实验七 基于 tflite 的深度模型应用

一、实验目的及要求

1、学习和掌握 tflite 的安装方法

2、掌握观测 tflite 模型输入输出数据格式的方法。

3、掌握利用 tflite 模型实现图像识别的方法。

4、掌握利用 tflite 模型实现目标检测的方法。

5、掌握利用 tflite 模型实现人体关键点检测的方法。

6、掌握利用 tflite 模型实现图像风格转换的方法

二、预习要求

阅读本实验例程部分，实现 tflite 的安装与预训练模型的下载，以便能够充分利用

实验时间编程调试。

三、实验设备

硬件：PC 机。

软件：Python 及相关集成开发环境。

四、实验内容

（1）实现 tflite 的安装

安装 TensorFlow

安装 TFLite 前需要先安装 TensorFlow。可通过以下命令安装 TensorFlow：

**pip install tensorflow**

安装 TFLite

TFLite 可以通过以下命令进行安装：

**pip install tflite**

安装 TFLite Interpreter

TFLite Interpreter 是运行 TFLite 模型的 Python 库，可以通过以下命令进行安装：

**pip install tflite-runtime**

1. 利用预训练模型实现图像识别功能（tf\_cam\_classification.py）

import numpy as np

import cv2

import tflite\_runtime.interpreter as tflite

from PIL import Image,ImageFont,ImageDraw

def paint\_chinese\_opencv(im,chinese,pos,color):

img\_PIL = Image.fromarray(cv2.cvtColor(im,cv2.COLOR\_BGR2RGB))

font = ImageFont.truetype('NotoSansCJK-Bold.ttc',25,encoding="utf-8")

fillColor = color *#(255,0,0)*

position = pos *#(100,100)*

*# if not isinstance(chinese,unicode):*

*# chinese = chinese.decode('utf-8')*

draw = ImageDraw.Draw(img\_PIL)

draw.text(position,chinese,fillColor,font)

img = cv2.cvtColor(np.asarray(img\_PIL),cv2.COLOR\_RGB2BGR)

return img

if \_\_name\_\_ == "\_\_main\_\_":

*# 输出概率最大的三个分类结果*

Top\_K = 3

*# 分类模型*

file\_model = "mobileNet\_V1\\mobilenet\_v1\_1.0\_224\_quant.tflite"

*# 标签列表*

file\_label = "mobileNet\_V1\\labels\_mobilenet\_quant\_v1\_224\_cn\_baidu.txt"

*# 读取标签*

with open(file\_label, 'r',encoding='utf-8') as f:

labels = [line.strip() for line in f.readlines()]

*# 加载分类模型*

interpreter = tflite.Interpreter(model\_path=file\_model)

interpreter.allocate\_tensors()

*# 读取输入数据细节*

input\_details = interpreter.get\_input\_details()

print('Info of input')

print(input\_details)

*# 读取输出数据的细节*

output\_details = interpreter.get\_output\_details()

print('Info of output')

print(output\_details)

*# 获取输入图像的尺寸要求*

height = input\_details[0]['shape'][1]

width = input\_details[0]['shape'][2]

*# # 打开摄像头*

*# url = "http://admin:admin@192.168.3.27:8081"*

*# url = 0*

cap = cv2.VideoCapture(0)

*# 初始化帧率计算*

frame\_rate\_calc = 1

freq = cv2.getTickFrequency()

while True:

*# 获取起始时间*

t1 = cv2.getTickCount()

*# 读取一帧图像*

success, img = cap.read()

*# 获取它的尺寸*

imH,imW,\_ = np.shape(img)

*# BGR 转RGB*

img\_rgb = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

*# 尺寸缩放适应网络输入要求*

img\_resized = cv2.resize(img\_rgb, (width, height))

*# 维度扩张适应网络输入要求*

input\_data = np.expand\_dims(img\_resized, axis=0)

*# 数据输入网络*

interpreter.set\_tensor(input\_details[0]['index'],input\_data)

*# 进行识别*

interpreter.invoke()

