МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №1

Специальность ИИ(з)

Выполнил
Д. Д. Крупич,
студент группы ИИ-24

Проверил
Андренко К.В,
Преподаватель-стажер кафедры ИИТ,
«___k _____2025 г.

Брест 2025

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, построить график изменения ошибки (matplotlib);

2. Ознакомьтесь с state-of-the-art результатами для предлагаемых выборок (из материалов в сети Интернет). 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);

Вариант:

7	Fashion-MNIST	28X28	Adam

Выполнение:

Код программы

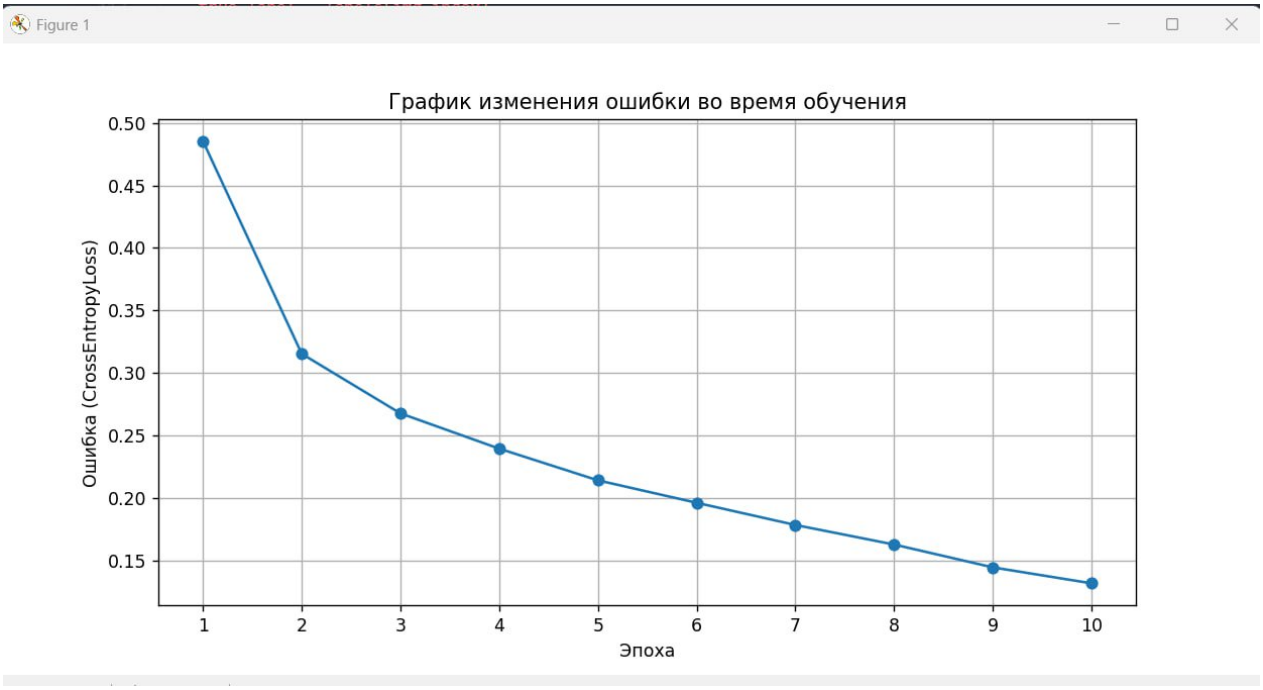
```
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision
import torchvision.transforms as transforms
import matplotlib.pyplot as plt
import numpy as np
import time

class SimpleCNN(nn.Module):
    def __init__(self):
        super(SimpleCNN, self).__init__()
        self.conv_block1 = nn.Sequential(
            nn.Conv2d(in_channels=1, out_channels=16, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2)
        )
        self.conv_block2 = nn.Sequential(
            nn.Conv2d(in_channels=16, out_channels=32, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2)
        )
        self.classifier = nn.Sequential(
            nn.Flatten(),
            nn.Linear(32 * 7 * 7, 128),
            nn.ReLU(),
            nn.Linear(128, 10)
        )
```

