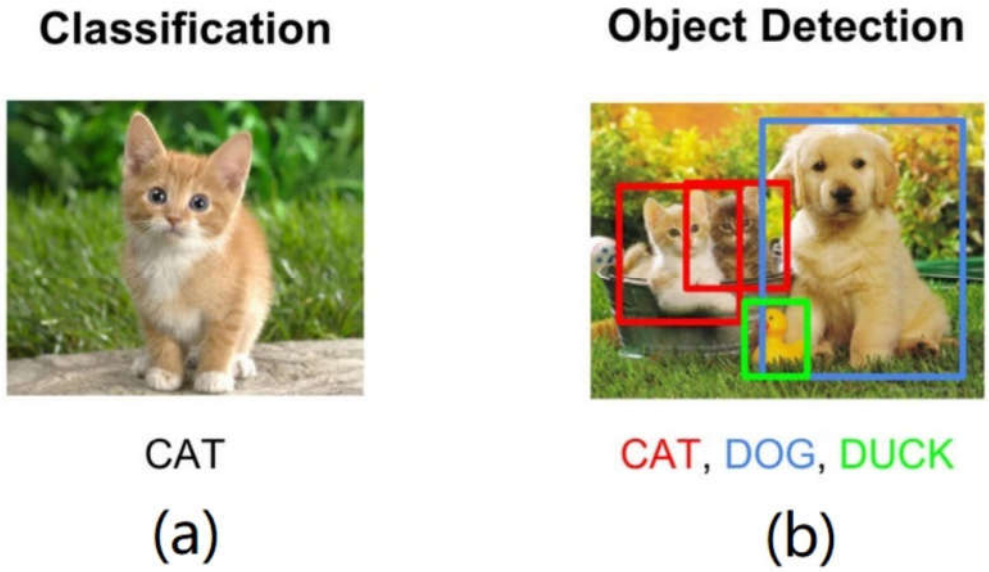
Python编程与人工智能实践

应用篇：基于tflite的目标检测 (Object Detection)

于泓
鲁东大学
信息与电气工程学院
2021.4.11

目标检测

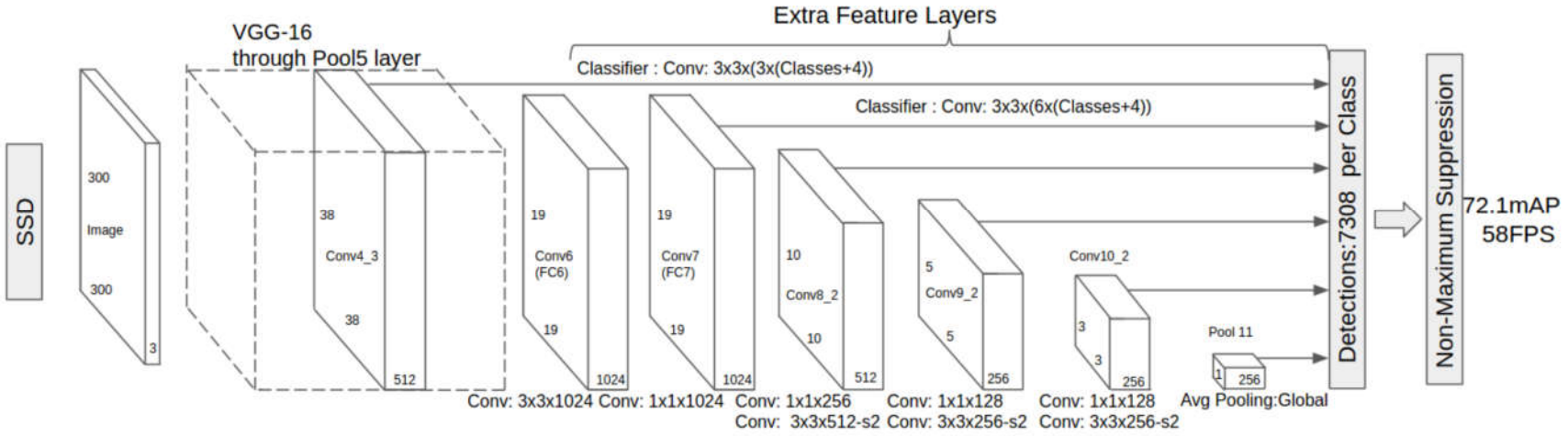
目标检测（Detection）任务关注特定的物体目标，要求同时获得这一目标的类别信息和位置信息。相比分类，检测给出的是对图片前景和背景的理解，我们需要从背景中分离出感兴趣的目标，并确定这一目标的描述（类别和位置），因而，检测模型的输出是一个列表，列表的每一项使用一个数据组给出检出目标的类别和位置。



