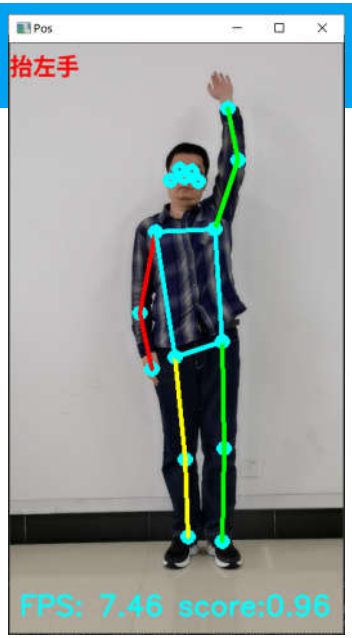
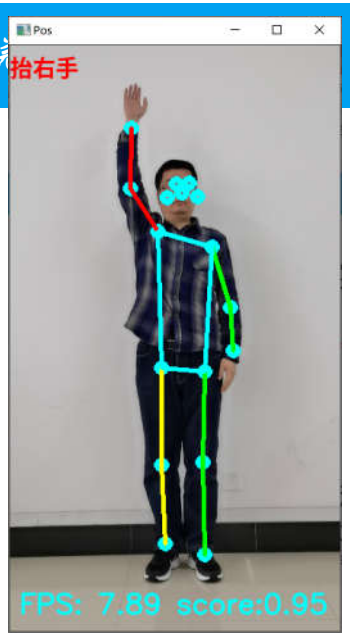
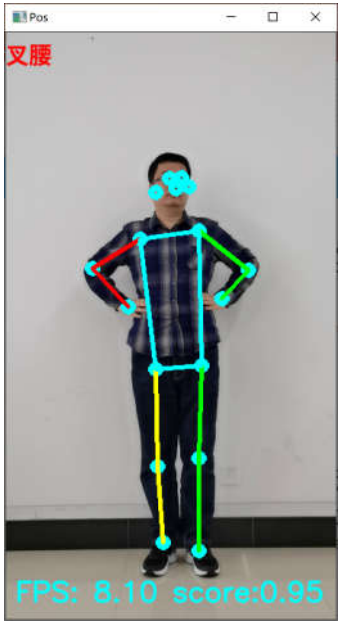
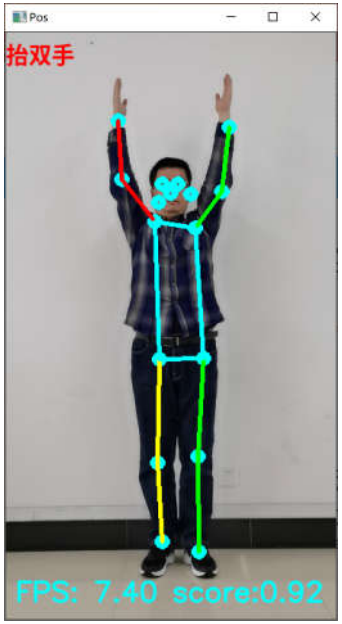




