**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

1. **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

1. **主要仪器设备**

**仪器：** PC

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

1. **源程序**

源程序在实验步骤与调试中给出。

1. **实验步骤与调试**
2. **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.")

1. **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}")

1. **训练与验证函数**
2. 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

1. 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

1. **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

1. **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)

1. **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)

1. **输出结果表格**

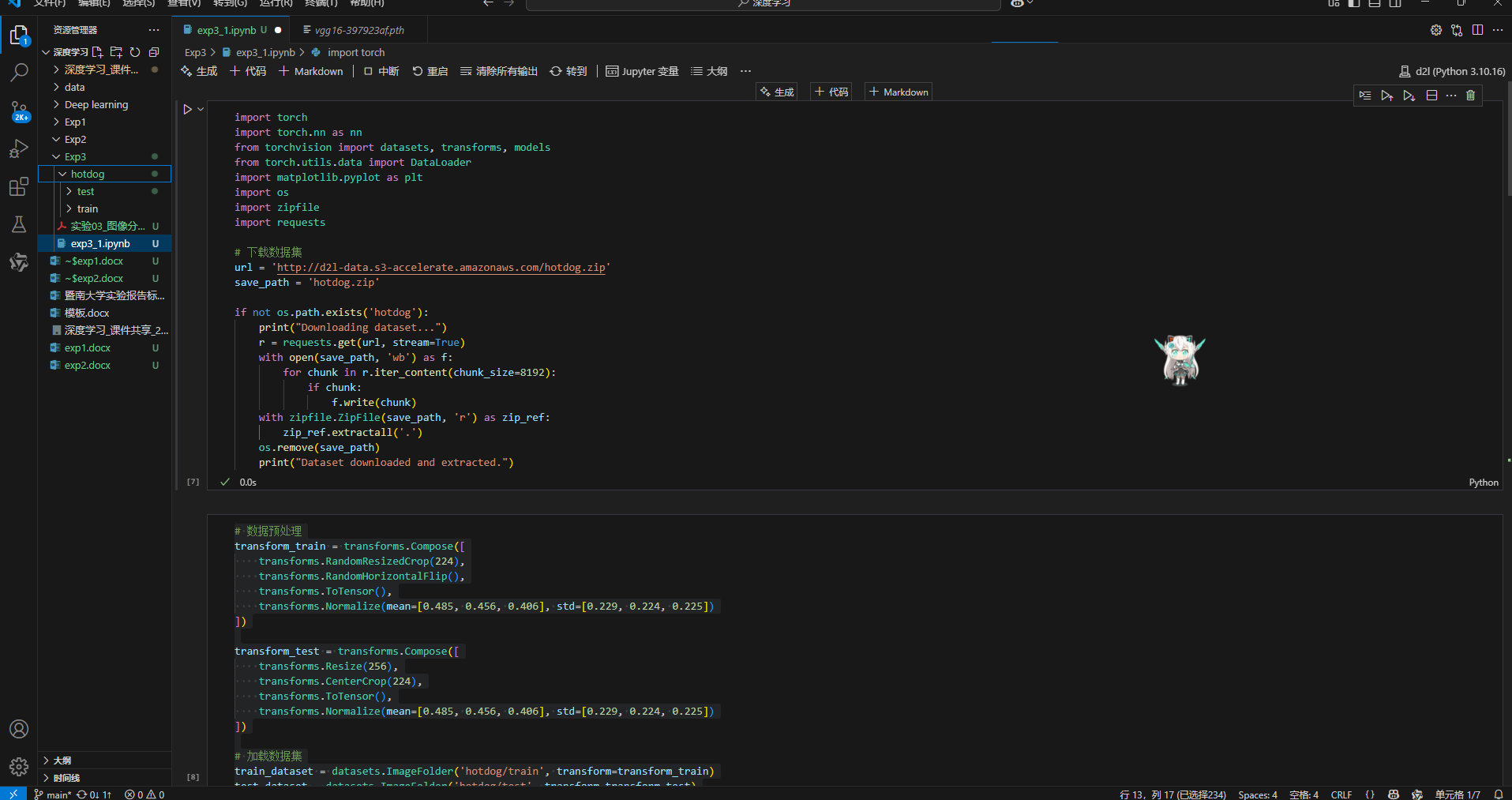
print("| 模型 | 训练准确率 | 测试准确率 |")

print("| --- | --- | --- |")

