YOLOv3目标检测算法

模型预测

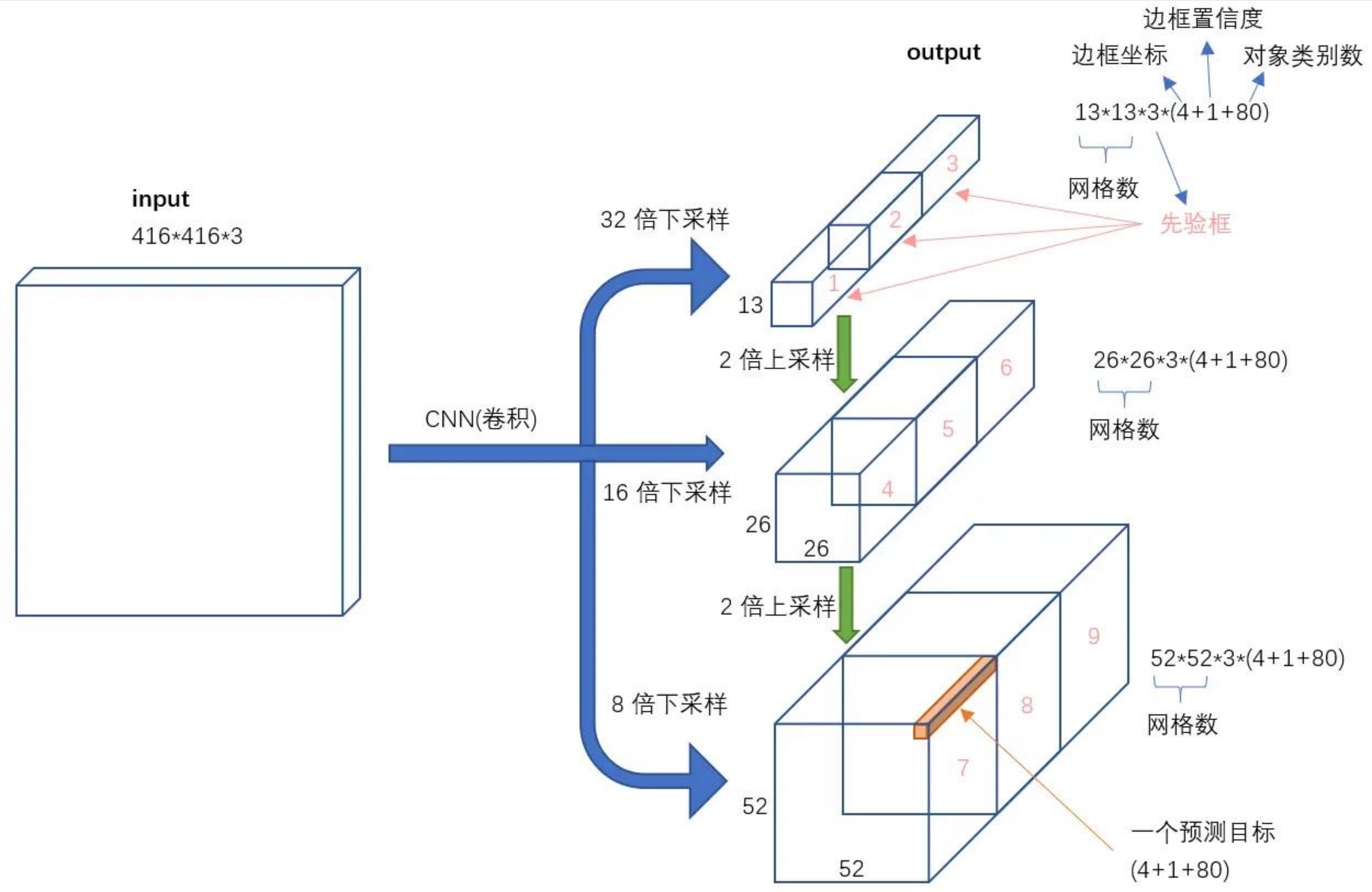
@tm9161

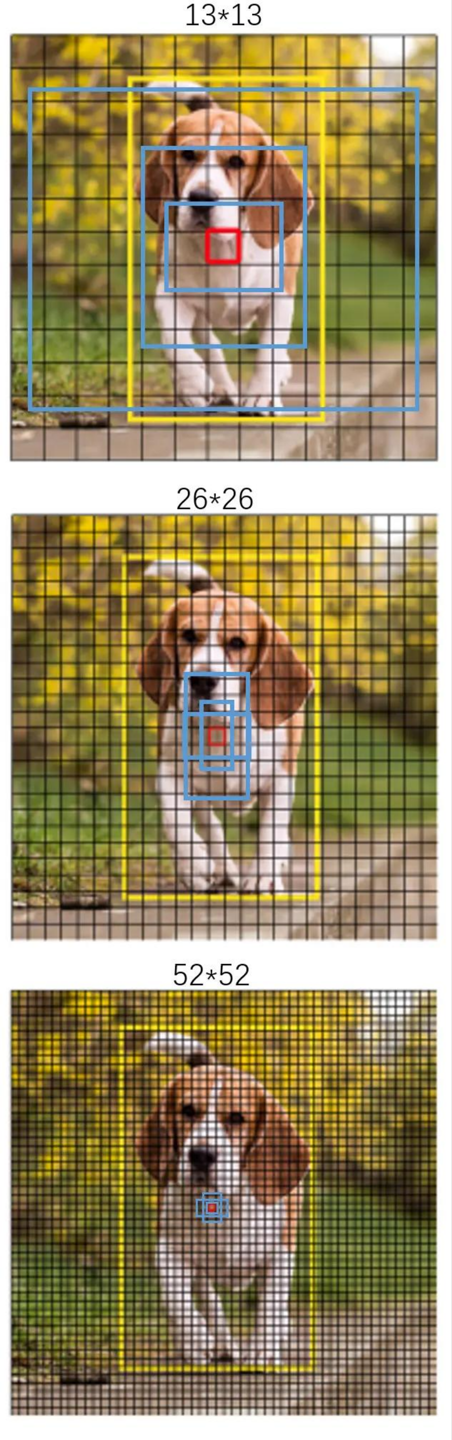
L4

模型预测

1. YOLO v3网络结构
2. 图像预处理
3. 模型预测
4. 绘制预测框

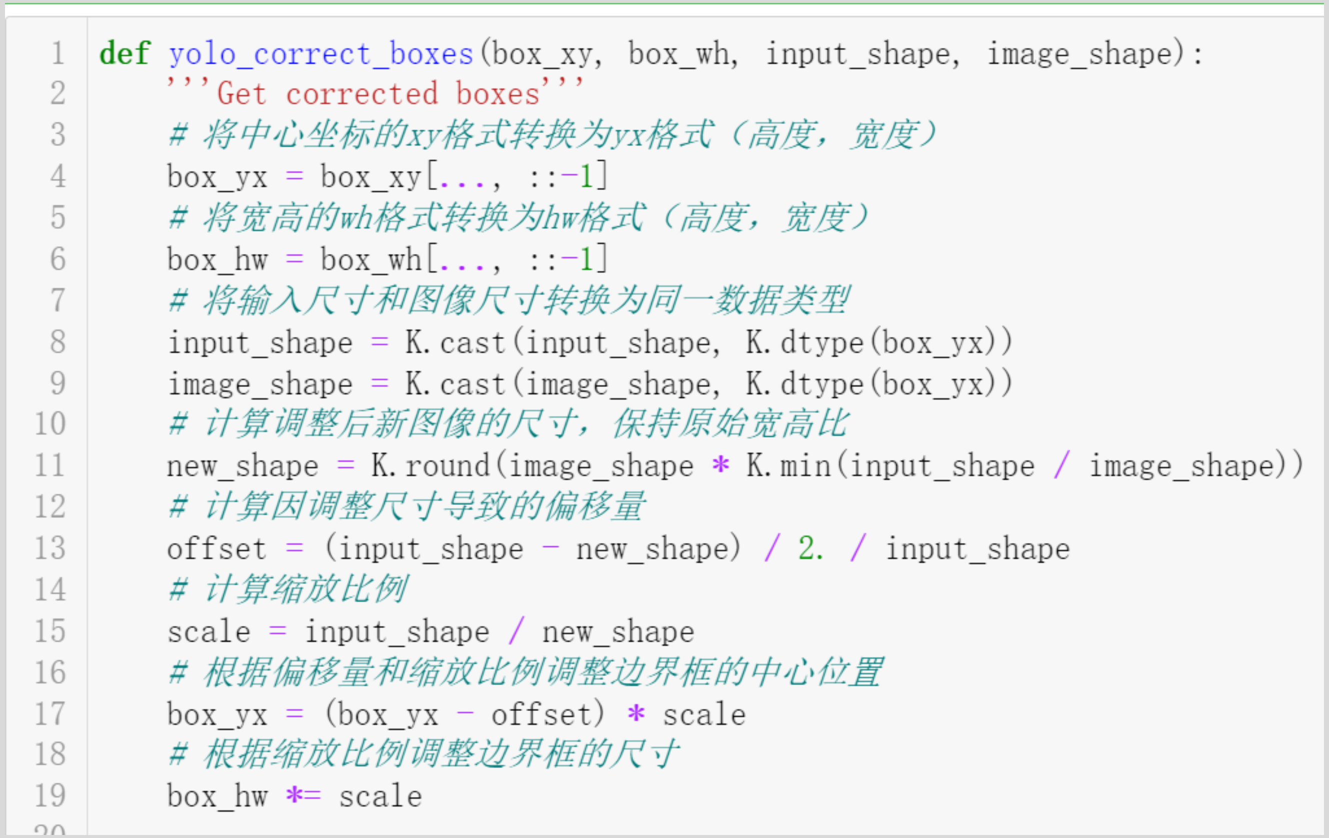
YOLO V3



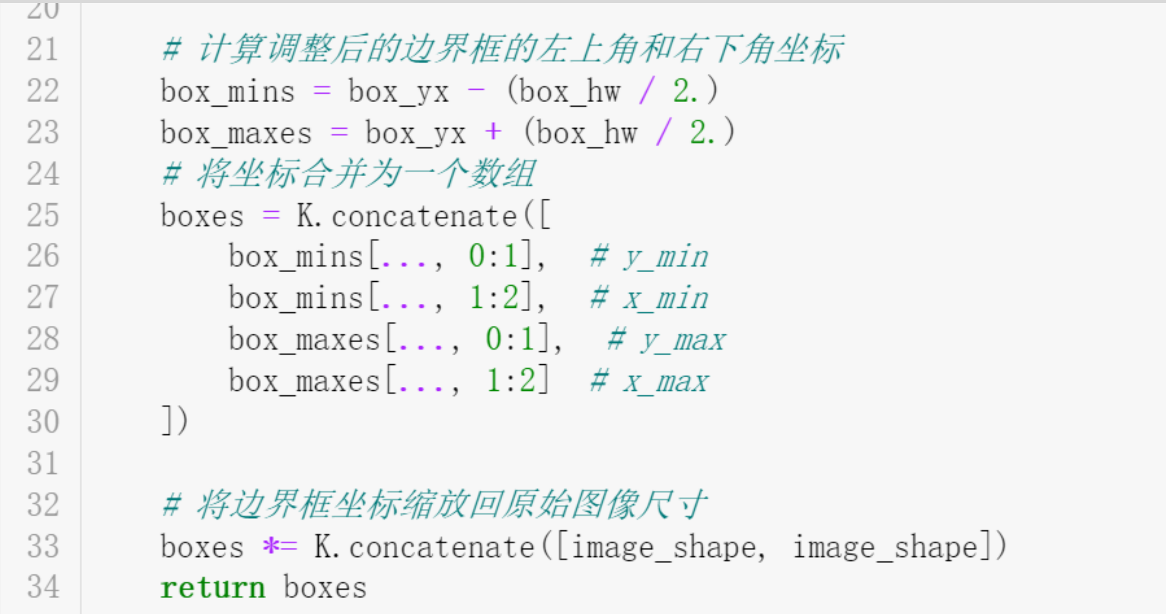


图片来源：https://www.jianshu.com/p/d13ae1055302?tdsourcetag=s_pcqq_aiomsg

图像预处理：边界框调整

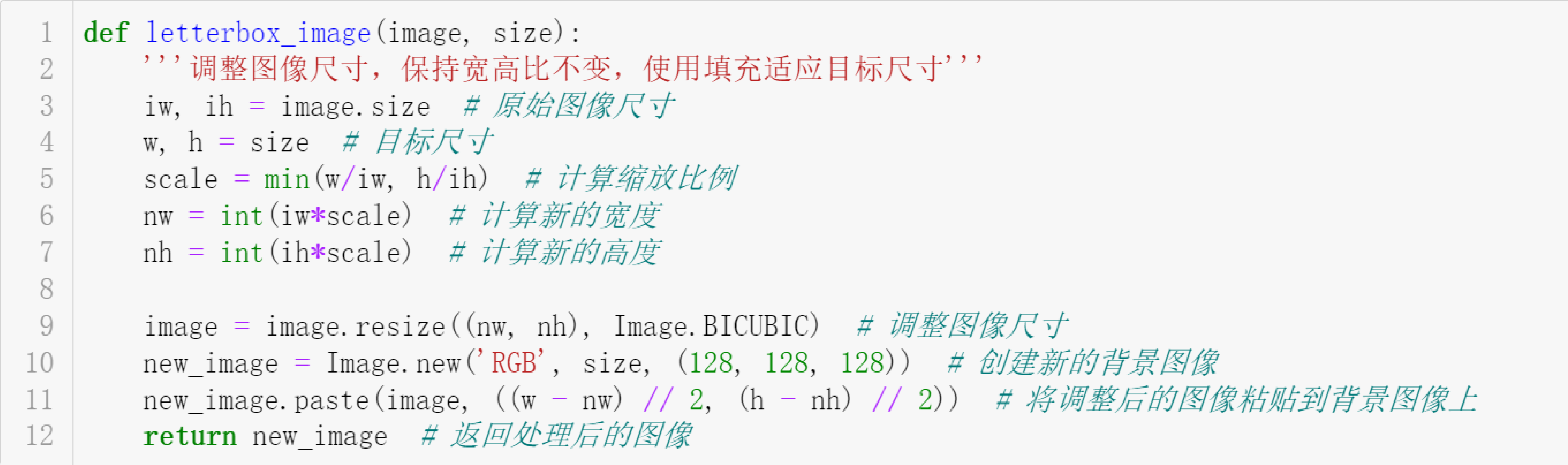