*# 获得输出*

outputs = interpreter.get\_tensor(output\_details[0]['index'])[0]

output = np.squeeze(outputs)

*# 根据量化情况对输出进行还原*

if output\_details[0]['dtype'] == np.uint8:

scale, zero\_point = output\_details[0]['quantization']

output = scale \* (output - zero\_point)

*# 找到Top-K 个最大值*

ordered = np.argpartition(-output, Top\_K-1)

*# 输出标签以及分类的概率输出*

for i in range(Top\_K):

str\_info = "%s %.2f%%"%(labels[ordered[i]],output[ordered[i]]\*100)

pos = (1,1+i\*25)

img = paint\_chinese\_opencv(img,str\_info,pos,(255,0,0))

cv2.putText(img,'FPS: %.2f'%(frame\_rate\_calc),(imW-200,imH-20),

cv2.FONT\_HERSHEY\_SIMPLEX,1,(255,255,0),2,cv2.LINE\_AA)

*# 显示结果*

cv2.imshow('Result', img)

*# 计算帧率*

t2 = cv2.getTickCount()

time1 = (t2-t1)/freq

frame\_rate\_calc= 1/time1

*# 按q退出*

if cv2.waitKey(1) & 0xFF == ord('q'):

break

cap.release()



1. 利用预训练模型实现目标检测功能（tf\_obj\_det\_cam.py）

import numpy as np

import cv2

import tflite\_runtime.interpreter as tflite

from PIL import Image,ImageFont,ImageDraw

def paint\_chinese\_opencv(im,chinese,pos,color):

img\_PIL = Image.fromarray(cv2.cvtColor(im,cv2.COLOR\_BGR2RGB))

font = ImageFont.truetype('NotoSansCJK-Bold.ttc',25,encoding="utf-8")

fillColor = color *#(255,0,0)*

position = pos *#(100,100)*

*# if not isinstance(chinese,unicode):*

*# chinese = chinese.decode('utf-8')*

draw = ImageDraw.Draw(img\_PIL)

draw.text(position,chinese,fillColor,font)

img = cv2.cvtColor(np.asarray(img\_PIL),cv2.COLOR\_RGB2BGR)

return img

if \_\_name\_\_=="\_\_main\_\_":

*# 设置检测阈值*

min\_conf\_threshold = 0.35

*# 检测模型*

file\_model = "model\_obj\_detect\\detect.tflite"

*# 标签*

file\_label = "model\_obj\_detect\\labelmap\_cn.txt"

*# 获取标签*

with open(file\_label, 'r',encoding="utf-8") as f:

labels = [line.strip() for line in f.readlines()]

if labels[0] == '???':

del(labels[0])

*# 载入模型*

interpreter = tflite.Interpreter(model\_path=file\_model)

interpreter.allocate\_tensors()

*# 获取输入、输出的数据的信息*

input\_details = interpreter.get\_input\_details()

print('input\_details\n',input\_details)

output\_details = interpreter.get\_output\_details()

print('output\_details',output\_details)

*# 获取输入图像的高和宽*

height = input\_details[0]['shape'][1]

width = input\_details[0]['shape'][2]

*# 打开摄像头*

url = "http://admin:admin@192.168.3.27:8081"

cap = cv2.VideoCapture(0)

*# 初始化帧率计算*

frame\_rate\_calc = 1

freq = cv2.getTickFrequency()

while True:

*# 获取起始时间*

t1 = cv2.getTickCount()

*# 读取一帧图像*

success, frame = cap.read()

*# 获取图像的宽和高*

imH,imW,\_ = np.shape(frame)

*# RGB 转 BGR*

frame\_rgb = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)

frame\_resized = cv2.resize(frame\_rgb, (width, height))

input\_data = np.expand\_dims(frame\_resized, axis=0)

*# 输入图像*

interpreter.set\_tensor(input\_details[0]['index'],input\_data)

*# 进行检测*

interpreter.invoke()

*# 获取检测结果*

*# 检测物体的边框*

boxes = interpreter.get\_tensor(output\_details[0]['index'])[0] *# Bounding box coordinates of detected objects*

