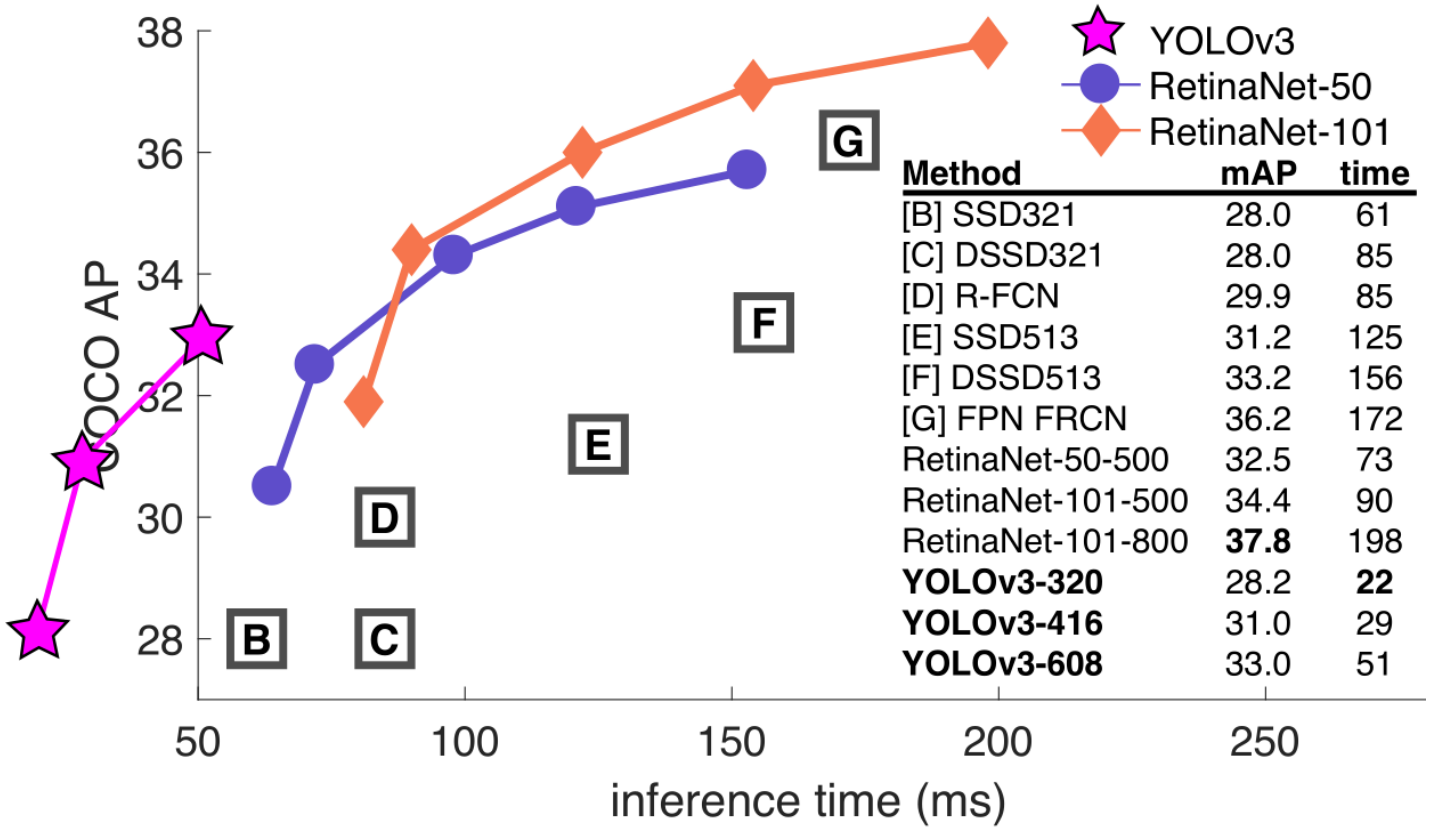
YOLO v3

Joseph Redmon , Ali Farhadi.

YOLOv3: An Incremental Improvement. 2018

https://arxiv.org/abs/1804.02767

YOLOv3



Method	mAP	time
[B] SSD321	28.0	61
[C] DSSD321	28.0	85
[D] R-FCN	29.9	85
[E] SSD513	31.2	125
[F] DSSD513	33.2	156
[G] FPN FRCN	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	**37.8**	198
YOLOv3-320	28.2	**22**
YOLOv3-416	31.0	29
YOLOv3-608	33.0	51