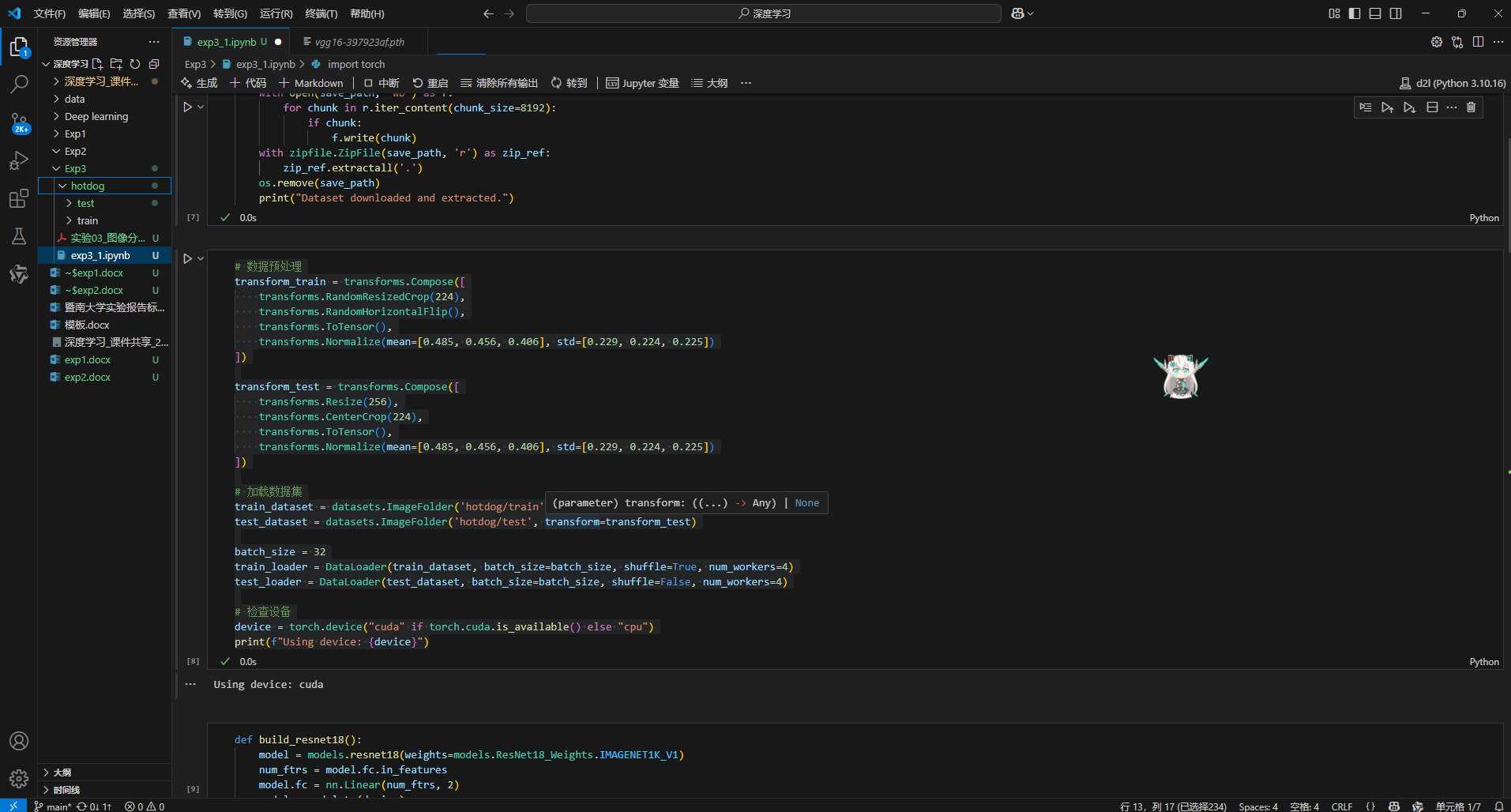
print(f"| ResNet-18 | {train\_acc\_res[-1]:.4f} | {val\_acc\_res[-1]:.4f} |")

print(f"| VGG-16 | {train\_acc\_vgg[-1]:.4f} | {val\_acc\_vgg[-1]:.4f} |")