*# 检测物体的类别*

classes = interpreter.get\_tensor(output\_details[1]['index'])[0] *# Class index of detected objects*

*# 检测物体的分数*

scores = interpreter.get\_tensor(output\_details[2]['index'])[0] *# Confidence of detected objects*

*# 对于概率大于 50%的进行显示*

for i in range(len(scores)):

if ((scores[i] > min\_conf\_threshold) and (scores[i] <= 1.0)):

*# 获取边框坐标*

ymin = int(max(1,(boxes[i][0] \* imH)))

xmin = int(max(1,(boxes[i][1] \* imW)))

ymax = int(min(imH,(boxes[i][2] \* imH)))

xmax = int(min(imW,(boxes[i][3] \* imW)))

*# 画框*

cv2.rectangle(frame, (xmin,ymin), (xmax,ymax), (10, 255, 0), 2)

*# 获取检测标签*

object\_name = labels[int(classes[i])] *# Look up object name from "labels" array using class index*

label = '%s: %d%%' % (object\_name, int(scores[i]\*100)) *# Example: 'person: 72%'*

*#显示标记*

frame = paint\_chinese\_opencv(frame,label,(xmin,ymin-5),(255,0,0))

*# 显示帧率*

cv2.putText(frame,'FPS: %.2f'%(frame\_rate\_calc),(imW-200,imH-20),cv2.FONT\_HERSHEY\_SIMPLEX,1,(255,255,0),2,cv2.LINE\_AA)

*# 显示结果*

cv2.imshow('object detect', frame)

*# 计算帧率*

t2 = cv2.getTickCount()

time1 = (t2-t1)/freq

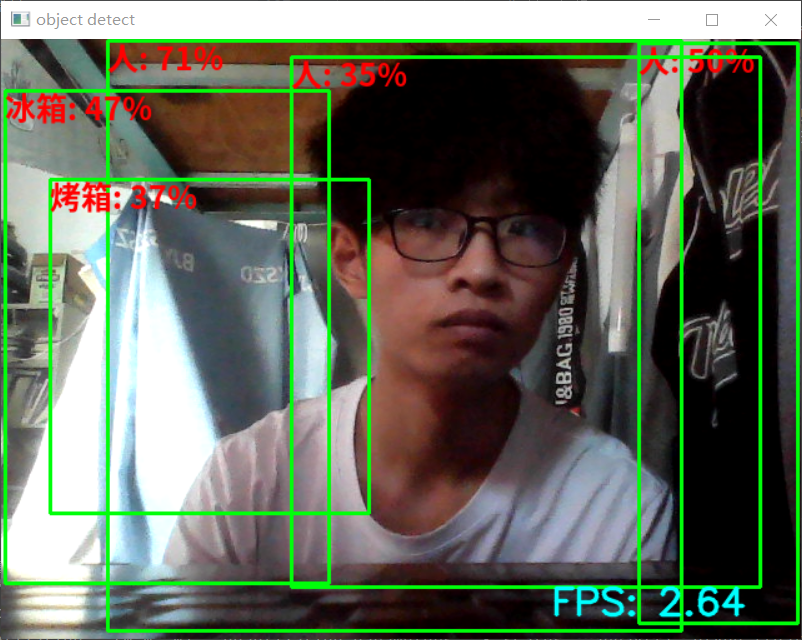
frame\_rate\_calc= 1/time1

*# 按q退出*

if cv2.waitKey(1) & 0xFF == ord('q'):

break

cap.release()