- YOLOv3非常快速准确。 YOLOv3的mAP可以与RetinaNet相当，但速度提高约4倍(51ms vs 198ms)。

YOLO算法的基本思想

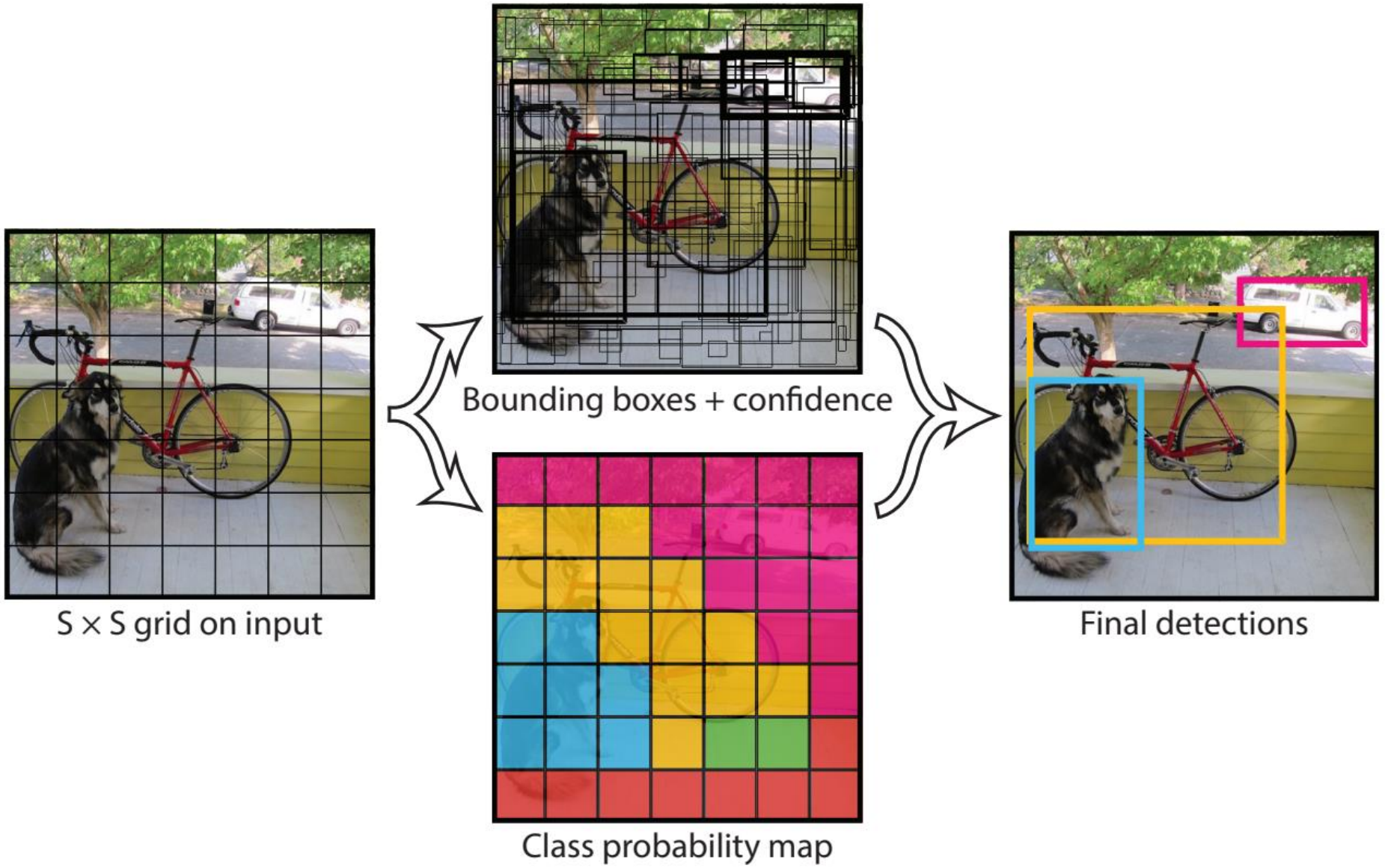
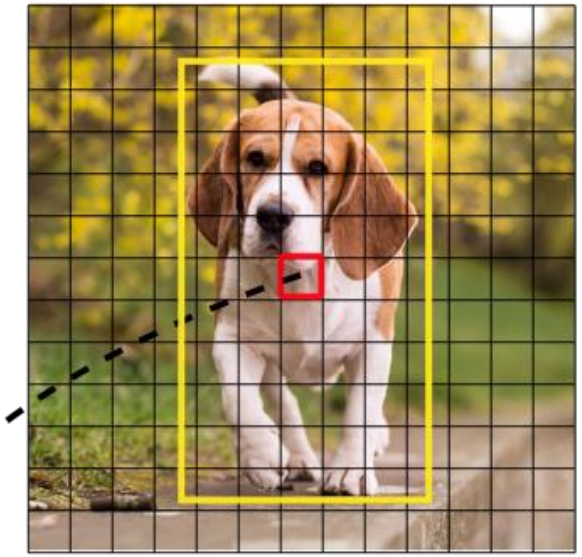
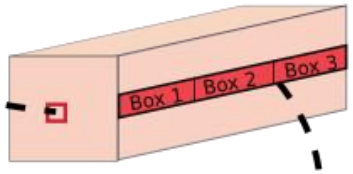


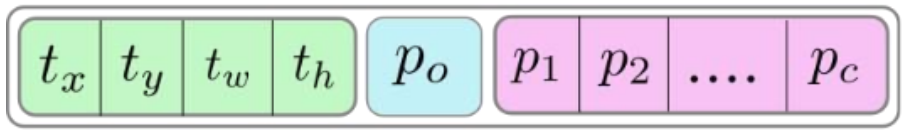
Image Grid. The Red Grid is responsible for detecting the dog



Prediction Feature Map



Attributes of a bounding box



x B

Box Co-ordinates

Objectness Score

Class Scores

preprocessed image
(608, 608, 3)

Deep CNN

reduction
factor: 32

19

19

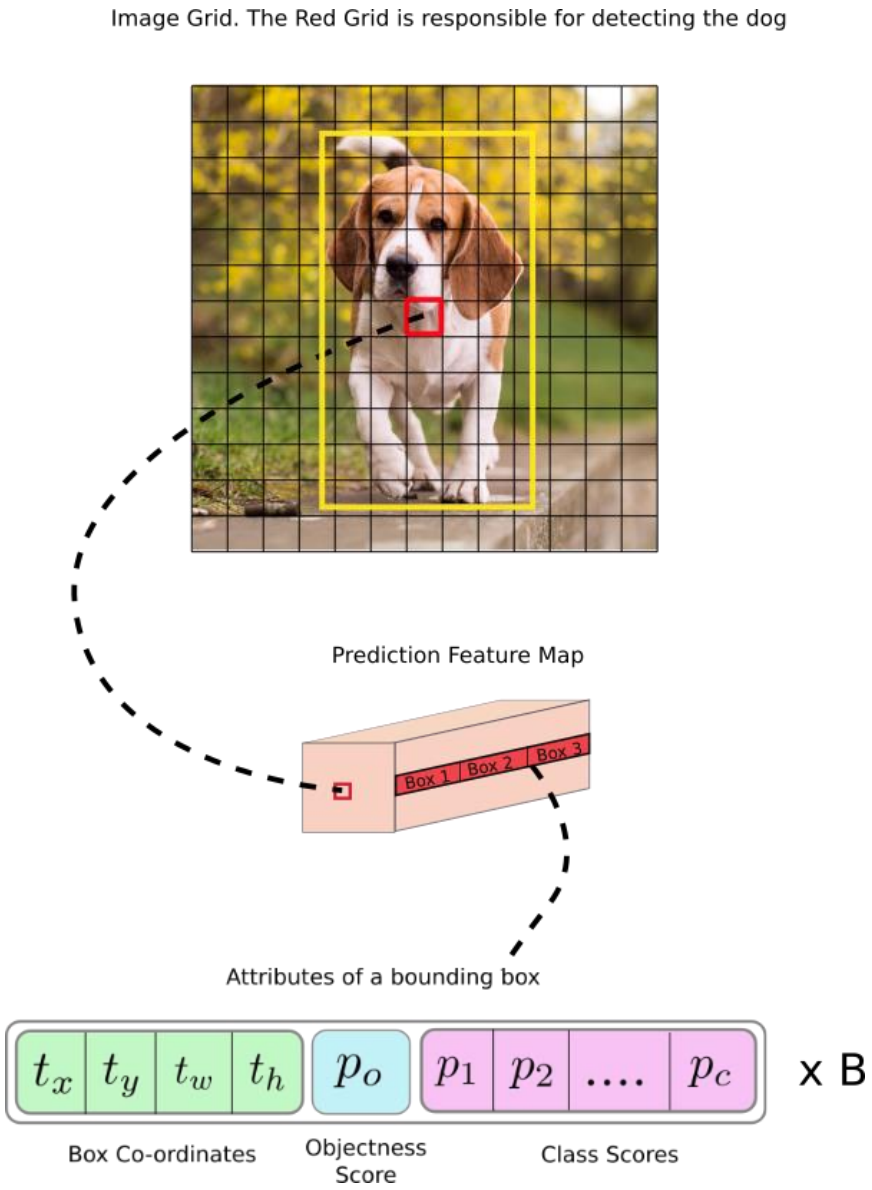
p_c b_x b_y b_h b_w 80 class probabilities

box 1 ...