```python
def yolo_correct_boxes(box_xy, box_wh, input_shape, image_shape):
    '''Get corrected boxes'''
    # 将中心坐标的xy格式转换为yx格式（高度，宽度）
    box_yx = box_xy[..., ::-1]
    # 将宽高的wh格式转换为hw格式（高度，宽度）
    box_hw = box_wh[..., ::-1]
    # 将输入尺寸和图像尺寸转换为同一数据类型
    input_shape = K.cast(input_shape, K.dtype(box_yx))
    image_shape = K.cast(image_shape, K.dtype(box_yx))
    # 计算调整后新图像的尺寸，保持原始宽高比
    new_shape = K.round(image_shape * K.min(input_shape / image_shape))
    # 计算因调整尺寸导致的偏移量
    offset = (input_shape - new_shape) / 2. / input_shape
    # 计算缩放比例
    scale = input_shape / new_shape
    # 根据偏移量和缩放比例调整边界框的中心位置
    box_yx = (box_yx - offset) * scale
    # 根据缩放比例调整边界框的尺寸
    box_hw *= scale
```



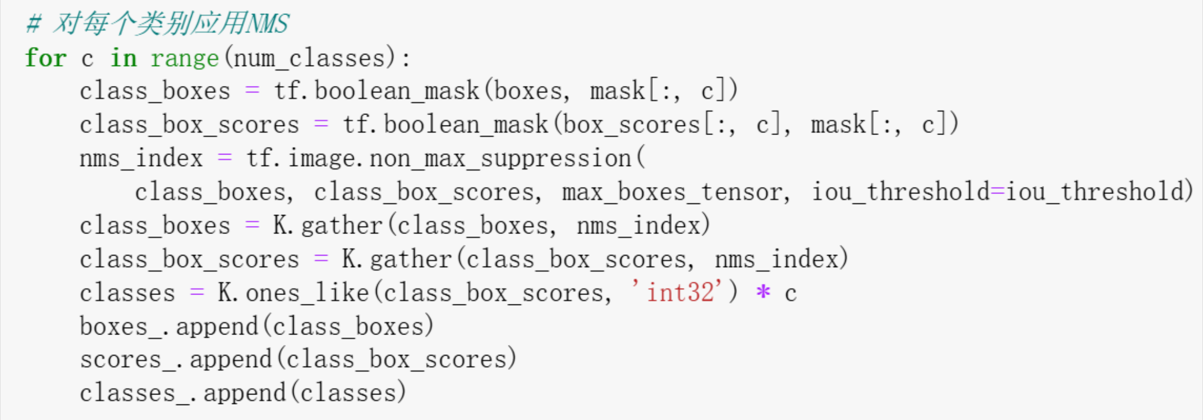
```python
    # 计算调整后的边界框的左上角和右下角坐标
    box_mins = box_yx - (box_hw / 2.)
    box_maxes = box_yx + (box_hw / 2.)
    # 将坐标合并为一个数组
    boxes = K.concatenate([
        box_mins[..., 0:1],  # y_min
        box_mins[..., 1:2],  # x_min
        box_maxes[..., 0:1],  # y_max
        box_maxes[..., 1:2]  # x_max
    ])

    # 将边界框坐标缩放回原始图像尺寸
    boxes *= K.concatenate([image_shape, image_shape])
    return boxes
```

