Lab 7: 深度学习

第一部分：基于ResNet的图像分类任务--迁移学习

在第一部分中，我们将在tensorflow/keras框架下，利用ResNet进行图像分类。我们将采用keras中预设的ResNet权值和模型框架，并修改最后的全连接层，使其适应我们自己的任务。这种方法叫做“迁移学习”，即利用已有的模型，在其基础上进行局部改造，加速并优化新任务学习过程的方法。

图片数据有五类：螃蟹、海豚、龙虾、海马和海星。数据取材于加州理工学院的教学用公开图片数据集101 Object Categories的一小部分。完整数据集可以在此处获得：https://www.kaggle.com/datasets/bryceyu/101-objectcategories (https://www.kaggle.com/datasets/bryceyu/101-objectcategories)

In [ ]:
```
import tensorflow as tf
from tensorflow.keras.models import Model
from tensorflow.keras.models import Sequential
from tensorflow.keras import layers

from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import load_model

import numpy as np
import matplotlib.pyplot as plt
import pathlib

# 从keras导入ResNet50的模型
from tensorflow.keras.applications.resnet import ResNet50, preprocess_input
MODEL_FILE = 'resnet50.model'
```

第一步：获取数据集

In [ ]:
```
data_dir = pathlib.Path('lab7_img')
batch_size = 32
img_height = 180
img_width = 180
# 按文件夹的字母排序，定义类别
class_names=['crab',
             'dolphin',
             'lobster',
             'seahorse',
             'starfish']
num_classes = len(class_names)
print("类别名称: ",class_names)
print("类别数量: ",num_classes)
```

第二步：数据预处理

1. 拆分训练集和验证集（测试图片在单独的文件夹里）
2. 数据样板增强（旋转、缩放、翻转等）

In [ ]:
```
print("训练集: ")

train_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input,
    dtype = 'float32',
    rotation_range=90, #随机旋转
    zoom_range=[1,2], #随机缩放
    horizontal_flip=True, #水平翻转
    fill_mode='nearest',
    validation_split=0.3 #训练集和验证集的拆分比例
)

train_ds = train_datagen.flow_from_directory(
    data_dir,
    subset="training",
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical',
)

print("验证集: ")

val_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input,
    dtype = 'float32',
    rotation_range=90,
    zoom_range=[1,2],
    horizontal_flip=True,
    fill_mode='nearest',
    validation_split=0.3
)

val_ds = val_datagen.flow_from_directory(
    data_dir,
    subset="validation",
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical'
)
```

In [ ]:
```
data_dir_test = pathlib.Path('lab7_img_test')
class_names=['crab',
             'dolphin',
             'lobster',
             'seahorse',
             'starfish']
num_classes = len(class_names)

print("测试集: ")
test_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input,
    dtype = 'float32',
    fill_mode='nearest'
)
test_ds = test_datagen.flow_from_directory(
    data_dir_test,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical'
)
```

第三步：加载预训练的模型

加载ImageNet比赛的ResNet预训练模型，权重使用GoogLeNet v3的权重（当时ResNet的做法）。但是输出层需要更替，因此设置 `include_top=False` 。

In [ ]:
```
base_model = ResNet50(
    weights='imagenet', #使用imagenet任务的预训练权重
    include_top=False, #取消掉原模型的输出层（因为imagenet比赛的任务类别为1000，我们只有5个类别
    input_shape= (img_height,img_width,3)
)
```

第四步：替换掉输出层

将我们自己的输出层放进模型里

GoogLeNet和ResNet的卷积权值已经获得了很好的效果，我们只希望训练网络的全连接层以适应我们自己的任务，因此我们要设置 `layer.trainable = False` 确保预训练层保持原样。

In [ ]:
```
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Dropout

x = base_model.output                                  # ResNet的输出层
x = GlobalAveragePooling2D()(x)                        # 用平均池化做降采样
x = Dropout(0.5)(x)                                    # 随机失活

# 定义输出层（在全连接层之后，用softmax创建概率分布向量作为得分输出，dense指全连接层）
predictions = Dense(num_classes, activation='softmax')(x)
model = Model(inputs=base_model.input, outputs=predictions)

# 关闭对其他层的训练
for layer in base_model.layers:
    layer.trainable = False #如果希望重新训练整个网络，就把这里改为True
```

第五步：设置网络的训练细节

In [ ]:
```
ACCURACY_THRESHOLD = 0.9 #设置accuracy的阈值，当accuracy达到这个阈值，就采用当前模型（本次训练的数据量较少，难以达到这

model.compile(optimizer='rmsprop', #优化使用RMSProp策略
              loss='categorical_crossentropy', #使用交叉熵损失作为损失函数
              metrics=['accuracy'] #用accuracy指标来评估模型性能
             )

# 利用callback来获取超过阈值的模型
from tensorflow.keras.callbacks import Callback as cb
class myCallback(cb):
    def on_epoch_end(self, epoch, logs={}):
        if(logs.get('val_accuracy') >= ACCURACY_THRESHOLD):
            print("\n Accuracy达到了 %2.2f%%, 停止训练" %(ACCURACY_THRESHOLD*100))
            self.model.stop_training = True

callbacks = myCallback()
```

第六步：训练网络

In [ ]:
```
# 设置训练轮次
EPOCHS = 50
STEPS_PER_EPOCH = 1
VALIDATION_STEPS = 1

# 记录每一个轮次的历史信息
history = model.fit(
    train_ds,
    epochs=EPOCHS,
    steps_per_epoch=STEPS_PER_EPOCH,
    validation_data=val_ds,
    callbacks=[callbacks]
)

model.save(MODEL_FILE)
```