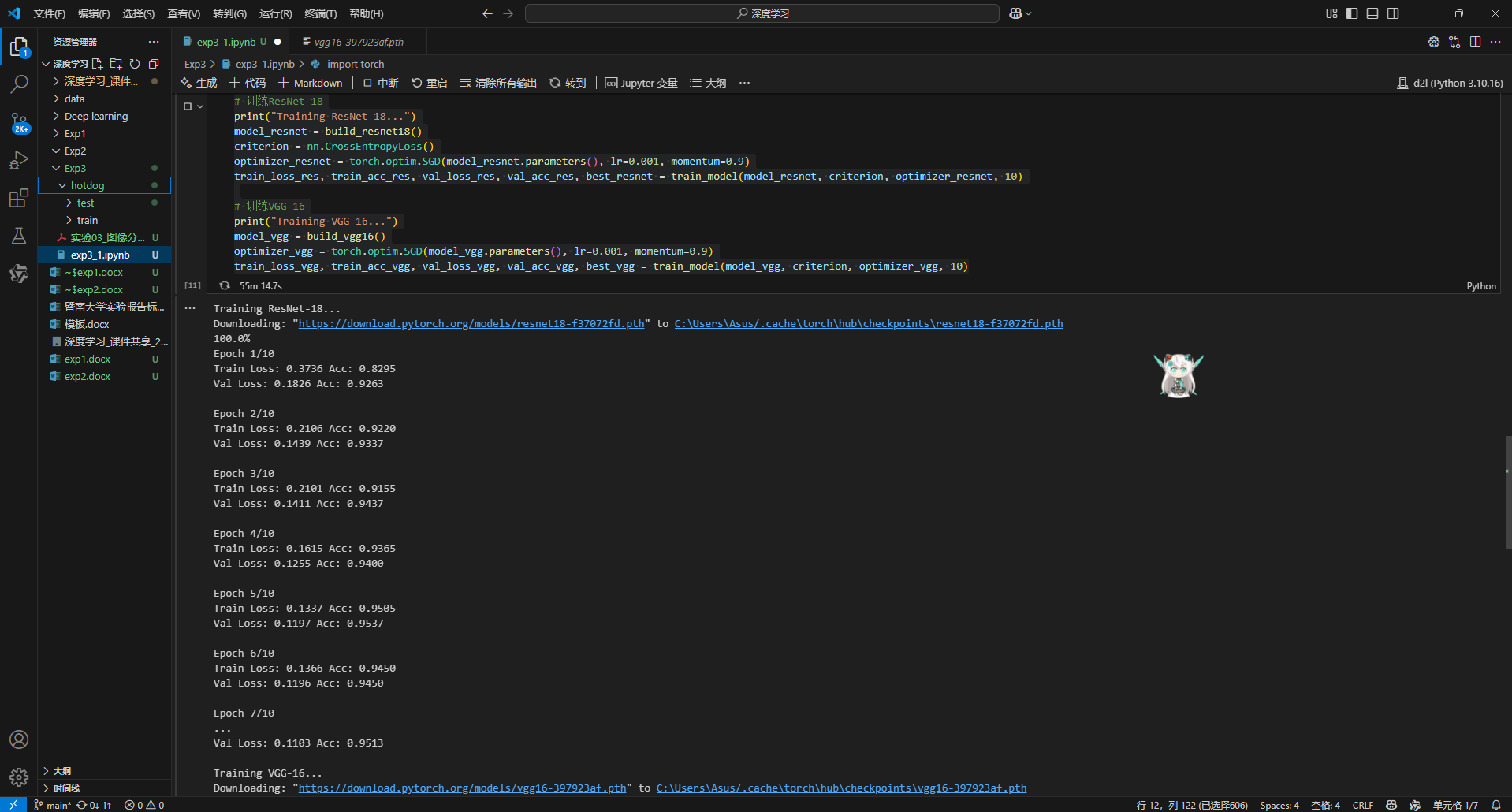
1. **实验结果与分析**
2. **热狗数据集下载**



