

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»

Выполнил:

Студент 4 курса

Группы ИИ-21

Пучинский А.А.

Проверил:

Крощенко А.А.

Брест 2024

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

3	CIFAR-10	32X32	SGD
---	----------	-------	-----

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

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

transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])

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

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

class SimpleCNN(nn.Module):
    def __init__(self):
        super(SimpleCNN, self).__init__()
        self.conv1 = nn.Conv2d(3, 32, 3, padding=1)
        self.conv2 = nn.Conv2d(32, 64, 3, padding=1)
        self.pool = nn.MaxPool2d(2, 2)
        self.fc1 = nn.Linear(64 * 8 * 8, 512)
        self.fc2 = nn.Linear(512, 10)
        self.relu = nn.ReLU()

    def forward(self, x):
        x = self.pool(self.relu(self.conv1(x)))
        x = self.pool(self.relu(self.conv2(x)))
        x = x.view(-1, 64 * 8 * 8)
        x = self.relu(self.fc1(x))
        x = self.fc2(x)
        return x

net = SimpleCNN().to(device)
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(net.parameters(), lr=0.001,
momentum=0.9)

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

epochs = 10
train_losses = train_model(epochs)

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

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) #
Перенос данных на устройство
            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}%')

evaluate_model()

def imshow(img):
    img = img / 2 + 0.5
    npimg = img.numpy()
    plt.imshow(np.transpose(npimg, (1, 2, 0)))
    plt.show()

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

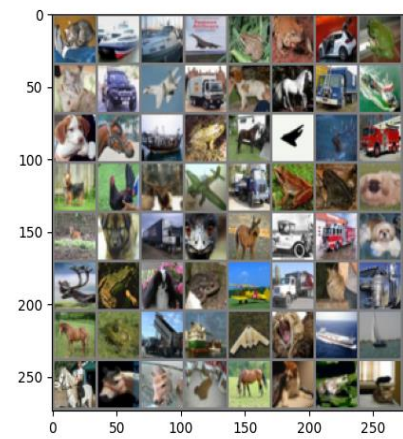
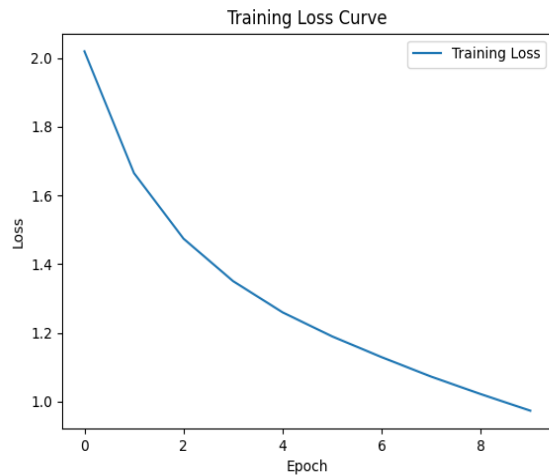
imshow(torchvision.utils.make_grid(images))

print('GroundTruth: ', ' '.join('%5s' % trainset.classes[labels[j]] for j
in range(4)))

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

print('Predicted: ', ' '.join('%5s' % trainset.classes[predicted[j]] for j
in range(4)))
```

Figure 1



(x, y) = (5.60, 1.650)



```

Using device: cpu
Files already downloaded and verified
Files already downloaded and verified
Epoch 1, Loss: 2.0854258555585465
Epoch 2, Loss: 1.67141945038915
Epoch 3, Loss: 1.4542370252597057
Epoch 4, Loss: 1.344760413060103
Epoch 5, Loss: 1.2577967312939637
Epoch 6, Loss: 1.1823473764807366
Epoch 7, Loss: 1.121997518155276
Epoch 8, Loss: 1.060880561161529
Epoch 9, Loss: 1.0083717703819275
Epoch 10, Loss: 0.9585007263723847
Accuracy on test set: 65.35%
GroundTruth: cat ship ship airplane
Predicted: cat automobile airplane airplane
  
```

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