1. 利用预训练模型实现位姿检测功能，并进行简单的动作识别（tf\_cam\_posnet.py）

import numpy as np

import cv2

import tflite\_runtime.interpreter as tflite

from PIL import Image,ImageFont,ImageDraw

def paint\_chinese\_opencv(im,chinese,pos,color):

img\_PIL = Image.fromarray(cv2.cvtColor(im,cv2.COLOR\_BGR2RGB))

font = ImageFont.truetype('NotoSansCJK-Bold.ttc',25,encoding="utf-8")

fillColor = color *#(255,0,0)*

position = pos *#(100,100)*

*# if not isinstance(chinese,unicode):*

*# chinese = chinese.decode('utf-8')*

draw = ImageDraw.Draw(img\_PIL)

draw.text(position,chinese,fillColor,font)

img = cv2.cvtColor(np.asarray(img\_PIL),cv2.COLOR\_RGB2BGR)

return img

def get\_angle(v1,v2):

angle = np.dot(v1,v2)/(np.sqrt(np.sum(v1\*v1))\*np.sqrt(np.sum(v2\*v2)))

angle = np.arccos(angle)/3.14\*180

cross = v2[0]\*v1[1] - v2[1]\*v1[0]

if cross<0:

angle = - angle

return angle

def get\_pos(keypoints):

*# 计算右臂与水平方向的夹角*

keypoints = np.array(keypoints)

v1 = keypoints[5]- keypoints[6]

v2 = keypoints[8]- keypoints[6]

angle\_right\_arm = get\_angle(v1,v2)

*# 计算左臂与水平方向的夹角*

v1 = keypoints[7]- keypoints[5]

v2 = keypoints[6]- keypoints[5]

angle\_left\_arm = get\_angle(v1,v2)

*# 计算左肘的夹角*

v1 = keypoints[6]- keypoints[8]

v2 = keypoints[10]- keypoints[8]

angle\_right\_elbow = get\_angle(v1,v2)

*# 计算右肘的夹角*

v1 = keypoints[5]- keypoints[7]

v2 = keypoints[9]- keypoints[7]

angle\_left\_elbow = get\_angle(v1,v2)

str\_pos = ""

*# 设计动作识别规则*

if angle\_right\_arm<0 and angle\_left\_arm<0:

str\_pos = "正常"

if abs(angle\_left\_elbow)<120 and abs(angle\_right\_elbow)<120:

str\_pos = "叉腰"

elif angle\_right\_arm<0 and angle\_left\_arm>0:

str\_pos = "抬左手"

elif angle\_right\_arm>0 and angle\_left\_arm<0:

str\_pos = "抬右手"

elif angle\_right\_arm>0 and angle\_left\_arm>0:

str\_pos = "抬双手"

if abs(angle\_left\_elbow)<120 and abs(angle\_right\_elbow)<120:

str\_pos = "三角形"

return str\_pos

if \_\_name\_\_=="\_\_main\_\_":

*# 检测模型*

file\_model = "posenet\_mobilenet\_v1\_100\_257x257\_multi\_kpt\_stripped.tflite"

interpreter = tflite.Interpreter(model\_path=file\_model)

interpreter.allocate\_tensors()

*# 获取输入、输出的数据的信息*

input\_details = interpreter.get\_input\_details()

print('input\_details\n',input\_details)

output\_details = interpreter.get\_output\_details()

print('output\_details',output\_details)

*# 获取PosNet 要求输入图像的高和宽*

height = input\_details[0]['shape'][1]

width = input\_details[0]['shape'][2]

*# 初始化帧率计算*

frame\_rate\_calc = 1

freq = cv2.getTickFrequency()

video = "pos.mp4"

*# 打开摄像头*

cap = cv2.VideoCapture(video)

while True:

*# 获取起始时间*

t1 = cv2.getTickCount()

*# 读取一帧图像*

success, img = cap.read()

if not success:

break

*# 获取图像帧的尺寸*

imH,imW,\_ = np.shape(img)

*# 适当缩放*

img = cv2.resize(img,(int(imW\*0.5),int(imH\*0.5)))

*# 获取图像帧的尺寸*

imH,imW,\_ = np.shape(img)

*# BGR 转RGB*

img\_rgb = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

*# 尺寸缩放适应PosNet 网络输入要求*

img\_resized = cv2.resize(img\_rgb, (width, height))

*# 维度扩张适应网络输入要求*

input\_data = np.expand\_dims(img\_resized, axis=0)

