暨南大学本科实验报告专用纸

课程名称 深度学习实验 成绩评定

实验项目名称 图像分类的迁移学习 指导教师 林聪

实验项目编号 03 实验项目类型 验证型

学生姓名 赵俊文 学号 2022104002

学院 智能科学与工程学院 专业 人工智能

实验时间 2025 年 5 月 2 日～ 5 月 9 日 上 午

（一） 实验目的

- 分别加载 ResNet-18 和 VGG-16 模型的预训练权重，并对现有框架进行修改
- 通过在新数据集上训和练验证,利用模型微调方式实现迁移学习.输出训练曲线图. 对比曲线图与准确率

（二） 主要仪器设备

仪器： PC

实验环境： Windows11,Python3.10,Pytorch11.8

（三） 源程序

源程序在实验步骤与调试中给出。

（四） 实验步骤与调试

1. 下载热狗数据集

```
import torch
import torch.nn as nn
from torchvision import datasets, transforms, models
from torch.utils.data import DataLoader
import matplotlib.pyplot as plt
import os
import zipfile
import requests

# 下载数据集
url = 'http://d2l-data.s3-accelerate.amazonaws.com/hotdog.zip'
save_path = 'hotdog.zip'
```

```
if not os.path.exists('hotdog'):
    print("Downloading dataset...")
    r = requests.get(url, stream=True)
    with open(save_path, 'wb') as f:
        for chunk in r.iter_content(chunk_size=8192):
            if chunk:
                f.write(chunk)
    with zipfile.ZipFile(save_path, 'r') as zip_ref:
        zip_ref.extractall('.')
    os.remove(save_path)
    print("Dataset downloaded and extracted.")
```

2. 数据预处理与加载

```
# 数据预处理
transform_train = transforms.Compose([
    transforms.RandomResizedCrop(224),
    transforms.RandomHorizontalFlip(),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224,
0.225])
])

transform_test = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224,
0.225])
])

# 加载数据集
train_dataset = datasets.ImageFolder('hotdog/train',
transform=transform_train)
test_dataset = datasets.ImageFolder('hotdog/test',
transform=transform_test)

batch_size = 32
train_loader = DataLoader(train_dataset, batch_size=batch_size,
shuffle=True, num_workers=4)
```

```
test_loader = DataLoader(test_dataset, batch_size=batch_size,
shuffle=False, num_workers=4)

# 检查设备
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(f"Using device: {device}")
```

3. 训练与验证函数

(1) ResNet-18

```
def build_resnet18():
    model =
models.resnet18(weights=models.ResNet18_Weights.IMAGENET1K_V1)
    num_ftrs = model.fc.in_features
    model.fc = nn.Linear(num_ftrs, 2)
    model = model.to(device)
    return model
```

(2) VGG-16

```
def build_vgg16():
    model = models.vgg16(weights=models.VGG16_Weights.IMAGENET1K_V1)
    num_ftrs = model.classifier[6].in_features
    model.classifier[6] = nn.Linear(num_ftrs, 2)
    model = model.to(device)
    return model
```

4. 训练与验证函数

```
def train_model(model, criterion, optimizer, num_epochs=10):
    train_losses, train_accs, val_losses, val_accs = [], [], [], []
    best_acc = 0.0

    for epoch in range(num_epochs):
        model.train()
        running_loss = 0.0
        correct = 0
        total = 0

        # 训练阶段
```

```
        for inputs, labels in train_loader:
            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()
            _, predicted = outputs.max(1)
            total += labels.size(0)
            correct += predicted.eq(labels).sum().item()

        # 计算训练指标
        train_loss = running_loss / len(train_loader)
        train_acc = correct / total
        train_losses.append(train_loss)
        train_accs.append(train_acc)

        # 验证阶段
        model.eval()
        val_loss = 0.0
        val_correct = 0
        val_total = 0

        with torch.no_grad():
            for inputs, labels in test_loader:
                inputs, labels = inputs.to(device), labels.to(device)
                outputs = model(inputs)
                loss = criterion(outputs, labels)
                val_loss += loss.item()
                _, predicted = outputs.max(1)
                val_total += labels.size(0)
                val_correct += predicted.eq(labels).sum().item()

        # 计算验证指标
        val_loss = val_loss / len(test_loader)
        val_acc = val_correct / val_total
        val_losses.append(val_loss)
        val_accs.append(val_acc)

        # 更新最佳准确率
        if val_acc > best_acc:
            best_acc = val_acc
```

```
        print(f'Epoch {epoch+1}/{num_epochs}')
        print(f'Train Loss: {train_loss:.4f} Acc: {train_acc:.4f}')
        print(f'Val Loss: {val_loss:.4f} Acc: {val_acc:.4f}\n')

    return train_losses, train_accs, val_losses, val_accs, best_acc
```