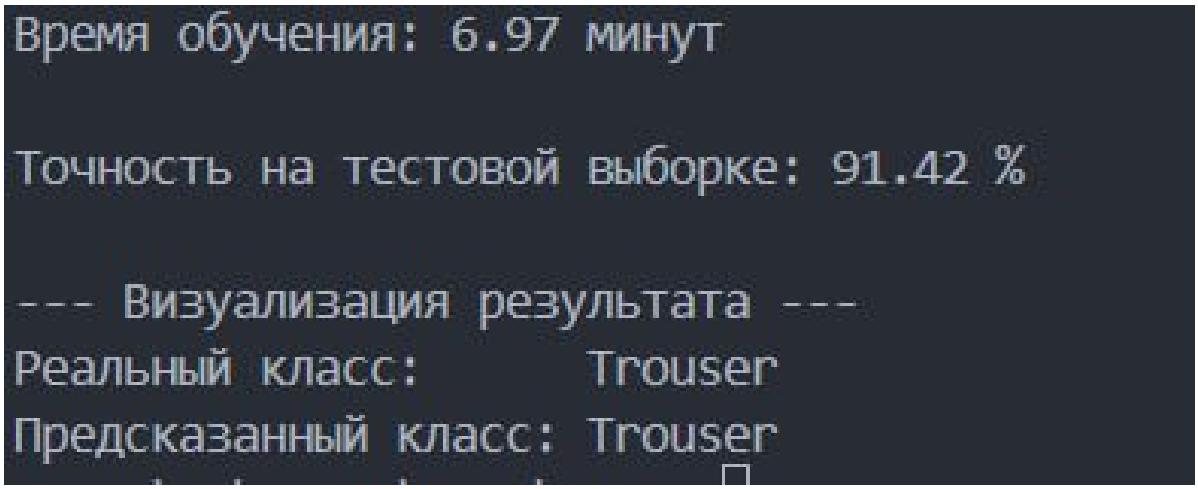
```
    def forward(self, x):
        x = self.conv_block1(x)
        x = self.conv_block2(x)
        x = self.classifier(x)
        return x

if __name__ == '__main__':
    device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
    print(f"Используемое устройство: {device}")

    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, num_workers=2)

    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, num_workers=2)

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

    model = SimpleCNN()
    model.to(device)

    criterion = nn.CrossEntropyLoss()
    optimizer = optim.Adam(model.parameters(), lr=0.001)

    start_time = time.time()
    loss_history = []
    num_epochs = 10

    for epoch in range(num_epochs):
        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()
```

```
    epoch_loss = running_loss / len(train_loader)
    loss_history.append(epoch_loss)
    print(f'Эпоха {epoch + 1}/{num_epochs}, Ошибка (Loss): {epoch_loss:.4f}')

end_time = time.time()
print('Обучение завершено')
print(f'Время обучения: {((end_time - start_time) / 60):.2f} минут')

model.eval()
correct = 0
total = 0
with torch.no_grad():
    for data in test_loader:
        images, labels = data[0].to(device), data[1].to(device)
        outputs = model(images)
        _, predicted = torch.max(outputs.data, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()

accuracy = 100 * correct / total
print(f'\nТочность на тестовой выборке: {accuracy:.2f} %')

plt.figure(figsize=(10, 5))
plt.plot(range(1, num_epochs + 1), loss_history, marker='o', linestyle='-')
plt.title('График изменения ошибки во время обучения')
plt.xlabel('Эпоха')
plt.ylabel('Ошибка (CrossEntropyLoss)')
plt.xticks(range(1, num_epochs + 1))
plt.grid(True)
plt.show()

print("\n--- Визуализация результата ---")

dataiter = iter(test_loader)
images, labels = next(dataiter)

img_index = 5
image_to_show = images[img_index]
true_label = labels[img_index]

def imshow(img):
    img = img / 2 + 0.5
    npimg = img.numpy()
    plt.imshow(np.transpose(npimg, (1, 2, 0)))
    plt.axis('off')
    plt.show()

model.eval()
with torch.no_grad():
    output = model(image_to_show.to(device).unsqueeze(0))
```

```
    _, predicted_index = torch.max(output, 1)

print(f'Реальный класс:      {classes[true_label]}')
print(f'Предсказанный класс: {classes[predicted_index.item()]}')

imshow(image_to_show)
```

График ошибки и вывод программы:





SOTA-результат из статьи MDPI "State-of-the-Art Results with the Fashion-MNIST Dataset" (https://doi.org/10.3390/math12203174) — точность до 99.65% с моделью CNN-3-128, использующей 3 сверточных слоя, dropout, и аугментацию данных.

Разница в точности обусловлена:
- более сложной архитектурой модели с большим числом фильтров и слоёв,
- применением регуляризации (dropout) для снижения переобучения,
- аугментацией данных для улучшения обобщающей способности,
- оптимизацией гиперпараметров через серию вычислительных экспериментов,

Вывод: Я изучил построение модели СНС и ее обучение.