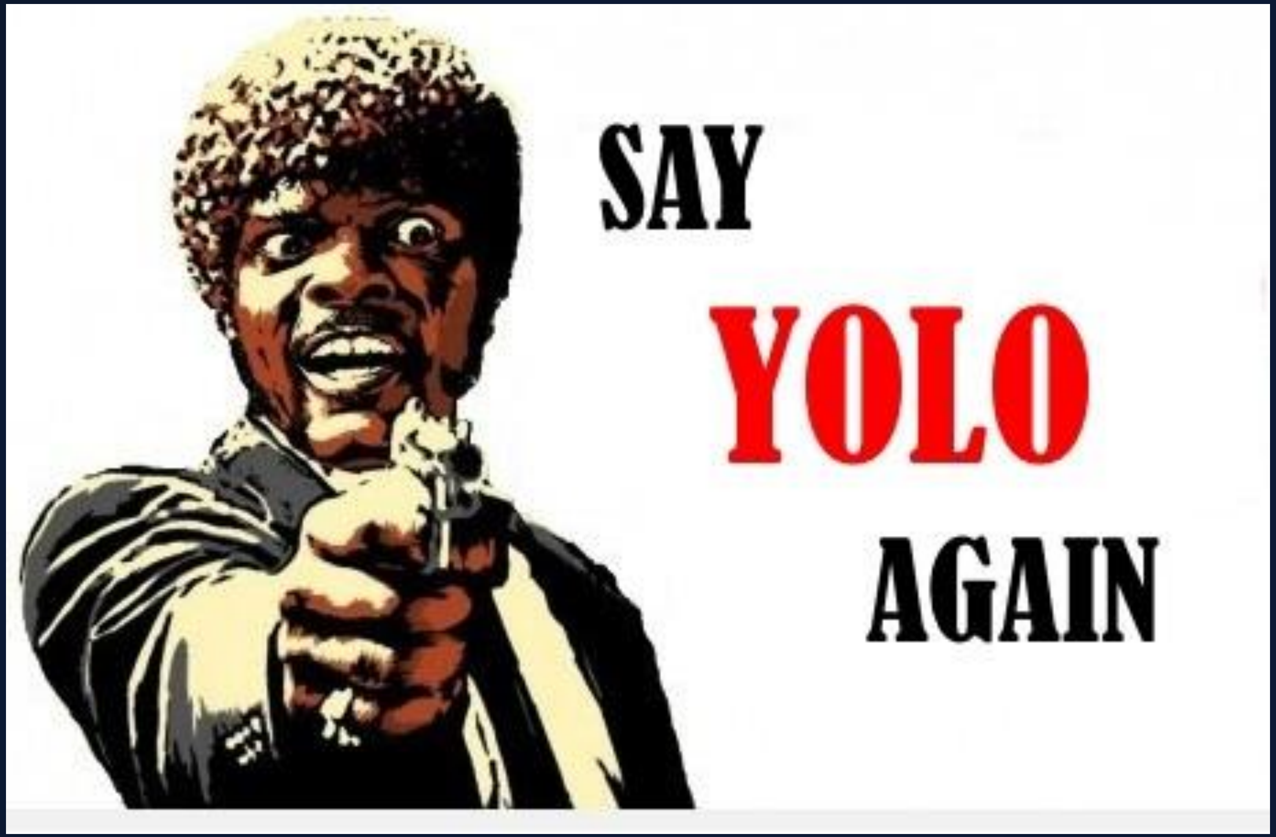


Joseph Redmon

Yolov1

Yolov2

Yolov3

Alexey Bochkovskiy

Yolov4

ultralytics

Yolov5

Yolo网络讲解

https://github.com/YunYang1994/tensorflow-yolov3

CONTENTS

1

测试

1.输入
2.网络结构
3.输出
4.解码
5.后处理与非极大值抑制(NMS)

2

训练

1.标签
2.置信度损失
3.定位损失
4.类别概率损失


测试

1 输入

1. 长宽相等的正方形
2. 32的倍数

```
ih, iw    = target_size
h,  w, _  = image.shape

scale = min(iw/w, ih/h)
nw, nh  = int(scale * w), int(scale * h)
image_resized = cv2.resize(image, (nw, nh))

image_paded = np.full(shape=[ih, iw, 3], fill_value=128.0)
dw, dh = (iw - nw) // 2, (ih-nh) // 2
image_paded[dh:nh+dh, dw:nw+dw, :] = image_resized
image_paded = image_paded / 255.
```

