关于2022年广东省人工智能开发员职业技能竞赛赛前培训和变更初赛形式的通知

各参赛选手:

根据《关于举办2022年广东省人工智能开发员职业技能竞赛的通知》（粤软协字〔2022〕22号）要求，2022年广东省人工智能开发员职业技能竞赛赛前培训（技术说明会）将于2022年11月8日以线上直播的方式举办。因受疫情影响，原定于11月12日上午在广东邮电职业技术学院举办的2022年广东省人工智能开发员职业技能竞赛初赛（理论考试）将变更为线上考试。现将赛前培训和初赛有关安排通知如下:

一、赛前培训（技术说明会）

1.培训时间：2022年11月8日（周二）晚上19:30-21:30

2.培训形式：在线直播，未参与直播的人员，可关注广东软件行业协会微信视频号回看视频。

3.直播平台：广东软件行业协会微信视频号同步直播

腾讯会议（会议号：916-215-468）

4.培训内容: 介绍2022年广东省人工智能开发员职业技能竞赛初赛（理论考试）及决赛（实际操作）考核的基本知识与能力要求；介绍试题范围、比赛样题（附件3）与评判标准；介绍华为云开发者认证线上训练营等。

二、初赛（理论考试）

（一）时间和考题

1.初赛时间：2022 年 11 月 12 日（周六）上午 10:00-11:30，考试时长为 90 分钟。

2.考题均为客观题（单选题、多选题和判断题），满分 100 分。

（二）初赛要求

1.参加初赛（理论考试）人员需提前做好有关准备，确保考试环境、考试设备和监考设备满足要求。

2.参加竞赛人员须严格遵守线上考试纪律和相关要求，发现违反规则的行为，将取消竞赛成绩。发现作弊行为，将取消竞赛资格。

3.初赛当天，参加竞赛人员需通过考试系统登录考试页面，阅读《线上考试须知》后进行答题。同时，监考手机需用分配的“选手编号+姓名”（例如“160-张三”）在上午 9:20 前进入指定的腾讯会议（软件需最新版），准备好身份证进行身份验证。迟到 20 分钟以上（即上午 9:40 后）不允许参加本次考试。