box 2 ...

box 3 ...

YOLOv3算法的基本思想



- 首先通过特征提取网络对输入图像提取特征，得到一定大小的特征图，比如13×13（相当于416×416图片大小），然后将输入图像分成13 × 13个grid cells，接着如果GT中某个目标的中心坐标落在哪个grid cell中，那么就由该grid cell来预测该目标。每个grid cell都会预测3固定数量的边界框(YOLO v1中是2个，YOLO v2中是5个，YOLO v3中是3个，这几个边界框的初始大小是不同的)

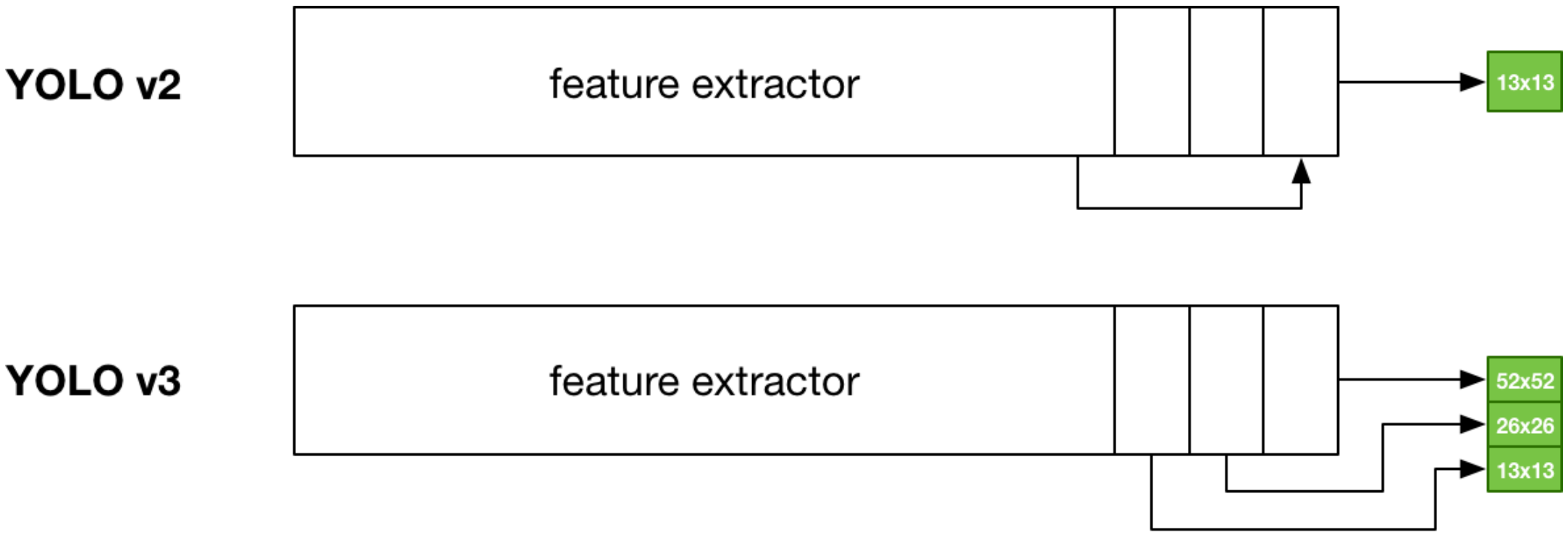
- 预测得到的输出特征图有两个维度是提取到的特征的维度，比如13 × 13，还有一个维度（深度）是 B × (5+C)，注：YOLO v1中是（B × 5+C），其中B表示每个grid cell预测的边界框的数量 （YOLO v3中是3个）；
C表示边界框的类别数（没有背景类，所以对于VOC数据集是20），
5表示4个坐标信息和一个目标性得分（objectness score）。

类别预测(Class Prediction)

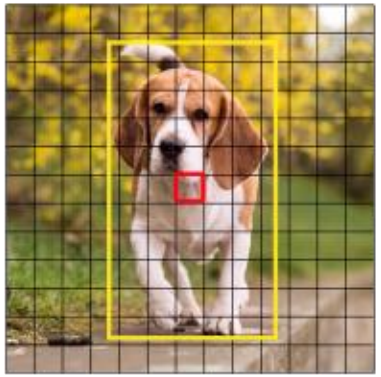
- 大多数分类器假设输出标签是互斥的。 如果输出是互斥的目标类别，则确实如此。 因此，YOLO应用softmax函数将得分转换为总和为1的概率。 而YOLOv3使用多标签分类。 例如，输出标签可以是“行人”和“儿童”，它们不是非排他性的。 （现在输出的总和可以大于1）

- YOLOv3用多个独立的逻辑（logistic）分类器替换softmax函数，以计算输入属于特定标签的可能性。
- 在计算分类损失进行训练时，YOLOv3对每个标签使用二元交叉熵损失。 这也避免使用softmax函数而降低了计算复杂度。

