Министерство образования Республики Беларусь
Учреждение образования
“Брестский государственный технический университет”
Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №1

Специальность ИИ-21

Выполнил:
Парфеевец И.А.
Студент группы ИИ-21

Проверил:
А. А. Крощенко
доц. кафедры ИИТ

Брест 2024

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

Общее задание

1. Выполнить конструирование своей модели СНС, обучить ее на выборке по заданию (использовать **torchvision.datasets**). Предпочтение отдавать как можно более простым архитектурам, базирующимся на базовых типах слоев (сверточный, полносвязный, подвыборочный, слой нелинейного преобразования). Оценить эффективность обучения на тестовой выборке

2. Ознакомьтесь с state-of-the-art результатами для предлагаемых выборок (https://paperswithcode.com/task/image-classification). Сделать выводы о результатах обучения СНС из п. 1;

3. Реализовать визуализацию работы СНС из пункта 1 (выбор и подачу на архитектуру произвольного изображения с выводом результата);

4. Оформить отчет по выполненной работе, загрузить исходный код и отчет в соответствующий репозиторий на github.

Задание по вариантам

№ варианта	Выборка	Размер исходного изображения	Оптимизатор
2	Fashion-MNIST	28X28	SGD

Код программы:

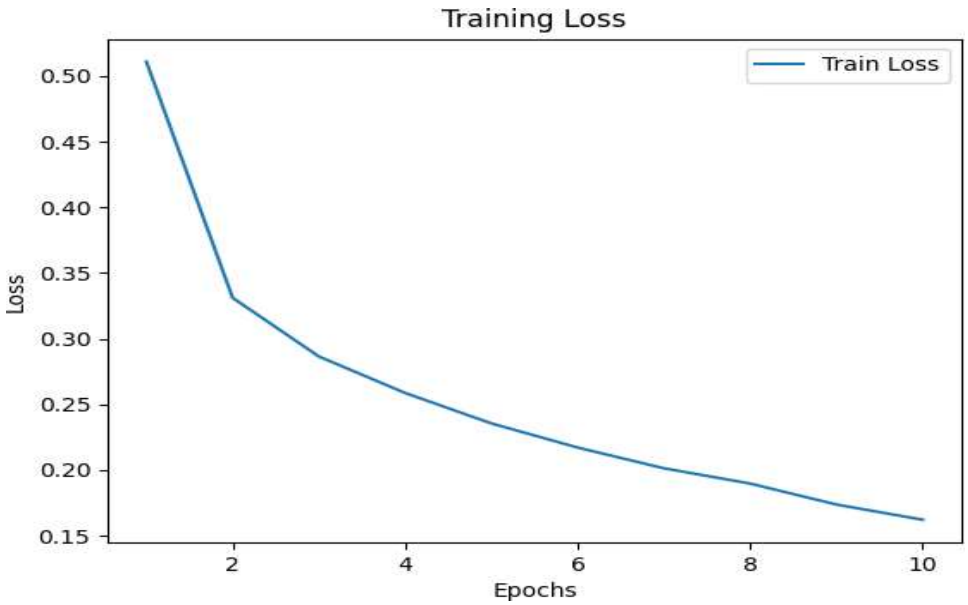
```
# Лабораторная работа №1 - Обучение классификаторов средствами библиотеки PyTorch
# Задание 2: Fashion-MNIST, 28x28, оптимизатор - SGD

# Шаг 1: Импорт необходимых библиотек
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision
import torchvision.transforms as transforms
import matplotlib.pyplot as plt
import numpy as np

# Шаг 2: Загрузка и подготовка данных
transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5,), (0.5,))  # Нормализация для улучшения сходимости
])

# Загрузка данных Fashion-MNIST
trainset = torchvision.datasets.FashionMNIST(root='./data', train=True, download=True, transform=transform)
```

```
trainloader = torch.utils.data.DataLoader(trainset, batch_size=64,
shuffle=True)

testset = torchvision.datasets.FashionMNIST(root='./data', train=False,
download=True, transform=transform)
testloader = torch.utils.data.DataLoader(testset, batch_size=64,
shuffle=False)

# Классы Fashion-MNIST
classes = ('T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat', 'Sandal',
'Shirt', 'Sneaker', 'Bag', 'Ankle boot')

# Шаг 3: Определение простой архитектуры CNC
class SimpleCNN(nn.Module):
    def __init__(self):
        super(SimpleCNN, self).__init__()
        self.conv1 = nn.Conv2d(1, 16, 3, padding=1)   # Входные каналы = 1,
выходные = 16
        self.conv2 = nn.Conv2d(16, 32, 3, padding=1)  # Выходные каналы = 32
        self.pool = nn.MaxPool2d(2, 2)                # Пулинг с ядром 2x2
        self.fc1 = nn.Linear(32 * 7 * 7, 128)         # Первый полносвязный
слой
        self.fc2 = nn.Linear(128, 10)                 # Выходной
полносвязный слой на 10 классов
        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, 32 * 7 * 7)  # Преобразование в одномерный вектор
        x = self.relu(self.fc1(x))
        x = self.fc2(x)
        return x

# Инициализация модели, функции потерь и оптимизатора
net = SimpleCNN()
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(net.parameters(), lr=0.01, momentum=0.9)

# Шаг 4: Обучение модели
num_epochs = 10
train_loss_history = []
test_loss_history = []

for epoch in range(num_epochs):
    running_loss = 0.0
    net.train()
    for inputs, labels in trainloader:
        optimizer.zero_grad()           # Обнуление градиентов
        outputs = net(inputs)           # Прямой проход
        loss = criterion(outputs, labels)  # Вычисление потерь
        loss.backward()                 # Обратное распространение
        optimizer.step()                # Шаг оптимизации
        running_loss += loss.item()
    train_loss_history.append(running_loss / len(trainloader))
```

```
    # Оценка на тестовой выборке
    net.eval()
    test_loss = 0.0
    with torch.no_grad():
        for inputs, labels in testloader:
            outputs = net(inputs)
            loss = criterion(outputs, labels)
            test_loss += loss.item()
    test_loss_history.append(test_loss / len(testloader))

    print(f"Epoch {epoch + 1}/{num_epochs}, Train Loss:
{train_loss_history[-1]}, Test Loss: {test_loss_history[-1]}")

# Шаг 5: Визуализация графиков ошибки
plt.plot(train_loss_history, label='Train Loss')
plt.plot(test_loss_history, label='Test Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.title('Train and Test Loss per Epoch')
plt.show()

# Шаг 6: Визуализация работы СНС на тестовом изображении
# Функция для отображения изображения
def imshow(img):
    img = img / 2 + 0.5  # Денормализация
    npimg = img.numpy()
    plt.imshow(np.transpose(npimg, (1, 2, 0)), cmap='gray')
    plt.show()

# Выбор одного изображения из тестовой выборки и отображение его класса
dataiter = iter(testloader)
images, labels = next(dataiter)

# Вывод изображения
imshow(images[0])
print(f'Actual Label: {classes[labels[0]]}')

# Предсказание модели
outputs = net(images[0].unsqueeze(0))
_, predicted = torch.max(outputs, 1)
print(f'Predicted Label: {classes[predicted[0]]}')
```