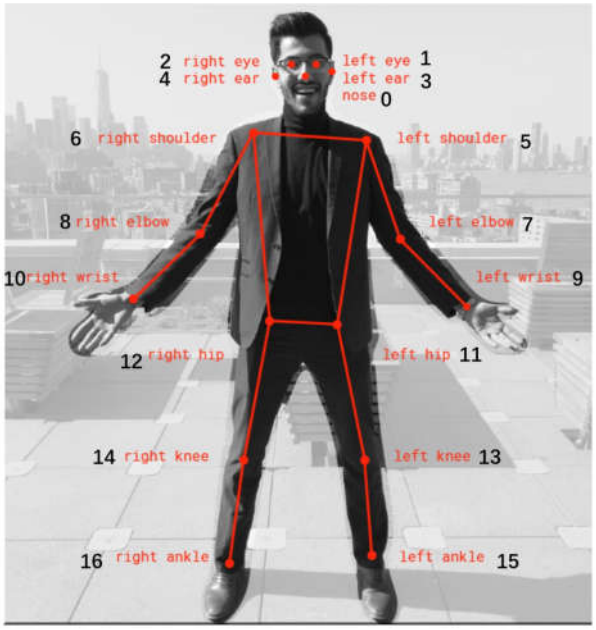
# Python编程与人工智能实践

## 应用篇：基于PosNet的位姿检测与动作识别

于泓
鲁东大学
信息与电气工程学院
2021.5.11

# 位姿检测

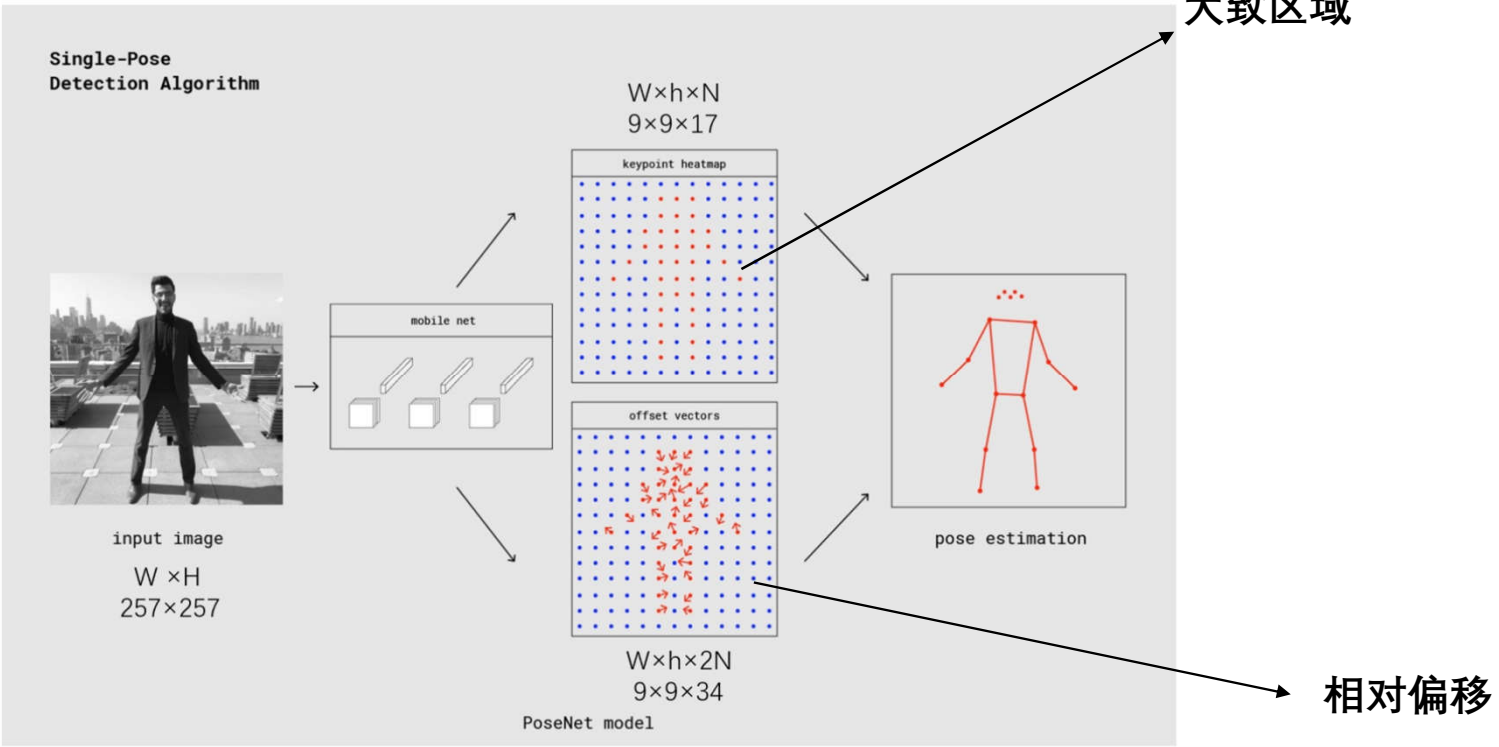
人体关键点检测（Human Keypoints Detection）又称为人体姿态估计（Pose Estimation），是计算机视觉中一个重要任务，是人体动作识别、行为分析、人机交互等的前置任务。




目前 COCO keypoint track 是人体姿态估计的权威公开比赛之一，COCO数据集中把人体关键点表示为 17 个关节，人体姿态估计的任务就是从输入的图片中检测到人体及对应的关键点位置。

# Posnet的基本原理

PoseNet是 Google 公司提出的一种基于深度学习的实时人体姿态模型，可以实现实时的人体位姿估计




https://storage.googleapis.com/download.tensorflow.org/models/tflite/posenet_mobilenet_v1_100_257x257_multi_kpt_stripped.tflite

```python
import numpy as np
import cv2
import tflite_runtime.interpreter as tflite
from PIL import Image,ImageFont,ImageDraw


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)
```