图像预处理：保持宽高比&填充



```python
def letterbox_image(image, size):
    '''调整图像尺寸，保持宽高比不变，使用填充适应目标尺寸'''
    iw, ih = image.size  # 原始图像尺寸
    w, h = size  # 目标尺寸
    scale = min(w/iw, h/ih)  # 计算缩放比例
    nw = int(iw*scale)  # 计算新的宽度
    nh = int(ih*scale)  # 计算新的高度

    image = image.resize((nw, nh), Image.BICUBIC)  # 调整图像尺寸
    new_image = Image.new('RGB', size, (128, 128, 128))  # 创建新的背景图像
    new_image.paste(image, ((w - nw) // 2, (h - nh) // 2))  # 将调整后的图像粘贴到背景图像上
    return new_image  # 返回处理后的图像
```

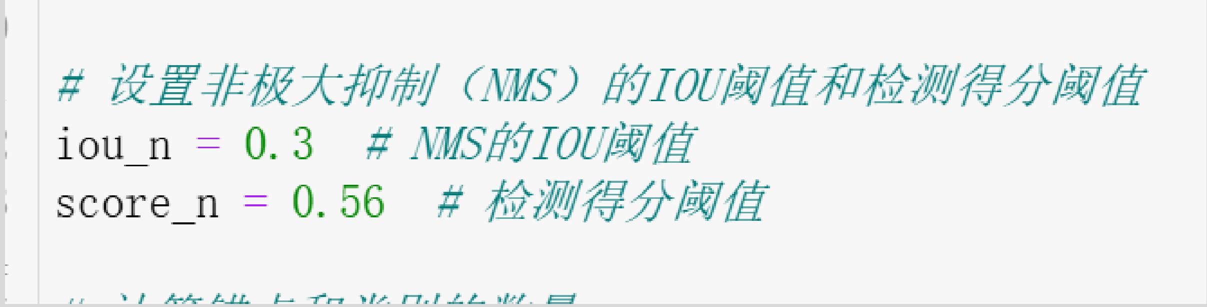
非极大值抑制:



```
# 对每个类别应用NMS
for c in range(num_classes):
    class_boxes = tf.boolean_mask(boxes, mask[:, c])
    class_box_scores = tf.boolean_mask(box_scores[:, c], mask[:, c])
    nms_index = tf.image.non_max_suppression(
        class_boxes, class_box_scores, max_boxes_tensor, iou_threshold=iou_threshold)
    class_boxes = K.gather(class_boxes, nms_index)
    class_box_scores = K.gather(class_box_scores, nms_index)
    classes = K.ones_like(class_box_scores, 'int32') * c
    boxes_.append(class_boxes)
    scores_.append(class_box_scores)
    classes_.append(classes)
```