4.《线上考试须知》等线上考试规则和具体要求见附件 1。线上考试流程见附件 2，考试系统网址、选手编号和考场分配等信息将另行通知。

（三）赛前测试（模拟考试）

1.2022 年 11 月 11 日上午将安排一次赛前测试（模拟考试），参加考试人员本人须参加以熟悉考试全流程，否则责任自负。

2.赛前测试与正式竞赛当天的流程安排基本一致。参加人

员须通过身份验证后进入竞赛作答环节，否则视为自动放弃赛前测试。

3.赛前测试中，参加考试人员应进行作答，并点击保存和交卷，以测试答题功能是否正常。

四、联系方式

联系人：张佩珊

电　话：020-38263106、18988936575

邮　箱：hyb@gdsia.org.cn

地　址：广州市天河区员村一横路七号广东软件大厦 5 楼

附件：

1.线上考试须知

2. 2022 年广东省人工智能开发员职业技能竞赛初赛线上考试流程安排

3.比赛样题

2022年广东省人工智能开发员职业技能竞赛组委会

（广东软件行业协会代章）

2022 年 11 月 8 日

附件 1：

线上考试须知

一、考试环境要求：

考试全过程严格禁止无关人员出入考试场所。

二、考试设备要求：

1.电脑设备：电源稳定、网络通畅，首选笔记本电脑。

2.选手进入考试系统前应关闭电脑上与考试无关网页和软件，包括安全卫士、电脑管家及各类通讯软件，以免由于被动弹窗导致被系统判定为作弊。

三、监考设备要求：

1.手机设备：考生需准备一台手机为监考设备，手机摄像头确保覆盖考生本人、考试屏幕、书桌的桌面和大部分考试环境，以使得其满足监控视角的要求（见图1）。

监考设备摆放位置说明　　监考设备摆放视角　　腾讯会议监考手机画面

图1-监控视角说明

2.用于监考的手机应提前安装好腾讯会议（最新版），选

手须在比赛前将手机调成飞行模式或设置来电转移后再连接无线网络进行实时监控，以保障考试顺利完成。手机屏幕保持常亮。

四、考试纪律要求：

1.确保视频设备（监考设备）按规定的角度和位置摆放，考试全程不偏离视频采集范围。

2.考试过程中，不佩戴任何类型的耳机，不戴口罩、墨镜、帽子等遮挡面部。

3.不翻阅书籍、资料，不使用实物计算器等电子设备。

4.在考试时不做与考试无关的事情，不找人协助作答，不使用与考试无关的设备协助作答，不对试题和答案进行截屏，不传递答案等。

5.在考试过程中因个人设备或网络等客观原因影响考试结果，将不予补时；按统一考试结束时间结束考试。

6.在作答前请仔细阅读考试相关信息，了解考试流程；对于未能正确进行考试流程或操作，自行承担后果。

附件 2：

2022年广东省人工智能开发员职业技能竞赛初赛线上考试流程安排

序号	时间	事项
1	8:30-9:00	1. 参加考试人员调试好答题设备和监考设备，确认网络、应用程序和电源方面没有故障； 2. 答题主机登录考试平台，确认完成考前须知阅读，等待进入考试页面。 3. 参加考试人员通过监考手机，用“选手编号+姓名”登录指定的腾讯会议，进入会议等候室。
2	9:00-9:50	1. 监考人员开启会议录制，并审批等候室人员进入腾讯会议； 2. 进入腾讯会议人员调试好监考手机并摆放到指定位置，**保证摄像、声音功能正常**； 3. 监考人员对进入腾讯会议的人员进行身份验证。
3	9:40	腾讯会议等候区停止审批进入会议的申请。
4	9:50	1. 监考人员停止进行身份验证，未通过验证人员取消本次考试资格； 2. 参加考试人员确认已完成考前须知阅读，准备开始考试。
5	10:00	考试系统开通答题通道，参加考试人员点击“答卷”按钮，开始答题。
6	10:00-11:30	1. 参加考试人员进行线上答题（注意及时点击保存，确保答题内容完成上传）； 2. 考试开始 30 分钟后允许点击提交试卷，结束考试。建议提交试卷前，先点击预览确认答题内容已经完成上传。
7	11:30	1. 考试系统关闭答题通道； 2. 监考人员停止会议录制； 3. 参加考试人员退出腾讯会议。

附件 3

比赛样题

一、初赛（理论考试）样题

理论考试试题分三个类型：单选题、多选题和判断题。其中单选题 25 道、多选题 10 道、判断题 20 道。单选题每题 2 分，多选题每题 3 分，判断题每题 1 分。考点占比：机器学习占 20%，深度学习占 25%，计算机视觉占 30%，自然语言处理占 15%，语音处理占 10%。各类题型的样题见表 1。

表 1-理论考试样题

序号	题干	题型	考点	正确答案	选项 A	选项 B	选项 C	选项 D
1	InceptionNet 通过加入残差块来增加模型深度	判断	计算机视觉	B	对	错		
2	TensorFlow1.0 版本的代码可以用于 TensorFlow2.0 版本	判断	深度学习	B	对	错		
3	卷积神经网络可以对一个输入完成不同种类的变换（旋转或缩放）	判断	计算机视觉	B	对	错		
4	HMM 可以做的所有事情，CRF 都能做	判断	机器学习	A	对	错		
5	以下不属于深度学习神经网络的是	单选	深度学习	C	CNN	RNN	Logistic	LSTM
6	以下不属于灰度变换映射函数的是	单选	计算机视觉	D	反转	对比度增强	伽马矫正	高斯滤波
7	以下哪个选项不属于自然语言处理技术的三个层面	单选	自然语言处理	C	词法分析	句法分析	语音分析	语义分析
8	以下哪些属于常见的深度学习开发框架	多选	深度学习	ABCD	MXNet	CNTK	TensorFlow	Pytorch
9	以下选项中，哪些是自然语言处理的应用	多选	自然语言处理	ABC	情感分析	智能问答	文本生成	风格迁移
10	基于深度学习的语音识别算法有哪些优点	多选	语音处理	BD	不需要训练数据	端到端处理任务	需要强制对齐标注数据	自动特征提取

二、决赛（实际操作）样题

样题：计算机视觉技术与应用

实验简介

实验介绍

随着新冠疫情的爆发，口罩已经改变了人们的生活习惯，为了防止新冠疫情的传播，口罩已经成为出门必需品，进入公共场合、坐公交地铁等都要求佩戴口罩。那么检测人们是否佩戴口罩就至关重要。

实验目的

本实验使用了几百条戴口罩和没戴口罩的数据集，先进行图像预处理，再构建卷积神经网络训练模型，最后导入模型进行预测。

实验环境

conda-py3.7.6-tf2.0

任务一：导入数据集

在当前目录创建一个notebook，kernel选择conda-py3.7.6-tf2.0即可，

输入以下代码下载数据集，进入数据集所在目录：

```
cd /data/cvlky/data
```

将数据集拷贝至当前文件夹：

```
cp CNN_mask_detection.zip  /ddhome/data
```

回到当前目录：

```
cd /ddhome/data
```

输入以下代码解压数据集：

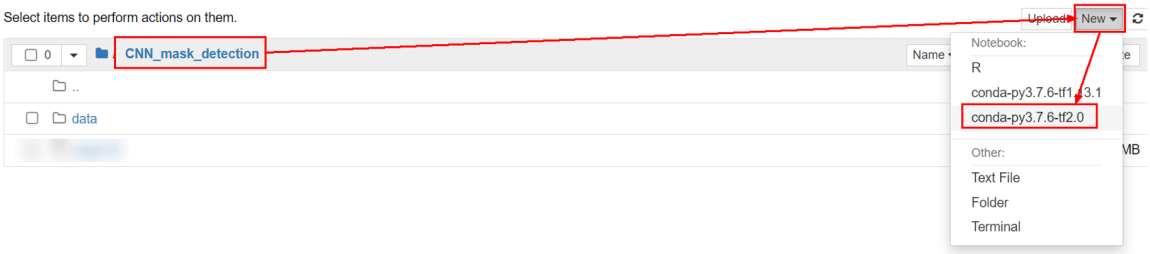
```
!unzip CNN_mask_detection.zip
```

如果没有unzip命令，可以输入如下代码安装unzip：

```
!yum install unzip -y
```

关闭此notebook，可以观察到解压后的data文件夹中有with_mask和without_mask两种类型数据。

进入CNN_mask_detection文件夹，**重新新建notebook**，即可开始实验。



任务二：图像的获取和预处理

步骤1：导入相关包

```
import os
import glob
import cv2
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Activation, Dropout,
Dense, Flatten
```