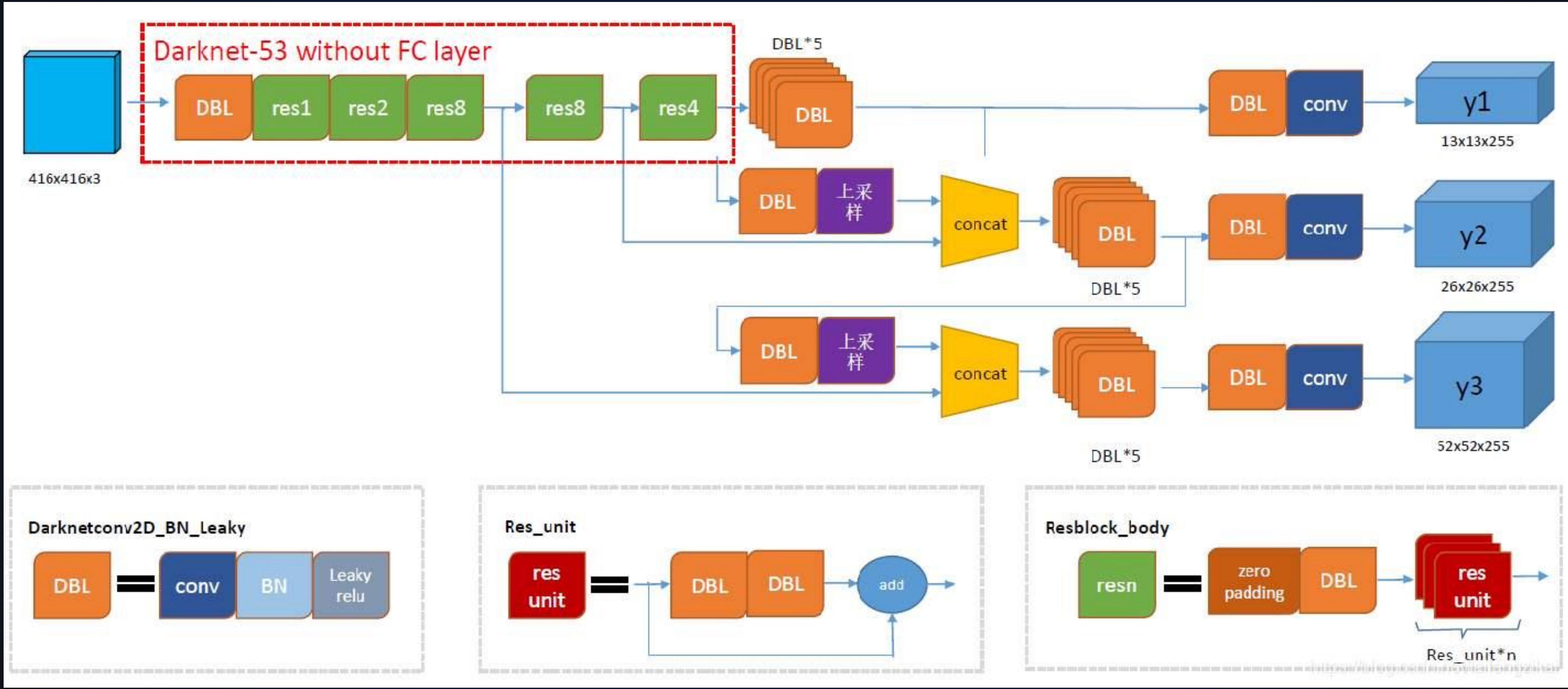
1 输入

Data augmentation

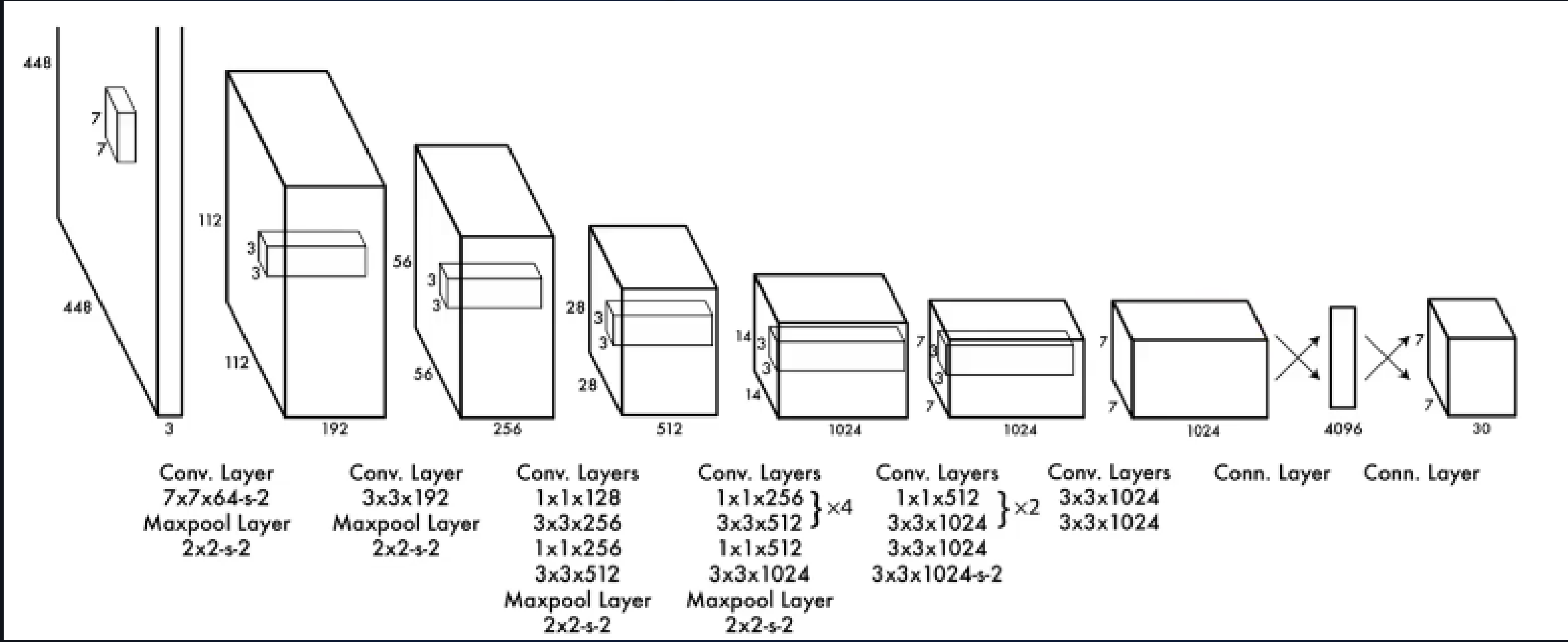
mosaic

2

网络结构

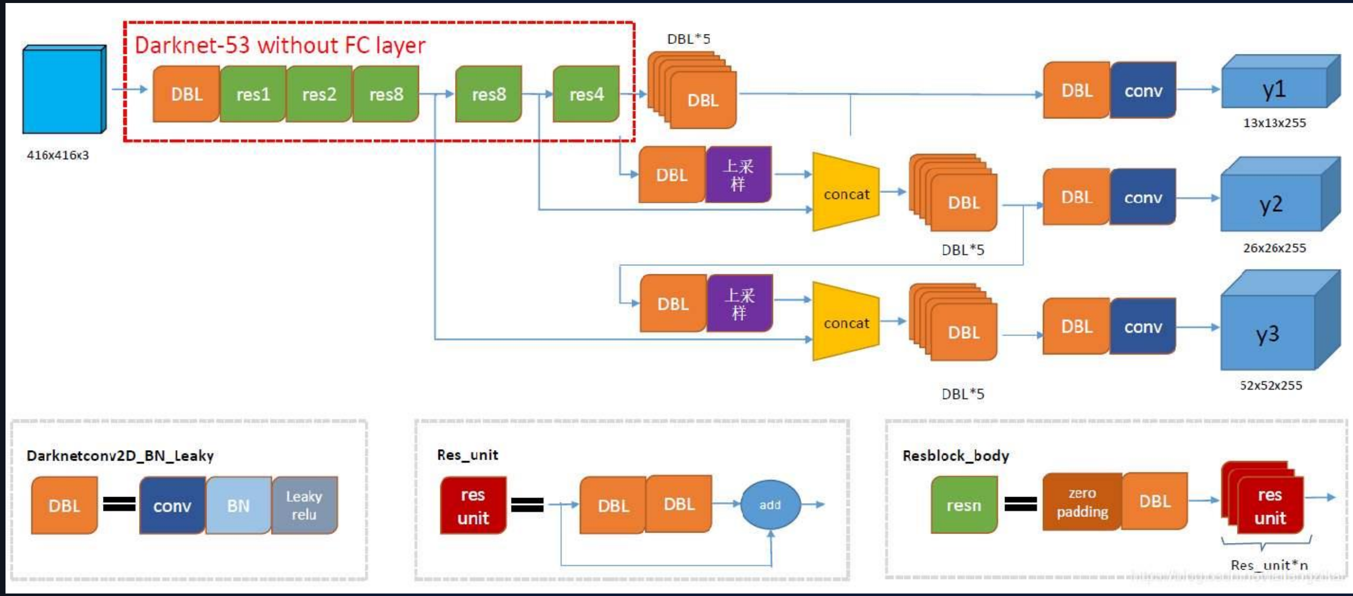

Darknet-53 without FC layer
416x416x3
DBL
res1
res2
res8
res8
res4
DBL*5
DBL
DBL
上采样
concat
DBL
DBL*5
DBL
上采样
concat
DBL
DBL*5
DBL
conv
y1
13x13x255
DBL
conv
y2
26x26x255
DBL
conv
y3
52x52x255
Darknetconv2D_BN_Leaky
DBL = conv BN Leaky relu
Res_unit
res unit = DBL DBL add
Resblock_body
resn = zero padding DBL res unit
Res_unit*n