5. 训练模型并保存结果

```
# 训练 ResNet-18
print("Training ResNet-18...")
model_resnet = build_resnet18()
criterion = nn.CrossEntropyLoss()
optimizer_resnet = torch.optim.SGD(model_resnet.parameters(), lr=0.001,
momentum=0.9)
train_loss_res, train_acc_res, val_loss_res, val_acc_res, best_resnet =
train_model(model_resnet, criterion, optimizer_resnet, 10)

# 训练 VGG-16
print("Training VGG-16...")
model_vgg = build_vgg16()
optimizer_vgg = torch.optim.SGD(model_vgg.parameters(), lr=0.001,
momentum=0.9)
train_loss_vgg, train_acc_vgg, val_loss_vgg, val_acc_vgg, best_vgg =
train_model(model_vgg, criterion, optimizer_vgg, 10)
```

6. 绘制训练曲线

```
def plot_curves(model_name, train_loss, train_acc, val_loss, val_acc):
    plt.figure(figsize=(12, 5))
    plt.subplot(1, 2, 1)
    plt.plot(train_loss, label='Train Loss')
    plt.plot(val_loss, label='Val Loss')
    plt.title(f'{model_name} Loss')
    plt.xlabel('Epoch')
    plt.legend()

    plt.subplot(1, 2, 2)
```

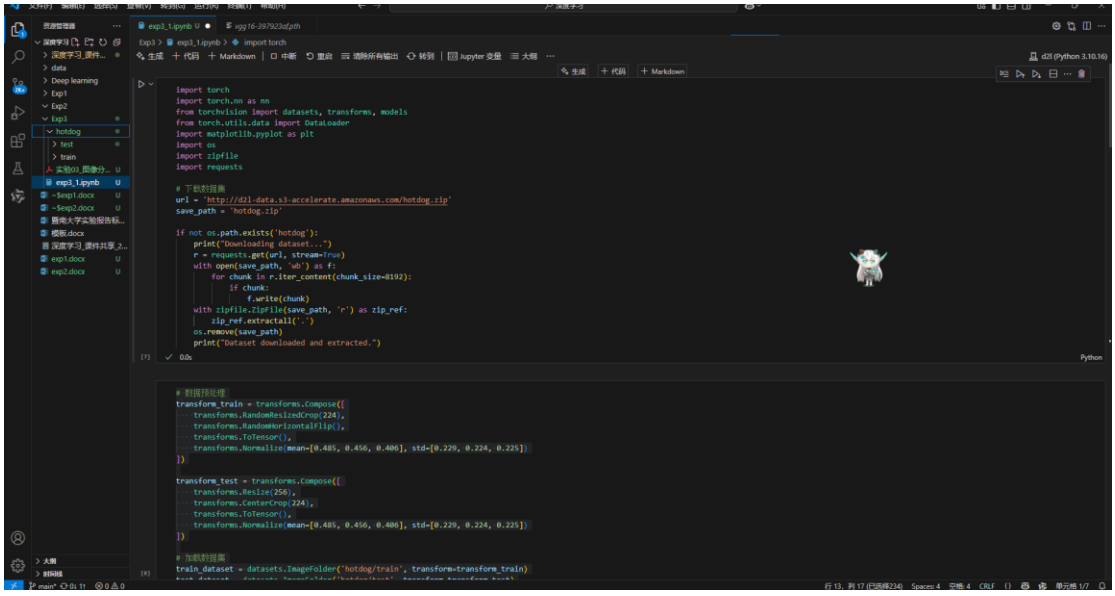
```
    plt.plot(train_acc, label='Train Acc')
    plt.plot(val_acc, label='Val Acc')
    plt.title(f'{model_name} Accuracy')
    plt.xlabel('Epoch')
    plt.legend()
    plt.show()

plot_curves('ResNet-18', train_loss_res, train_acc_res, val_loss_res,
val_acc_res)
plot_curves('VGG-16', train_loss_vgg, train_acc_vgg, val_loss_vgg,
val_acc_vgg)
```