步骤2：获取数据

补充代码后，请将结果截图1保存到本地，用于实验报告上传。

```
# data是数据集的目录
data = r'./data'
# 下面2个路径分别是2种数据的目录
with_mask = os.path.join(data, 'with_mask')
without_mask = os.path.join(data, 'without_mask')
# 遍历文件夹，将每一个图片的地址依次存入列表中
X_with_mask = [os.path.abspath(fp) for fp in glob.glob(os.path.join(with_mask,
'*.jpg'))]
X_without_mask = [os.path.abspath(fp) for fp in
glob.glob(os.path.join(without_mask, '*.jpg'))]
X = X_with_mask + X_without_mask
# 生成标签，1代表佩戴了口罩，0代表未佩戴口罩
Y =  [1] * len(X_with_mask) + /*此处由考生进行代码填写*/
# 十字交叉分割，把所有的图片路径和标签通过shuffle=True打乱顺序，然后分0.9比例的给训练集图片和
训练集标签，剩下的作为测试集
x_train, x_test, y_train, y_test = /*此处由考生进行代码填写*/
```

步骤3：图像预处理

定义图像预处理函数，把所有图像的尺寸缩放为（64.64,3），再将图像进行归一化：

```
def image_preprocessing(path, label):
```

```
    data_set = []
    label_set = []
    # 遍历路径列表中的路径以及标签列表里的标签
    for (one_path, one_label) in zip(path,label):
        # cv2读取图片为numpy矩阵
        image = cv2.imread(one_path)
        # 缩放图片为指定大小
        image_resize = cv2.resize(image, (64,64), cv2.INTER_AREA)

        # cv2读取的矩阵值在0-255，我们选择除以255将图片归一化
        image = image_resize/255

        data_set.append(image)
        label_set.append(one_label)
    # 转为矩阵形式返回
    return np.array(data_set), np.array(label_set)
```

调用上述函数，将训练集和测试集中的图像进行预处理，再输出处理好的图像尺寸和数量，**补充代码后，请将结果截图2保存到本地，用于实验报告上传。**

输入：

```
x_train,y_train = image_preprocessing(x_train,y_train)
x_test, y_test = image_preprocessing(x_test, y_test)
#查看训练集和测试集的图像和标签尺寸
print( /*此处由考生进行代码填写*/)
```

输出：

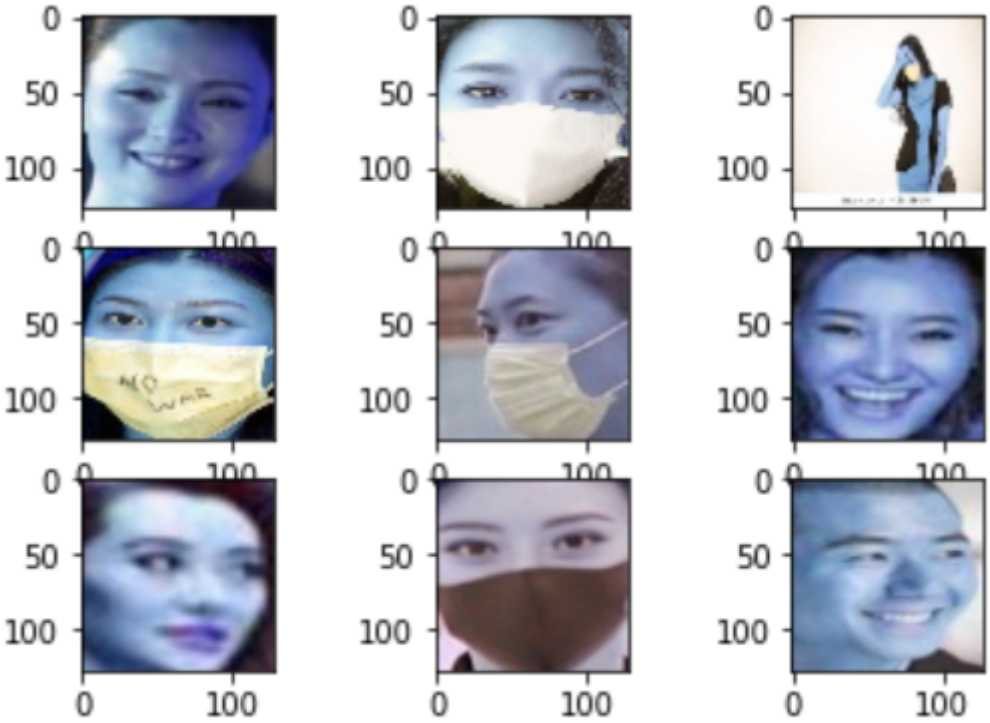
```
(418, 64, 64, 3) (418,) (47, 64, 64, 3) (47,)
```

