73.602 %
Epoch [13/30], Loss: 1.0115
Accuracy of the network on the 50000 train images: 76.066 %
Epoch [14/30], Loss: 0.8062
Accuracy of the network on the 50000 train images: 76.714 %
Epoch [15/30], Loss: 1.0698
Accuracy of the network on the 50000 train images: 77.114 %
Epoch [16/30], Loss: 0.5280
Accuracy of the network on the 50000 train images: 79.11 %
Epoch [17/30], Loss: 0.6226
Accuracy of the network on the 50000 train images: 80.958 %
Epoch [18/30], Loss: 0.7331
Accuracy of the network on the 50000 train images: 81.148 %
Epoch [19/30], Loss: 0.8234
Accuracy of the network on the 50000 train images: 81.946 %
```

```
Epoch [20/30], Loss: 0.5830
Accuracy of the network on the 50000 train images: 83.606 %
Epoch [21/30], Loss: 1.0760
Accuracy of the network on the 50000 train images: 84.628 %
Epoch [22/30], Loss: 0.9413
Accuracy of the network on the 50000 train images: 83.894 %
Epoch [23/30], Loss: 0.3450
Accuracy of the network on the 50000 train images: 86.016 %
Epoch [24/30], Loss: 0.8272
Accuracy of the network on the 50000 train images: 85.78 %
Epoch [25/30], Loss: 0.5649
Accuracy of the network on the 50000 train images: 87.19 %
Epoch [26/30], Loss: 0.3770
Accuracy of the network on the 50000 train images: 88.528 %
Epoch [27/30], Loss: 0.6261
Accuracy of the network on the 50000 train images: 88.626 %
Epoch [28/30], Loss: 0.2759
Accuracy of the network on the 50000 train images: 89.134 %
Epoch [29/30], Loss: 0.1230
Accuracy of the network on the 50000 train images: 89.566 %
Epoch [30/30], Loss: 0.3492
4014.880949497223
```




0.9
0.8
0.7
0.6
0.5
0.4
0
5
10
15
20
25
30