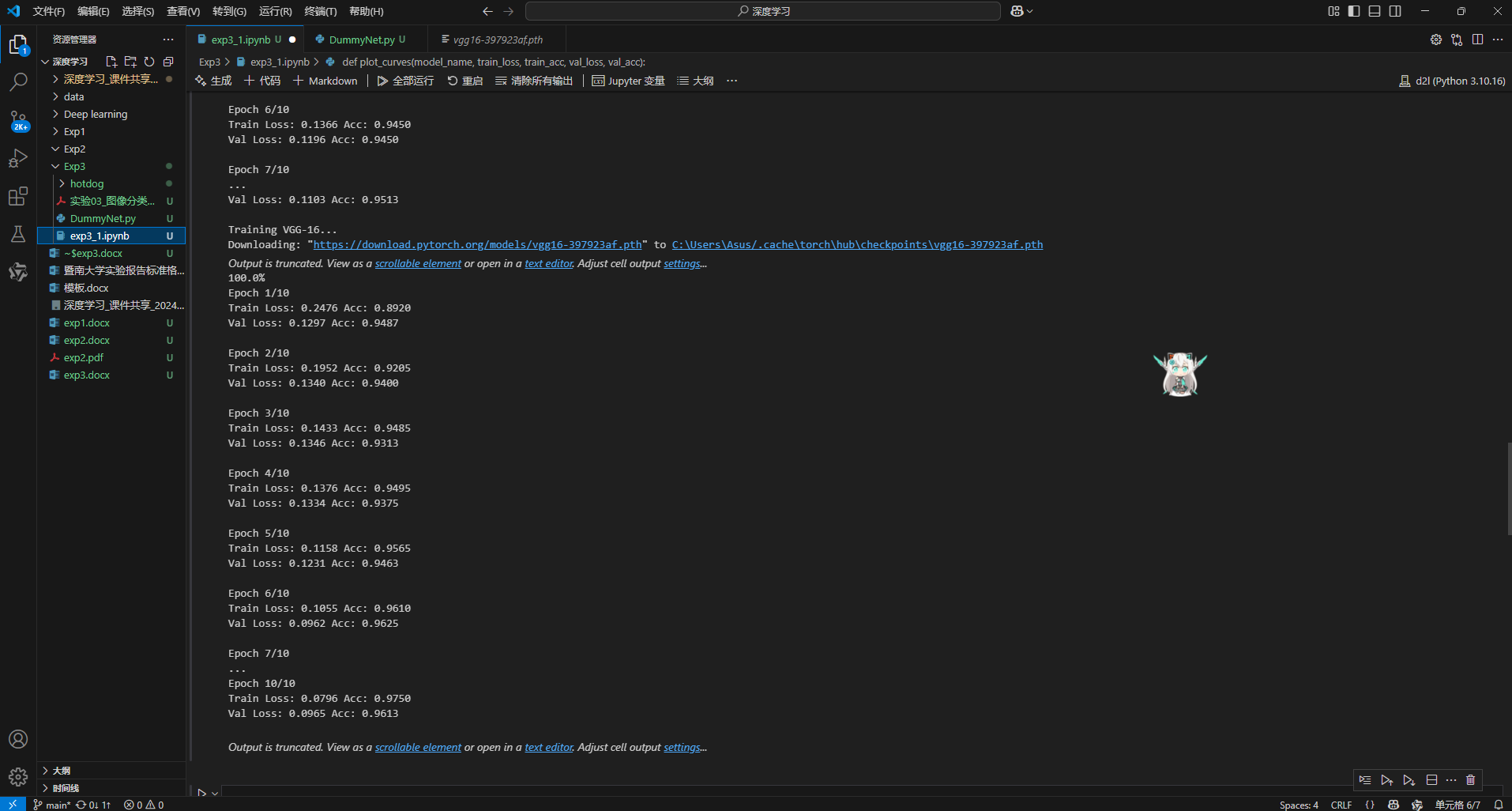
1. **数据预处理和加载数据集**



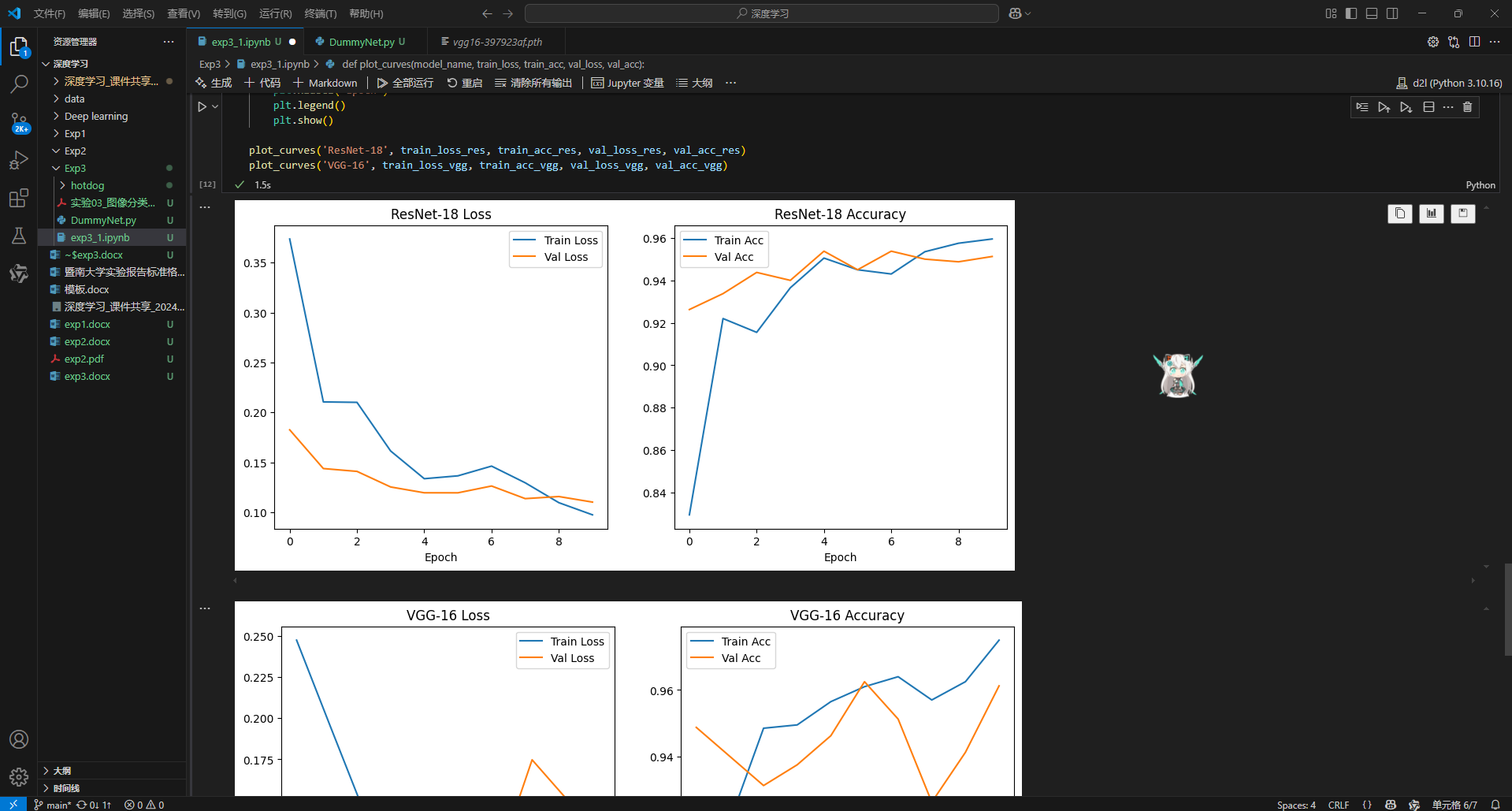