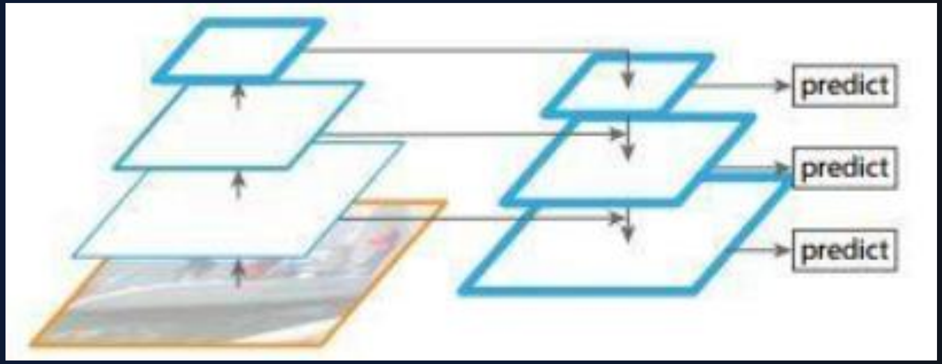
2 网络结构



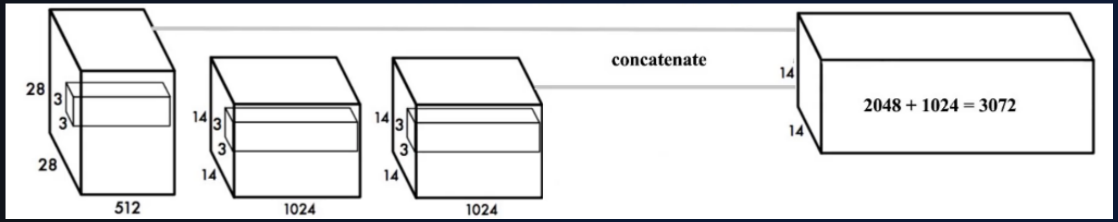
448
7
7
448
3
112
3
3
112
192
56
3
3
56
256
28
3
3
28
512
14
3
3
14
1024
7
3
3
7
1024
7
7
1024
4096
7
7
30
Conv. Layer
7x7x64-s-2
Maxpool Layer
2x2-s-2
Conv. Layer
3x3x192
Maxpool Layer
2x2-s-2
Conv. Layers
1x1x128
3x3x256
1x1x256
3x3x512
Maxpool Layer
2x2-s-2
Conv. Layers
1x1x256
3x3x512
×4
1x1x512
3x3x1024
Maxpool Layer
2x2-s-2
Conv. Layers
1x1x512
3x3x1024
×2
3x3x1024
3x3x1024-s-2
Conv. Layers
3x3x1024
3x3x1024
Conn. Layer
Conn. Layer

