实验题目：基于 CNN 的深度学习目标检测实验：花朵识别与分类

实验要求：

使用卷积神经网络（CNN）对各类花朵进行检测和分类。

数据集应包含多种花朵类别，每个类别应有足够的样本数量。

评估模型的准确率、召回率和 F1 分数等指标。

使用 Python 编写实现代码，并添加详细注释。

可视化模型训练过程中的损失和准确率，以及最终预测结果。

实验方案：

选择一个适用于花朵识别任务的数据集。

对数据集进行预处理，包括缩放、归一化、数据增强等操作。

构建卷积神经网络（CNN）模型，设置合适的层数和参数。

划分训练集和测试集，训练模型并评估性能。

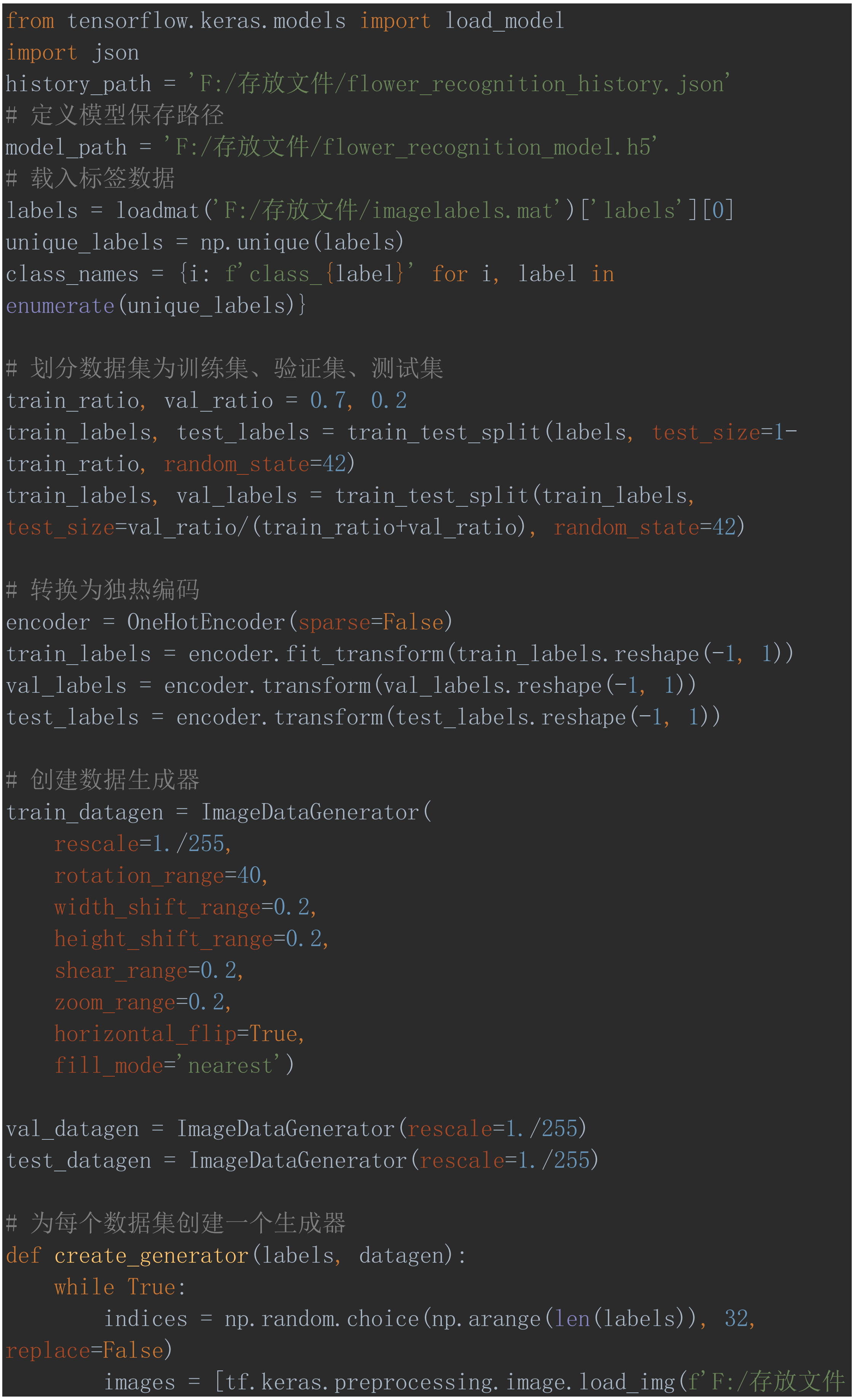
可视化训练过程和预测结果。

数据集：

Oxford 102 花卉数据集，包含 102 种花卉类别，共有 8189 张图像。

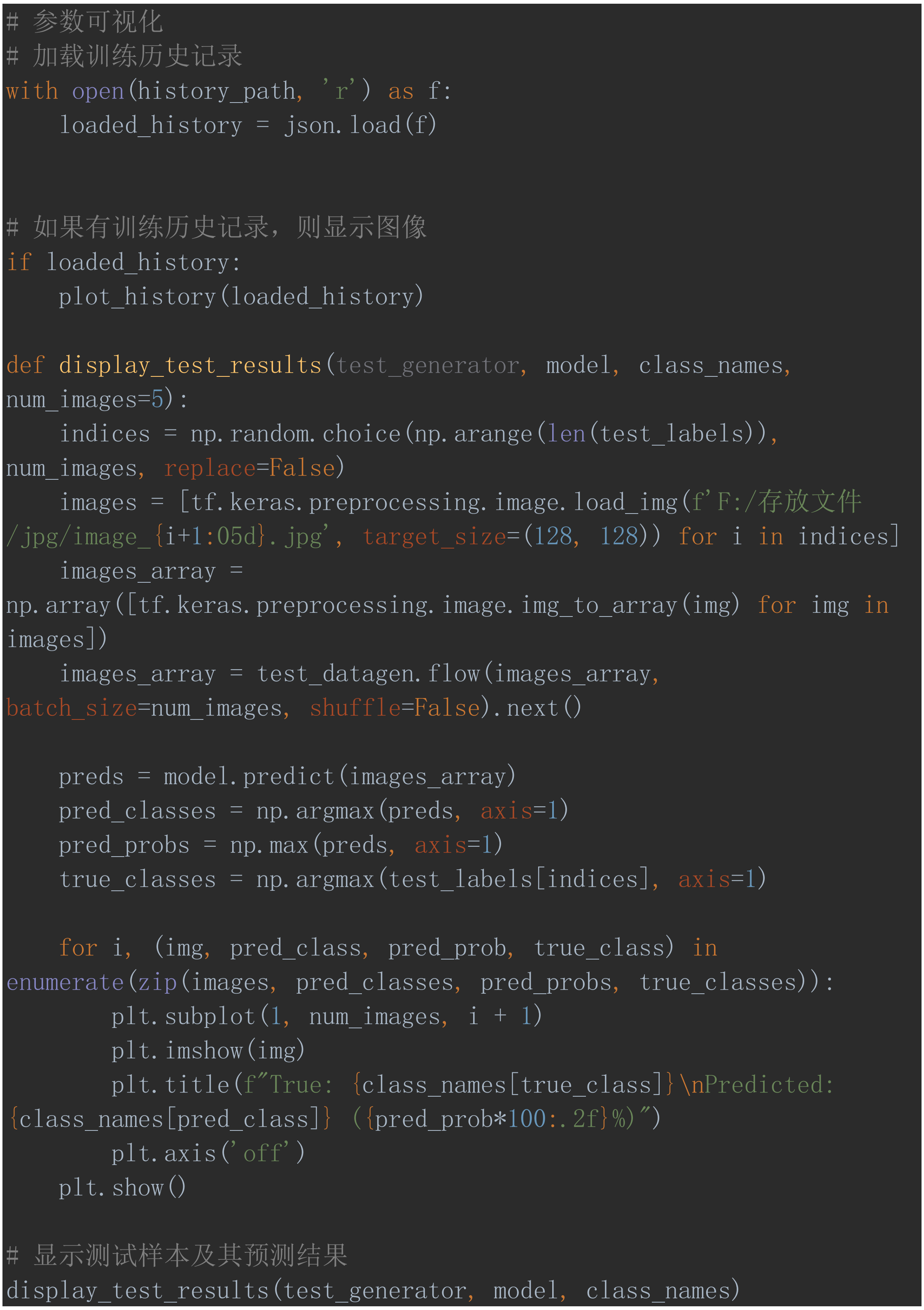
数据集下载地址：http://www.robots.ox.ac.uk/~vgg/data/flowers/102/index.html

```
import os
import random
import shutil
import numpy as np
import tensorflow as tf
from scipy.io import loadmat
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
import matplotlib.pyplot as plt
```



```
from tensorflow.keras.models import load_model
import json
history_path = 'F:/存放文件/flower_recognition_history.json'
# 定义模型保存路径
model_path = 'F:/存放文件/flower_recognition_model.h5'
# 载入标签数据
labels = loadmat('F:/存放文件/imagelabels.mat')['labels'][0]
unique_labels = np.unique(labels)
class_names = {i: f'class_{label}' for i, label in
enumerate(unique_labels)}

# 划分数据集为训练集、验证集、测试集
train_ratio, val_ratio = 0.7, 0.2
train_labels, test_labels = train_test_split(labels, test_size=1-
train_ratio, random_state=42)
train_labels, val_labels = train_test_split(train_labels,
test_size=val_ratio/(train_ratio+val_ratio), random_state=42)

# 转换为独热编码
encoder = OneHotEncoder(sparse=False)
train_labels = encoder.fit_transform(train_labels.reshape(-1, 1))
val_labels = encoder.transform(val_labels.reshape(-1, 1))
test_labels = encoder.transform(test_labels.reshape(-1, 1))

# 创建数据生成器
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest')

val_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)

# 为每个数据集创建一个生成器
def create_generator(labels, datagen):
    while True:
        indices = np.random.choice(np.arange(len(labels)), 32,
replace=False)
        images = [tf.keras.preprocessing.image.load_img(f'F:/存放文件
```

```python
/jpg/image_{i+1:05d}.jpg', target_size=(128, 128)) for i in indices]