7. 输出结果表格

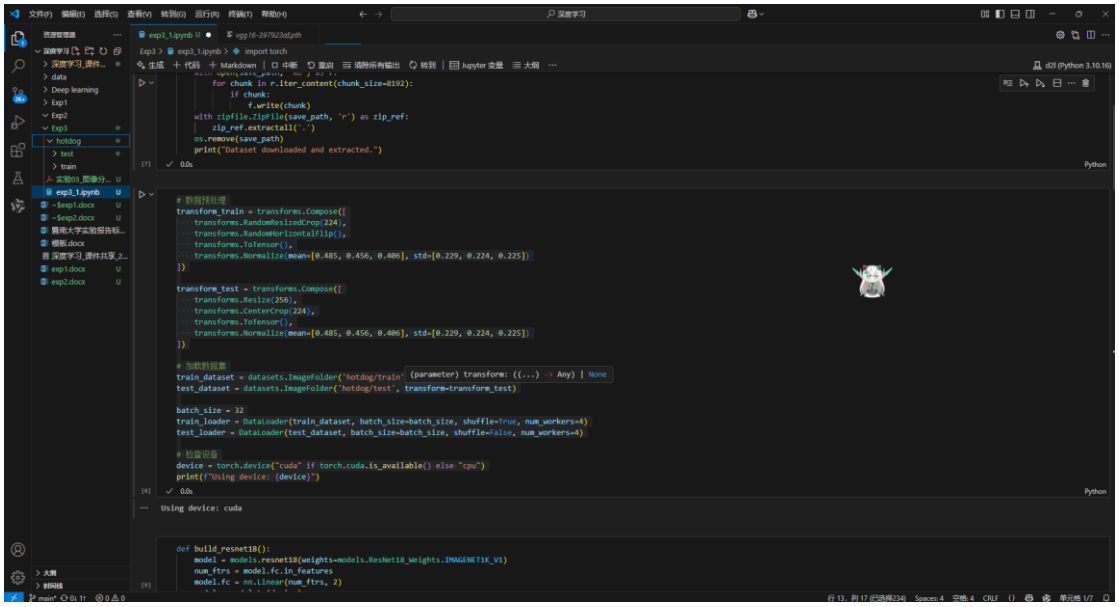
```
print("| 模型 | 训练准确率 | 测试准确率 |")
print("| --- | --- | --- |")
print(f"| ResNet-18 | {train_acc_res[-1]:.4f} | {val_acc_res[-1]:.4f} |")
print(f"| VGG-16 | {train_acc_vgg[-1]:.4f} | {val_acc_vgg[-1]:.4f} |")
```