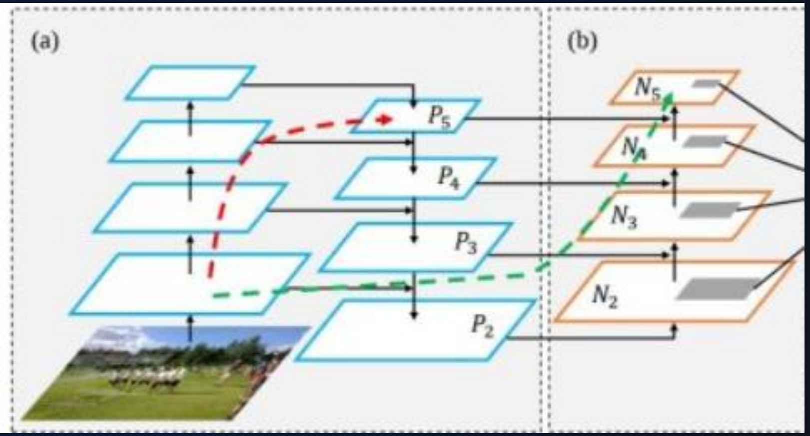
2 网络结构





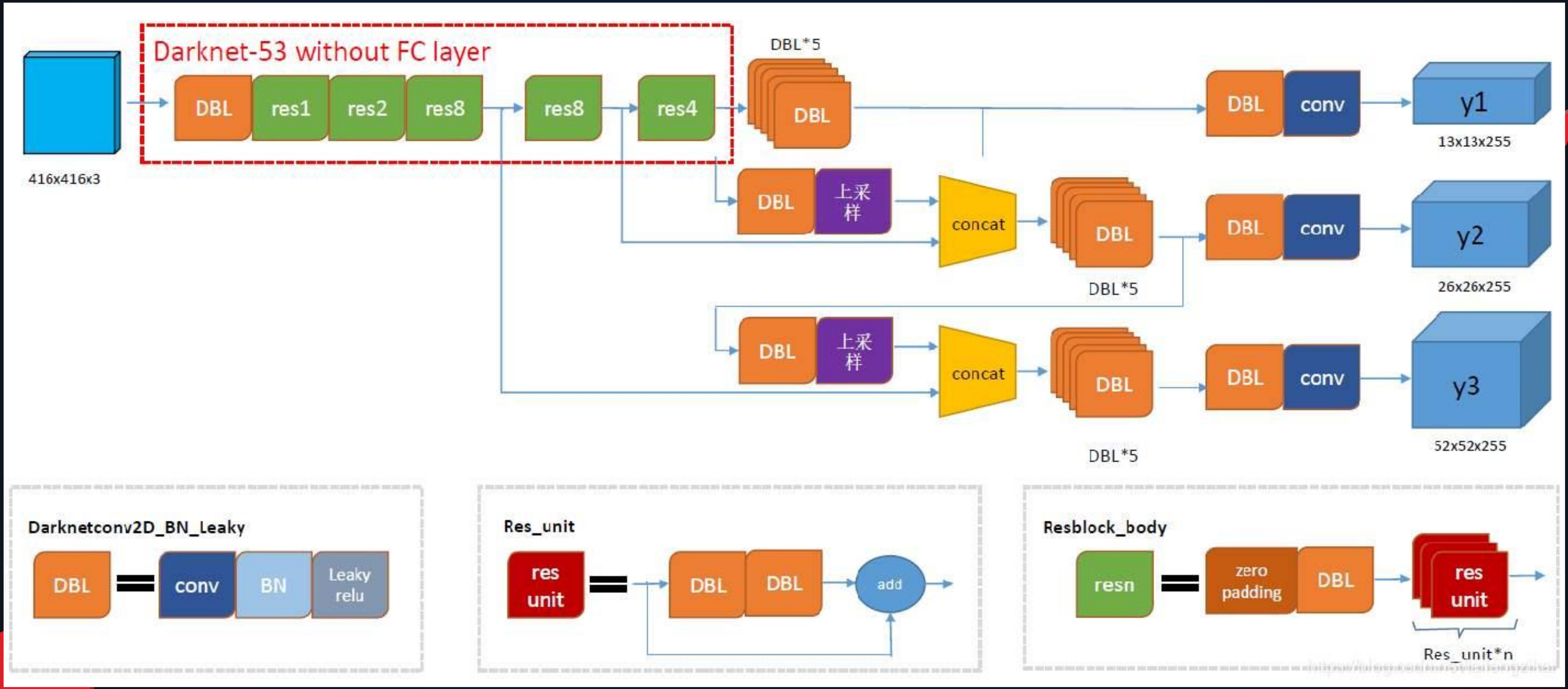
FPN (Feature Pyramid Networks)





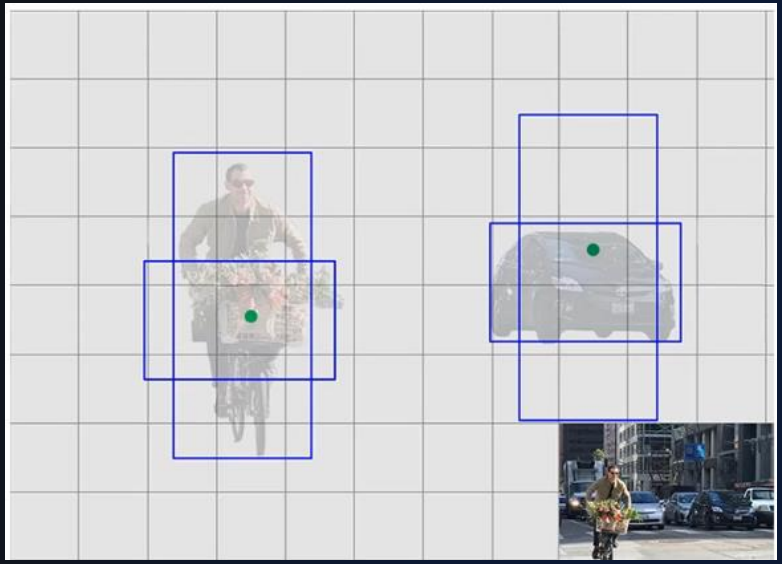
PAN(Path Aggregation Network)