*# 尺度缩放 变为 -1~+1*

input\_data = (np.float32(input\_data) - 128.0)/128.0

*# 数据输入网络*

interpreter.set\_tensor(input\_details[0]['index'],input\_data)

*# 进行关键点检测*

interpreter.invoke()

*# 获取hotmat*

hotmaps = interpreter.get\_tensor(output\_details[0]['index'])[0] *# Bounding box coordinates of detected objects*

*# 获取偏移量*

offsets = interpreter.get\_tensor(output\_details[1]['index'])[0] *# Class index of detected objects*

*# 获取hotmat的 宽 高 以及关键的数目*

h\_output,w\_output, n\_KeyPoints= np.shape(hotmaps)

*# 存储关键点的具体位置*

keypoints =[]

*# 关键点的置信度*

score =0

for i in range(n\_KeyPoints):

*# 遍历每一张hotmap*

hotmap = hotmaps[:,:,i]

*# 获取最大值 和最大值的位置*

max\_index = np.where(hotmap==np.max(hotmap))

max\_val = np.max(hotmap)

*# 获取y，x偏移量 前n\_KeyPoints张图是y的偏移 后n\_KeyPoints张图是x的偏移*

offset\_y = offsets[max\_index[0],max\_index[1],i]

offset\_x = offsets[max\_index[0],max\_index[1],i+n\_KeyPoints]

*# 计算在posnet输入图像中具体的坐标*

pos\_y = max\_index[0]/(h\_output-1)\*height + offset\_y

pos\_x = max\_index[1]/(w\_output-1)\*width + offset\_x

*# 计算在源图像中的坐标*

pos\_y = pos\_y/(height-1)\*imH

pos\_x = pos\_x/(width-1)\*imW

*# 取整获得keypoints的位置*

keypoints.append([int(round(pos\_x[0])),int(round(pos\_y[0]))])

*# 利用sigmoid函数计算置每一个点的置信度*

score = score + 1.0/(1.0+np.exp(-max\_val))

*# 取平均得到最终的置信度*

score = score/n\_KeyPoints

if score>0.5:

*# 标记关键点*

for point in keypoints:

cv2.circle(img,(point[0],point[1]),5,(255,255,0),5)

*# 画关节连接线*

*# 左臂*

cv2.polylines(img, [np.array([keypoints[5],keypoints[7],keypoints[9]])],False, (0,255,0), 3)

*# # 右臂*

cv2.polylines(img, [np.array([keypoints[6],keypoints[8],keypoints[10]])],False, (0,0,255), 3)

*# # 左腿*

cv2.polylines(img, [np.array([keypoints[11],keypoints[13],keypoints[15]])],False, (0,255,0), 3)

*# # 右腿*

cv2.polylines(img, [np.array([keypoints[12],keypoints[14],keypoints[16]])],False, (0,255,255), 3)

*# 身体部分*

cv2.polylines(img, [np.array([keypoints[5],keypoints[6],keypoints[12],keypoints[11],keypoints[5]])],False, (255,255,0), 3)

*# 计算位置角*

str\_pos = get\_pos(keypoints)

*# 显示动作识别结果*

img = paint\_chinese\_opencv(img,str\_pos,(0,5),(255,0,0))

*# 显示帧率*

cv2.putText(img,'FPS: %.2f score:%.2f'%(frame\_rate\_calc,score),(imW-350,imH-20),cv2.FONT\_HERSHEY\_SIMPLEX,1,(255,255,0),2,cv2.LINE\_AA)

*# 显示结果*

cv2.imshow('Pos', img)