        images = np.array([tf.keras.preprocessing.image.img_to_array(img) for img in images])
        yield datagen.flow(images, labels[indices], batch_size=32, shuffle=False).next()

train_generator = create_generator(train_labels, train_datagen)
val_generator = create_generator(val_labels, val_datagen)
test_generator = create_generator(test_labels, test_datagen)

def plot_history(history):
    plt.figure(figsize=(12, 4))

    plt.subplot(1, 2, 1)
    plt.plot(history['accuracy'], label='train accuracy')
    plt.plot(history['val_accuracy'], label='val accuracy')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.legend()

    plt.subplot(1, 2, 2)
    plt.plot(history['loss'], label='train loss')
    plt.plot(history['val_loss'], label='val loss')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.legend()

    plt.show()


if os.path.exists(model_path):
    # 加载已有模型
    model = load_model(model_path)
else:
    # 构建模型
    model = Sequential()
    model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 3)))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(64, (3, 3), activation='relu'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(128, (3, 3), activation='relu'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(128, (3, 3), activation=’relu’))
    model.add(MaxPooling2D((2, 2)))
    model.add(Flatten())
    model.add(Dense(512, activation=’relu’))
    model.add(Dropout(0.5))
    model.add(Dense(102, activation=’softmax’))
    # 编译模型
    model.compile(optimizer=’adam’, loss=’categorical_crossentropy’,
metrics=[’accuracy’])
    # 训练模型
    history = model.fit(train_generator, epochs=10,
steps_per_epoch=100, validation_data=val_generator,
validation_steps=50)
    # 保存模型
    model.save(’F:/存放文件/flower_recognition_model.h5’)
    with open(history_path, ’w’) as f:
        json.dump(history.history, f)
    plot_history(history)

# 测试模型
test_loss, test_acc = model.evaluate(test_generator, steps=100)