多尺度融合


YOLO v2
feature extractor
13x13
YOLO v3
feature extractor
52x52
26x26
13x13

Prediction Feature Maps at different Scales

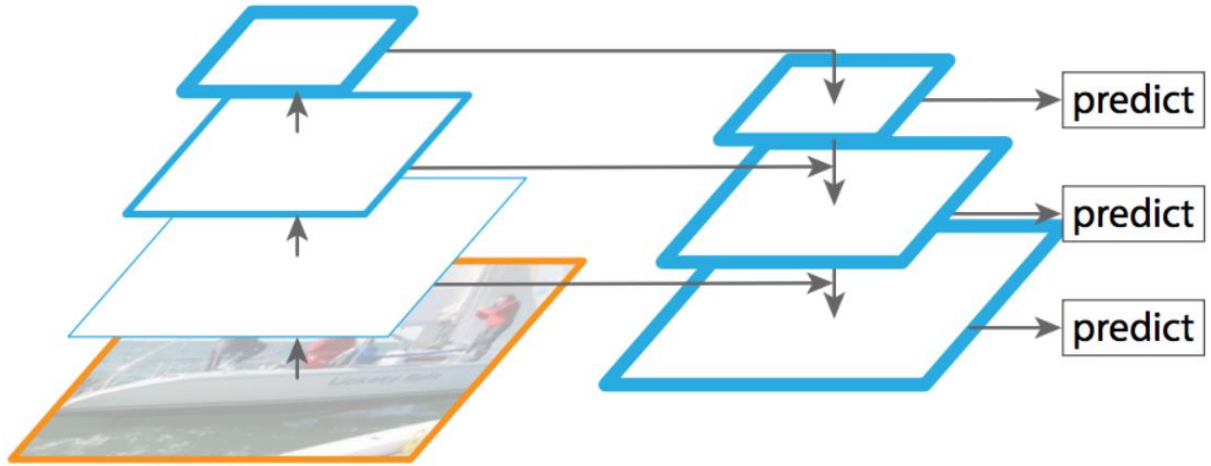


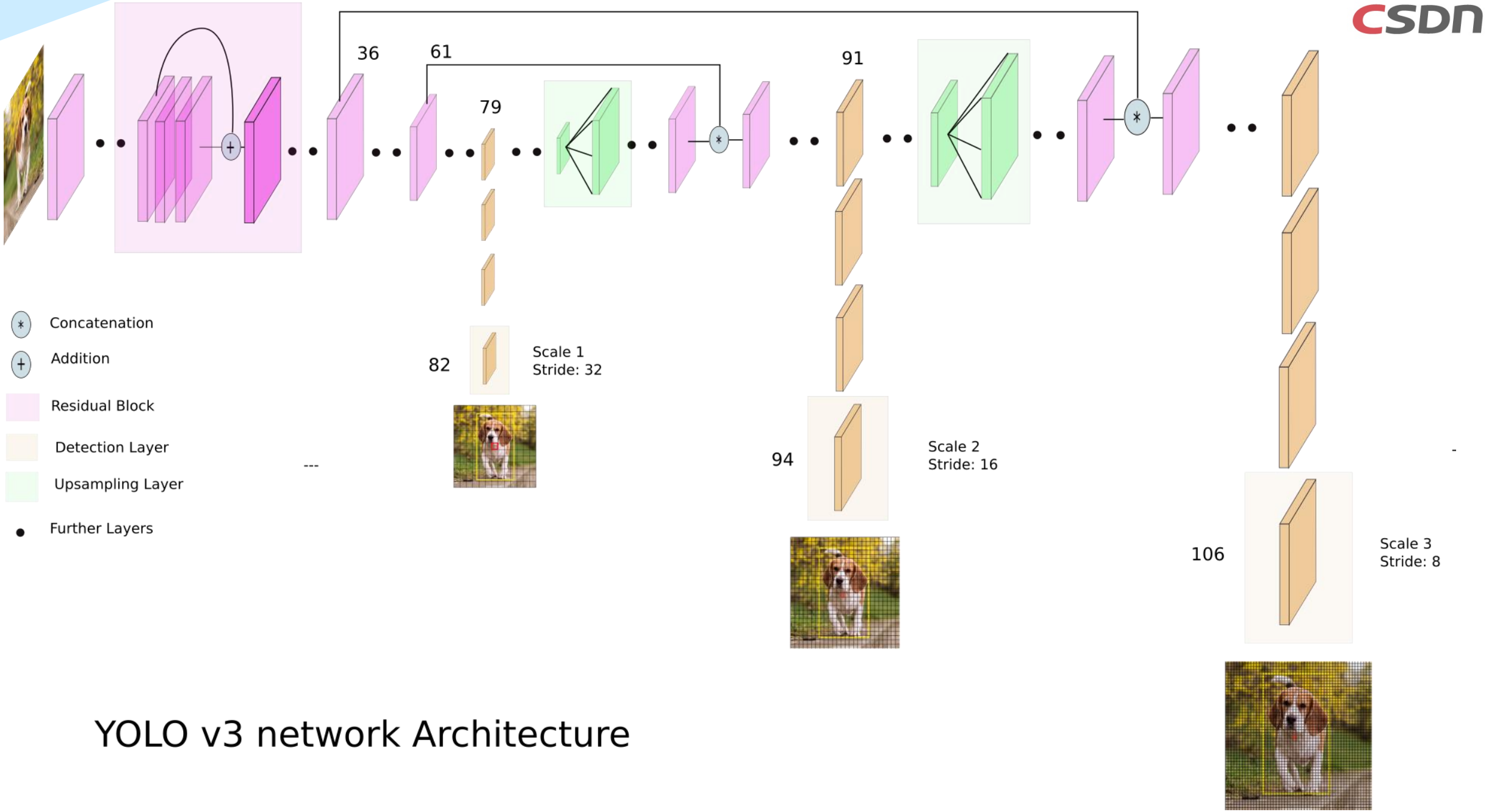
13 x 13

26 x 26

52 x 52

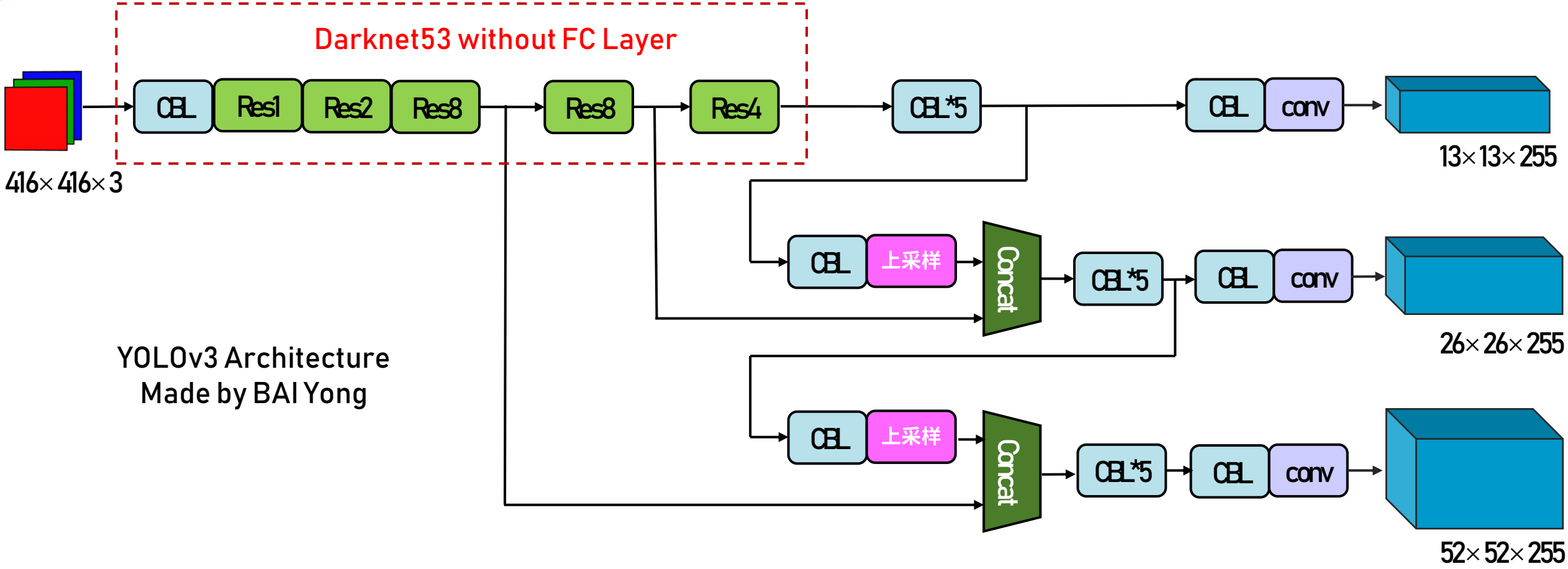
Feature Pyramid Network (FPN)



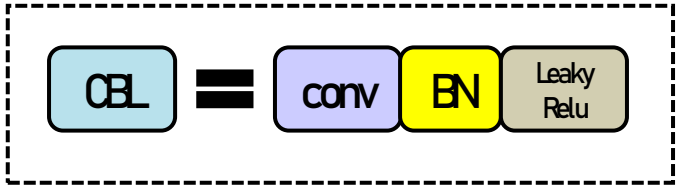


YOLO v3 network Architecture

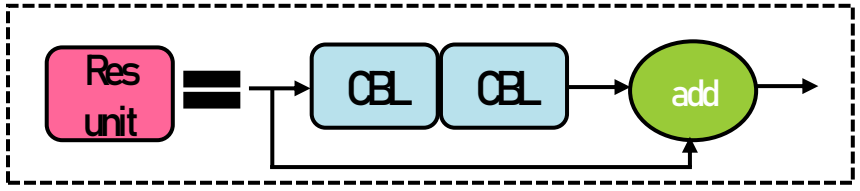
YOLO v3 网络结构



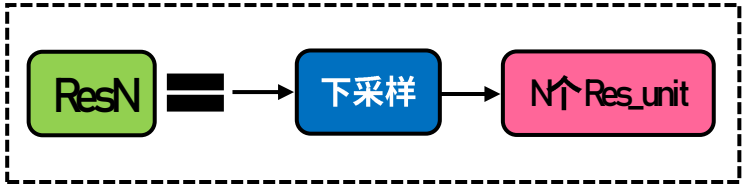
CBL (DarknetConv2D_BN_Leaky)



Res_unit



Resblock_body



Feature Pyramid Networks (FPN) like Feature Pyramid