目标：保留最好的框 去掉重复和重叠度过高的框

阈值：重合度+置信度



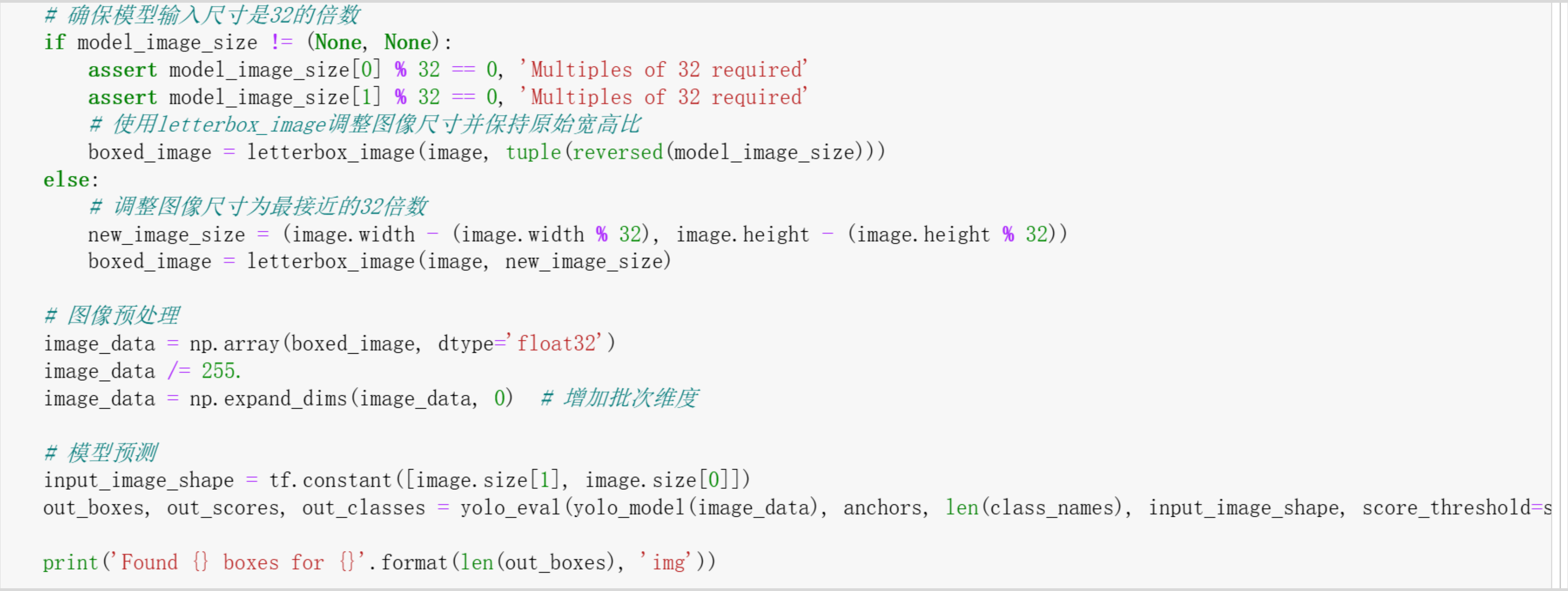
```
# 设置非极大抑制（NMS）的IOU阈值和检测得分阈值
iou_n = 0.3  # NMS的IOU阈值
score_n = 0.56  # 检测得分阈值

# 计算锚点和类别的数量
```