第七步：绘制准确率和损失值随轮次变化的折线图

通过观察损失值的折线图，确保模型没有发生过拟合现象。

In [ ]:
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_actual = len(history.history['loss'])
epochs_range = range(epochs_actual)

plt.figure(figsize=(18, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```

第八步：测试训练好的网络

In [ ]:
```
import numpy as np
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import load_model

model = load_model(MODEL_FILE)
```

In [ ]:
```
# 使用模型进行预测的方法：
# 定义Pred函数，输入图像和模型。
# 该函数对输入的图像进行预处理，使其符合网络预期的输入尺寸。
# 然后运行 model.predict，提供类别预测结果
def pred(model, img):
    img = img.resize((img_width, img_height))
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)
    x = preprocess_input(x)
    results = model.predict(x)
    return results[0]
```

单张图片测试

In [ ]:
```
# 单张图片:
img_name = 'lab7_img_test_individual/crab.jpg'

# 绘制图片
plt.figure()
img = image.load_img(img_name, target_size=(img_height, img_width))
result = pred(model, img)
plt.imshow(img)
print("文件名: ", img_name)

# 显示分类器得到的该图片在每个类别的概率
plt.figure()
labels = class_names
plt.barh(range(num_classes), result, alpha=0.5)
plt.yticks(range(num_classes),labels)
plt.xlabel('Raw Score')
plt.xlim(0,1)
plt.tight_layout()
plt.show()

score = tf.nn.softmax(result)
print(
    "这张图片的类别是: {} , 可能性为 {:.2f}%"
    .format(class_names[np.argmax(score)], 100 * np.max(score))
)
```

任务1

对lab7_img_test_individual文件夹中的其他所有图片做分类测试并展示结果

In [ ]:
```
# 在这里完成任务1，可以使用额外的cells
```

任务2：测试集整体测试

利用pred( )函数对测试集（test_ds）中的每一张图片进行分类预测，并记录预测结果。

绘制混淆矩阵，对模型在测试集上的性能进行可视化。

提示：混淆矩阵可以用sklearn中的confusion_matrix函数，以及seaborn中的heatmap函数。

In [ ]:
```
# 在这里完成任务2，可以使用额外的cells
```

任务3：使用ResNet152

从keras导入ResNet152模型，然后自由调整输入图像尺寸、batch size、优化策略等进行训练。

训练完成后，在测试集上进行测试并绘制混淆矩阵。

In [ ]:
```
# 从keras导入ResNet152的模型
from tensorflow.keras.applications.resnet import ResNet152, preprocess_input
MODEL_FILE = 'resnet152.model'
```

In [ ]:
```
# 在此处完成任务3，可以使用额外的cells
```

第二部分：经典任务--情绪识别

情绪识别是指利用深度学习技术来分析和识别人类的情绪状态。这种技术通过训练深度学习模型，使其能够自动从原始数据中学习和提取与情绪相关的特征，进而对人类的情绪进行准确分类和识别。它是当前计算机视觉技术的重要应用之一，在城市感知、人机交互、新闻传媒等领域具有较高的潜在应用价值。

请安装deepface并利用它对人像图片进行情绪识别。该工具通过预训练的模型，对人脸图像展现出的情绪进行分类，并根据7个情绪类别进行打分。这7个类别包括：angry（生气）、disgust（厌恶）、fear（恐惧）、happy（开心）、sad（悲伤）、surprise（吃惊）和 neutral（无情绪）。

In [ ]:
```
import cv2
from deepface import DeepFace
```

In [ ]:
```
# 导入人脸图像
img = cv2.imread('lab7_img_deepface/face1.jpg')
plt.imshow(img[:,:,::-1])
plt.show()

# 利用DeepFace识别情绪
result = DeepFace.analyze(img,actions=['emotion'])
```

In [ ]:
```
#查看各情绪的得分：
print(result[0]['emotion']) #“0”指的是序列中的第一个人，因为DeepFace可以识别拥有多张人脸的图片
```

In [ ]:
```
#查看最高得分（即情绪类别）：
print(result[0]['dominant_emotion'])
```

任务4

现有以下场景：

某团队新开发了另外一款情绪识别产品，分别识别出angry（生气）、disgust（厌恶）、fear（恐惧）、happy（开心）、sad（悲伤）、surprise（吃惊）和 neutral（无情绪）的图像各10张，储存在lab7_img_deepface/model_test路径下。现在，请你用DeepFace对这些图像再次进行分类，并回答下面的问题：

1. 该模型识别出的情绪，有多大比例与DeepFace的识别结果相同?
2. 这两个模型在哪些情绪类别拥有较高的一致性?
3. 这两个模型在哪些情绪类别的一致性较低?

In [ ]:
```
# 在这里完成任务4，可以用额外的cells
```

提交方式

本次作业有4个任务。完成所有cell的运行后，保存为ipynb和PDF格式（保留所有输出）。将导出的ipynb命名为“Lab7+姓名+学号.ipynb”，将导出的PDF命名为“Lab7+姓名+学号.pdf”，并将上述两个文件提交到学习通作业模块的相应位置（如果ipynb无法单独上传，请打包成zip格式）。请独立完成练习，截止时间：2024年6月21日23:59。超时1天之内将扣除5%的分数，超时1天以上将扣除10%的分数。