*# 计算帧率*

t2 = cv2.getTickCount()

time1 = (t2-t1)/freq

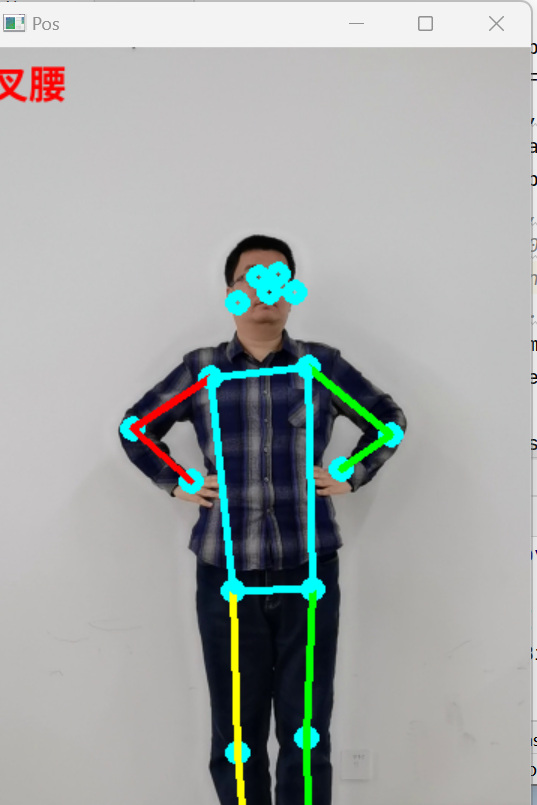
frame\_rate\_calc= 1/time1

*# 按q退出*

if cv2.waitKey(1) & 0xFF == ord('q'):

break

cap.release()



（5）利用预训练模型实现图像风格功能先运行 collect\_fea\_style.py 实现风格提取，再

运行 img\_style\_transform.py 实现风格转换

import numpy as np

import cv2

import tflite\_runtime.interpreter as tflite

import os

*# 风格特征提取模型*

file\_model\_prediction = "model/magenta\_arbitrary-image-stylization-v1-256\_int8\_prediction\_1.tflite"

base\_path ="img\_style"

def img\_preprocessing(img,size\_output):

*# 获取图像的尺寸*

imH,imW,\_ = np.shape(img)

*# BGR 转RGB*

img\_rgb = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

*# 尺寸缩放适应网络输入要求*

img\_resized = cv2.resize(img\_rgb, size\_output)

*# 维度扩张适应网络输入要求*

input\_data = np.expand\_dims(img\_resized, axis=0)

*# 正则化变为 0-1之间*

input\_data = input\_data/255.0

return (imW,imH), np.float32(input\_data)

*# 从风格图片中提取风格特征矢量.*

def run\_style\_predict(file\_model\_prediction,img):

*# 加载模型*

interpreter = tflite.Interpreter(model\_path=file\_model\_prediction)

interpreter.allocate\_tensors()

*# 获取输入的数据的信息*

input\_details = interpreter.get\_input\_details()

*# 获取输入图像的高和宽*

height = input\_details[0]['shape'][1]

width = input\_details[0]['shape'][2]

*# 对图像进行预处理*

t,preprocessed\_style\_image = img\_preprocessing(img,(width,height))

*# 特征输入网络*

interpreter.set\_tensor(input\_details[0]["index"], preprocessed\_style\_image)

*# 获取模型矢量.*

interpreter.invoke()

style\_bottleneck = interpreter.tensor(interpreter.get\_output\_details()[0]["index"])()

return style\_bottleneck

if \_\_name\_\_ == "\_\_main\_\_":

*# 遍历文件夹*

for f\_img in os.listdir(base\_path):

*# 遍历所有以.jpg为后缀的文件*

if os.path.splitext(f\_img)[-1] == ".jpg":

str\_style = os.path.splitext(f\_img)[0]

print(str\_style)

f\_img = os.path.join(base\_path,f\_img)

