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

Пучинский А.А.

Проверил:

Крощенко А.А.

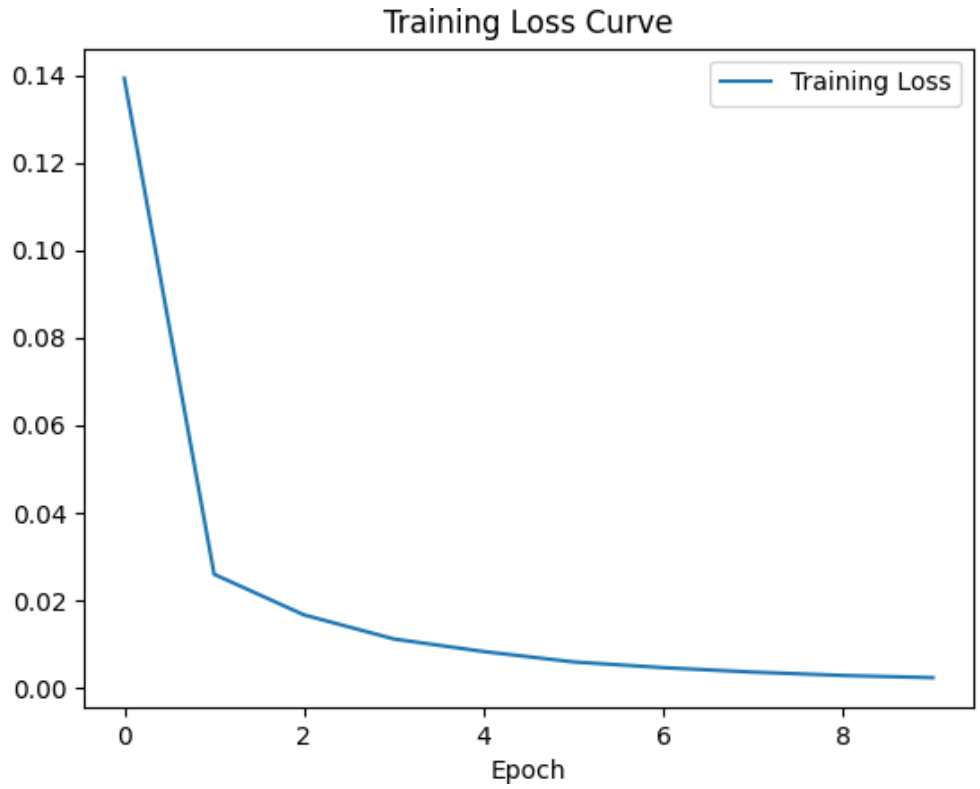
Брест 2024

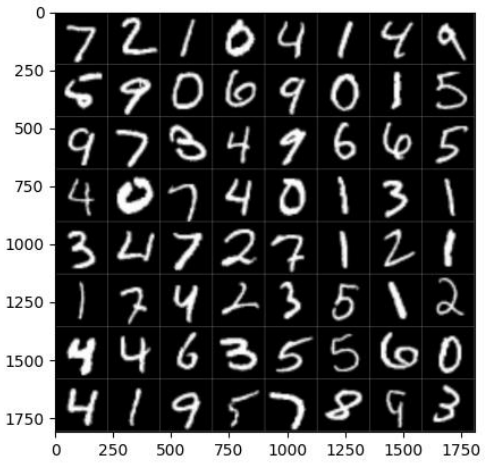
Цель: осуществлять обучение НС, сконструированных на базе предобученных архитектур НС.

3	MNIST	SGD	ResNet18

Код программы:

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

# Устройство для вычислений
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(f"Using device: {device}")

# Подготовка данных
transform = transforms.Compose([
    transforms.Grayscale(num_output_channels=3),  # Преобразование в трехканальные изображения
    transforms.Resize((224, 224)),  # Изменение размера для совместимости с ResNet
    transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
])

trainset = torchvision.datasets.MNIST(root='./data', train=True, download=True, transform=transform)
trainloader = torch.utils.data.DataLoader(trainset, batch_size=64, shuffle=True)

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

# Подготовка модели ResNet18 с изменением структуры
net = models.resnet18(pretrained=True)  # Загружаем предобученную модель
net.fc = nn.Linear(net.fc.in_features, 10)  # Замена выходного слоя для 10 классов (MNIST)
net = net.to(device)

# Критерий и оптимизатор
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(net.parameters(), lr=0.001, momentum=0.9)

# Функция обучения модели
def train_model(epochs=10):
    train_losses = []
    for epoch in range(epochs):
        running_loss = 0.0
        for i, data in enumerate(trainloader, 0):
            inputs, labels = data
            inputs, labels = inputs.to(device), labels.to(device)

            optimizer.zero_grad()
            outputs = net(inputs)
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()

            running_loss += loss.item()
        train_losses.append(running_loss / len(trainloader))
        print(f'Epoch {epoch+1}, Loss: {running_loss / len(trainloader)}')
    return train_losses

# Запуск обучения
epochs = 10
train_losses = train_model(epochs)

# График изменения ошибки
plt.plot(range(epochs), train_losses, label='Training Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Training Loss Curve')
plt.legend()
plt.show()

# Функция оценки точности
def evaluate_model():
    correct = 0
    total = 0
    with torch.no_grad():
        for data in testloader:
            images, labels = data
            images, labels = images.to(device), labels.to(device)
            outputs = net(images)
            _, predicted = torch.max(outputs.data, 1)
            total += labels.size(0)
            correct += (predicted == labels).sum().item()
    accuracy = 100 * correct / total
    print(f'Accuracy on test set: {accuracy:.2f}%')

# Оценка точности модели
evaluate_model()

# Визуализация работы модели
def imshow(img):
    img = img / 2 + 0.5  # денормализация для отображения
    npimg = img.numpy()
    plt.imshow(np.transpose(npimg, (1, 2, 0)))
    plt.show()

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

# Показ изображений
imshow(torchvision.utils.make_grid(images))

print('GroundTruth: ', ' '.join(f'{trainset.classes[labels[j]]}' for j in range(4)))

images = images.to(device)
outputs = net(images)
_, predicted = torch.max(outputs, 1)

print('Predicted: ', ' '.join(f'{trainset.classes[predicted[j]]}' for j in range(4)))
```





```
Epoch 1, Loss: 0.13922648613394767
Epoch 2, Loss: 0.025970448586674754
Epoch 3, Loss: 0.016695320367629923
Epoch 4, Loss: 0.011201719136120303
Epoch 5, Loss: 0.008327237579652341
Epoch 6, Loss: 0.005957100715984388
Epoch 7, Loss: 0.0046492192194797595
Epoch 8, Loss: 0.003635173032893293
Epoch 9, Loss: 0.002867764023139820 4
Epoch 10, Loss: 0.0023633672227393707
Accuracy on test set: 99.49%
GroundTruth:  7 - seven 2 - two 1 - one 0 - zero
Predicted:  7 - seven 2 - two 1 - one 0 - zero
```

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