接下来展示一些样本，**补充代码后，请将结果截图3保存到本地，用于实验报告上传。**

输入：

```
#展示前9张图片
plt.figure()
for i in /*此处由考生进行代码填写*/:
    plt.subplot(3,3,i+1)
    /*此处由考生进行代码填写*/
plt.show()
```

输出：



任务三：构建模型并训练

步骤1：构建CNN网络模型

CNN网络模型可以自定义修改，下面是构造了四组两层卷积层+一层最大池化层，再加三层全连接层的模型，隐藏层全部使用了relu激活函数，在输出层使用softmax激活函数。**补充代码后，请将结果截图4保存到本地，用于实验报告上传。**

```
def CNN_classification_model():
    model = Sequential()
    #添加两层卷积层，卷积核尺寸为64*3*3，padding为same，激活函数选择relu
    #再添加一层池化层，窗口大小为2*2，步长为2，padding为same
    model.add(/*此处由考生进行代码填写*/,input_shape=(64,64,3)))
    model.add(Activation(/*此处由考生进行代码填写*/))
    model.add(/*此处由考生进行代码填写*/)
    model.add(Activation(/*此处由考生进行代码填写*/))
    model.add(/*此处由考生进行代码填写*/)

    #添加两层卷积层，卷积核尺寸为128*3*3，padding为same，激活函数选择relu
    #再添加一层池化层，窗口大小为2*2，步长为2，padding为same
    model.add(/*此处由考生进行代码填写*/)
    model.add(Activation(/*此处由考生进行代码填写*/))
    model.add(/*此处由考生进行代码填写*/)
    model.add(Activation(/*此处由考生进行代码填写*/))
    model.add(/*此处由考生进行代码填写*/)

    #添加两层卷积层，卷积核尺寸为256*3*3，padding为same，激活函数选择relu
    #再添加一层池化层，窗口大小为2*2，步长为2，padding为same
    model.add(/*此处由考生进行代码填写*/)
    model.add(Activation(/*此处由考生进行代码填写*/))
    model.add(/*此处由考生进行代码填写*/)
    model.add(Activation(/*此处由考生进行代码填写*/))
    model.add(/*此处由考生进行代码填写*/)
```

```
    #添加两层卷积层，卷积核尺寸为512*3*3，padding为same，激活函数选择relu
    #再添加一层池化层，窗口大小为2*2，步长为2，padding为same
    model.add(/*此处由考生进行代码填写*/)
    model.add(Activation(/*此处由考生进行代码填写*/))
    model.add(/*此处由考生进行代码填写*/)
    model.add(Activation(/*此处由考生进行代码填写*/))
    model.add(/*此处由考生进行代码填写*/)

    model.add(Flatten())
    model.add(Dense(1024))
    model.add(Activation('relu'))
    model.add(Dense(512))
    model.add(Activation('relu'))
    model.add(Dropout(0.3))
    #有无口罩两个分类，输出层只有两个神经元
    model.add(/*此处由考生进行代码填写*/)
    model.add(Activation(/*此处由考生进行代码填写*/))

    model.compile(optimizer=tf.keras.optimizers.Adam(lr=0.001, decay=0.1),
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])
    return model
```

模型初始化，并打印模型结构。**补充代码后，请将结果截图5保存到本地，用于实验报告上传。**

输入：

```
model=CNN_classification_model()
#打印模型结构
/*此处由考生进行代码填写*/
```

输出：

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 128, 128, 64)      1792
_________________________________________________________________
activation (Activation)      (None, 128, 128, 64)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 128, 128, 64)      36928
_________________________________________________________________
activation_1 (Activation)    (None, 128, 128, 64)      0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 64, 64, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 64, 64, 128)       73856
_________________________________________________________________
activation_2 (Activation)    (None, 64, 64, 128)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 64, 64, 128)       147584
_________________________________________________________________
activation_3 (Activation)    (None, 64, 64, 128)       0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 32, 32, 128)       0
_________________________________________________________________
```

```
conv2d_4 (Conv2D)            (None, 32, 32, 256)       295168
_________________________________________________________________
activation_4 (Activation)    (None, 32, 32, 256)       0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 32, 32, 256)       590080
_________________________________________________________________
activation_5 (Activation)    (None, 32, 32, 256)       0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 16, 16, 256)       0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 16, 16, 512)       1180160
_________________________________________________________________
activation_6 (Activation)    (None, 16, 16, 512)       0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 16, 16, 512)       2359808
_________________________________________________________________
activation_7 (Activation)    (None, 16, 16, 512)       0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 8, 8, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 32768)             0
_________________________________________________________________
dense (Dense)                (None, 1024)              33555456
_________________________________________________________________
activation_8 (Activation)    (None, 1024)              0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               524800
_________________________________________________________________
activation_9 (Activation)    (None, 512)               0
_________________________________________________________________
dropout (Dropout)            (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 1026
_________________________________________________________________
activation_10 (Activation)   (None, 2)                 0
=================================================================
Total params: 38,766,658
Trainable params: 38,766,658
Non-trainable params: 0
```

步骤2：模型训练

接下来进行模型训练。**补充代码后，请将结果截图6保存到本地，用于实验报告上传。**

输入：