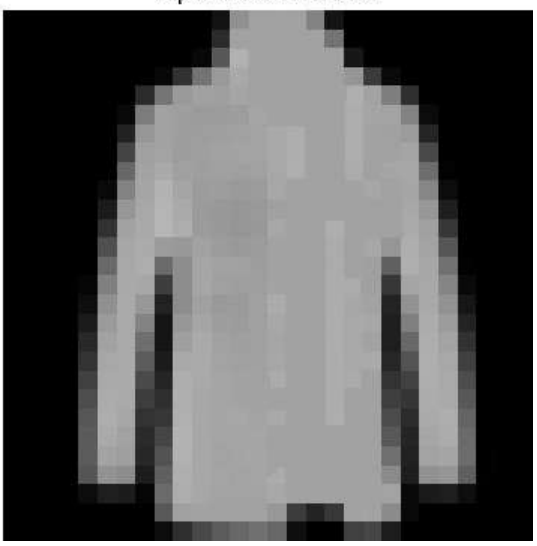
Выводы программы:

```
Epoch [1/10], Loss: 0.5109
Epoch [2/10], Loss: 0.3312
Epoch [3/10], Loss: 0.2865
Epoch [4/10], Loss: 0.2587
Epoch [5/10], Loss: 0.2355
Epoch [6/10], Loss: 0.2172
Epoch [7/10], Loss: 0.2014
Epoch [8/10], Loss: 0.1898
Epoch [9/10], Loss: 0.1738
Epoch [10/10], Loss: 0.1622
Test Accuracy: 91.51%
```