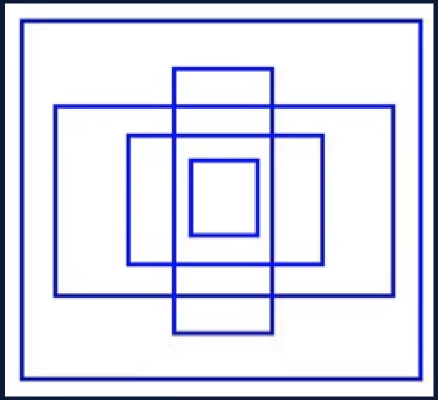
3 输出

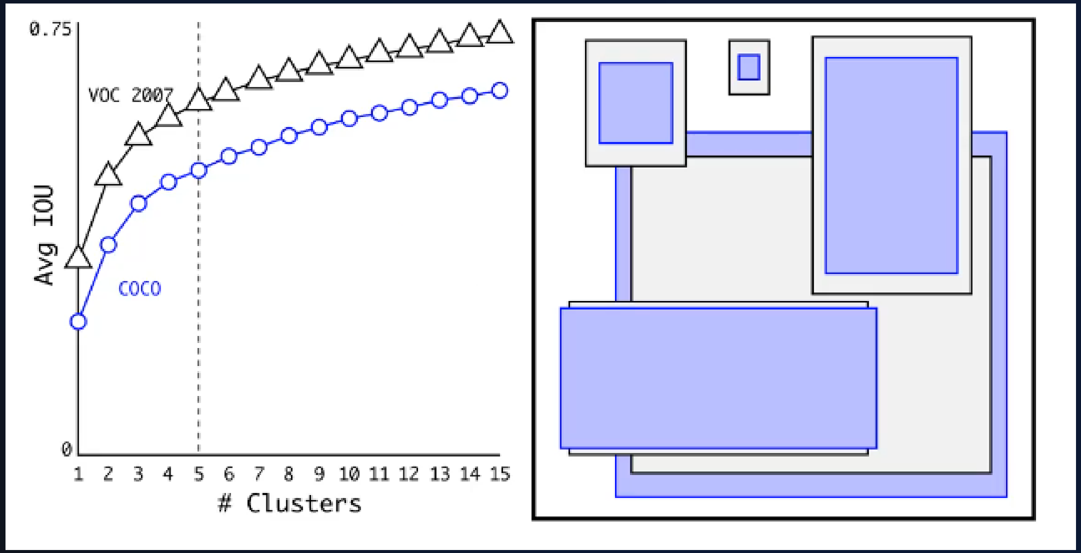


Darknet-53 without FC layer
416x416x3
DBL
res1
res2
res8
res8
res4
DBL*5
DBL
DBL
conv
y1
13x13x255
DBL
上采样
concat
DBL
DBL*5
DBL
conv
y2
26x26x255
DBL
上采样
concat
DBL
DBL*5
DBL
conv
y3
52x52x255
Darknetconv2D_BN_Leaky
DBL
conv
BN
Leaky relu
Res_unit
res unit
DBL
DBL
add
Resblock_body
resn
zero padding
DBL
res unit
Res_unit*n

3 输出







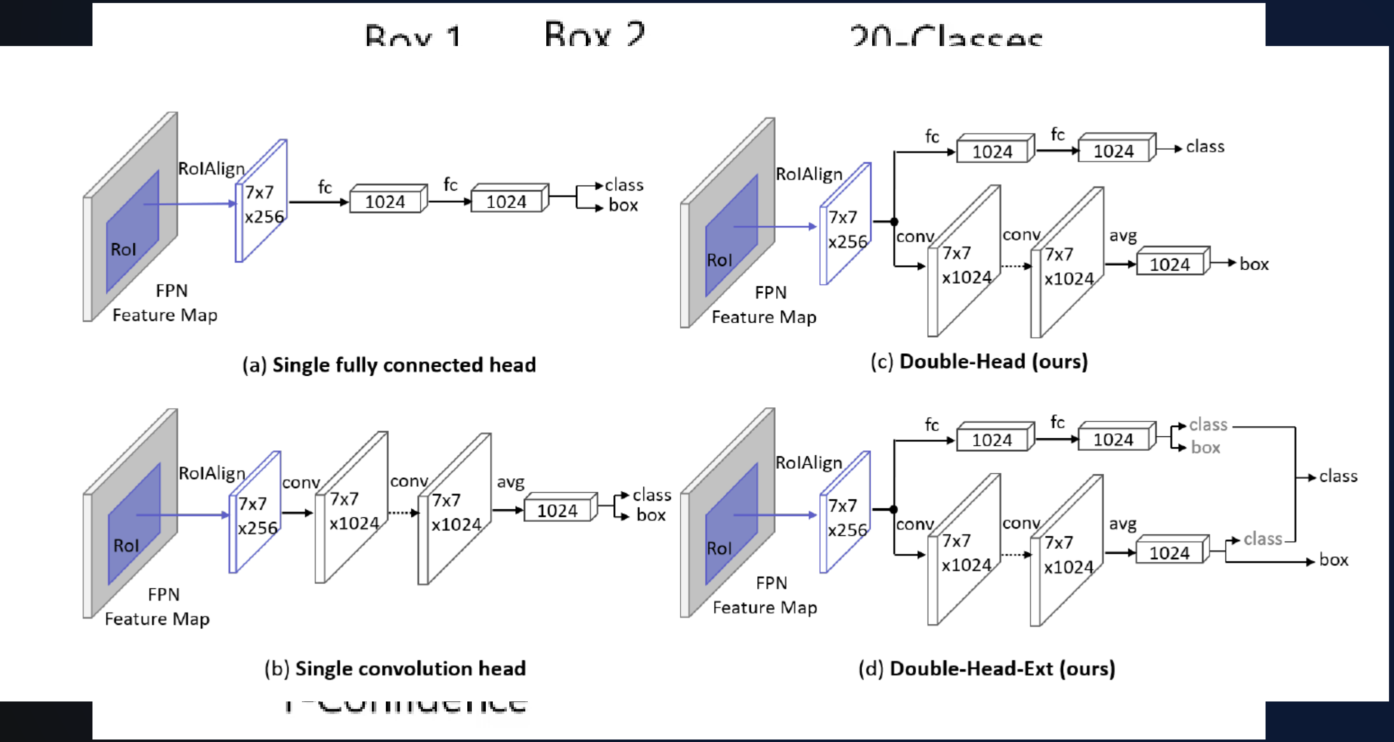
通道数

3*(num_class +5).

3 输出

类别数

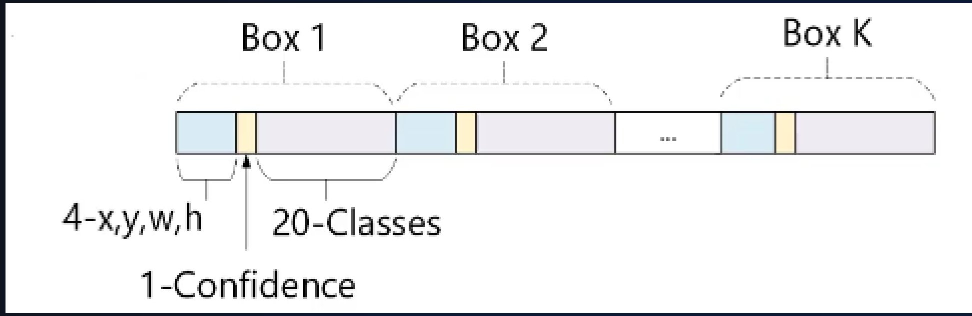
t_x, t_y, t_w, t_h和原始置信度



通道数

3*(num_class+5).

