**人工智能训练师（三级）操作技能考核**

**试题单**

准考证号：

试题代码：

试题名称：人脸AI智能检测系统交互流程设计

考核时间：20min

**1.**场地设备要求

（1）人工智能训练师主机 1 台；

**2.**工作任务

在安防监控、智能交通等领域，实时准确的人脸检测需求日益增长。传统的人脸检测方法在面对复杂光照、多角度、遮挡等情况时，检测效果往往不尽人意。随着深度学习技术的发展，基于深度神经网络的人脸检测模型展现出强大的性能优势。ONNX（Open Neural Network Exchange）作为一种开放式的神经网络交换格式，能够实现不同深度学习框架间的模型转换与共享，使得基于 ONNX 的人脸检测模型可以在多种环境下高效运行。本系统所使用的 “version-RFB-320.onnx” 模型，通过大量数据训练，能够快速准确地检测出图像中的人脸，在实际应用场景中具有重要价值。

AI 模型说明：“version-RFB-320.onnx” 模型是用于人脸检测的 ONNX 格式模型，对应的类别标签文件为 “voc-model-labels.txt” 。该模型的使用交互流程为：

（1）加载 “version-RFB-320.onnx” 模型和 “voc-model-labels.txt” 类别标签；

（2）加载本地测试图片文件夹 “imgs” 中的所有图片，并对每张图片进行预处理以符合模型输入要求；

（3）使用 “version-RFB-320.onnx” 模型对加载的图片进行人脸检测；

（4）在图片上绘制检测到的人脸框，并将处理后的图片保存到 “./detect\_imgs\_results\_onnx” 文件夹中；

（5）统计所有图片中检测到的人脸总数并输出。

你作为一名人工智能训练师，请完成以下工作任务：

（1）补全该模型的使用交互流程对应的 Python 代码（3.2.5.ipynb），实现本地测试图片文件夹 “imgs” 中所有图片的人脸检测，将运行结果截图保存到3.2.5-1.jpg中，并将检测结果的图片上传。

（2）在上面的使用交互流程基础上，结合实际应用场景，设计在人脸检测系统中使用 “version-RFB-320.onnx” 模型的一种人机交互优化方案，包括交互界面布局、操作流程等内容，将其保存为docx文件，命名为 3.2.5.docx。

所有结果文件储存在桌面新建的考生文件夹中，文件夹命名为 “准考证号 + 身份证号后六位”。

**3.**技能要求

（1）能确保模型在单一场景下稳定运行；   
（2）能通过分析，找到单一场景下人工和智能交互的最优方式。

（3）能对单一场景下人工和智能交互界面设计提出优化需求。

**4.**质量指标

（1）模型运行稳定，使用正常；   
（2）单一场景下人工和智能交互的最优方式切实可行。

**人工智能训练师（三级）操作技能考核**

**试题评分表**

准考证号：

试题代码：

试题名称：人脸AI智能检测系统交互流程设计

考核时间：20min

测量分评分表

| 细则编号 | 配分 | 评分细则描述 | 规定或  标称值 | 结果或  实际值 | 得分 |
| --- | --- | --- | --- | --- | --- |
| M1 | 2 | 从标签文件中读取每一行，并去除行首尾的空白字符，得到类别名称列表代码正确得2分； | 根据数据 |  |  |
| M2 | 2 | 创建 ONNX Runtime 的推理会话，用于运行模型进行推理代码正确得2分； | 根据数据 |  |  |
| M3 | 2 | 获取模型输入的名称代码正确得2分； | 根据数据 |  |  |
| M4 | 2 | 果保存结果的目录不存在，则创建该目录代码得2分； | 根据数据 |  |  |
| M5 | 2 | 使用 OpenCV 读取图像文件 2分代码正确得2分； | 根据数据 |  |  |
| M6 | 2 | 将图像调整为 320x240 的尺寸（符合模型输入的尺寸要求）代码正确得2分； | 根据数据 |  |  |
| M7 | 2 | 定义图像归一化的均值数组代码正确得2分； | 根据数据 |  |  |
| M8 | 1 | 在第一个维度上扩展一个维度，将图像变为 (1, 通道数, 高度, 宽度)，以符合模型输入的维度要求代码正确得2分； | 根据数据 |  |  |
| M9 | 2 | 使用 ONNX Runtime 运行模型，输入图像数据，得到模型输出的置信度和边界框代码正确得2分； | 根据数据 |  |  |
| M10 | 1 | 本地保存运行结果的截图3.2.5-1.jpg中人脸数量正确：得1分； | 根据数据 |  |  |
| M11 | 1 | 检测结果的图片中人脸标注框和数量都正确得1分； | 根据数据 |  |  |
| M12 | 1 | 人机交互最优流程（不少于3条）正确得1分； | 根据数据 |  |  |
| 合计配分 | 20 | 合计得分 | |  |  |