（五） 实验结果与分析

1. 热狗数据集下载

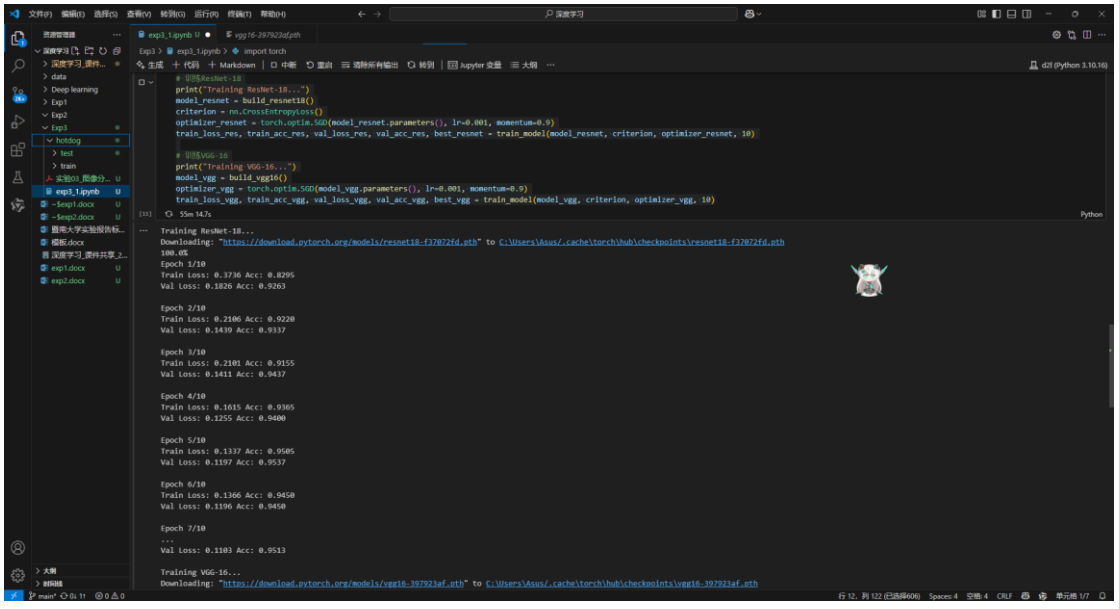


2. 数据预处理和加载数据集

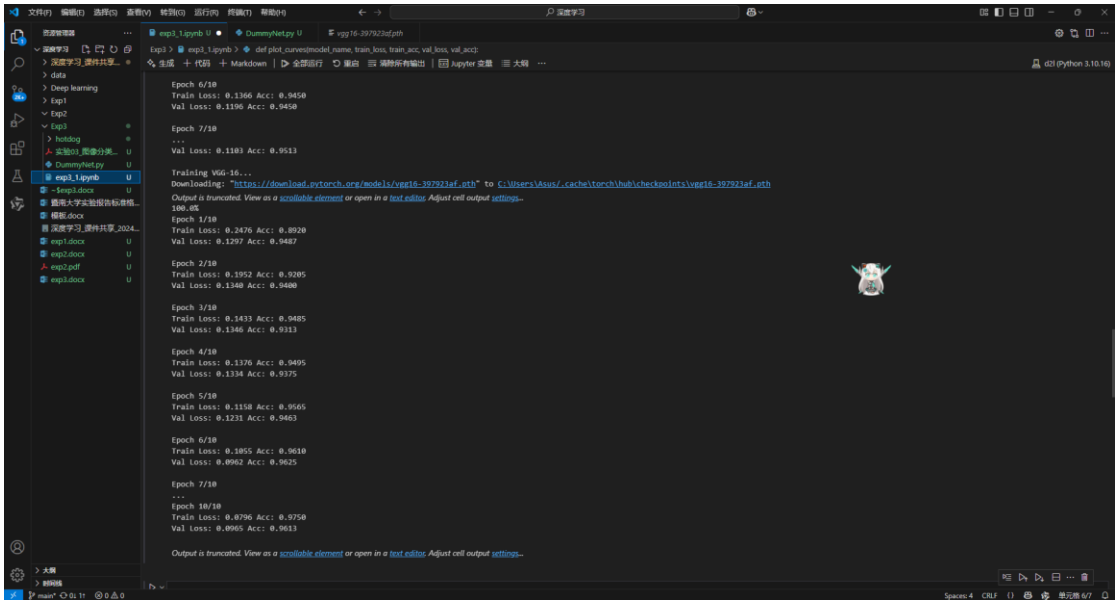


3. 模型训练与验证

(1) ResNet-18

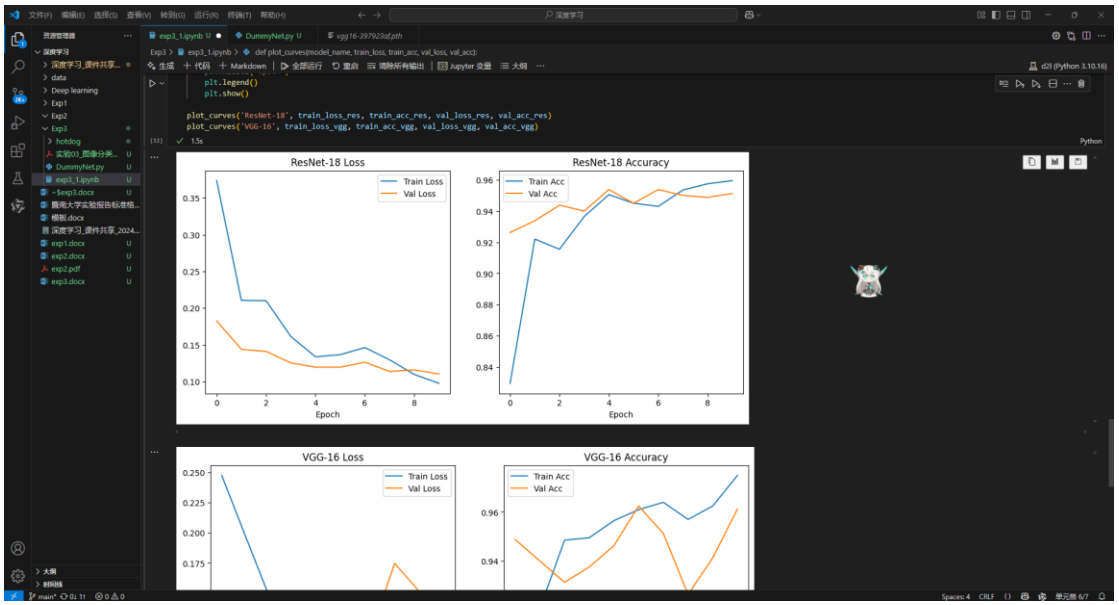


(2) VGG-16