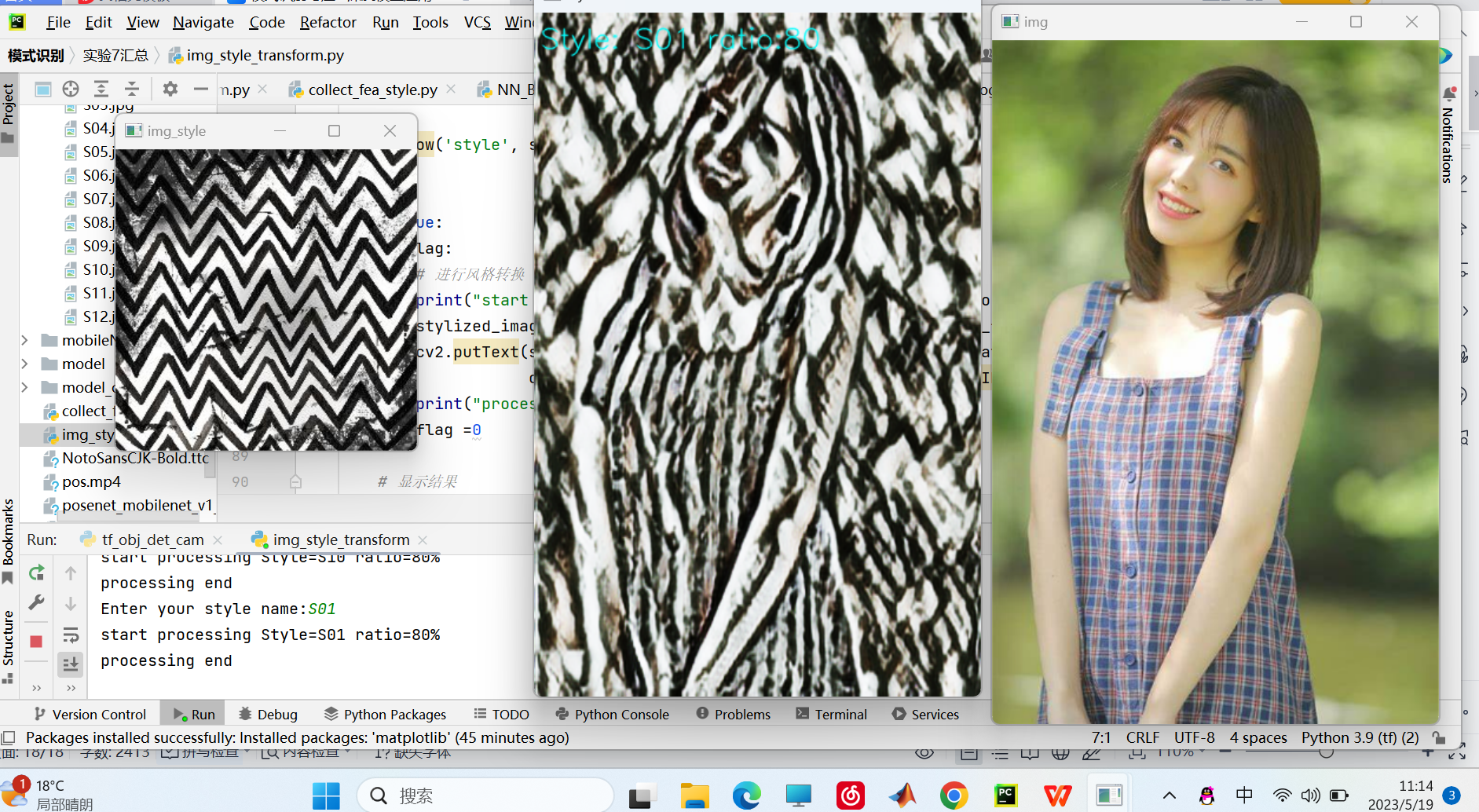
*# 提取风格特征*

style\_bottleneck = run\_style\_predict(file\_model\_prediction,cv2.imread(f\_img))

*# 特征保存*

print('save style feature of image %s as %s.npz'%(f\_img,os.path.join('fea\_style',str\_style)))

np.savez(os.path.join('fea\_style',str\_style),fea = style\_bottleneck)



五、实验报告内容要求

1. 列出编写的 python 代码。对主要的语句进行注释

2. 对实验结果进行截图，对结果进行必要的解释并说明实验中使用的参数。

3. 写出调试的过程，说明测试用例及调试中遇到的主要问题和解决方法。

4. 写出实验收获与不足，以及对实验的相关意见。