目标检测算法

YOLO: You Only Look Once

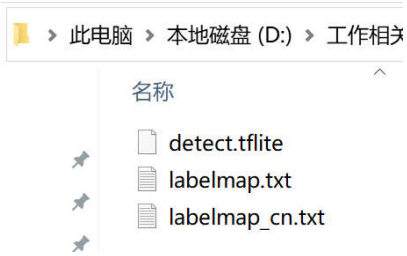
SSD： Single Shot Multibox Detector 采用算法



tflite中的SSD模型

“coco_ssd_mobilenet_v1”该模型也属于量化模型，使用MobileNet 神经网络进行图像深度特征提取，后端采用 SSD（Single Shot Multibox Detector）算法实现端到端的边界预测，该模型在微软提供的 COCO 数据上进行训练, 可以识别 90 类物体。

模型下载地址:
http://storage.googleapis.com/download.tensorflow.org/models/tflite/coco_ssd_mobilenet_v1_1.0_quant_2018_06_29.zip
COCO数据集
https://cocodataset.org/home



labelmap.txt

```
1  ???
2  person
3  bicycle
4  car
5  motorcycle
6  airplane
7  bus
8  train
9  truck
10 boat
11 traffic light
12 fire hydrant
13 ???
14 stop sign
15 parking meter
16 bench
17 bird
18 cat
19 dog
20 horse
```

labelmap_cn.txt

```
1  ???
2  人
3  自行车
4  汽车
5  摩托车
6  飞机
7  公交车
8  火车
9  卡车
10 船
11 红绿灯
12 消防栓
13 ???
14 停止标志
15 停车收费表
16 板凳
17 鸟
18 猫
19 狗
20 马
```

代码实现

```python
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  = 0.5

    # 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)
```

```
input_details
 [{'name': 'normalized_input_image_tensor',
  'index': 175, 'shape': array([  1, 300, 300,   3]),
  'shape_signature': array([  1, 300, 300,   3]),
  'dtype': <class 'numpy.uint8'>,
  'quantization': (0.0078125, 128),
  'quantization_parameters': {'scales': array([0.0078125], dtype=float32),
  'zero_points': array([128]),
  'quantized_dimension': 0}, 'sparsity_parameters': {}}]
```

```
output_details
[{'name': 'TFLite_Detection_PostProcess',
  'index': 167, 'shape': array([ 1, 10,  4]),
   'shape_signature': array([ 1, 10,  4]),
   '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': 'TFLite_Detection_PostProcess:1',
   'index': 168, 'shape': array([ 1, 10]), 'shape_signature': array([ 1, 10]),
   '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': 'TFLite_Detection_PostProcess:2',
   'index': 169, 'shape': array([ 1, 10]), 'shape_signature': array([ 1, 10]),
   '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': 'TFLite_Detection_PostProcess:3', 'index': 170, 'shape': array([1]),
   'shape_signature': array([1]), 'dtype': <class 'numpy.float32'>,
   'quantization': (0.0, 0), 'quantization_parameters': {'scales': array([], dtype=float32),
   'zero_points': array([], dtype=int32), 'quantized_dimension': 0}, 'sparsity_parameters': {}}]
```

10个目标的边界

10个目标的类别

10个目标的置信度（概率）

置信度最大目标的类别

```
# 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
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)
```

获取网络输入
需要图像的大小

```
input_details
[{'name': 'normalized_input_image_tensor',
  'index': 175, 'shape': array([  1, 300, 300,   3]),
  'shape_signature': array([  1, 300, 300,   3]),
  'dtype': <class 'numpy.uint8'>,
  'quantization': (0.0078125, 128),
  'quantization_parameters': {'scales': array([0.0078125], dtype=float32),
  'zero_points': array([128]),
  'quantized_dimension': 0}, 'sparsity_parameters': {}}]
```

图像缩放，满足网络需求

```
# 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()
```