```
# 进行训练
output = model./*此处由考生进行代码填写*/(/*此处由考生进行代码填写*/,/*此处由考生进行代码填写*/,batch_size=64,epochs=10,verbose=1,validation_split=0.1)
```

输出：

```
Train on 376 samples, validate on 42 samples
Epoch 1/10
376/376 [==============================] - 18s 48ms/sample - loss: 1.0247 - accuracy: 0.6596 - val_loss: 0.6768 - val_accuracy: 0.6667
```

```
Epoch 2/10
376/376 [==============================] - 17s 46ms/sample - loss: 0.6445 -
accuracy: 0.6676 - val_loss: 0.6075 - val_accuracy: 0.6667
Epoch 3/10
376/376 [==============================] - 17s 46ms/sample - loss: 0.5818 -
accuracy: 0.6676 - val_loss: 0.5413 - val_accuracy: 0.6667
Epoch 4/10
376/376 [==============================] - 17s 46ms/sample - loss: 0.5034 -
accuracy: 0.6676 - val_loss: 0.5071 - val_accuracy: 0.6667
Epoch 5/10
376/376 [==============================] - 17s 45ms/sample - loss: 0.5242 -
accuracy: 0.6676 - val_loss: 0.5354 - val_accuracy: 0.6667
Epoch 6/10
376/376 [==============================] - 17s 45ms/sample - loss: 0.4961 -
accuracy: 0.6676 - val_loss: 0.4816 - val_accuracy: 0.6667
Epoch 7/10
376/376 [==============================] - 17s 46ms/sample - loss: 0.4314 -
accuracy: 0.6676 - val_loss: 0.4324 - val_accuracy: 0.6667
Epoch 8/10
376/376 [==============================] - 17s 45ms/sample - loss: 0.4294 -
accuracy: 0.6702 - val_loss: 0.4139 - val_accuracy: 0.6667
Epoch 9/10
376/376 [==============================] - 17s 46ms/sample - loss: 0.4006 -
accuracy: 0.7101 - val_loss: 0.4017 - val_accuracy: 0.7619
Epoch 10/10
376/376 [==============================] - 17s 46ms/sample - loss: 0.3774 -
accuracy: 0.8218 - val_loss: 0.3868 - val_accuracy: 0.8095
```

将模型和权重文件保存至当前文件夹，**补充代码后，请将结果截图7保存到本地，用于实验报告上传。**

```
model./*此处由考生进行代码填写*/('./mask.h5')
model./*此处由考生进行代码填写*/('./weights.h5')
```

任务四：模型预测

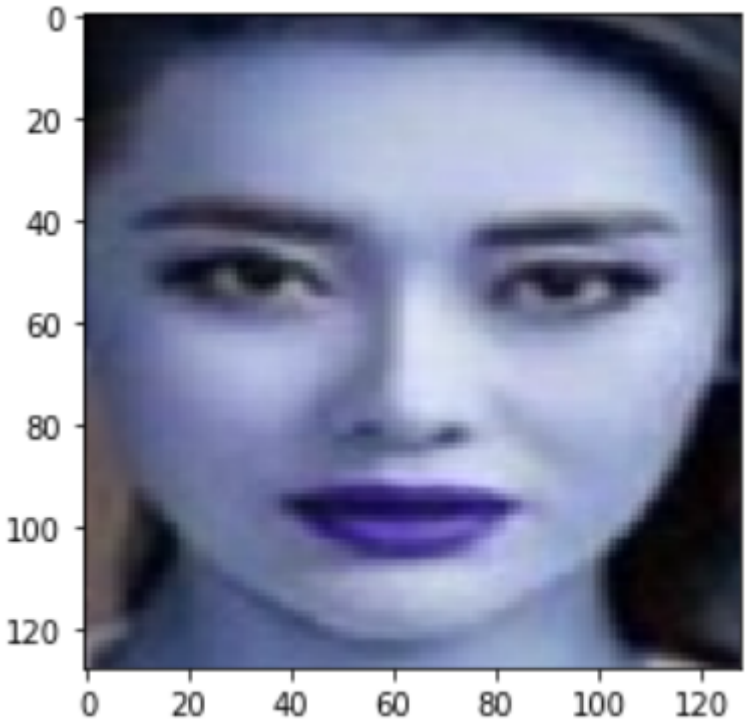
加载上一步保存的权重文件并输出模型的预测结果。**补充代码后，请将结果截图8保存到本地，用于实验报告上传。**

输入：

```
new_model = CNN_classification_model()
new_model./*此处由考生进行代码填写*/('./weights.h5')
predictions = new_model.predict(x_test)
# 预测一张图片
# 查看第n个样本
plt.imshow(x_test[46])

