Министерство образования Республики Беларусь
Учреждение образования
«Брестский Государственный технический университет»
Кафедра ИИТ

Лабораторная работа №2

По дисциплине: «Модели решения задач в интеллектуальных системах»

Тема: «Конструирование моделей на базе предобученных нейронных сетей»

Выполнил:
Студент 4 курса
Группы ИИ-21
Ясюкевич В.С.
Проверил:
Крощенко А. А.

Брест 2024

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

Ход работы:

Вариант 17

В-т	Выборка	Оптимизатор	Предобученная архитектура
17	STL-10 (размеченная часть)	RMSprop	MobileNet v2

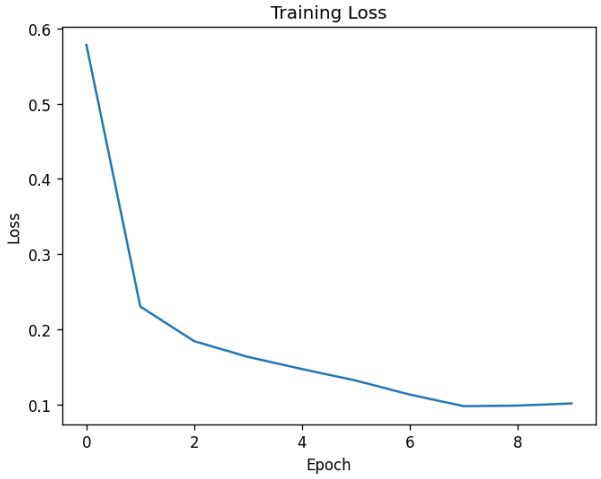
Код программы:

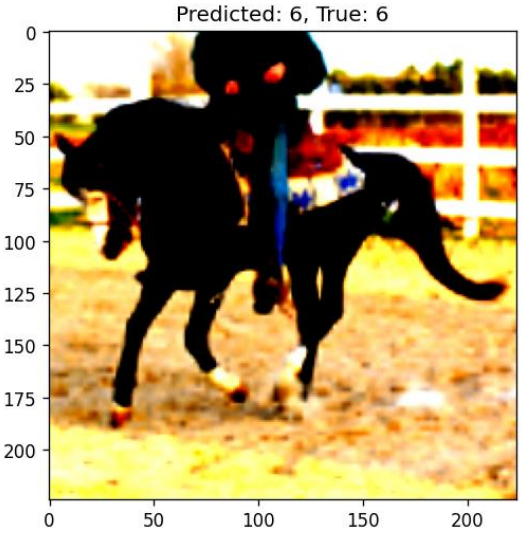
```
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision
import torchvision.transforms as
transforms
import torchvision.models as models
import matplotlib.pyplot as plt
print(torch.cuda.is_available())
device = torch.device("cuda" if
torch.cuda.is_available() else "cpu")
print(f"Using device: {device}")
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 =
torchvision.datasets.STL10(root='./data
', split='train', transform=transform,
download=True)
test_dataset =
torchvision.datasets.STL10(root='./data
', split='test', transform=transform,
download=True)
train_loader =
torch.utils.data.DataLoader(dataset=tra
in_dataset, batch_size=64,
shuffle=True)
test_loader =
torch.utils.data.DataLoader(dataset=tes
t_dataset, batch_size=1000,
shuffle=False)
model =
models.mobilenet_v2(weights=models.Mobi
leNet_V2_Weights.IMAGENET1K_V1)
for param in model.parameters():
    param.requires_grad = False
model.classifier[1] =
nn.Linear(model.classifier[1].in_featur
es, 10)
model = model.to(device)
criterion = nn.CrossEntropyLoss()
optimizer =
optim.RMSprop(model.parameters(),
lr=0.001)
scheduler =
torch.optim.lr_scheduler.StepLR(optimiz
er, step_size=7, gamma=0.25)
def train_model(num_epochs):
    model.train()
    train_loss_history = []
    for epoch in range(num_epochs):
        running_loss = 0.0
        for images, labels in
train_loader:
            images, labels =
images.to(device), labels.to(device)

            optimizer.zero_grad()
            outputs = model(images)
            loss = criterion(outputs,
labels)
            loss.backward()
            optimizer.step()
            running_loss += loss.item()
        epoch_loss = running_loss /
len(train_loader)
        train_loss_history.append(epoch
_loss)
        scheduler.step()
        print(f'Epoch
[{epoch+1}/{num_epochs}], Loss:
{epoch_loss:.4f}')
    return train_loss_history
def test_model():
    model.eval()
    correct = 0
    total = 0
```

```
    with torch.no_grad():
        for images, labels in
test_loader:
            images, labels =
images.to(device), labels.to(device)
            outputs = model(images)
            _, predicted =
torch.max(outputs.data, 1)
            total += labels.size(0)
            correct += (predicted ==
labels).sum().item()
    accuracy = 100 * correct / total
    print(f'Accuracy on the test set:
{accuracy:.2f}%')
    return accuracy
def plot_loss_history(loss_history):
    plt.plot(loss_history)
    plt.title('Training Loss')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.show()
num_epochs = 10
loss_history = train_model(num_epochs)
test_model()
plot_loss_history(loss_history)
def visualize_prediction(image_index):
    image, label =
test_dataset[image_index]
    model.eval()
    with torch.no_grad():
        image =
image.to(device).unsqueeze(0)
        output = model(image)
        _, predicted =
torch.max(output.data, 1)
        plt.imshow(image.cpu().squeeze(
).permute(1, 2, 0))
        plt.title(f'Predicted:
{predicted.item()}, True: {label}')
        plt.show()
visualize_prediction(0)
```

Результат программы:





```
Epoch [1/10], Loss: 0.5783
Epoch [2/10], Loss: 0.2302
Epoch [3/10], Loss: 0.1842
Epoch [4/10], Loss: 0.1632
Epoch [5/10], Loss: 0.1471
Epoch [6/10], Loss: 0.1318
Epoch [7/10], Loss: 0.1132
Epoch [8/10], Loss: 0.0978
Epoch [9/10], Loss: 0.0985
Epoch [10/10], Loss: 0.1014
Accuracy on the test set: 94.51%
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers). Got range
[-1.9124069..2.5005665].
```

Вывод: осуществил обучение НС, сконструированных на базе предобученных архитектур НС.