4 解码



$bx=(sigmoid(t_x)+cx)*stride$ (1)

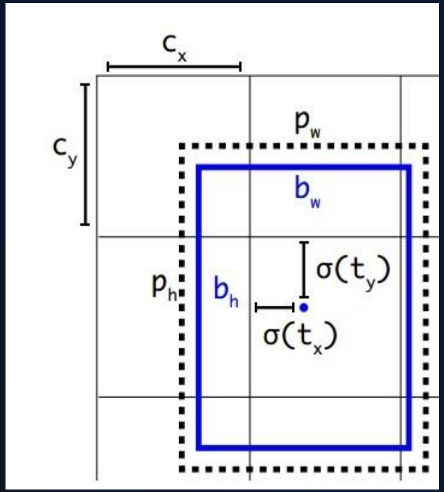
$by=(sigmoid(t_y)+cy)*stride$ (2)

$bw=p_w e^{tw}*stride$ (3)

$bh=p_h e^{th}*stride$ (4)

$conf=sigmoid(raw\_conf)$ (5)

$prob=sigmoid(raw\_prob)$ (6)



b_x、b_y、b_h、b_w——中心横纵坐标与高宽
p_h和p_w——先验框的高和宽
t_x和t_y——物体中心距离网格左上角位置的预测偏移量
t_w和t_h——物体相对于先验框的预测偏移量
c_x和c_y——网格左上角的坐标
Stride——最后特征图缩放的比例

5 后处理与NMS

1. 还原

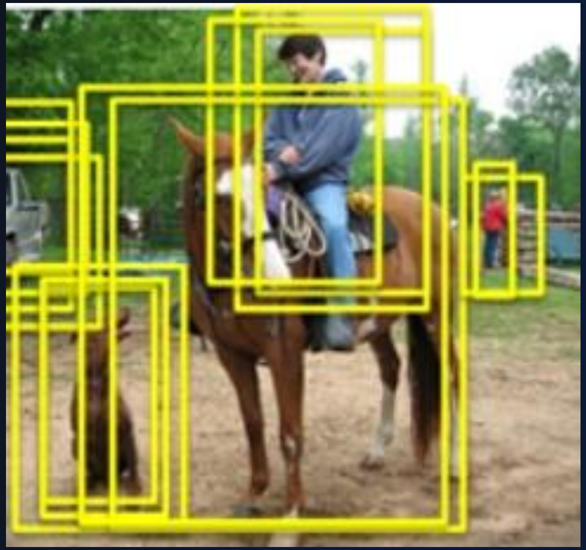
2. 筛选

(52*52+26*26+13*13)*3 = 10647个

类别置信度(Score) = 置信度(Conf)*类别概率(Prob)< threshold

`core.utils.postprocess_boxes`

5 后处理与NMS



Diou、Ciou

Fast NMS、Cluster NMS、Matrix NMS

```
classes_in_img = list(set(bboxes[:, 5]))
best_bboxes = []

for cls in classes_in_img:
    cls_mask = (bboxes[:, 5] == cls)
    cls_bboxes = bboxes[cls_mask]

    while len(cls_bboxes) > 0:
        max_ind = np.argmax(cls_bboxes[:, 4])
        best_bbox = cls_bboxes[max_ind]
        best_bboxes.append(best_bbox)
        cls_bboxes = np.concatenate([cls_bboxes[: max_ind], cls_bboxes[max_ind + 1:]])
        iou = bboxes_iou(best_bbox[np.newaxis, :4], cls_bboxes[:, :4])
        weight = np.ones((len(iou),), dtype=np.float32)

        assert method in ['nms', 'soft-nms']

        if method == 'nms':
            iou_mask = iou > iou_threshold
            weight[iou_mask] = 0.0

        if method == 'soft-nms':
            weight = np.exp(-(1.0 * iou ** 2 / sigma))

        cls_bboxes[:, 4] = cls_bboxes[:, 4] * weight
        score_mask = cls_bboxes[:, 4] > 0.
        cls_bboxes = cls_bboxes[score_mask]
```