1. **模型训练与验证**
2. ResNet-18



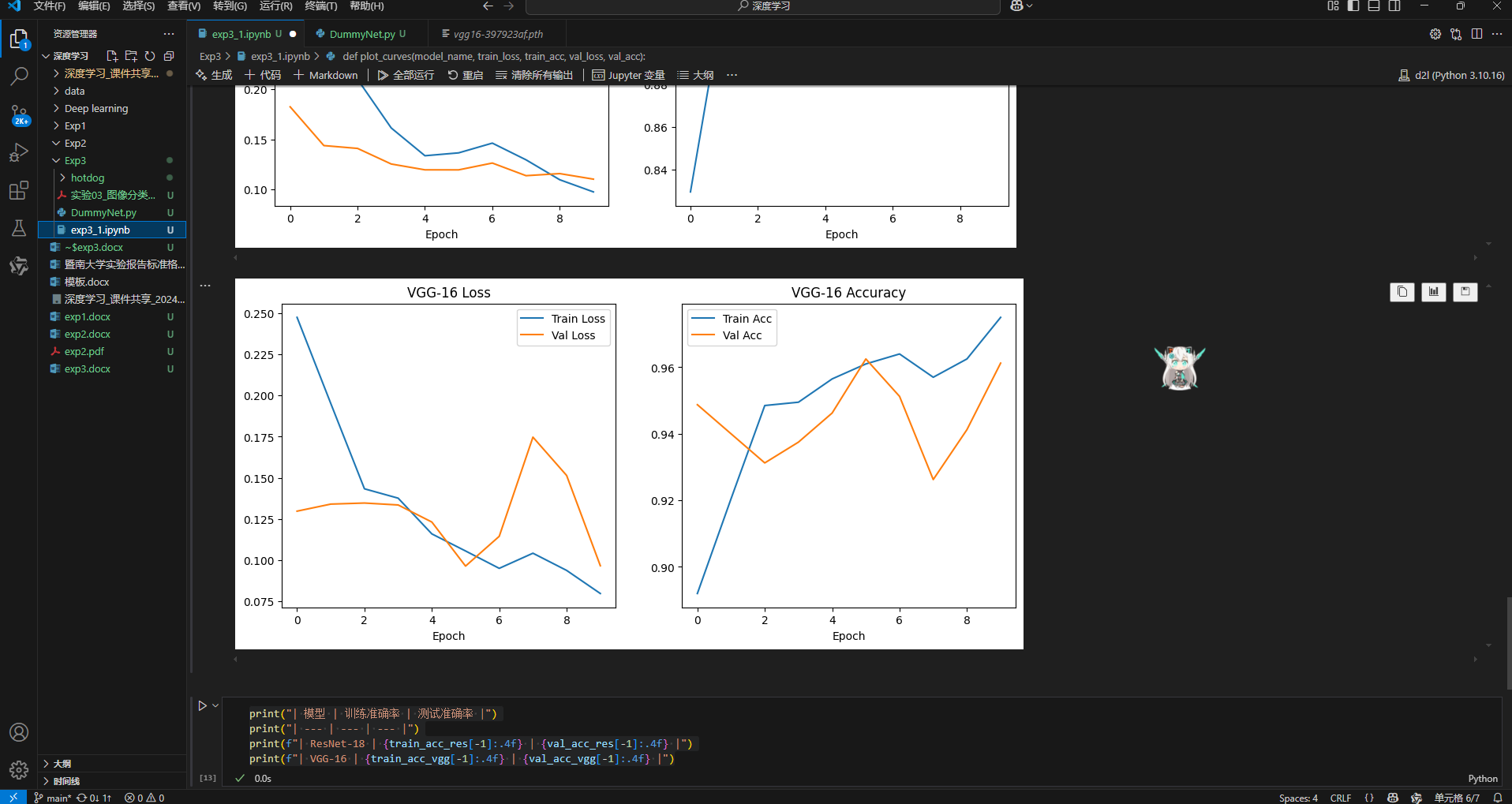