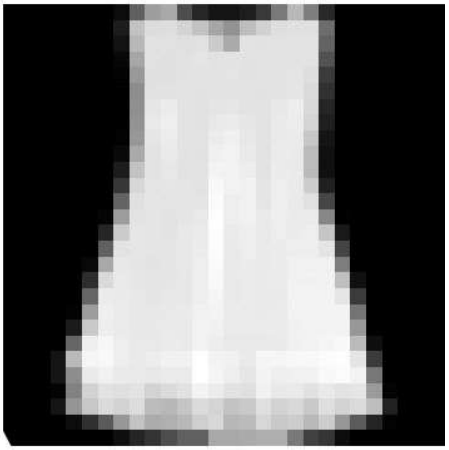
Исходное изображение

Измененное изображение
Прогноз: Пальто



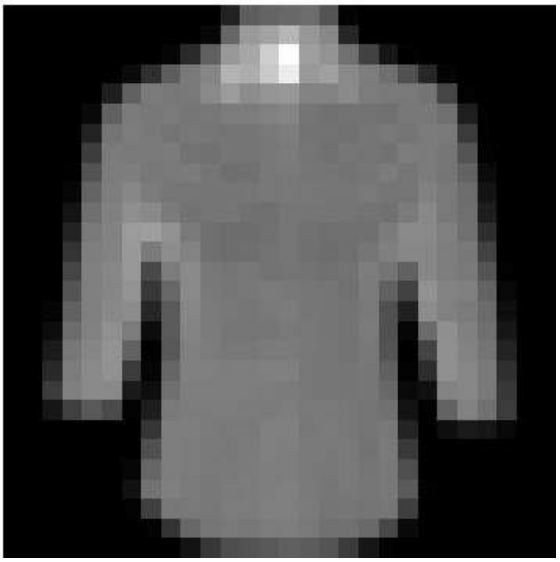
Исходное изображение

Измененное изображение
Прогноз: Платье



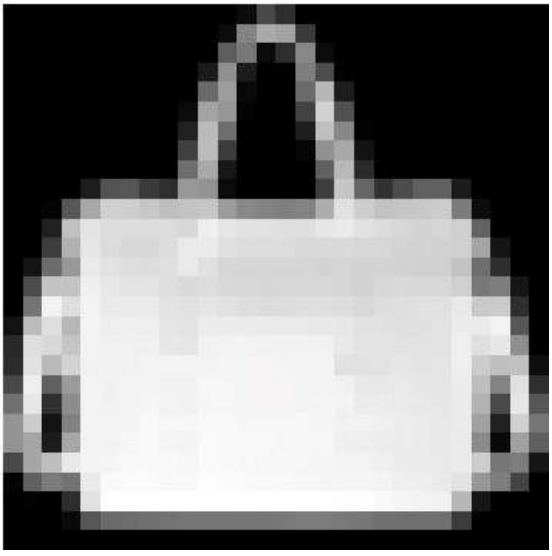
Исходное изображение

Измененное изображение
Прогноз: Рубашка



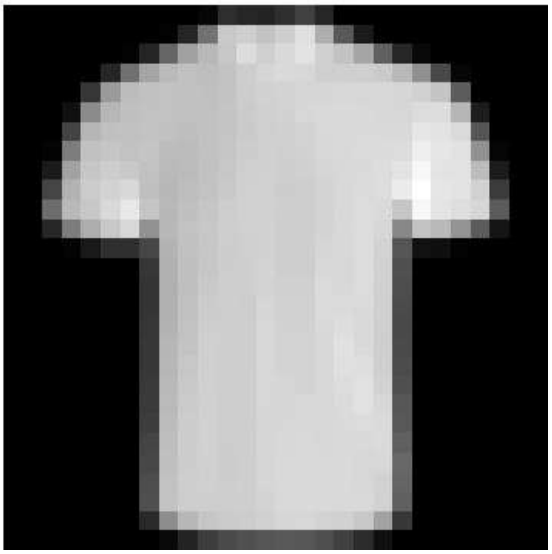
Исходное изображение

Измененное изображение
Прогноз: Сумка



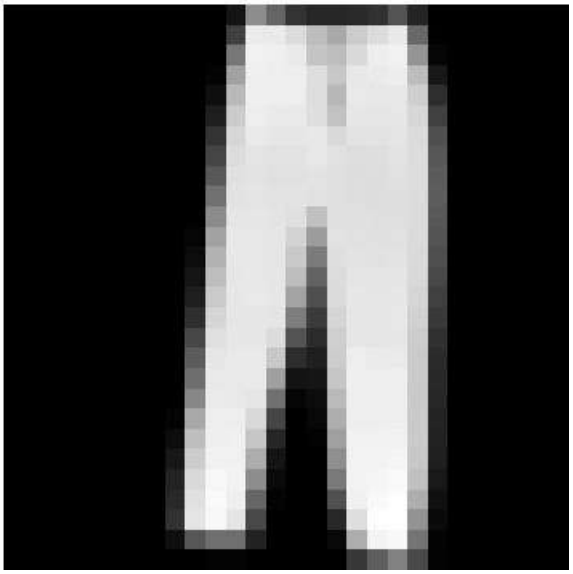
Исходное изображение

Измененное изображение
Прогноз: Футболка/топ



Исходное изображение

Измененное изображение
Прогноз: Брюки



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