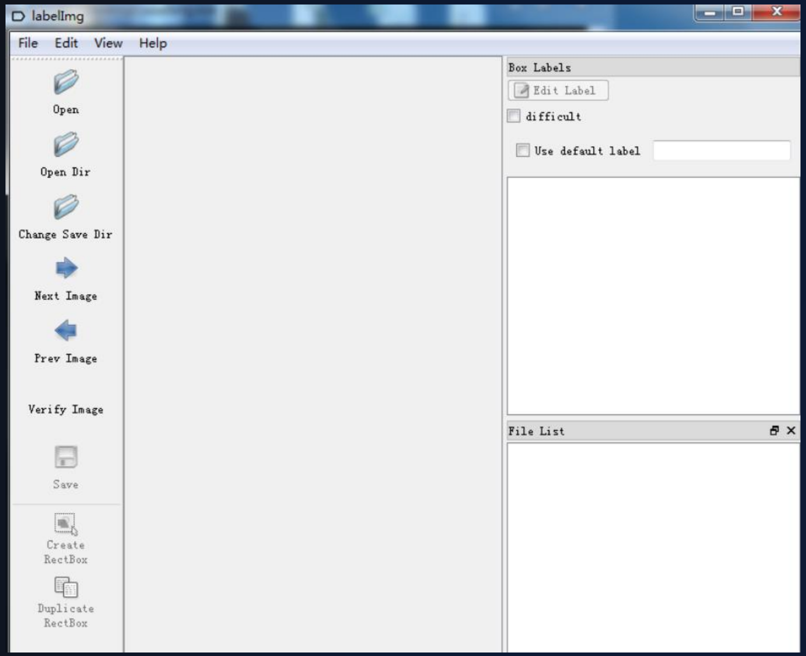
图像预处理 → 网络输出偏移值 → 解码

↓

后处理与非极大值抑制

训练

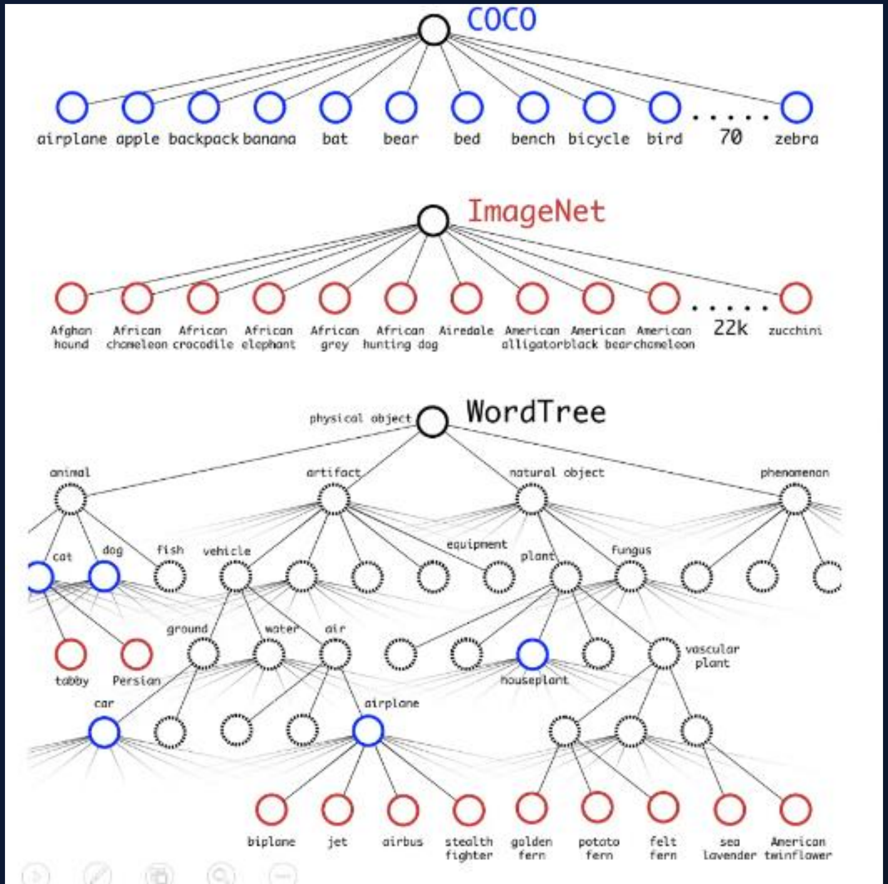
1 标签



image_path x_min, y_min, x_max, y_max, class_id x_min, y_min ,..., class_id

ImageNet >> COCO

Yolo9000



损失函数

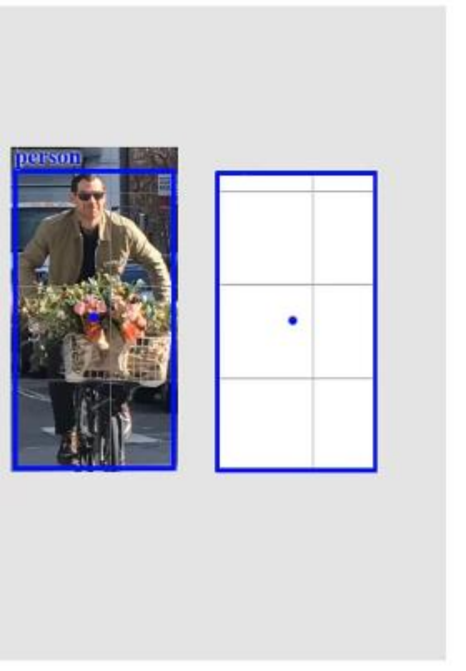
- 置信度损失
- 定位损失
- 类别概率损失

2 置信度损失

对照的标准

(52*52+26*26+13*13)*3 = 10647个

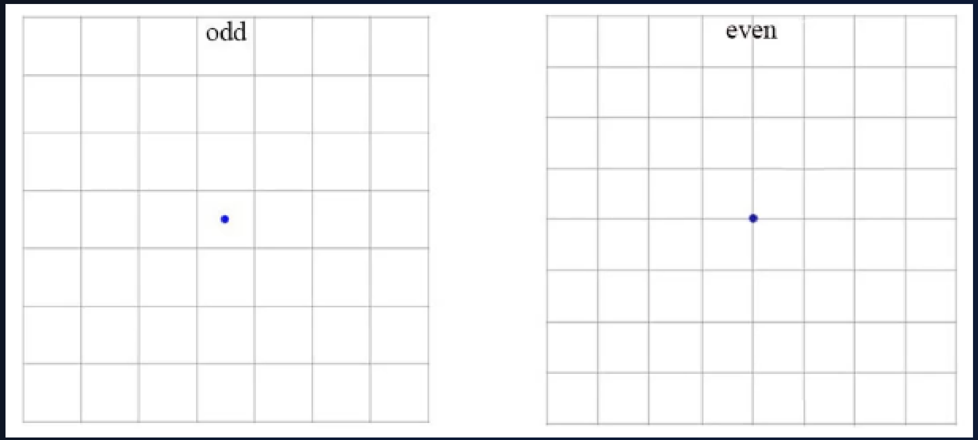
Iou与阈值进行比较



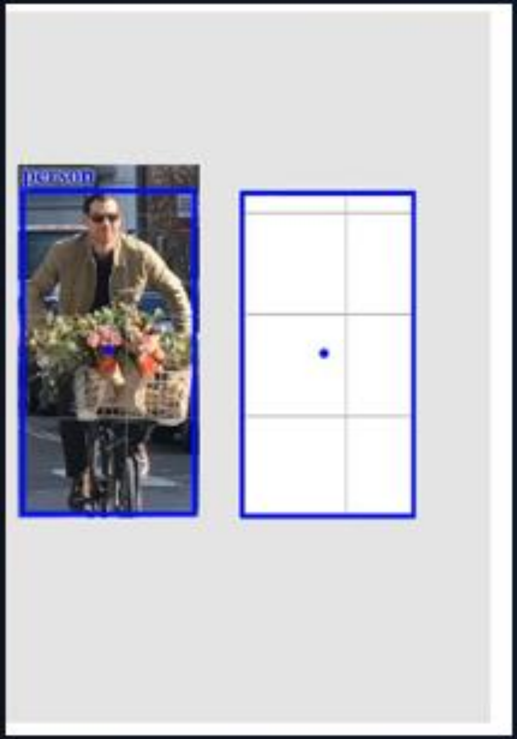

```
best_anchor_ind = np.argmax(np.array(iou).reshape(-1), axis=-1)
best_detect = int(best_anchor_ind / self.anchor_per_scale)
best_anchor = int(best_anchor_ind % self.anchor_per_scale)
```