1. VGG-16



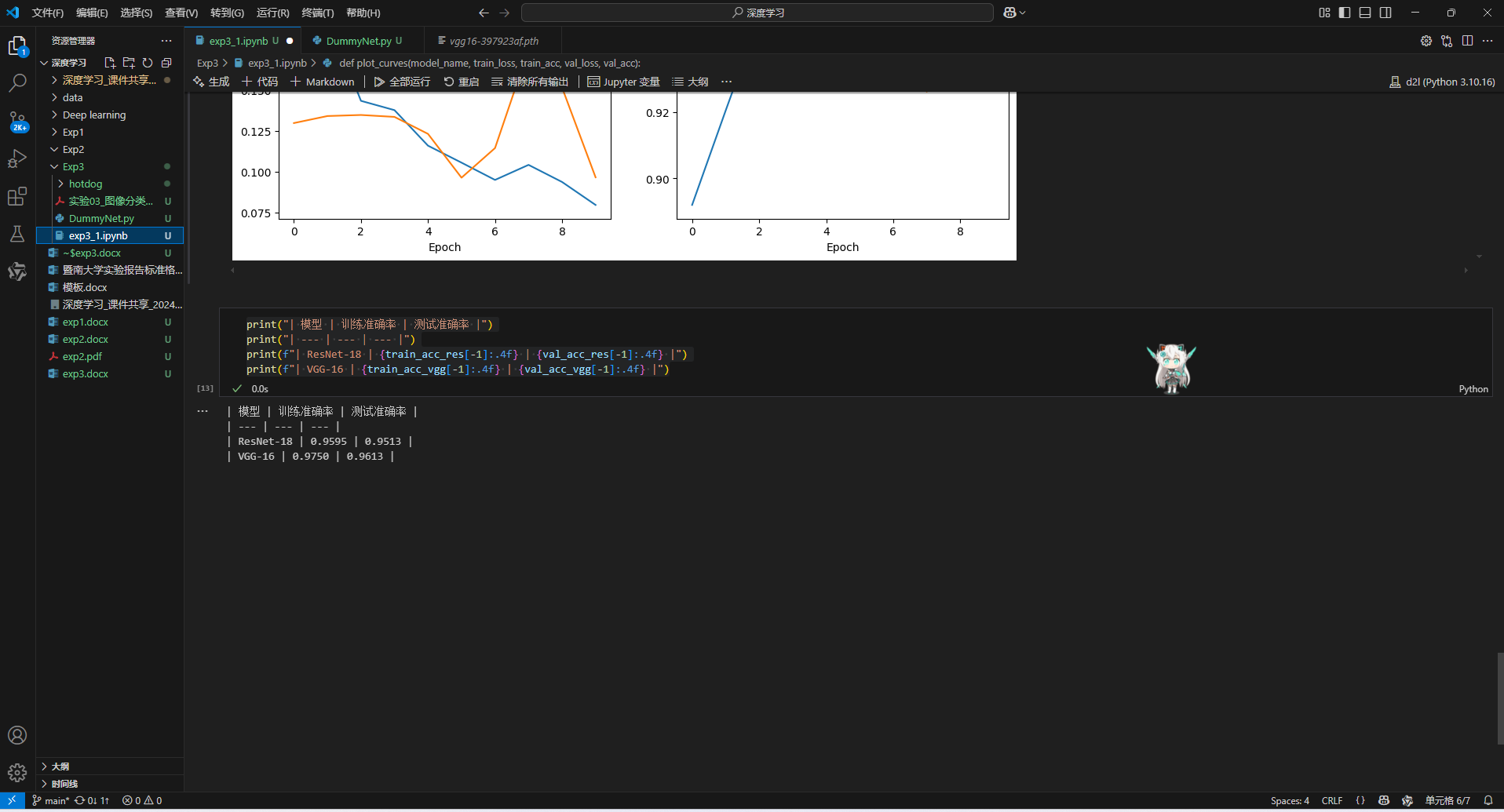
1. **训练曲线**
2. ResNet-18



1. VGG-16



1. **结果输出**



1. **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 |