模型预测

1. 检查尺寸

2. 图像预处理

3. 模型预测

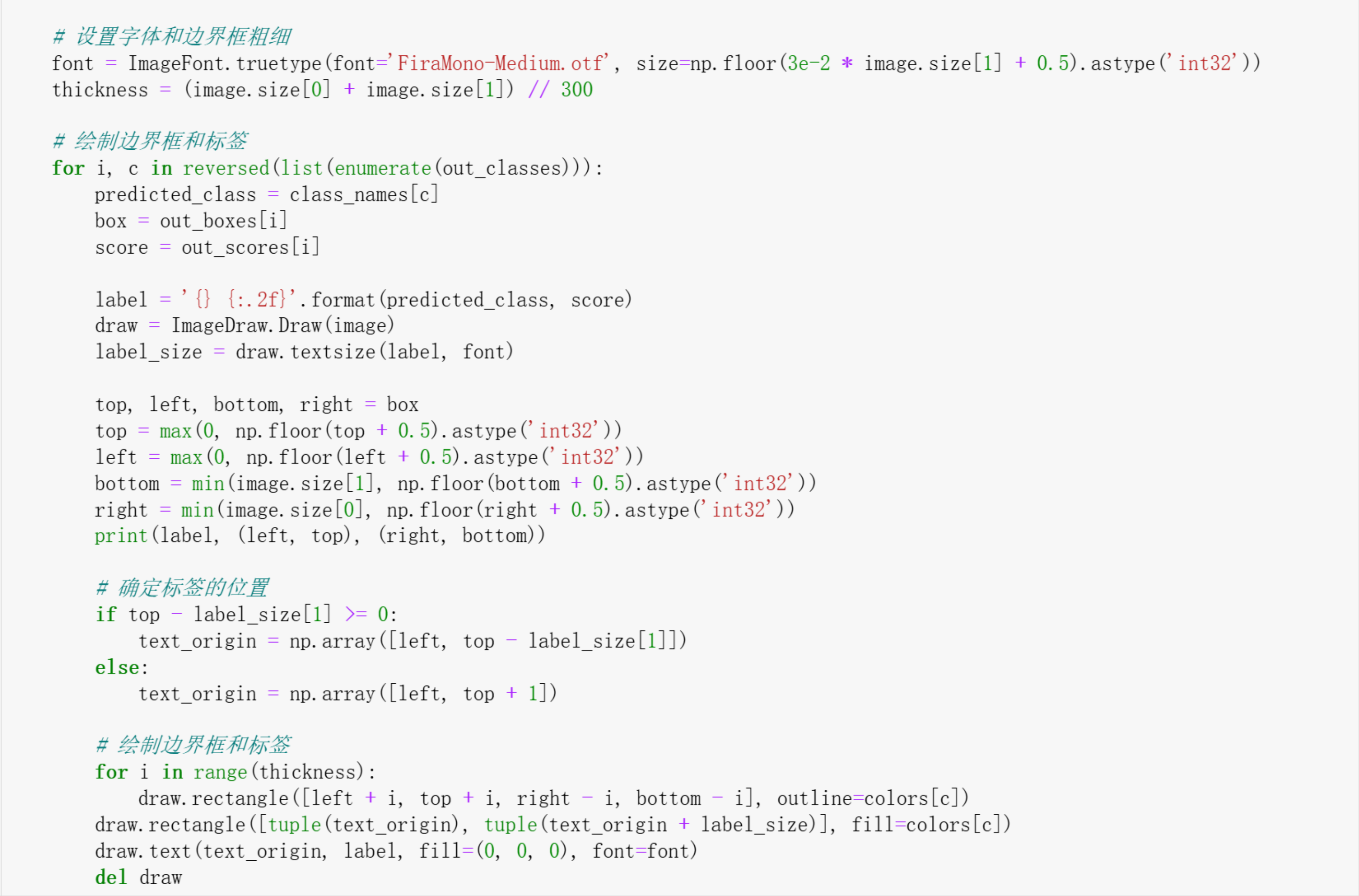


```
# 确保模型输入尺寸是32的倍数
if model_image_size != (None, None):
    assert model_image_size[0] % 32 == 0, 'Multiples of 32 required'
    assert model_image_size[1] % 32 == 0, 'Multiples of 32 required'
    # 使用letterbox_image调整图像尺寸并保持原始宽高比
    boxed_image = letterbox_image(image, tuple(reversed(model_image_size)))
else:
    # 调整图像尺寸为最接近的32倍数
    new_image_size = (image.width - (image.width % 32), image.height - (image.height % 32))
    boxed_image = letterbox_image(image, new_image_size)

# 图像预处理
image_data = np.array(boxed_image, dtype='float32')
image_data /= 255.
image_data = np.expand_dims(image_data, 0)  # 增加批次维度

# 模型预测
input_image_shape = tf.constant([image.size[1], image.size[0]])
out_boxes, out_scores, out_classes = yolo_eval(yolo_model(image_data), anchors, len(class_names), input_image_shape, score_threshold=s

print('Found {} boxes for {}'.format(len(out_boxes), 'img'))
```

