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

Годынюк И.А.

Проверил:

Крощенко А.А.

Брест 2024

Вариант 9

9	CIFAR-10	Adam	MobileNet v3

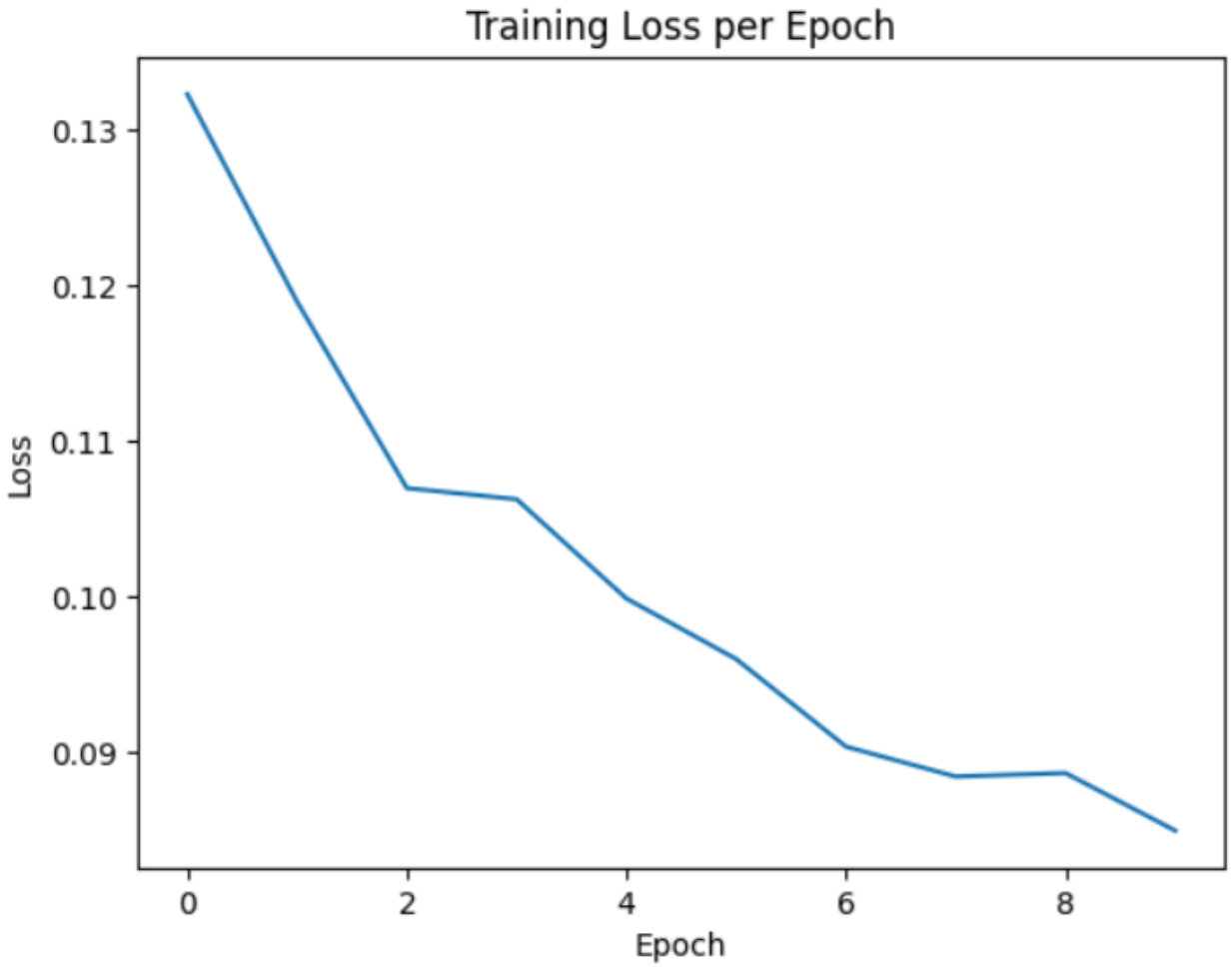
Цель: научиться конструировать нейросетевые классификаторы и выполнять их обучение на известных выборках компьютерного зрения.

Код программы:

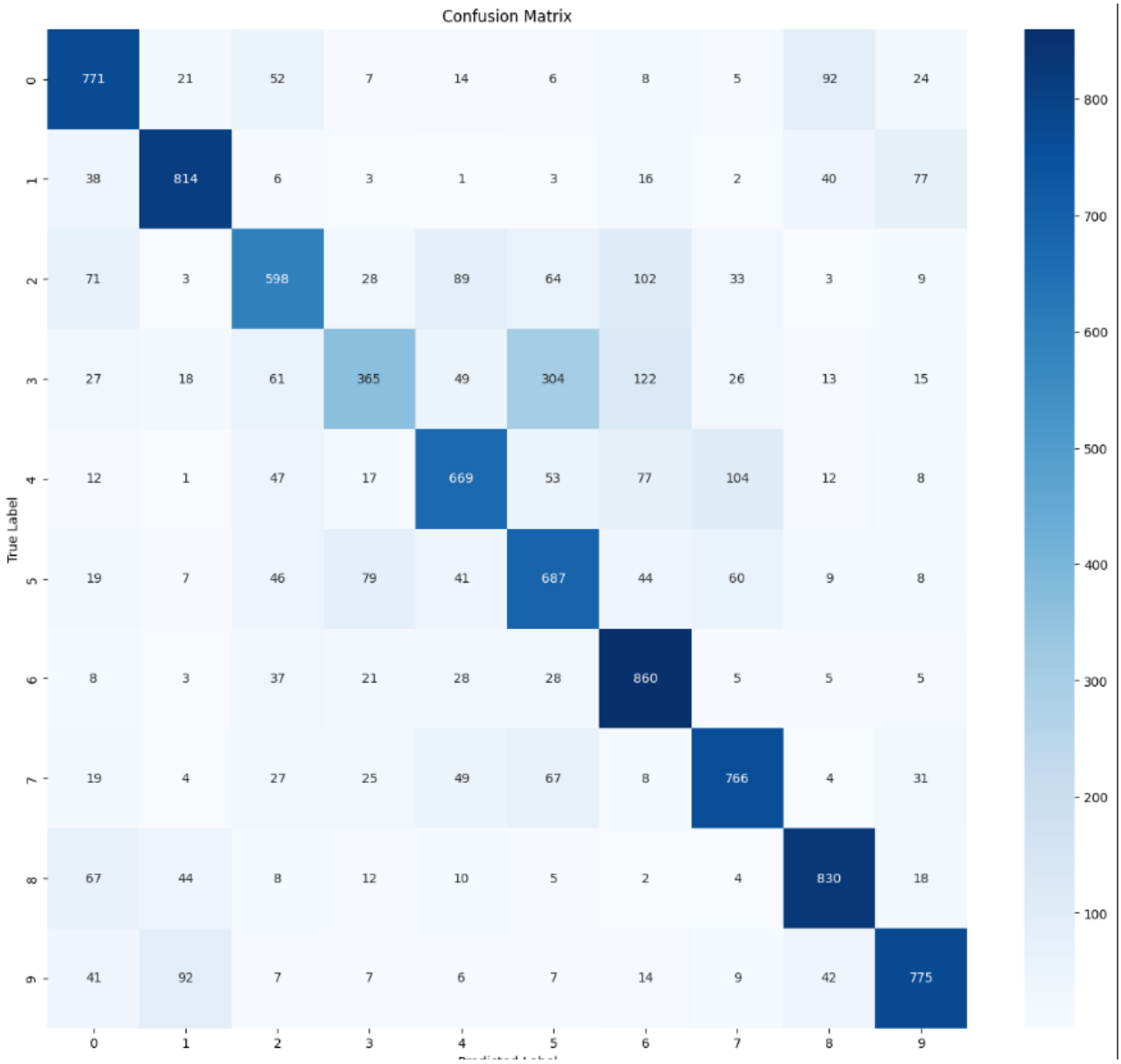
```
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision
import torchvision.transforms as transforms
from torchvision import models
import matplotlib.pyplot as plt

# 1. Загрузка и предобработка данных с аугментацией для CIFAR-10
transform_train = transforms.Compose([
    transforms.RandomHorizontalFlip(),
    transforms.RandomCrop(32, padding=4),
    transforms.ToTensor(),
    transforms.Normalize((0.4914, 0.4822, 0.4465), (0.2023, 0.1994, 0.2010))  # Нормализация
для CIFAR-10
])

transform_test = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.4914, 0.4822, 0.4465), (0.2023, 0.1994, 0.2010))
])

trainset = torchvision.datasets.CIFAR10(root='./data', train=True, download=True,
transform=transform_train)
trainloader = torch.utils.data.DataLoader(trainset, batch_size=64, shuffle=True)

testset = torchvision.datasets.CIFAR10(root='./data', train=False, download=True,
transform=transform_test)
testloader = torch.utils.data.DataLoader(testset, batch_size=64, shuffle=False)

# 2. Загрузка MobileNet v3
model = models.mobilenet_v3_small(pretrained=True)

# Изменяем последний слой под количество классов CIFAR-10
model.classifier[3] = nn.Linear(model.classifier[3].in_features, 10)

# 3. Инициализация устройства (GPU/CPU), функции потерь и оптимизатора
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model = model.to(device)

criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)
scheduler = optim.lr_scheduler.StepLR(optimizer, step_size=30, gamma=0.1)

# 4. Функция обучения модели (аналогична той, что у тебя была)
def train_model(model, trainloader, criterion, optimizer, scheduler, epochs=10):
    train_loss = []
    model.train()
    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 = model(inputs)
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()

            running_loss += loss.item()
            if i % 100 == 99:  # Печать каждые 100 batch'ей
                print(f'[Epoch {epoch + 1}, Batch {i + 1}] Loss: {running_loss / 100:.3f}')
                running_loss = 0.0

        scheduler.step()
        train_loss.append(running_loss / len(trainloader))

    print('Finished Training')
    return train_loss

# 5. Обучение модели
train_loss = train_model(model, trainloader, criterion, optimizer, scheduler, epochs=10)

# 6. Тестирование модели и построение матрицы ошибок (как в ЛР №1)
test_model_with_confusion_matrix(model, testloader)

# 7. Построение графика ошибки
plt.plot(train_loss)
plt.title('Training Loss per Epoch')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.show()
```

График изменения ошибки при обучении



Точность на тестовой выборке



Точность на предобученной:

```
Accuracy of the network on the 10000 test images: 71.35%
```

Точность на не предобученной:

```
Accuracy of the network on the 10000 test images: 50.39%
```

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