1. 补全该模型的使用交互流程对应的 Python 代码（3.2.5.ipynb），实现本地测试图片文件夹 “imgs” 中所有图片的人脸检测，将运行结果截图保存到3.2.5-1.jpg中，并将检测结果的图片上传。

代码3.2.5.ipynb：

import os

import time

import cv2

import numpy as np

import vision.utils.box\_utils\_numpy as box\_utils

import onnxruntime as ort

# 定义预测函数，对模型输出的边界框和置信度进行后处理

def predict(width, height, confidences, boxes, prob\_threshold, iou\_threshold=0.3, top\_k=-1):

boxes = boxes[0]

confidences = confidences[0]

picked\_box\_probs = []

picked\_labels = []

for class\_index in range(1, confidences.shape[1]):

probs = confidences[:, class\_index]

mask = probs > prob\_threshold

probs = probs[mask]

if probs.shape[0] == 0:

continue

subset\_boxes = boxes[mask, :]

box\_probs = np.concatenate([subset\_boxes, probs.reshape(-1, 1)], axis=1)

box\_probs = box\_utils.hard\_nms(box\_probs,

iou\_threshold=iou\_threshold,

top\_k=top\_k,

)

picked\_box\_probs.append(box\_probs)

picked\_labels.extend([class\_index] \* box\_probs.shape[0])

if not picked\_box\_probs:

return np.array([]), np.array([]), np.array([])

picked\_box\_probs = np.concatenate(picked\_box\_probs)

picked\_box\_probs[:, 0] \*= width

picked\_box\_probs[:, 1] \*= height

picked\_box\_probs[:, 2] \*= width

picked\_box\_probs[:, 3] \*= height

return picked\_box\_probs[:, :4].astype(np.int32), np.array(picked\_labels), picked\_box\_probs[:, 4]

# 从标签文件中读取每一行，并去除行首尾的空白字符，得到类别名称列表 2分

class\_names = [name.strip() for name in open('voc-model-labels.txt').readlines()]

# 创建 ONNX Runtime 的推理会话，用于运行模型进行推理 2分

ort\_session = ort.InferenceSession('version-RFB-320.onnx')

# 获取模型输入的名称 2分

input\_name = ort\_session.get\_inputs()[0].name

# 定义保存检测结果图像的目录路径

result\_path = "./detect\_imgs\_results\_onnx"

# 定义置信度阈值，用于筛选出置信度较高的检测结果

threshold = 0.7

# 定义存储待检测图像的目录路径

path = "imgs"

# 用于统计所有图像中检测到的目标框总数，初始化为 0

sum = 0

# 如果保存结果的目录不存在，则创建该目录 2分

if not os.path.exists(result\_path):

os.makedirs(result\_path)

# 获取指定目录下的所有文件和文件夹名称列表

listdir = os.listdir(path)

# 遍历目录下的每个文件

for file\_path in listdir:

# 拼接图像文件的完整路径

img\_path = os.path.join(path, file\_path)