- yolo v3输出了3个不同尺度的特征，如上图所示的y1, y2, y3。借鉴了FPN(feature pyramid networks)，采用多尺度来对不同大小的目标进行检测，越精细的grid cell就可以检测出越精细的物体。

  y1,y2和y3的深度都是255，边长的规律是13:26:52。

  对于COCO类别而言，有80个类别，所以每个box应该对每个类别都输出一个概率。yolo v3设定的是每个网格单元预测3个边界框，所以每个边界框有(x, y, w, h, confidence)五个基本参数，然后还要有80个类别的概率。所以3×(5 + 80) = 255。

- YOLO v3采用了多尺度的特征融合，所以边界框的数量要比之前多很多，以输入图像为416 × 416为例：(13×13+26×26+52×52)×3=10647比13×13×5更多。

为确定priors，YOLOv3应用k均值聚类。然后它预先选择9个聚类簇。

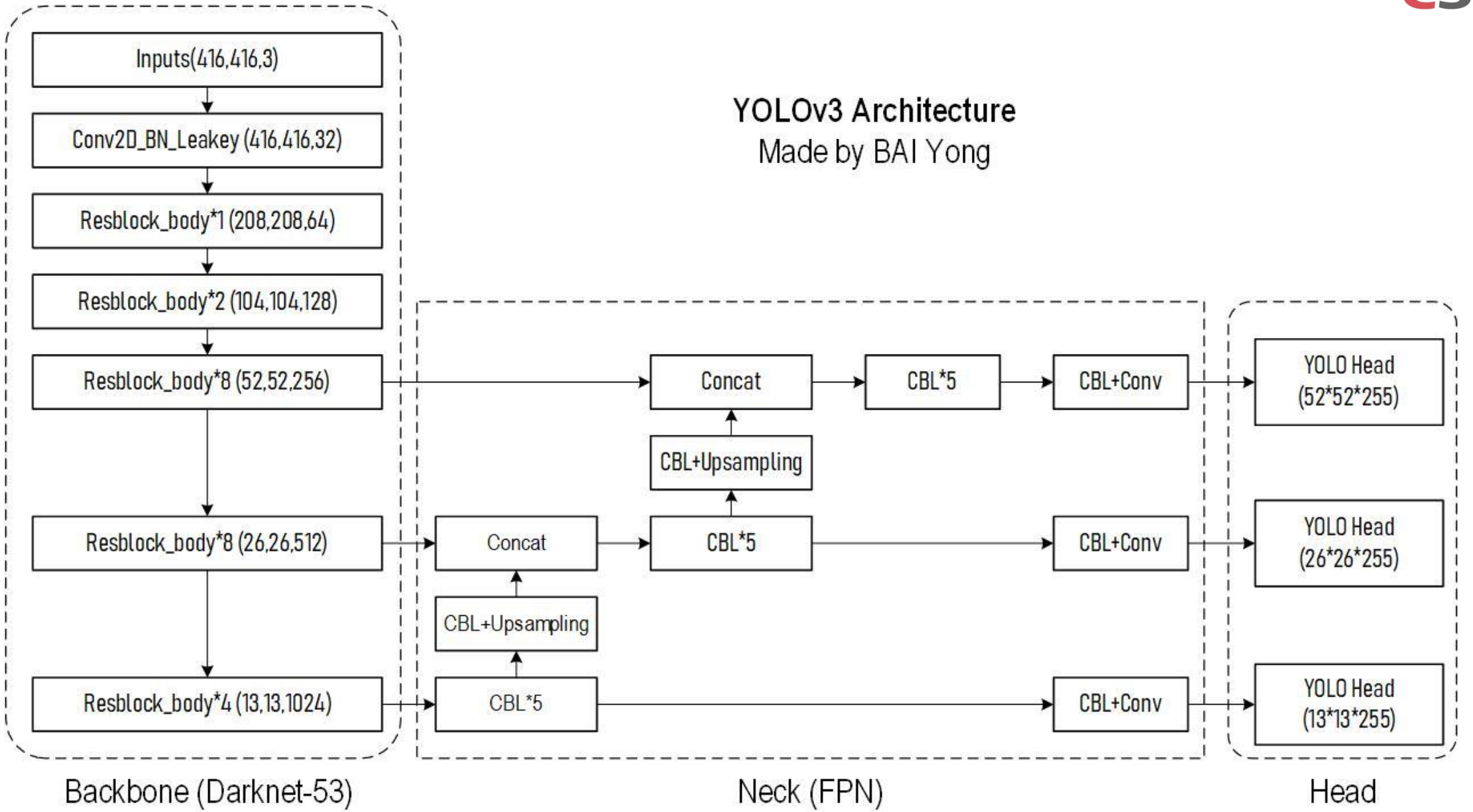
对于COCO，锚定框的宽度和高度为（10×13），（16×30），（33×23），（30×61），（62×45），（59×119），（116× 90），（156×198），（373×326）。这应该是按照输入图像的尺寸是416×416计算得到的。这9个priors根据它们的尺度分为3个不同的组。在检测目标时，给一个特定的特征图分配一个组。