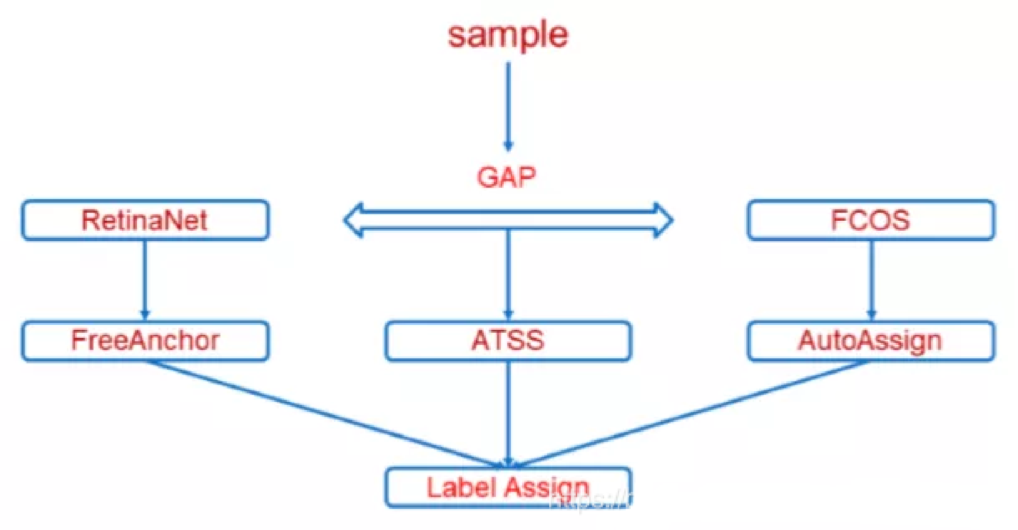
1. 一个网格负责一个物体检测
2. 得到的特征图的边长最好是一个奇数
3. smooth_onehot

2 置信度损失



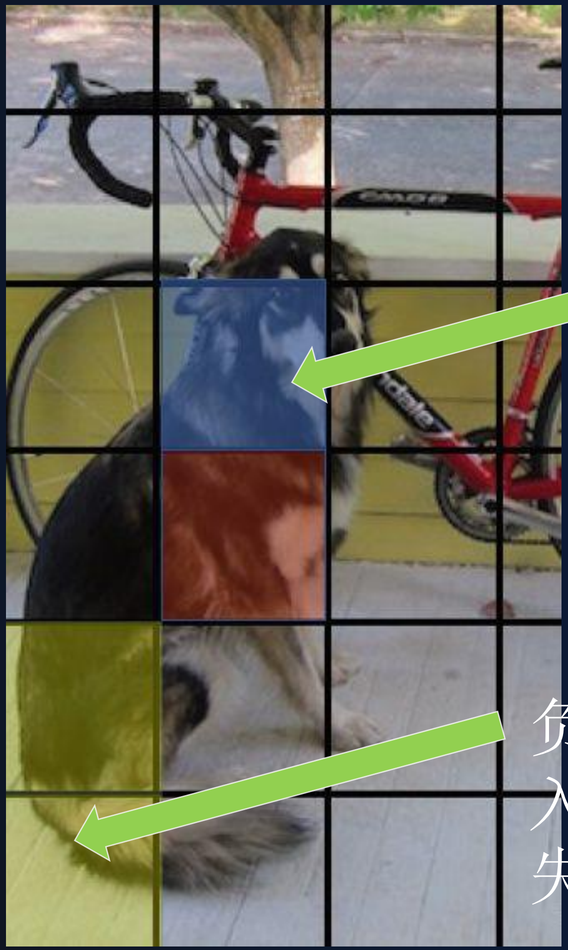
Iou与阈值进行比较





2 置信度损失

负样本与忽略样本的确定



忽略样本，不计入置信度损失

负样本，计入置信度损失

$$L_{focalloss} = -\alpha_t (1 - p_t)^{\mu * \gamma} \log(p_t)$$

Retinanet Yolo
19161 vs. 6300

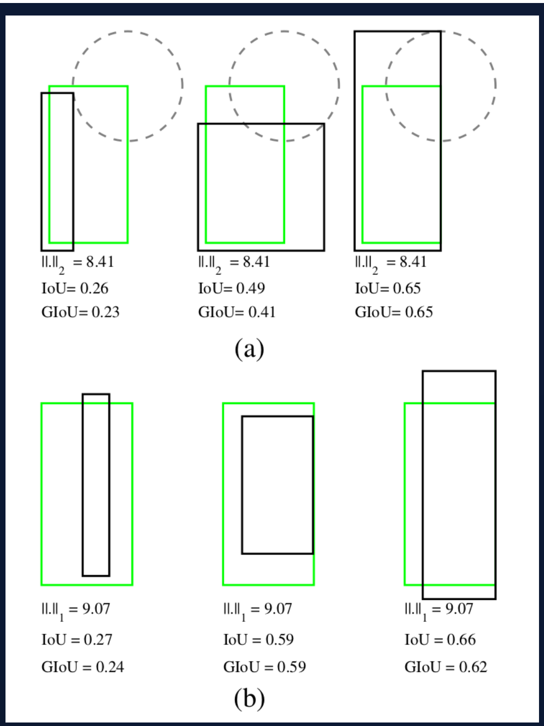
$$\alpha_t$$

3 定位损失

预测框与标签框的中心横纵坐标、高宽之间的损失

$$\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{obj}\left(x_i-\hat{x}_i)^2+(y_i-\hat{y_i})^2\right)\Big]+\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{obj}\left[(\sqrt{w_i}-\sqrt{\hat{w}_i})^2+(\sqrt{h_i}-\sqrt{\hat{h}_i})^2\right]$$

MSE,L1 →IoU →GIoU



平衡大小框的位置损失

(2 - label.h*label.w)

4 类别概率损失

```
prob_loss = respond_bbox * tf.nn.sigmoid_cross_entropy_with_logits(labels=label_prob, logits=conv_raw_prob)
```


THANKS