Министерство образования Республики Беларусь
Учреждение образования
«Брестский Государственный технический университет»
Кафедра ИИТ

Лабораторная работа №1

По дисциплине: «Модели решения задач в интеллектуальных системах»

Тема: «Обучение классификаторов средствами библиотеки PyTorch»

Выполнил:
Студент 4 курса
Группы ИИ-21
Ясюкевич В.С.
Проверил:
Крощенко А. А.

Брест 2024

Цель: научиться конструировать нейросетевые классификаторы и выполнять их обучение на известных выборках компьютерного зрения.

Ход работы:

Вариант 17

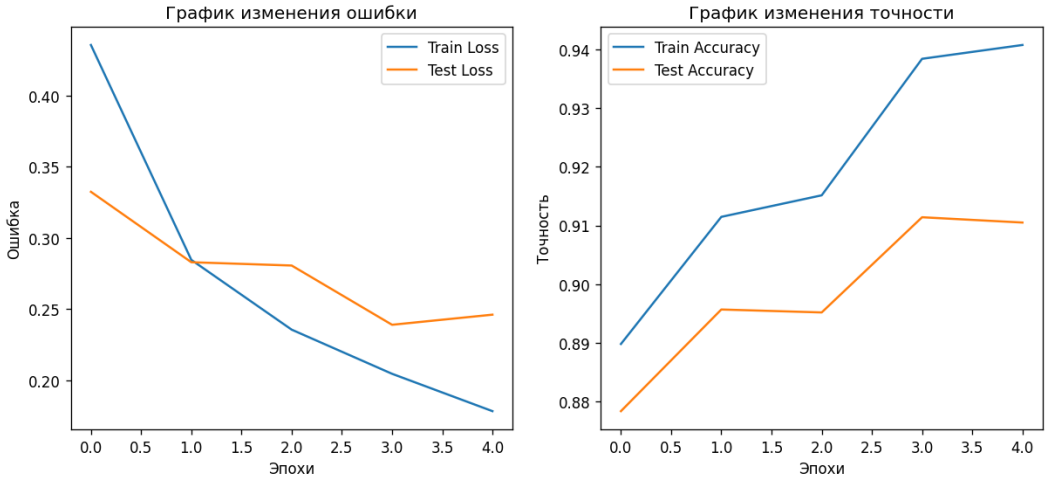
17	Fashion-MNIST	28X28	RMSprop

Код программы:

```
import torch

import torch.nn as nn

import torch.optim as optim

import torchvision

import torchvision.transforms as
transforms

import matplotlib.pyplot as plt

transform =
transforms.Compose([transforms.ToTensor
(), transforms.Normalize((0.5,),
(0.5,))])

train_set =
torchvision.datasets.FashionMNIST(root=
'./data', train=True, download=True,
transform=transform)

train_loader =
torch.utils.data.DataLoader(train_set,
batch_size=64, shuffle=True)

test_set =
torchvision.datasets.FashionMNIST(root=
'./data', train=False, download=True,
transform=transform)

test_loader =
torch.utils.data.DataLoader(test_set,
batch_size=64, shuffle=False)

class SimpleCNN(nn.Module):

    def __init__(self):

        super(SimpleCNN,
self).__init__()

        self.conv1 = nn.Conv2d(1, 32,
kernel_size=3, padding=1)

        self.pool =
nn.MaxPool2d(kernel_size=2, stride=2)

        self.conv2 = nn.Conv2d(32, 64,
kernel_size=3, padding=1)

        self.fc1 = nn.Linear(64 * 7 *
7, 128)

        self.fc2 = nn.Linear(128, 10)

        self.relu = nn.ReLU()

    def forward(self, x):

        x =
self.pool(self.relu(self.conv1(x)))

        x =
self.pool(self.relu(self.conv2(x)))

        x = x.view(-1, 64 * 7 * 7)

        x = self.relu(self.fc1(x))

        x = self.fc2(x)

        return x

def calculate_accuracy(loader, model):

    correct = 0

    total = 0

    with torch.no_grad():

        for images, labels in loader:

            images, labels =
images.to(device), labels.to(device)

            outputs = model(images)

            _, predicted =
torch.max(outputs, 1)

            total += labels.size(0)

            correct += (predicted ==
labels).sum().item()

    return correct / total

device = torch.device("cuda" if
torch.cuda.is_available() else "cpu")

model = SimpleCNN().to(device)

criterion = nn.CrossEntropyLoss()
```

```
optimizer =
optim.Adam(model.parameters(),
lr=0.001)

epochs = 5

train_loss_list = []

test_loss_list = []

train_acc_list = []

test_acc_list = []

for epoch in range(epochs):

    model.train()

    running_loss = 0.0

    for images, labels in train_loader:

        images, labels =
images.to(device), labels.to(device)

        optimizer.zero_grad()

        outputs = model(images)

        loss = criterion(outputs,
labels)

        loss.backward()

        optimizer.step()

        running_loss += loss.item()

    train_loss_list.append(running_loss
/ len(train_loader))

    train_acc =
calculate_accuracy(train_loader, model)

    train_acc_list.append(train_acc)

    model.eval()

    test_loss = 0.0

    with torch.no_grad():

        for images, labels in
test_loader:

            images, labels =
images.to(device), labels.to(device)

            outputs = model(images)

            loss = criterion(outputs,
labels)

            test_loss += loss.item()

    test_loss_list.append(test_loss /
len(test_loader))

    test_acc =
calculate_accuracy(test_loader, model)

    test_acc_list.append(test_acc)

    print(f'Epoch [{epoch +
1}/{epochs}], Train Loss:
{train_loss_list[-1]:.4f}, Test Loss:
{test_loss_list[-1]:.4f}, '

          f'Train Accuracy: {train_acc
* 100:.2f}%, Test Accuracy: {test_acc *
100:.2f}%')

plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)

plt.plot(train_loss_list, label='Train
Loss')

plt.plot(test_loss_list, label='Test
Loss')

plt.title('График изменения ошибки')

plt.xlabel('Эпохи')

plt.ylabel('Ошибка')

plt.legend()

plt.subplot(1, 2, 2)

plt.plot(train_acc_list, label='Train
Accuracy')

plt.plot(test_acc_list, label='Test
Accuracy')

plt.title('График изменения точности')

plt.xlabel('Эпохи')

plt.ylabel('Точность')

plt.legend()

plt.show()
```

Результат программы:




```
Epoch [1/5], Train Loss: 0.4354, Test Loss: 0.3323, Train Accuracy: 88.98%, Test Accuracy: 87.84%
Epoch [2/5], Train Loss: 0.2845, Test Loss: 0.2828, Train Accuracy: 91.15%, Test Accuracy: 89.57%
Epoch [3/5], Train Loss: 0.2354, Test Loss: 0.2805, Train Accuracy: 91.51%, Test Accuracy: 89.52%
Epoch [4/5], Train Loss: 0.2044, Test Loss: 0.2389, Train Accuracy: 93.84%, Test Accuracy: 91.14%
Epoch [5/5], Train Loss: 0.1782, Test Loss: 0.2460, Train Accuracy: 94.07%, Test Accuracy: 91.05%
```

Вывод: научился конструировать нейросетевые классификаторы и выполнять их обучение на известных выборках компьютерного зрения.