正则化

```
input_details
 [{'name': 'sub_2', 'index': 93, 'shape': array([  1, 257, 257,   3]),
 'shape_signature': array([  1, 257, 257,   3]),
 'dtype': <class 'numpy.float32'>,
 'quantization': (0.0, 0),
 'quantization_parameters': {'scales': array([], dtype=float32),
 'zero_points': array([], dtype=int32),
 'quantized_dimension': 0}, 'sparsity_parameters': {}}]
```

```
output_details
[{'name': 'MobilenetV1/heatmap_2/BiasAdd', 'index': 87,
  'shape': array([ 1,  9,  9, 17]),
  'shape_signature': array([ 1,  9,  9, 17]),
  'dtype': <class 'numpy.float32'>,
  'quantization': (0.0, 0),
  'quantization_parameters': {'scales': array([], dtype=float32),
  'zero_points': array([], dtype=int32), 'quantized_dimension': 0},
  'sparsity_parameters': {}},

  {'name': 'MobilenetV1/offset_2/BiasAdd', 'index': 90,
  'shape': array([ 1,  9,  9, 34]),
  'shape_signature': array([ 1,  9,  9, 34]),
  'dtype': <class 'numpy.float32'>,
  'quantization': (0.0, 0),
  'quantization_parameters': {'scales': array([], dtype=float32),
  'zero_points': array([], dtype=int32), 'quantized_dimension': 0},
  'sparsity_parameters': {}},
```

Hot Map 大致位置

查找最大值（$x,y$）

Offset Map 偏移量 （$dx,dy$）

在 W×H上的实际坐标

257×257

$$X_{pos} = x/w * W + dx$$

$$Y_{pos} = y/h * H + dy$$

```
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)
```

视频尺寸较大
为了方便显示
进行图像缩放

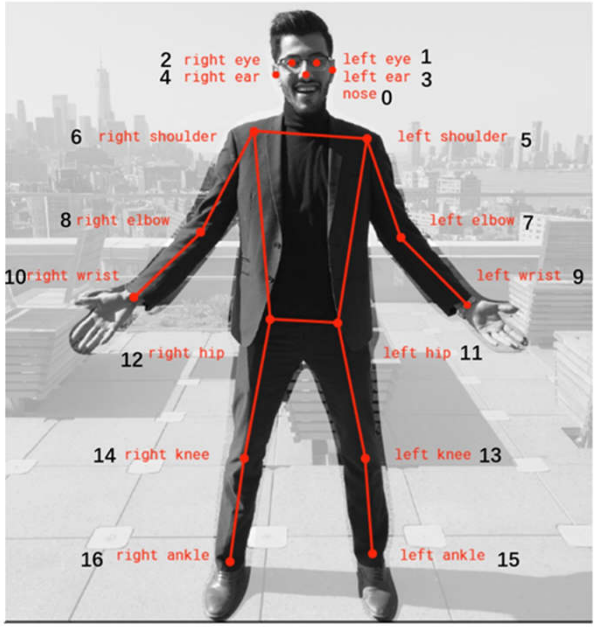
对像素值进行正则化
变为（-1,1）之间

```
# 进行关键点检测
interpreter.invoke()

# 获取hotmat
hotmaps = interpreter.get_tensor(output_details[0]['index'])[0] # Bounding bo

# 获取偏移量
offsets = interpreter.get_tensor(output_details[1]['index'])[0] # Class index

# 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]
```

获取第*i*个通道的最大值的位置（max_index）
即第*i*个关键点的大致区域

根据 max_index 从offset map中
获取偏移量（y在前，x在后）

```
    # 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)
```



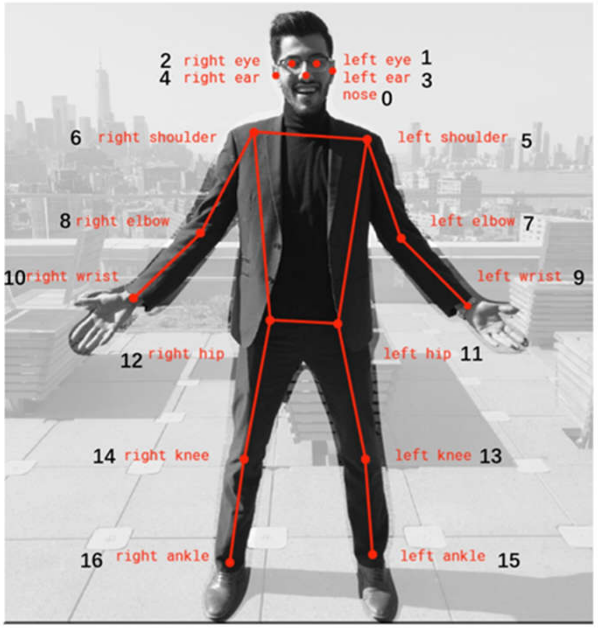
根据夹角规则，进行动作识别

```
        # 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),

     # 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()
```

```python
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
```

计算矢量之间的夹角

```python
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 = ""
```



设计动作规则

```
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
```