# 模型预测每个类别的概率
print(predictions[46])

# 模型预测类别，选择概率最大的作为模型预测的类别
print(/*此处由考生进行代码填写*/)

# 目标类别
print(y_test[46])
```

输出：

```
[0.9887921  0.01120787]
0
0
```



实验小结

本实验通过构建CNN网络模型预测是否佩戴口罩，可以自己任意修改网络构造方式来比较模型的准确率。

样题：计算机视觉技术与应用-答案

实验简介

实验介绍

随着新冠疫情的爆发，口罩已经改变了人们的生活习惯，为了防止新冠疫情的传播，口罩已经成为出门必需品，进入公共场合、坐公交地铁等都要求佩戴口罩。那么检测人们是否佩戴口罩就至关重要。

实验目的

本实验使用了几百条戴口罩和没戴口罩的数据集，先进行图像预处理，再构建卷积神经网络训练模型，最后导入模型进行预测。

实验环境

conda-py3.7.6-tf2.0

任务一：导入数据集

在当前目录创建一个notebook，kernel选择conda-py3.7.6-tf2.0即可，

输入以下代码下载数据集，进入数据集所在目录：

```
cd /data/cvlky/data
```

将数据集拷贝至当前文件夹：

```
cp CNN_mask_detection.zip  /ddhome/data
```

回到当前目录：

```
cd /ddhome/data
```

输入以下代码解压数据集：

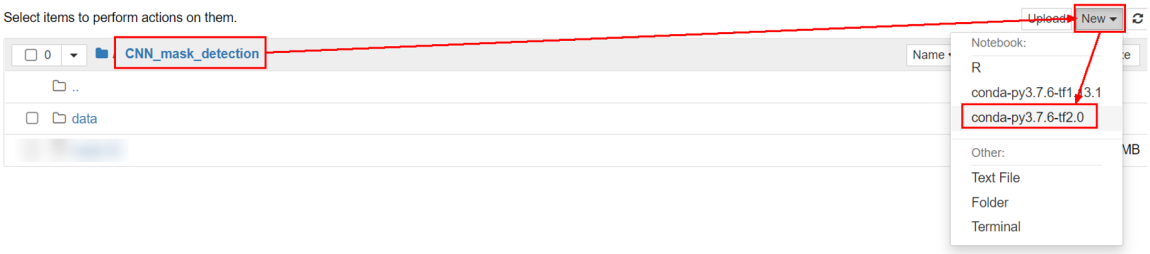
```
!unzip CNN_mask_detection.zip
```

如果没有unzip命令，可以输入如下代码安装unzip：

```
!yum install unzip -y
```

关闭此notebook，可以观察到解压后的data文件夹中有with_mask和without_mask两种类型数据。

进入CNN_mask_detection文件夹，**重新新建notebook**，即可开始实验。



任务二：图像的获取和预处理

步骤1：导入相关包

```
import os
import glob
import cv2
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Activation, Dropout,
Dense, Flatten
```

步骤2：获取数据

```
# data是数据集的目录
data = r'./data'
# 下面2个路径分别是2种数据的目录
with_mask = os.path.join(data, 'with_mask')
without_mask = os.path.join(data, 'without_mask')
# 遍历文件夹，将每一个图片的地址依次存入列表中
X_with_mask = [os.path.abspath(fp) for fp in glob.glob(os.path.join(with_mask,
'*.jpg'))]
X_without_mask = [os.path.abspath(fp) for fp in
glob.glob(os.path.join(without_mask, '*.jpg'))]
X = X_with_mask + X_without_mask
# 生成标签，1代表佩戴了口罩，0代表未佩戴口罩
Y = [1] * len(X_with_mask) + [0] * len(X_without_mask)
# 十字交叉分割，把所有的图片路径和标签通过shuffle=True打乱顺序，然后分0.9比例的给训练集图片和
训练集标签，剩下的作为测试集
x_train, x_test, y_train, y_test = train_test_split(X, Y, shuffle=True,
train_size=0.9)
```

步骤3：图像预处理

定义图像预处理函数，把所有图像的尺寸缩放为（64,64,3），再将图像进行归一化：

```
def image_preprocessing(path, label):
    data_set = []
```

```
    label_set = []
    # 遍历路径列表中的路径以及标签列表里的标签
    for (one_path, one_label) in zip(path,label):
        # cv2读取图片为numpy矩阵
        image = cv2.imread(one_path)
        # 缩放图片为指定大小
        image_resize = cv2.resize(image, (64, 64), cv2.INTER_AREA)

        # cv2读取的矩阵值在0-255，我们选择除以255将图片归一化
        image = image_resize/255

        data_set.append(image)
        label_set.append(one_label)
    # 转为矩阵形式返回
    return np.array(data_set), np.array(label_set)
```

调用上述函数，将训练集和测试集中的图像进行预处理，再输出处理好的图像尺寸和数量：

输入：