# 使用 OpenCV 读取图像文件 2分

orig\_image = cv2.imread(img\_path)

# 将图像从 BGR 颜色空间转换为 RGB 颜色空间（许多模型要求输入为 RGB 格式）

image = cv2.cvtColor(orig\_image, cv2.COLOR\_BGR2RGB)

# 将图像调整为 320x240 的尺寸（符合模型输入的尺寸要求） 2分

image = cv2.resize(image, (320, 240))

# 定义图像归一化的均值数组 2分

image\_mean = np.array([127, 127, 127])

# 对图像进行归一化处理，减去均值并除以 128

image = (image - image\_mean) / 128

# 将图像的维度从 (高度, 宽度, 通道数) 转换为 (通道数, 高度, 宽度)

image = np.transpose(image, [2, 0, 1])

# 在第一个维度上扩展一个维度，将图像变为 (1, 通道数, 高度, 宽度)，以符合模型输入的维度要求 1分

image = np.expand\_dims(image, axis=0)

# 将图像数据类型转换为 float32 类型

image = image.astype(np.float32)

# 记录开始时间，用于计算模型推理的耗时

time\_time = time.time()

# 使用 ONNX Runtime 运行模型，输入图像数据，得到模型输出的置信度和边界框 2分

confidences, boxes = ort\_session.run(None, {input\_name: image})

# 计算并打印模型推理的耗时

print("cost time:{}".format(time.time() - time\_time))

# 调用 predict 函数对模型输出的边界框和置信度进行后处理，得到最终的边界框、类别标签和置信度

boxes, labels, probs = predict(orig\_image.shape[1], orig\_image.shape[0], confidences, boxes, threshold)

# 遍历每个检测到的目标框

for i in range(boxes.shape[0]):

# 获取当前目标框的坐标

box = boxes[i, :]

# 生成当前目标框的标签字符串，包含类别名称和置信度

label = f"{class\_names[labels[i]]}: {probs[i]:.2f}"

# 在原始图像上绘制目标框，颜色为 (255, 255, 0)，线条粗细为 4

cv2.rectangle(orig\_image, (box[0], box[1]), (box[2], box[3]), (255, 255, 0), 4)

# 将绘制了目标框的图像保存到结果目录中

cv2.imwrite(os.path.join(result\_path, file\_path), orig\_image)

# 累加当前图像中检测到的目标框数量到总数中

sum += boxes.shape[0]

# 打印所有图像中检测到的目标框总数

print("sum:{}".format(sum))

截图3.2.5-1.jpg：



（2）在上面的使用交互流程基础上，结合实际应用场景，设计在人脸检测系统中使用 “version-RFB-320.onnx” 模型的一种人机交互优化方案，包括交互界面布局、操作流程等内容，将其保存为docx文件，命名为 3.2.5.docx。

参考答案： （答对3条得满分）

1. 交互主界面布局设计可以遵循 **"功能分区明确、操作流程线性、信息层级清晰"** 三个核心原则， 通过空间分区映射功能逻辑，让用户无需思考即可完成「选择→处理→保存」的标准流程。
2. 操作流程上，实施标准检测流程：

1）初始化（系统自动加载模型和标签文件，检查必要的依赖项（OpenCV等））

2）选择输入源： 用户点击"选择文件夹"按钮

3） 执行检测：系统依次处理每张图片，实时显示进度， 处理完成后播放提示音并显示统计信息

4）最后点击"保存结果"按钮，将结果保存指定位置

1. 用户体验上持续优化，可以采取：
2. 视觉反馈：检测中的人脸框使用渐变色（如黄→红表示置信度低→高）， 处理进度条和预估剩余时间显示
3. 性能优化：增加多线程处理，保持UI响应； 图片预处理和模型推理分离
4. 对于错误处理的处理，增加友好的错误提示（如模型加载失败、图片格式不支持等）， 提供重试机制
5. 增加一些辅助功能， 比如支持图片缩放和拖动查看细节，增加一些快捷键支持。