МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙГОСУДАРСТВЕННЫЙТЕХНИЧЕСКИЙУНИВЕРСИТЕТ» ФАКУЛЬТЕТ

ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №2

Специальность ИИ-22

Выполнила
Леваневская Н.И.
студентка группы ИИ-22

Проверил
А.А. Крощенко,
ст. преп.
кафедры ИИТ,
«—» ———— 2024 г.

Брест 2024

Вариант 11

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

Задания:
Для заданной выборки и архитектуры предобученной нейронной организовать процесс обучения НС, предварительно изменив структуру слоев, в соответствии с предложенной выборкой. Использовать тот же оптимизатор, что и в ЛР №1. Построить график изменения ошибки и оценить эффективность обучения на тестовой выборке.

Выборка	**Предобученная архитектура**	**Оптимизатор**
CIFAR-10	DenseNet121	Adadelta

Код программы:

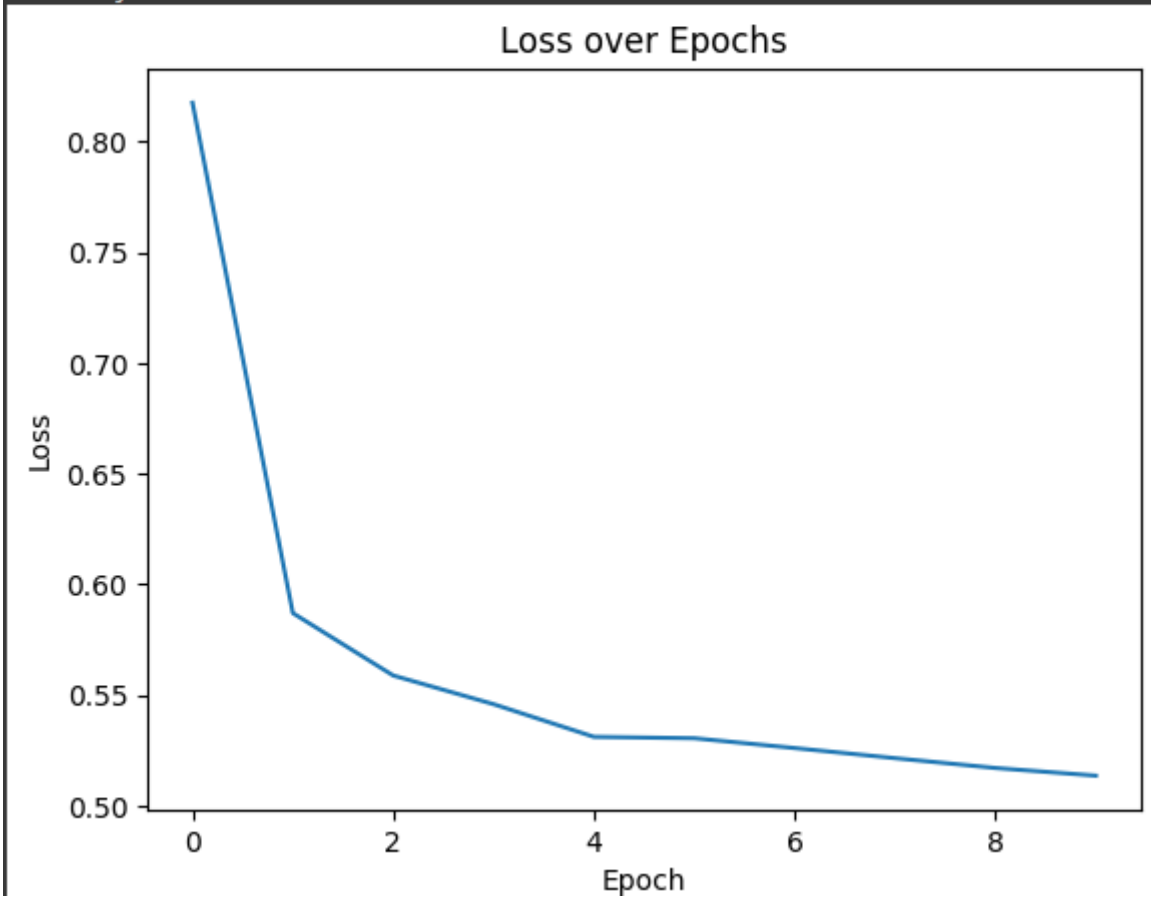
```
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision.transforms as transforms
from torchvision import datasets, models
from torch.utils.data import DataLoader
from tqdm import tqdm
import matplotlib.pyplot as plt

device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
print(f'Using device: {device}')
print(torch.version.cuda)

transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])

train_dataset = datasets.CIFAR10(root='./data', train=True, download=True,
transform=transform)
test_dataset = datasets.CIFAR10(root='./data', train=False, download=True,
transform=transform)

train_loader = DataLoader(train_dataset, batch_size=128, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=128, shuffle=False)

model = models.densenet121(pretrained=True)

for param in model.parameters():
    param.requires_grad = False

model.classifier = nn.Linear(model.classifier.in_features, 10)
```

```
model = model.to(device)

criterion = nn.CrossEntropyLoss()
optimizer = optim.Adadelta(model.parameters())

def train_model(num_epochs):
    train_losses = []
    for epoch in range(num_epochs):
        model.train()
        running_loss = 0.0
        progress_bar = tqdm(train_loader, desc=f'Epoch {epoch+1}/{num_epochs}',
leave=False)

        for inputs, labels in progress_bar:
            inputs, labels = inputs.to(device), labels.to(device)

            outputs = model(inputs)
            loss = criterion(outputs, labels)

            optimizer.zero_grad()
            loss.backward()
            optimizer.step()

            running_loss += loss.item()
            progress_bar.set_postfix({'loss': loss.item()})

        epoch_loss = running_loss / len(train_loader)
        train_losses.append(epoch_loss)
        print(f'Epoch {epoch+1}/{num_epochs}, Loss: {epoch_loss}')

    return train_losses

def evaluate_model():
    model.eval()
    correct = 0
    total = 0
    with torch.no_grad():
        for inputs, labels in test_loader:
            inputs, labels = inputs.to(device), labels.to(device)
            outputs = model(inputs)
            _, 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}%')

def plot_loss_curve(train_losses):
    plt.plot(train_losses)
    plt.title('Loss over Epochs')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.show()
```

```
num_epochs = 10
train_losses = train_model(num_epochs)
evaluate_model()
plot_loss_curve(train_losses)
```

Результаты обучения:

```
Using device: cuda
12.1
Files already downloaded and verified
Files already downloaded and verified
Epoch 1/10, Loss: 0.8175325515629995
Epoch 2/10, Loss: 0.5869439331162006
Epoch 3/10, Loss: 0.5587420123617363
Epoch 4/10, Loss: 0.5457849147374673
Epoch 5/10, Loss: 0.5310159446028493
Epoch 6/10, Loss: 0.5303959966163196
Epoch 7/10, Loss: 0.5259580541297298
Epoch 8/10, Loss: 0.5213889450673252
Epoch 9/10, Loss: 0.5169552611115643
Epoch 10/10, Loss: 0.513492008990339
Accuracy on test set: 78.67%
```



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