```
x_train,y_train = image_preprocessing(x_train,y_train)
x_test, y_test = image_preprocessing(x_test, y_test)
print(x_train.shape, y_train.shape, x_test.shape, y_test.shape)
```

输出：

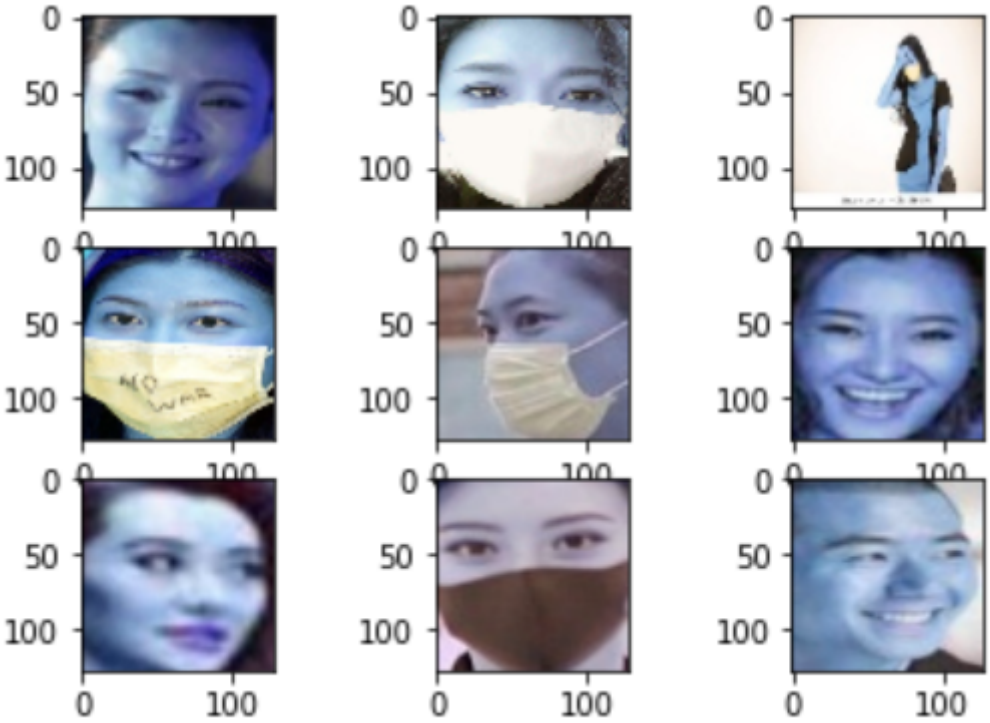
```
(418, 64, 64, 3) (418,) (47, 64, 64, 3) (47,)
```

接下来展示一些样本。

输入：

```
#展示前9张图片
plt.figure()
for i in range(9):
    plt.subplot(3,3,i+1)
    plt.imshow(x_train[i])
plt.show()
```

输出：



任务三：构建模型并训练

步骤1：构建CNN网络模型

CNN网络模型可以自定义修改，下面是构造了四组两层卷积层+一层最大池化层，再加三层全连接层的模型，隐藏层全部使用了relu激活函数，在输出层使用softmax激活函数。

```
def CNN_classification_model():
    model = Sequential()
    model.add(Conv2D(64,(3,3), padding='same',input_shape=(64,64,3)))
    model.add(Activation('relu'))
    model.add(Conv2D(64,(3,3), padding='same'))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2,2), strides=2, padding='same'))

    model.add(Conv2D(128,(3,3), padding='same'))
    model.add(Activation('relu'))
    model.add(Conv2D(128,(3,3), padding='same'))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2,2), strides=2, padding='same'))

    model.add(Conv2D(256,(3,3), padding='same'))
    model.add(Activation('relu'))
    model.add(Conv2D(256,(3,3), padding='same'))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2,2), strides=2, padding='same'))

    model.add(Conv2D(512,(3,3), padding='same'))
    model.add(Activation('relu'))
    model.add(Conv2D(512,(3,3), padding='same'))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2,2), strides=2, padding='same'))

    model.add(Flatten())
```

```
    model.add(Dense(1024))
    model.add(Activation('relu'))
    model.add(Dense(512))
    model.add(Activation('relu'))
    model.add(Dropout(0.3))
    #有无口罩两个分类，输出层只有两个神经元
    model.add(Dense(2))
    model.add(Activation('softmax'))

    model.compile(optimizer=tf.keras.optimizers.Adam(lr=0.001, decay=0.1),
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])
    return model
```

模型初始化，并打印模型结构。

输入：

```
model=CNN_classification_model()
model.summary()
```

输出：

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 128, 128, 64)      1792
_________________________________________________________________
activation (Activation)      (None, 128, 128, 64)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 128, 128, 64)      36928
_________________________________________________________________
activation_1 (Activation)    (None, 128, 128, 64)      0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 64, 64, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 64, 64, 128)       73856
_________________________________________________________________
activation_2 (Activation)    (None, 64, 64, 128)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 64, 64, 128)       147584
_________________________________________________________________
activation_3 (Activation)    (None, 64, 64, 128)       0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 32, 32, 128)       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 32, 32, 256)       295168
_________________________________________________________________
activation_4 (Activation)    (None, 32, 32, 256)       0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 32, 32, 256)       590080
_________________________________________________________________
activation_5 (Activation)    (None, 32, 32, 256)       0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 16, 16, 256)       0
_________________________________________________________________
```

```
conv2d_6 (Conv2D)            (None, 16, 16, 512)       1180160
_________________________________________________________________
activation_6 (Activation)    (None, 16, 16, 512)       0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 16, 16, 512)       2359808
_________________________________________________________________
activation_7 (Activation)    (None, 16, 16, 512)       0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 8, 8, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 32768)             0
_________________________________________________________________
dense (Dense)                (None, 1024)              33555456
_________________________________________________________________
activation_8 (Activation)    (None, 1024)              0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               524800
_________________________________________________________________
activation_9 (Activation)    (None, 512)               0
_________________________________________________________________
dropout (Dropout)            (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 1026
_________________________________________________________________
activation_10 (Activation)   (None, 2)                 0
=================================================================
Total params: 38,766,658
Trainable params: 38,766,658
Non-trainable params: 0
```