Darknet-53 Feature extractor

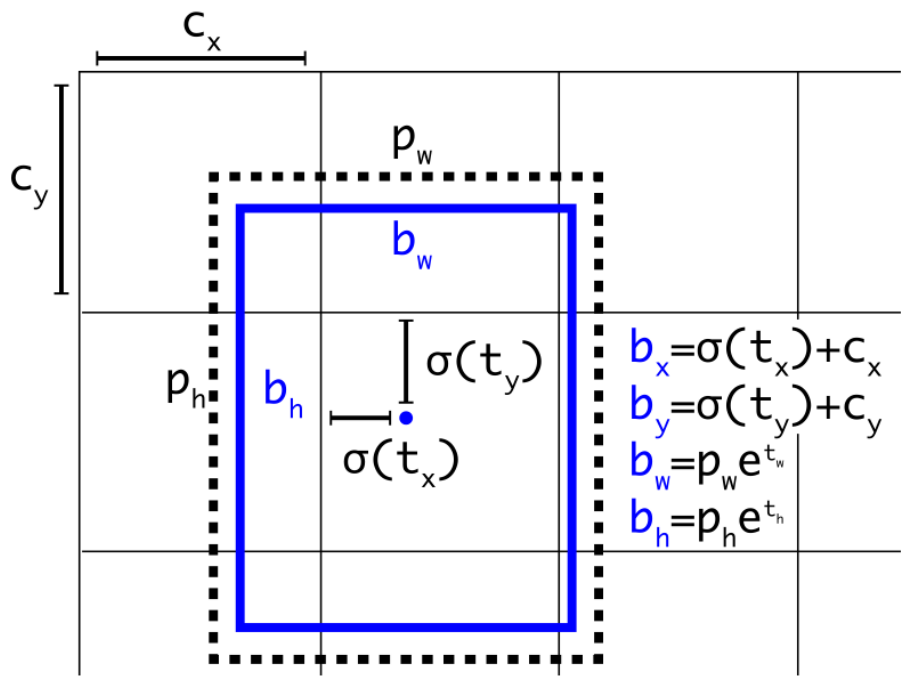
	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

- 一个新的53层Darknet-53用于取代Darknet-19作为特征提取器。
- Darknet-53主要由3×3和1×1滤波器组成，具有residual连接，如ResNet中的残差网络。
- Darknet-53比ResNet-152具有更少的BFLOP（ billion floating point operations ），但实现了相同的分类准确度，速度快了2倍。

注：整个v3结构里面，是没有池化层和全连接层的。前向传播过程中，张量的尺寸变换是通过改变卷积核的步长来实现的，比如stride=(2, 2)，这就等于将图像边长缩小了一半(即面积缩小到原来的1/4)。v3也和v2一样，backbone都会将输出特征图缩小到输入的1/32。所以，通常都要求输入图片是32的倍数 。


YOLOv3 Architecture
Made by BAI Yong
Inputs(416,416,3)
Conv2D_BN_Leakey (416,416,32)
Resblock_body*1 (208,208,64)
Resblock_body*2 (104,104,128)
Resblock_body*8 (52,52,256)
Resblock_body*8 (26,26,512)
Resblock_body*4 (13,13,1024)
Concat
CBL*5
CBL+Conv
CBL+Upsampling
Concat
CBL*5
CBL+Conv
CBL+Upsampling
CBL*5
CBL+Conv
YOLO Head
(52*52*255)
YOLO Head
(26*26*255)
YOLO Head
(13*13*255)
Backbone (Darknet-53)
Neck (FPN)
Head

- 网络在特征图（13×13 =169）的每个Cell上预测3个边界框，每一个边界框预测5个值：t_x,t_y,t_w,t_h，t_o，其中前四个是坐标的offset值，t_o是置信度。
- 如果这个Cell距离图像左上角的边距为(c_x,c_y)以及该Cell对应的边界框先验维度（Bounding Box Prior）的宽和高分别为(p_w,p_h)，则预测边界框的实际值见下图：



只对tx,ty作激活(sigma)处理,不对tw,th作激活处理。

边界框预测和代价函数计算
(Bounding box prediction & cost function calculation)

- YOLOv3使用逻辑回归（logistic）预测每个边界框的目标性得分(objectness score)。
- YOLOv3改变了计算代价函数的方式。
  - 如果边界框先验（锚定框）与GT目标比其他目标重叠多，则相应的目标性得分应为1。
  - 对于重叠大于预定义阈值（默认值0.5）的其他先验框，不会产生任何代价。
  - 每个GT目标仅与一个先验边界框相关联。 如果没有分配先验边界框，则不会导致分类和定位损失，只会有目标性的置信度损失。
  - 使用tx和ty（而不是bx和by）来计算损失。