绘制预测框

绘制边界框

绘制标签

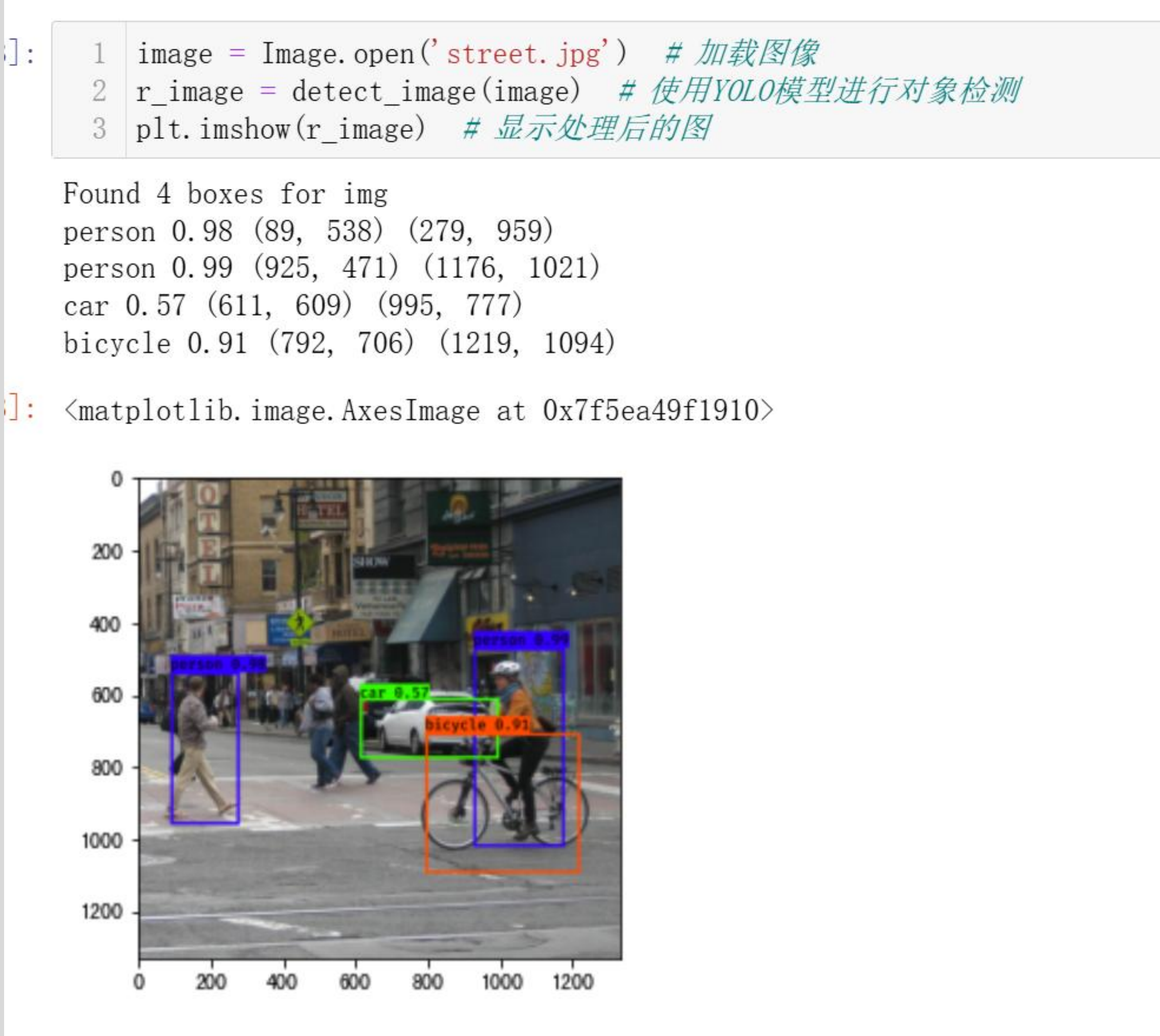


```python
# 设置字体和边界框粗细
font = ImageFont.truetype(font='FiraMono-Medium.otf', size=np.floor(3e-2 * image.size[1] + 0.5).astype('int32'))
thickness = (image.size[0] + image.size[1]) // 300

# 绘制边界框和标签
for i, c in reversed(list(enumerate(out_classes))):
    predicted_class = class_names[c]
    box = out_boxes[i]
    score = out_scores[i]

    label = '{} {:.2f}'.format(predicted_class, score)
    draw = ImageDraw.Draw(image)
    label_size = draw.textsize(label, font)

    top, left, bottom, right = box
    top = max(0, np.floor(top + 0.5).astype('int32'))
    left = max(0, np.floor(left + 0.5).astype('int32'))
    bottom = min(image.size[1], np.floor(bottom + 0.5).astype('int32'))
    right = min(image.size[0], np.floor(right + 0.5).astype('int32'))
    print(label, (left, top), (right, bottom))

    # 确定标签的位置
    if top - label_size[1] >= 0:
        text_origin = np.array([left, top - label_size[1]])
    else:
        text_origin = np.array([left, top + 1])

    # 绘制边界框和标签
    for i in range(thickness):
        draw.rectangle([left + i, top + i, right - i, bottom - i], outline=colors[c])
    draw.rectangle([tuple(text_origin), tuple(text_origin + label_size)], fill=colors[c])
    draw.text(text_origin, label, fill=(0, 0, 0), font=font)
    del draw
```

结果输出

加载图像

输出预测结果



```
image = Image.open('street.jpg')   # 加载图像
r_image = detect_image(image)   # 使用YOLO模型进行对象检测
plt.imshow(r_image)   # 显示处理后的图
```

```
Found 4 boxes for img
person 0.98 (89, 538) (279, 959)
person 0.99 (925, 471) (1176, 1021)
car 0.57 (611, 609) (995, 777)
bicycle 0.91 (792, 706) (1219, 1094)
```

```
<matplotlib.image.AxesImage at 0x7f5ea49f1910>
```

参考资料：

1. 3.1 YOLO系列理论合集(YOLOv1~v3)
https://www.bilibili.com/video/BV1yi4y1g7ro

2. AaronJny/tf2-keras-yolo3
https://github.com/AaronJny/tf2-keras-yolo3

3.大语言模型