步骤2：模型训练

接下来进行模型训练。

输入：

```
# 进行训练
output =
model.fit(x_train,y_train,batch_size=64,epochs=10,verbose=1,validation_split=0.1
)
```

输出：

```
Train on 376 samples, validate on 42 samples
Epoch 1/10
376/376 [==============================] - 18s 48ms/sample - loss: 1.0247 -
accuracy: 0.6596 - val_loss: 0.6768 - val_accuracy: 0.6667
Epoch 2/10
376/376 [==============================] - 17s 46ms/sample - loss: 0.6445 -
accuracy: 0.6676 - val_loss: 0.6075 - val_accuracy: 0.6667
Epoch 3/10
376/376 [==============================] - 17s 46ms/sample - loss: 0.5818 -
accuracy: 0.6676 - val_loss: 0.5413 - val_accuracy: 0.6667
Epoch 4/10
376/376 [==============================] - 17s 46ms/sample - loss: 0.5034 -
accuracy: 0.6676 - val_loss: 0.5071 - val_accuracy: 0.6667
```

```
Epoch 5/10
376/376 [==============================] - 17s 45ms/sample - loss: 0.5242 -
accuracy: 0.6676 - val_loss: 0.5354 - val_accuracy: 0.6667
Epoch 6/10
376/376 [==============================] - 17s 45ms/sample - loss: 0.4961 -
accuracy: 0.6676 - val_loss: 0.4816 - val_accuracy: 0.6667
Epoch 7/10
376/376 [==============================] - 17s 46ms/sample - loss: 0.4314 -
accuracy: 0.6676 - val_loss: 0.4324 - val_accuracy: 0.6667
Epoch 8/10
376/376 [==============================] - 17s 45ms/sample - loss: 0.4294 -
accuracy: 0.6702 - val_loss: 0.4139 - val_accuracy: 0.6667
Epoch 9/10
376/376 [==============================] - 17s 46ms/sample - loss: 0.4006 -
accuracy: 0.7101 - val_loss: 0.4017 - val_accuracy: 0.7619
Epoch 10/10
376/376 [==============================] - 17s 46ms/sample - loss: 0.3774 -
accuracy: 0.8218 - val_loss: 0.3868 - val_accuracy: 0.8095
```

将模型和权重文件保存至当前文件夹：

```
model.save('./mask.h5')
model.save_weights('./weights.h5')
```

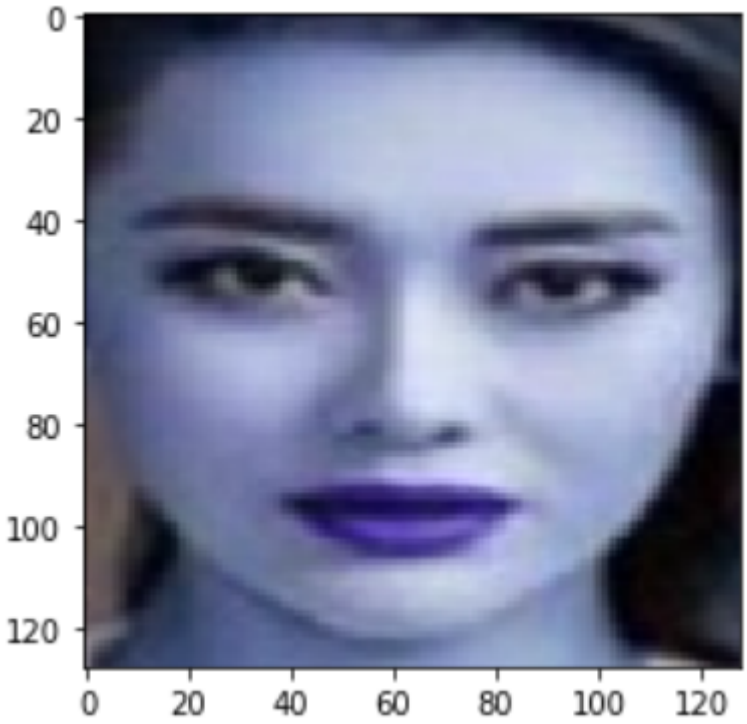
任务四：模型预测

加载上一步保存的权重文件并输出模型的预测结果：

```
new_model = CNN_classification_model()
new_model.load_weights('./weights.h5')
predictions = new_model.predict(x_test)
# 预测一张图片
# 查看第n个样本
plt.imshow(x_test[46])

# 模型预测每个类别的概率
print(predictions[46])

# 模型预测类别，选择概率最大的作为模型预测的类别
print(np.argmax(predictions[46]))

# 目标类别
print(y_test[46])
```

输出：

```
[0.9887921  0.01120787]
0
0
```



实验小结

本实验通过构建CNN网络模型预测是否佩戴口罩，可以自己任意修改网络构造方式来比较模型的准确率。