正样本：与GT的IOU最大的框

负样本：与GT的IOU<0.5的框

忽略的样本：与GT的IOU>0.5但不是最大的框

损失函数(Loss function)

- YOLO每个网格单元预测多个边界框。

  为了计算true positive的损失，只希望其中一个框负责该目标。

  为此，选择与GT具有最高IoU的那个框。

YOLO使用预测值和GT之间的误差平方的求和来计算损失。 损失函数包括：

- **classification loss,** 分类损失
- **localization loss,** 定位损失（预测边界框与GT之间的误差）
- **confidence loss,** 置信度损失（框的目标性；objectness of the box）

YOLOv1损失 (Loss)

最终损失将定位损失、置信度损失和分类损失相加在一起。

定位损失

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

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

置信度损失

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

分类损失

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2$$

YOLOv1定位损失（Localization loss）

定位损失测量预测的边界框位置和框大小的误差。 只计算负责检测目标的框。

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

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

where

$\mathbb{1}_{ij}^{obj} = 1$ if the j th boundary box in cell i is responsible for detecting the object, otherwise 0.

λ_{coord} increase the weight for the loss in the boundary box coordinates.

- 不希望在大框和小框中同等地加权绝对误差。 即不认为大框中的2像素误差对于小框是相同的。为了部分解决这个问题，YOLO预测边界框宽度和高度的平方根，而不是宽度和高度。 另外，为更加强调边界框的精度，将损失乘以λ_{coord}（默认值：5）。

YOLOv3的定位损失

GT框的偏移(offsets)

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} \left[\left(t_{xi} - \hat{t}_{xi} \right)^2 + \left(t_{yi} - \hat{t}_{yi} \right)^2 \right]$$

$$+\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} \left[(t_{wi} - \hat{t}_{wi})^2 + (t_{hi} - \hat{t}_{hi})^2 \right]$$

where

$\mathbb{1}_{ij}^{obj} = 1$ if the j th boundary box in cell i is responsible for detecting the object, otherwise 0.

λ_{coord} increase the weight for the loss in the boundary box coordinates.

- $\lambda_{coord} = 2 - w \times h$

YOLOv3置信度损失(Confidence loss)

- 如果在框中检测到目标，则置信度损失（测量框的目标性，measuring the objectness of the box）为：

$$\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{1}_{ij}^{\text{obj}}\left(C_i-\hat{C}_i\right)^2$$

where

$\hat{C}_i$ is the box confidence score of the box j in cell i.

$\mathbb{1}_{ij}^{obj}=1$ if the j th boundary box in cell i is responsible for detecting the object, otherwise 0.

- 如果在框中没有检测到目标，则置信度损失为：

$$\lambda_{\text{noobj}}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{1}_{ij}^{\text{noobj}}\left(C_i-\hat{C}_i\right)^2$$

大多数框不包含任何目标。 这导致类不平衡问题，即训练模型时更频繁地检测到背景而不是检测目标。为了解决这个问题，YOLOv1 将这个损失用因子 λ_{noobj}（默认值：0.5）降低。

where

$\mathbb{1}_{ij}^{noobj}$ is the complement of $\mathbb{1}_{ij}^{obj}$.

$\hat{C}_i$ is the box confidence score of the box j in cell i.

λ_{noobj} weights down the loss when detecting background.

YOLOv3的置信度损失和YOLOv1基本一样

YOLOv3分类损失 (Classification loss)

如果检测到目标，则预测框的分类损失是每个类别的条件类别概率的交叉熵损失：

YOLOv3: During training we use **binary cross-entropy loss** for the class predictions.

$$-\sum_{i=0}^{S^2}\sum_{j=0}^{B^2}1_{ij}^{\text{obj}}\sum_{c\in classes}[\hat{p}_i(c)\log(p_i(c))+(1-\hat{p}_i(c))\log(1-p_i(c))]$$

$1_{ij}^{obj} = 1$ if the j th boundary box in cell i is responsible for detecting the object, otherwise 0.

注意：负样本的objectness是零，故不会有分类损失。

YOLOv3的损失函数

定位损失

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} \left[(t_{xi} - \hat{t}_{xi})^2 + (t_{yi} - \hat{t}_{yi})^2 \right] + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} \left[(t_{wi} - \hat{t}_{wi})^2 + (t_{hi} - \hat{t}_{hi})^2 \right]$$

置信度损失

$$+\sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} (C_i - \hat{C}_i)^2 + \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{noobj}} (C_i - \hat{C}_i)^2$$

分类损失

$$-\sum_{i=0}^{S^2} \sum_{j=0}^{B^2} 1_{ij}^{\text{obj}} \sum_{c \in classes} [\hat{p}_i(c) \log(p_i(c)) + (1 - \hat{p}_i(c)) \log(1 - p_i(c))]$$

- $\lambda_{coord} = 2 - w \times h$

小结

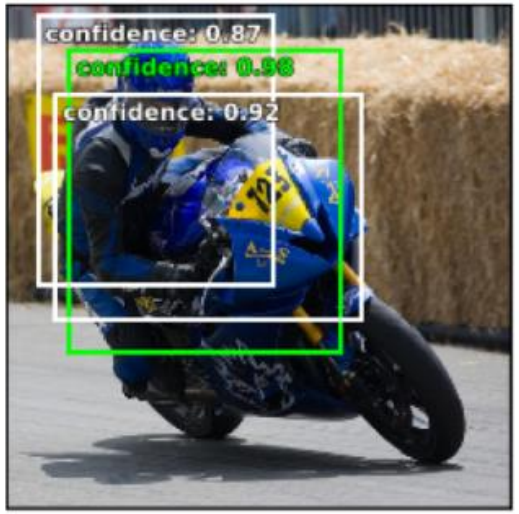
- 无论训练和测试都以anchor为基准计算出坐标偏移值，得出预测框坐标。
- 训练时可以计算预测框和GT框的IOU
- 但测试时没有GT框，只能比较多个预测框，比较相互之间的IOU，做NMS。
- Anchor框只是用来回归坐标值。

NMS (Non-Maximum Suppression)

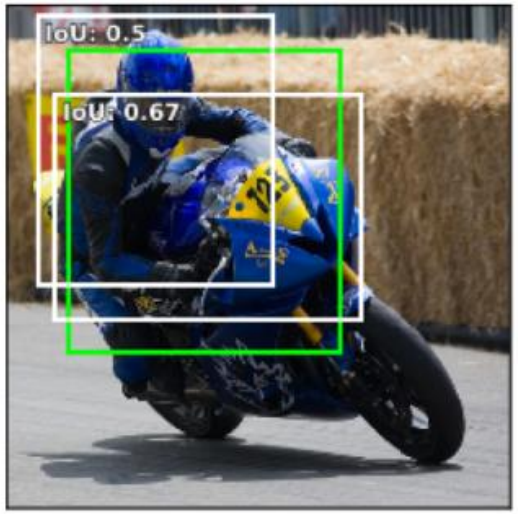
Repeat with next highest confidence prediction until no more boxes are being suppressed

For each class...

