

# YOLO v3

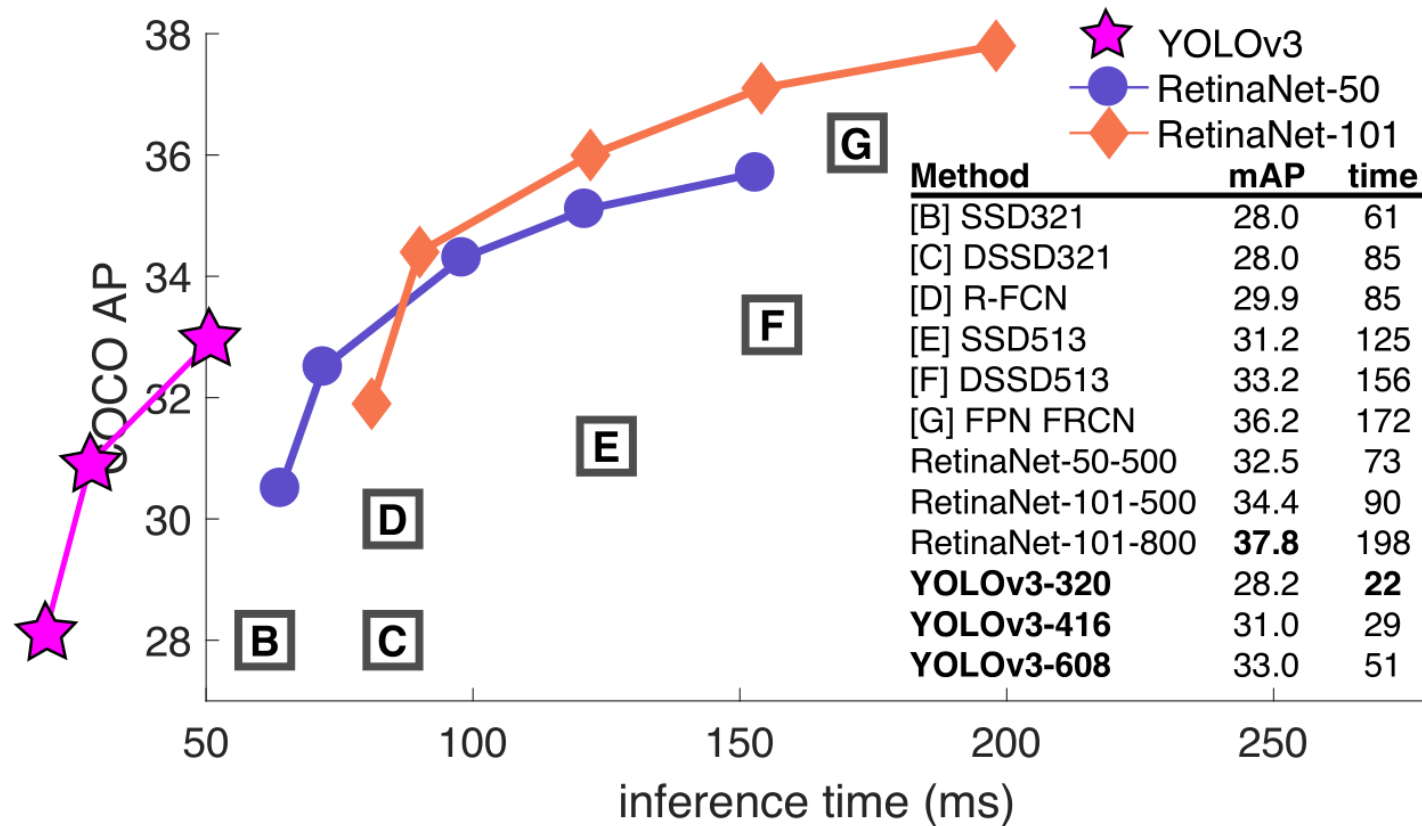
---

Joseph Redmon , Ali Farhadi.

