Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

Лабораторная работа №2

По дисциплине «Обработка изображений в ИС»

Тема: «Конструирование моделей на базе предобученных нейронных сетей»

Выполнил:

Студент 4 курса

Группы ИИ-21

Худик А.А.

Проверил:

Крощенко А.А.

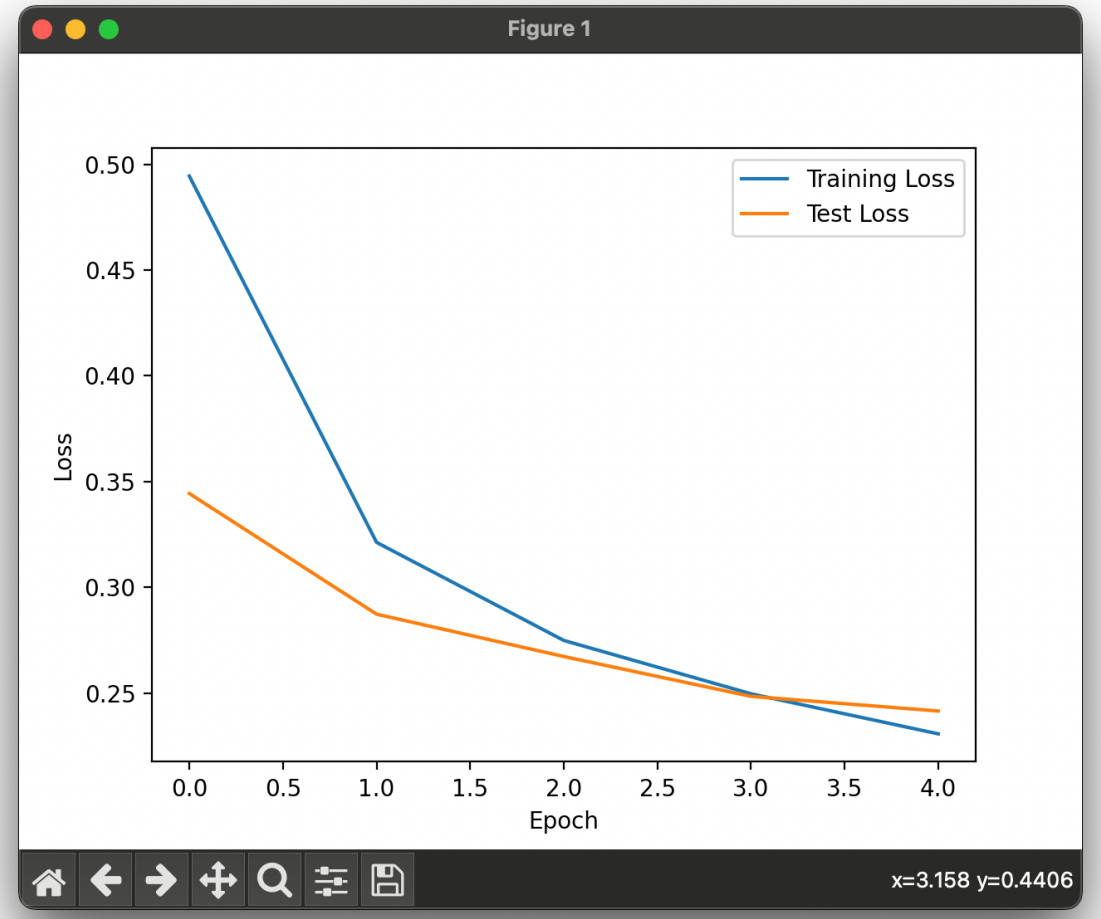
Брест 2024

Цель: осуществлять обучение НС, сконструированных на базе предобученных архитектур НС.

5	Fashion-MNIST	SGD	AlexNet

Код программы:

```
import torch
import torchvision
import torchvision.transforms as transforms
from torch import nn, optim
from torchvision.models import alexnet
import matplotlib.pyplot as plt
import matplotlib

matplotlib.use("MacOSX")

# Устройство для вычислений
device = torch.device("mps" if torch.mps.is_available() else "cpu")

# Подготовка данных с изменением размера и нормализацией
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize((0.5,), (0.5,))
])

train_data = torchvision.datasets.FashionMNIST(root='./data', train=True,
download=True, transform=transform)
test_data = torchvision.datasets.FashionMNIST(root='./data', train=False,
download=True, transform=transform)
train_loader = torch.utils.data.DataLoader(train_data, batch_size=32, shuffle=True)
test_loader = torch.utils.data.DataLoader(test_data, batch_size=32, shuffle=False)

# Подготовка модели AlexNet с изменением первого сверточного слоя и
выходного слоя
model = alexnet()
model.load_state_dict(torch.load('/Users/andrewhudik/Downloads/AlexNet Model
Weights.pth'))
model.features[0] = nn.Conv2d(1, 64, kernel_size=11, stride=4, padding=2)  #
Меняем первый слой на 1 канал
model.classifier[6] = nn.Linear(4096, 10)  # Меняем выходной слой под 10
классов
model = model.to(device)

# Настройка функции потерь и оптимизатора
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(model.parameters(), lr=0.001, momentum=0.9)

# Обучение модели
num_epochs = 5
train_losses, test_losses = [], []

for epoch in range(num_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_losses.append(running_loss / len(train_loader))

    # Оценка на тестовом наборе
    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_losses.append(test_loss / len(test_loader))
    print(f"Epoch {epoch+1}/{num_epochs}, Training Loss: {running_loss /
len(train_loader)}, Test Loss: {test_loss / len(test_loader)}")

# Построение графиков ошибки
plt.plot(train_losses, label='Training Loss')
plt.plot(test_losses, label='Test Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()

# Визуализация работы модели на тестовых изображениях
def visualize_model(model, dataloader):
    model.eval()
    with torch.no_grad():
        images, labels = next(iter(dataloader))
        images = images.to(device)
        outputs = model(images)
        _, preds = torch.max(outputs, 1)

        fig = plt.figure(figsize=(15, 10))
        for i in range(10):
            ax = fig.add_subplot(2, 5, i+1)
            ax.imshow(images[i].cpu().squeeze(), cmap='gray')
            ax.set_title(f'Predicted: {preds[i].item()}, True: {labels[i].item()}')
            ax.axis('off')
        plt.show()

visualize_model(model, test_loader)
```



Вывод: научился осуществлять обучение НС, сконструированных на базе предобученных архитектур НС.