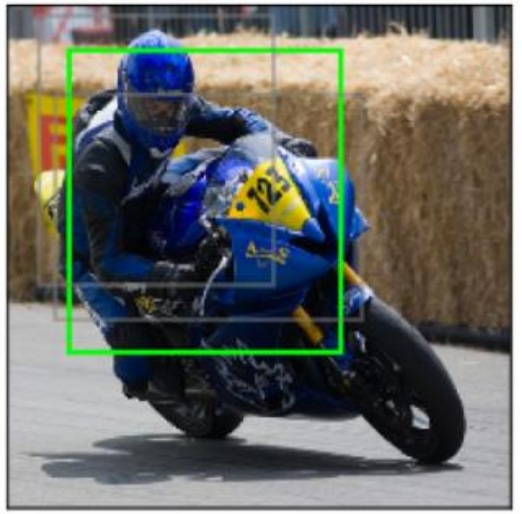
After filtering out low confidence predictions, we may still be left with ***redundant*** **detections**



Select the bounding box prediction with the **highest confidence**



Calculate the IoU between the selected box and all remaining predictions



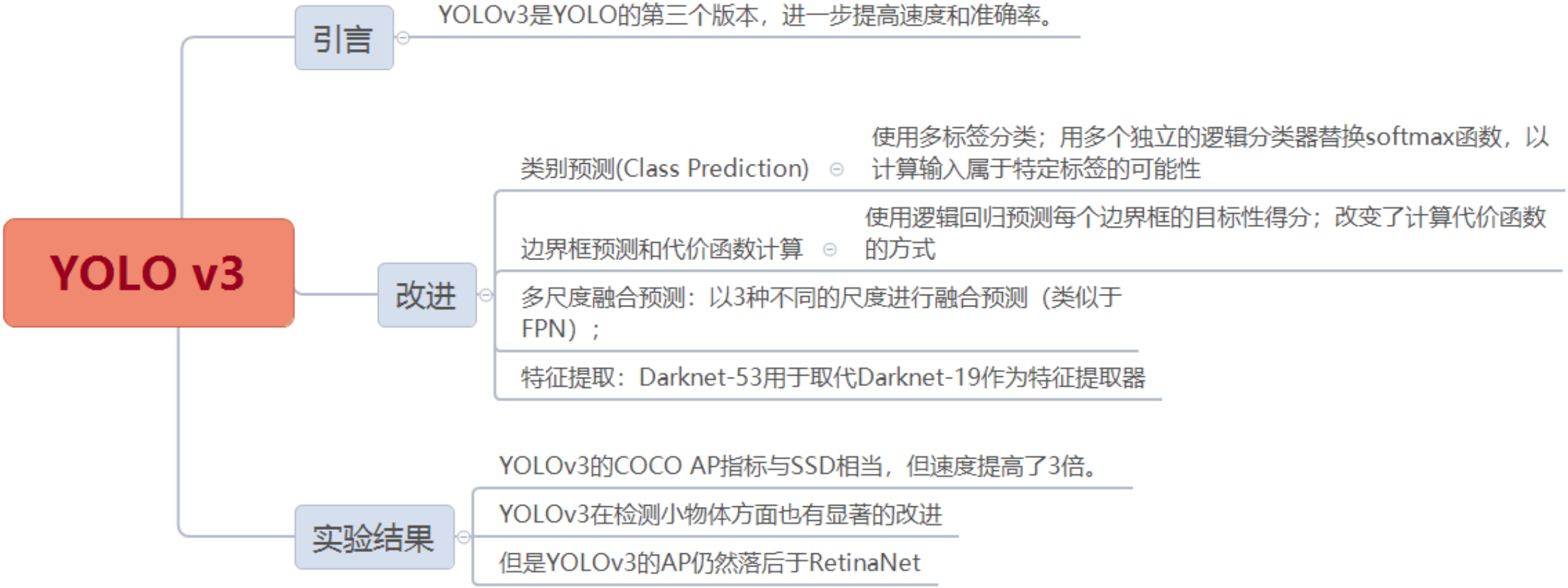
Remove any boxes which have an IoU score above some defined threshold

YOLOv3 performance

- YOLOv3的COCO AP指标与SSD相当，但速度提高了3倍。
- 但是YOLOv3的AP仍然落后于RetinaNet。 特别是，与RetinaNet相比，AP @ IoU = .75显著下降，这表明YOLOv3具有更高的定位误差。
- YOLOv3在检测小目标方面也有显著的改进。

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage methods							
Faster R-CNN+++ [5]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [8]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [6]	Inception-ResNet-v2 [21]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [20]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
One-stage methods							
YOLOv2 [15]	DarkNet-19 [15]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [11, 3]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [3]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet [9]	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet [9]	ResNeXt-101-FPN	**40.8**	**61.1**	**44.1**	**24.1**	**44.2**	51.2
YOLOv3 608 × 608	Darknet-53	33.0	57.9	34.4	18.3	35.4	41.9

YOLOv3 performs very well in the fast detector category when speed is important.


YOLO v3
引言
YOLOv3是YOLO的第三个版本，进一步提高速度和准确率。
改进
类别预测(Class Prediction)
使用多标签分类；用多个独立的逻辑分类器替换softmax函数，以计算输入属于特定标签的可能性
边界框预测和代价函数计算
使用逻辑回归预测每个边界框的目标性得分；改变了计算代价函数的方式
多尺度融合预测：以3种不同的尺度进行融合预测（类似于FPN）；
特征提取：Darknet-53用于取代Darknet-19作为特征提取器
实验结果
YOLOv3的COCO AP指标与SSD相当，但速度提高了3倍。
YOLOv3在检测小物体方面也有显著的改进
但是YOLOv3的AP仍然落后于RetinaNet