YOLOv3: An Incremental Improvement. 2018

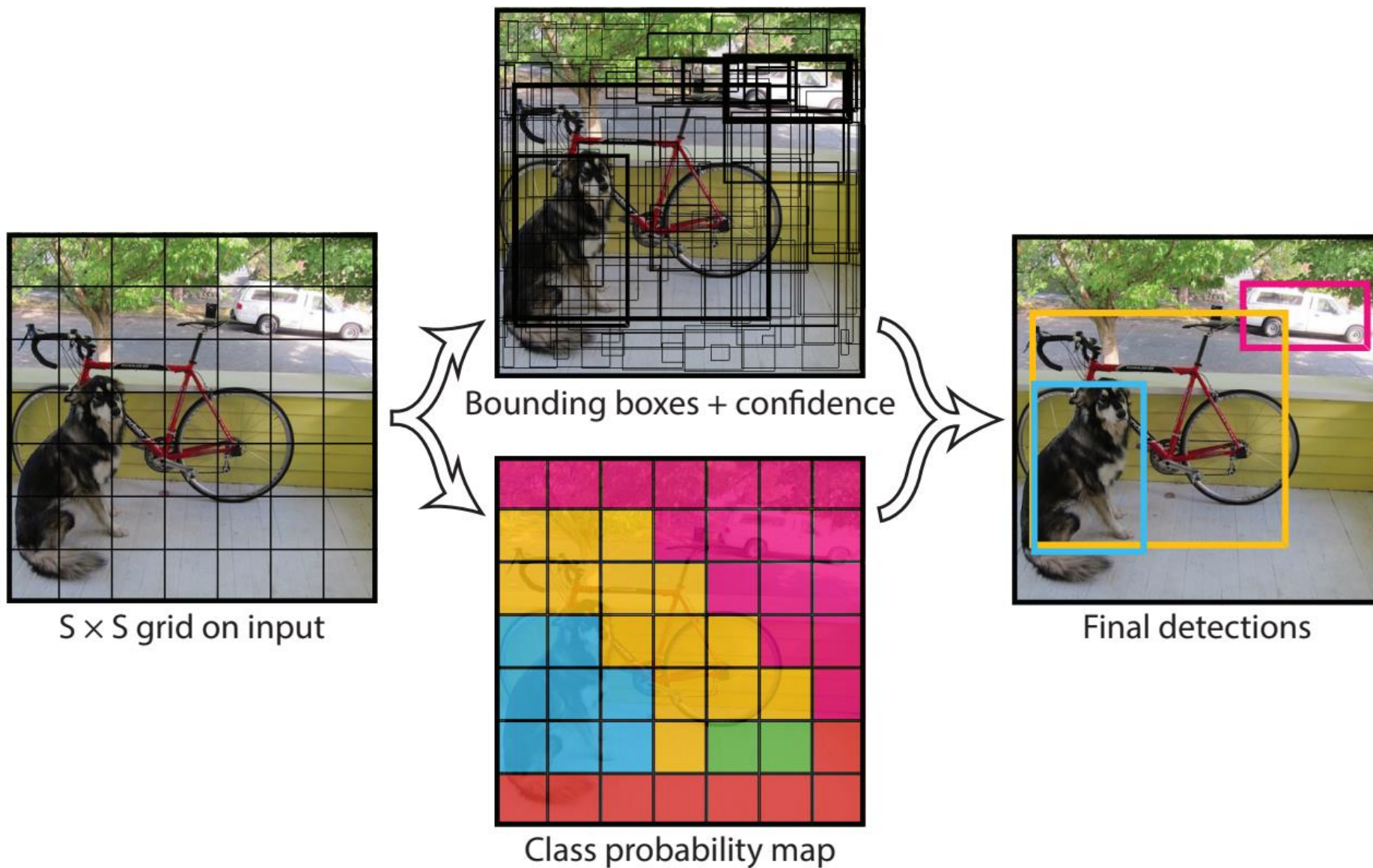
<https://arxiv.org/abs/1804.02767>

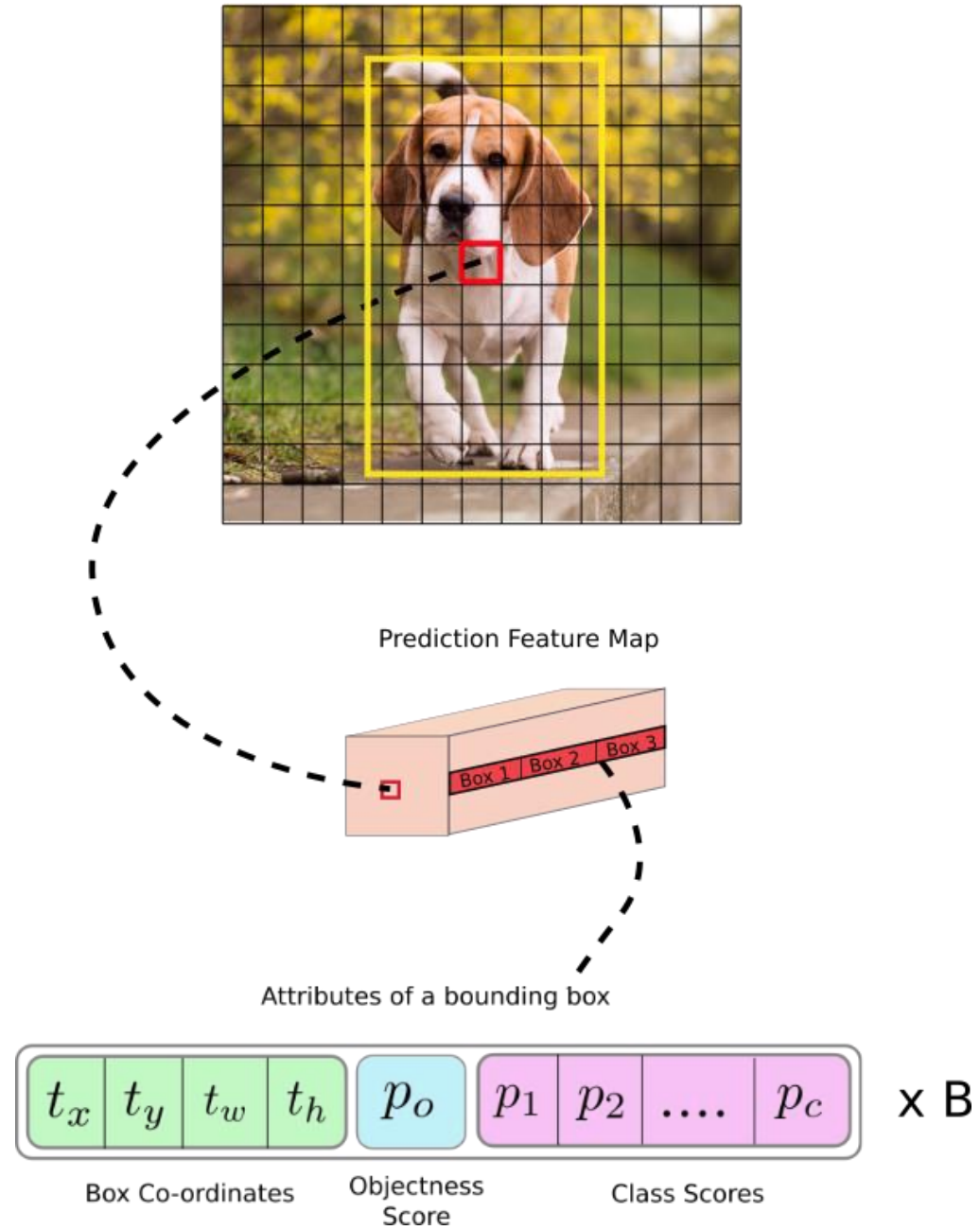