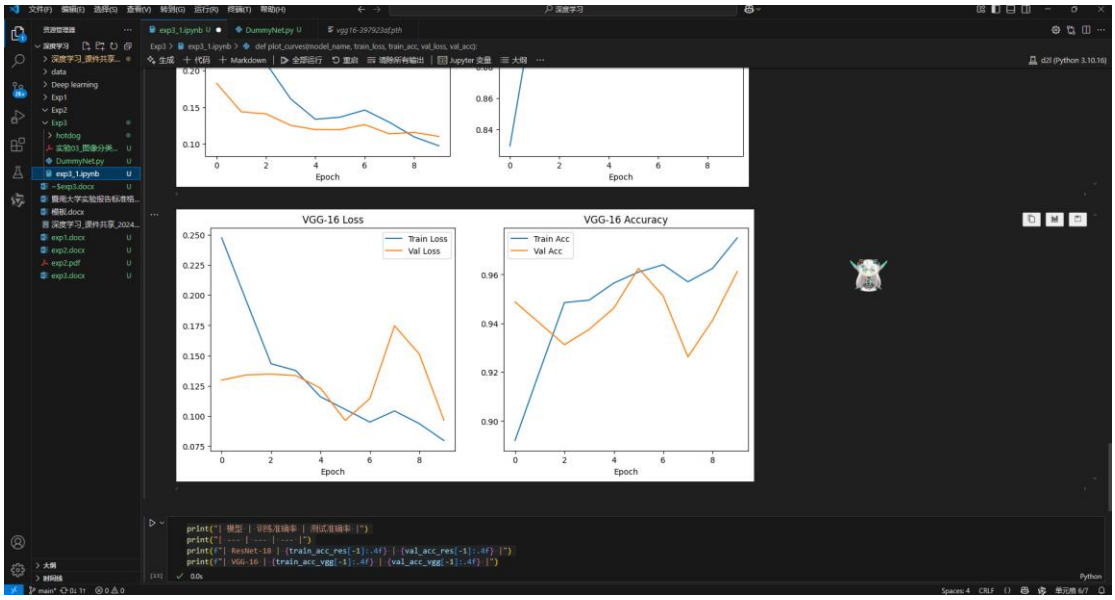


4. 训练曲线

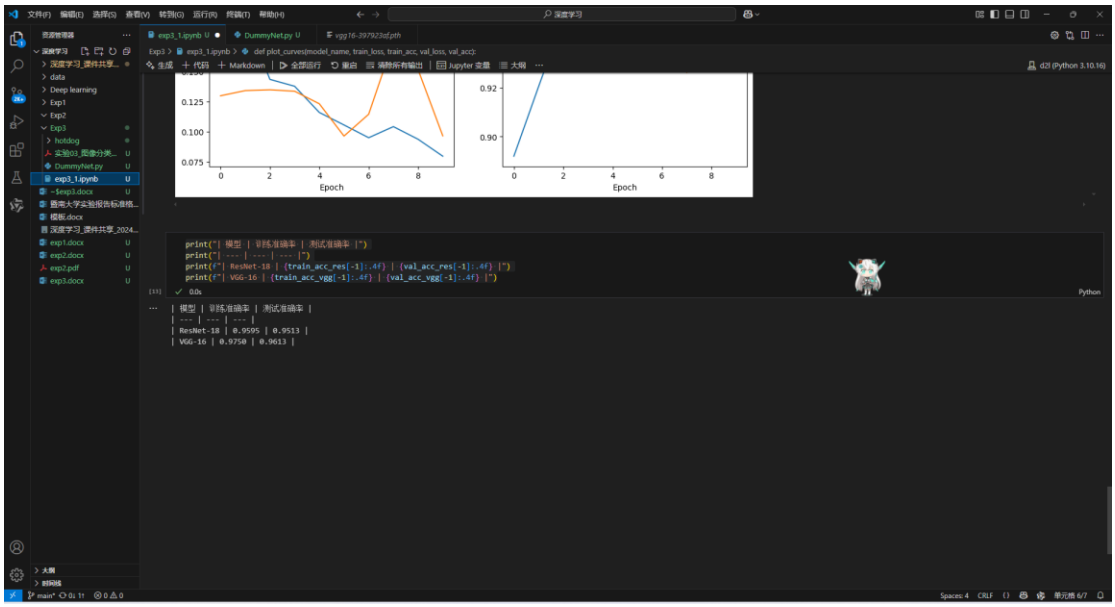
(1) ResNet-18



(2) VGG-16



5. 结果输出



6. 完成表格

模型	训练准确率	测试准确率
ResNet-18	0.9595	0.9513
VGG-16	0.9750	0.9613