Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №1

По дисциплине: «Обработка изображений в ИС»

Тема: «Обучение классификаторов средствами библиотеки PyTorch»

Выполнила:

Студентка 4 курса

Группы ИИ-21

Шнур А.А.

Проверил:

Крощенко А.А.

Брест 2024

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

Ход работы:

Вариант 15

№ варианта	**Выборка**	**Размер исходного изображения**	**Оптимизатор**
15	STL-10 (размеченная часть)	96X96	Adadelta

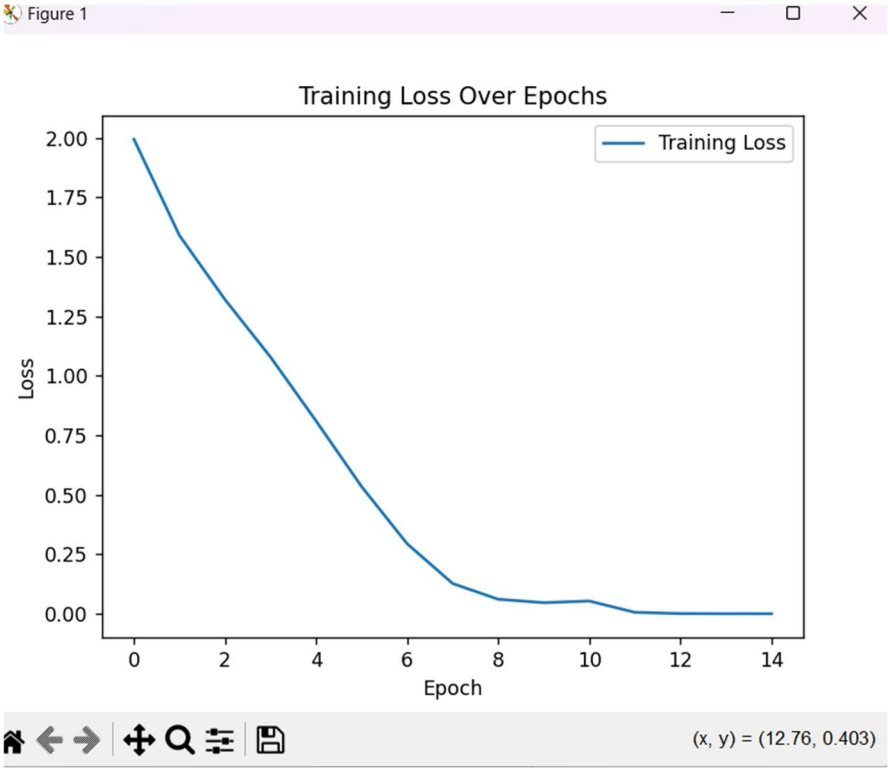
Код программы:

```
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

# Параметры обучения
batch_size = 64
learning_rate = 1.0  # для оптимизатора Adadelta не нужно устанавливать низкое значение
num_epochs = 15

# Загрузка STL-10 с преобразованием изображений
transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))  # нормализация
])

train_dataset = torchvision.datasets.STL10(root='./data', split='train', download=True,
transform=transform)
test_dataset = torchvision.datasets.STL10(root='./data', split='test', download=True,
transform=transform)

train_loader = torch.utils.data.DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
test_loader = torch.utils.data.DataLoader(test_dataset, batch_size=batch_size, shuffle=False)

# Определение простой архитектуры CNN
class SimpleCNN(nn.Module):
    def __init__(self):
        super(SimpleCNN, self).__init__()
        self.conv1 = nn.Conv2d(3, 32, kernel_size=3, stride=1, padding=1)
        self.pool = nn.MaxPool2d(kernel_size=2, stride=2, padding=0)
        self.conv2 = nn.Conv2d(32, 64, kernel_size=3, stride=1, padding=1)
        self.fc1 = nn.Linear(64 * 24 * 24, 128)
        self.fc2 = nn.Linear(128, 10)
        self.relu = nn.ReLU()

    def forward(self, x):
```

```
        x = self.relu(self.conv1(x))
        x = self.pool(x)
        x = self.relu(self.conv2(x))
        x = self.pool(x)
        x = x.view(-1, 64 * 24 * 24)  # изменение формы тензора
        x = self.relu(self.fc1(x))
        x = self.fc2(x)
        return x

# Инициализация модели, оптимизатора и функции потерь
model = SimpleCNN()
optimizer = optim.Adadelta(model.parameters(), lr=learning_rate)
criterion = nn.CrossEntropyLoss()

# Обучение модели
train_losses = []
for epoch in range(num_epochs):
    running_loss = 0.0
    for images, labels in train_loader:
        optimizer.zero_grad()
        outputs = model(images)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        running_loss += loss.item()

    avg_loss = running_loss / len(train_loader)
    train_losses.append(avg_loss)
    print(f'Epoch [{epoch+1}/{num_epochs}], Loss: {avg_loss:.4f}')

# График ошибки обучения
plt.plot(train_losses, label='Training Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Training Loss Over Epochs')
plt.legend()
plt.show()

# Оценка эффективности на тестовой выборке
model.eval()
correct = 0
total = 0
with torch.no_grad():
    for images, labels in test_loader:
        outputs = model(images)
        _, predicted = torch.max(outputs.data, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()

print(f'Accuracy of the model on the test images: {100 * correct / total:.2f}%')

# Визуализация классификации произвольного изображения из тестового набора
sample_image, sample_label = test_dataset[0]
model.eval()
with torch.no_grad():
    output = model(sample_image.unsqueeze(0))  # добавление размерности batch
    _, prediction = torch.max(output.data, 1)

# Отображение изображения и результата предсказания
```

