**1：**

import os

import numpy as np

import matplotlib.pyplot as plt

import tensorflow as tf

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Conv2D, MaxPooling2D, Dropout, Flatten

*# 设置参数*

train\_directory = 'train\_set/'

test\_directory = 'test\_set/'

validation\_directory = 'val\_set/'

img\_height, img\_width = 224, 224

batch\_size = 128

epochs = 30

learning\_rate = 0.0001

num\_classes = 6

*# 创建数据增强器*

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)

test\_datagen = ImageDataGenerator(rescale=1./255)

*# 加载数据*

train\_data = train\_datagen.flow\_from\_directory(train\_directory,

target\_size=(img\_width, img\_height),

batch\_size=batch\_size,

class\_mode='categorical')

test\_data = test\_datagen.flow\_from\_directory(test\_directory,

target\_size=(img\_width, img\_height),

batch\_size=batch\_size,

class\_mode='categorical')

val\_data = test\_datagen.flow\_from\_directory(validation\_directory,

target\_size=(img\_width, img\_height),

batch\_size=batch\_size,

class\_mode='categorical')

*# 定义模型*

model = Sequential()

model.add(Conv2D(32, (3, 3), activation='relu', input\_shape=(img\_width, img\_height, 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(256, (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(num\_classes, activation='softmax'))

*# 编译模型*

optimizer = tf.keras.optimizers.Adam(lr=learning\_rate)

model.compile(optimizer=optimizer,

loss='categorical\_crossentropy',

metrics=['accuracy'])

*# 训练模型*

history = model.fit(train\_data,

epochs=epochs,

validation\_data=val\_data)

*# 测试模型*

score = model.evaluate(test\_data)

print('Test accuracy:', score[1])

*# 绘制训练和验证损失图*

acc = history.history['accuracy']

val\_acc = history.history['val\_accuracy']

loss = history.history['loss']

val\_loss = history.history['val\_loss']

epochs\_range = range(len(acc))

fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(10, 5))

ax[0].plot(epochs\_range, acc, label='Training Accuracy')

ax[0].plot(epochs\_range, val\_acc, label='Validation Accuracy')

ax[0].set\_title('Training and Validation Accuracy')

ax[0].legend(loc='lower right')

ax[0].set\_xlabel('Epoch')

ax[0].set\_ylabel('Accuracy')

ax[1].plot(epochs\_range, loss, label='Training Loss')

ax[1].plot(epochs\_range, val\_loss, label='Validation Loss')

ax[1].set\_title('Training and Validation Loss')

ax[1].legend(loc='upper right')

ax[1].set\_xlabel('Epoch')

ax[1].set\_ylabel('Loss')

plt.show()

**2：**

import tensorflow as tf

from tensorflow.keras import layers

*# 定义模型*

model = tf.keras.Sequential([

layers.Conv2D(32, (3, 3), activation='relu', input\_shape=(512, 384, 3)),

layers.MaxPooling2D((2, 2)),

layers.Conv2D(64, (3, 3), activation='relu'),

layers.MaxPooling2D((2, 2)),

layers.Conv2D(64, (3, 3), activation='relu'),

layers.Flatten(),

layers.Dense(64, activation='relu'),

layers.Dense(6, activation='softmax')

])

*# 编译模型*

model.compile(optimizer='adam',

loss='sparse\_categorical\_crossentropy',

metrics=['accuracy'])

*# 加载数据集*

train\_ds = tf.keras.preprocessing.image\_dataset\_from\_directory(

'/path/to/dataset/',

validation\_split=0.2,

subset="training",

seed=123,

image\_size=(512, 384),

batch\_size=32)

val\_ds = tf.keras.preprocessing.image\_dataset\_from\_directory(

'/path/to/dataset/',

validation\_split=0.2,

subset="validation",

seed=123,

image\_size=(512, 384),

batch\_size=32)

*# 训练模型*

model.fit(train\_ds, epochs=10, validation\_data=val\_ds)

*# 加载测试数据*

test\_ds = tf.keras.preprocessing.image\_dataset\_from\_directory(

'/path/to/dataset/test',

seed=123,

image\_size=(512, 384),

batch\_size=32)

*# 进行预测*

predictions = model.predict(test\_ds)

*# 打印预测结果*

for i, prediction in enumerate(predictions):

print('Prediction for image {}: {}'.format(i, prediction.argmax()))