ECE371 Neural Networks and Deep Learning

Assignment 1

卢钰临 22308127
School of Electronics and Communication Engineering
Sun Yat-sen University, Shenzhen Campus
Luylin25@mail2.sysu.edu.cn

Abstract：通过学习深度神经网络和 Pytorch 的相关知识，我已拥有基础的神经网络理论知识。通过我所学的知识完善了代码，做了图片的预处理，模型全连接层的修改，损失函数和学习速率函数的调用，反向传播函数的调用。结合本次课程作业，我将所学的理论知识与实际项目结合，学习项目的代码框架，理解了图片分类项目的基本程序。

Introduction

本项目是对五种类型的花卉进行分类学习，需要实现数据的导入和预处理。完善代码框架并运行程序。我已完成对图片的处理以及部分深度学习相关函数的调用，并解决了部分程序报错问题，成功运行了该项目而实现了基本的花卉图片分类功能。

Related Work

通过上课以及课后知识查阅学习，我运用了相关知识，包括：逻辑回归和损失函数、梯度下降算法、交叉熵损失、浅层神经网络搭建。主要参考 吴恩达 CNN、RNN、GAN、GNN、DQN、Transformer、LSTM 深度学习课程，结合本课程的 PPT 进行学习。

Method

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

----------
Learning Rate: 0.000001
train Loss: 0.2315 Acc: 0.9258
val Loss: 0.2269 Acc: 0.9193

Epoch 23/24
----------
Learning Rate: 0.000001
train Loss: 0.2231 Acc: 0.9223
val Loss: 0.2591 Acc: 0.9105

val Loss: 0.2591 Acc: 0.9105
val Loss: 0.2591 Acc: 0.9105

Epoch 24/24
----------
Learning Rate: 0.000001
train Loss: 0.2175 Acc: 0.9188
val Loss: 0.2153 Acc: 0.9193

Training complete in 14m 56s
Best val Acc: 0.926316
PS C:\Users\23831\Desktop\中山大學\VSCodeData>
```

代码运行结果：其中学习速率为 0.000001，训练集损失 0.2175，准确度 0.9188，测试集损失 0.2153 ，准确度 0.9193，与训练集几乎一致，且准确度均在 90%以上，说明模型学习效果好且未发生过拟合。

Experiments

```python
data_transforms = transforms.Compose([
        # GRADED FUNCTION: Add five data augmentation methods, Normalizating and Tranform to tensor
        ### START SOLUTION HERE ###

        # 随机裁剪为224x224大小
        transforms.RandomResizedCrop(224),
        # 随机旋转±30度
        transforms.RandomRotation(20),
        # 随机水平翻转
        transforms.RandomHorizontalFlip(),
        # 随机垂直翻转
        transforms.RandomVerticalFlip(),
        # 色彩抖动（亮度/对比度/饱和度/色相）
        transforms.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2),
        # 转换为张量
        transforms.ToTensor(),
        # 标准化处理（使用ImageNet均值和标准差）
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])

        # Add five data augmentation methods, Normalizating and Tranform to tensor
        ### END SOLUTION HERE ###
])
```

此部分是对数据的预处理。该神经网络架构对输入变量数量需要统一，而图片的大小及尺寸不一，若直接输入会造成变量数目错误的问题。因此对图片进行大小统一以及合适的裁切，并进行标准化以便于模型对数据的处理。同时对图片输入进行旋转、翻转、色彩抖动等处理，是为了丰富训练集的训练数据，使其可以涵盖更多的数据可能性，同时这也是减少过拟合的方法之一。

```
# GRADED FUNCTION: Modify the last fully connected layer of model
### START SOLUTION HERE ###
# Modify the last fully connected layer of model

# 获取原始全连接层的输入特征数
num_ftrs = model.fc.in_features
# 获取数据集中的类别数量
num_classes = len(class_names)
# 替换全连接层，输出维度改为实际类别数
model.fc = nn.Linear(num_ftrs, num_classes)

### END SOLUTION HERE ###
```

全连接层是神经网络的最后一层，一般采用 sigmoid 函数实现类别检测概率。这里需要将全连接层的输出数量改为预测的花卉类型数量，即 5 种，而全连接层的输入可以通过 model.fc.in_features 获取，修改参数后替换全连接层。

```
# 使用交叉熵损失函数
criterion = nn.CrossEntropyLoss()
```

交叉熵损失函数是衡量模型训练效果的一项函数，可以直接调用。

```
# 使用随机梯度下降优化器（SGD）
# lr=0.001 初始学习率，momentum=0.9 动量参数
optimizer = optim.SGD(model.parameters(), lr=0.001, momentum=0.9)
```

梯度下降优化器 SGD 可以自适应优化学习的速率，在参数初始值开始的时候采用较高的学习速率可以加速模型的训练。随着训练后参数逐渐接近理想，学习率会逐步下降（本实验学习速率最后只有 0.000001）。在梯度下降算法中，每次更新参数时都需要计算当前点的梯度，计算量大且易受噪声干扰。引入动量参数后，可以在每次更新时考虑之前梯度的信息，使得更新更加平滑，降低噪声干扰。

```
# Backward pass and optimization

# 计算梯度
loss.backward()
# 更新模型参数
optimizer.step()
```

参数w和b的更新公式为：

$$w := w - \alpha \frac{dJ(w,b)}{dw}, \quad b := b - \alpha \frac{dJ(w,b)}{db}$$

反相传播与优化是对参数的迭代更新。先调用 Pytorch 中的 loss.backward()函数计算该模型当前训练的梯度，根据公式可以用梯度计算更新后的参数。再调用 optimizer.step()函数更新模型的参数。

Reference

吴恩达《神经网络和深度学习》、《改善深层神经网络》