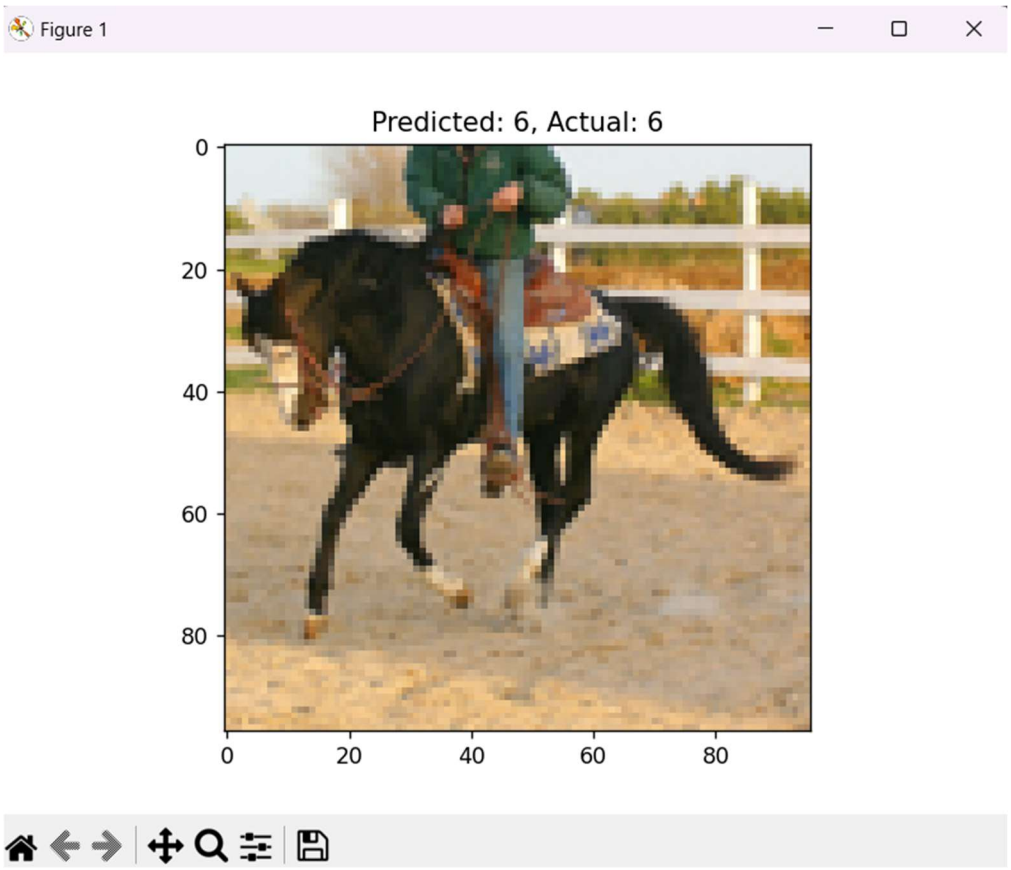
```
plt.imshow(np.transpose(sample_image.numpy(), (1, 2, 0)) * 0.5 + 0.5)  # обратная нормализация
plt.title(f'Predicted: {prediction.item()}, Actual: {sample_label}')
plt.show()
```

График изменения ошибки:



Среднее значение потерь для каждой эпохи и точность на тестовой выборке:

```
PS D:\Лабы 4 курс\ОиИС\Lab1> & C:/Users/HP/AppData
1.py"
Files already downloaded and verified
Files already downloaded and verified
Epoch [1/15], Loss: 1.9929
Epoch [2/15], Loss: 1.5888
Epoch [3/15], Loss: 1.3196
Epoch [4/15], Loss: 1.0781
Epoch [5/15], Loss: 0.8097
Epoch [6/15], Loss: 0.5340
Epoch [7/15], Loss: 0.2939
Epoch [8/15], Loss: 0.1269
Epoch [9/15], Loss: 0.0609
Epoch [10/15], Loss: 0.0466
Epoch [11/15], Loss: 0.0534
Epoch [12/15], Loss: 0.0060
Epoch [13/15], Loss: 0.0008
Epoch [14/15], Loss: 0.0003
Epoch [15/15], Loss: 0.0002
Accuracy of the model on the test images: 56.55%
```



Получили:

```
Accuracy of the model on the test images: 56.55%
```

Вывод: в ходе лабораторной работы научилась конструировать нейросетевые классификаторы и выполнять их обучение на известных выборках компьютерного зрения.