print(f”Test accuracy: {test_acc:.4f}, Test loss: {test_loss:.4f}”)

def display_test_results(test_generator, model, class_names,
num_images=5):
    # 随机选择测试样本
    batch = np.random.choice(test_generator.samples, num_images)
    for i, idx in enumerate(batch):
        x, y = test_generator[idx]
        x = np.expand_dims(x, axis=0)
        preds = model.predict(x)
        pred_class = np.argmax(preds, axis=1)
        pred_prob = np.max(preds, axis=1)
        true_class = np.argmax(y)

        # 显示图像和预测结果
        plt.subplot(1, num_images, i + 1)
        plt.imshow(x[0].astype(np.uint8))
        plt.title(f”True: {class_names[true_class]}\nPredicted:
{class_names[pred_class[0]]} ({pred_prob[0]*100:.2f}%)”)
        plt.axis(’off’)
    plt.show()
```



```
# 参数可视化
# 加载训练历史记录
with open(history_path, ’r’) as f:
    loaded_history = json.load(f)


# 如果有训练历史记录，则显示图像
if loaded_history:
    plot_history(loaded_history)

def display_test_results(test_generator, model, class_names, num_images=5):
    indices = np.random.choice(np.arange(len(test_labels)), num_images, replace=False)
    images = [tf.keras.preprocessing.image.load_img(f’F:/存放文件/jpg/image_{i+1:05d}.jpg’, target_size=(128, 128)) for i in indices]
    images_array = np.array([tf.keras.preprocessing.image.img_to_array(img) for img in images])
    images_array = test_datagen.flow(images_array, batch_size=num_images, shuffle=False).next()

    preds = model.predict(images_array)
    pred_classes = np.argmax(preds, axis=1)
    pred_probs = np.max(preds, axis=1)
    true_classes = np.argmax(test_labels[indices], axis=1)

    for i, (img, pred_class, pred_prob, true_class) in enumerate(zip(images, pred_classes, pred_probs, true_classes)):
        plt.subplot(1, num_images, i + 1)
        plt.imshow(img)
        plt.title(f”True: {class_names[true_class]}\nPredicted: {class_names[pred_class]} ({pred_prob*100:.2f}%)”)
        plt.axis(’off’)
    plt.show()

# 显示测试样本及其预测结果
display_test_results(test_generator, model, class_names)
```