## YOLOv3



- 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

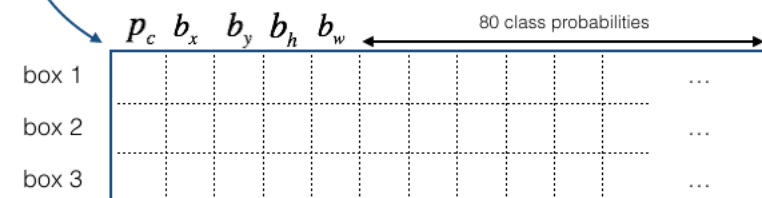
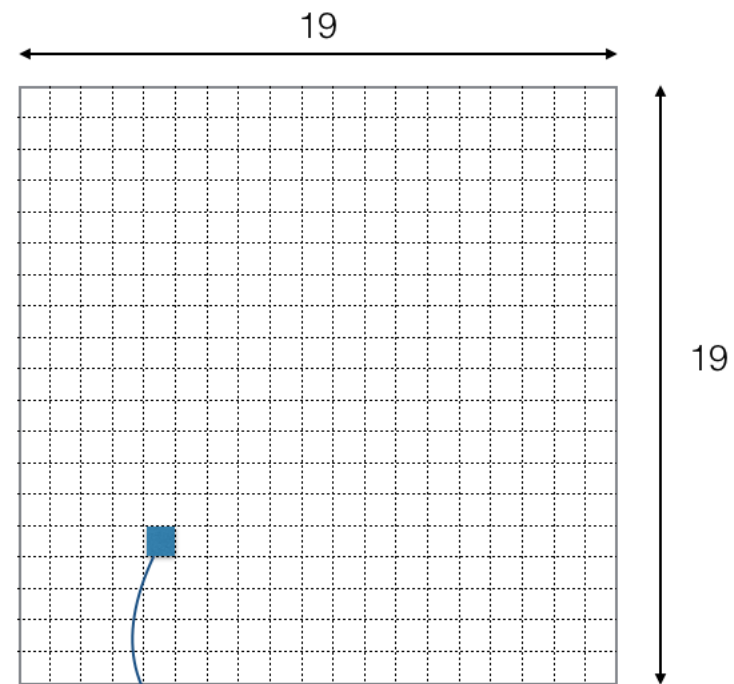




preprocessed image  
(608, 608, 3)



Deep CNN  
reduction  
factor: 32



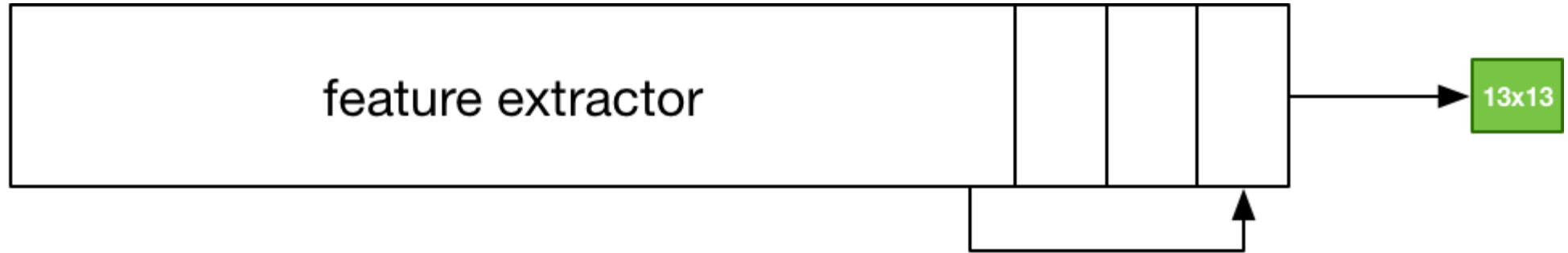
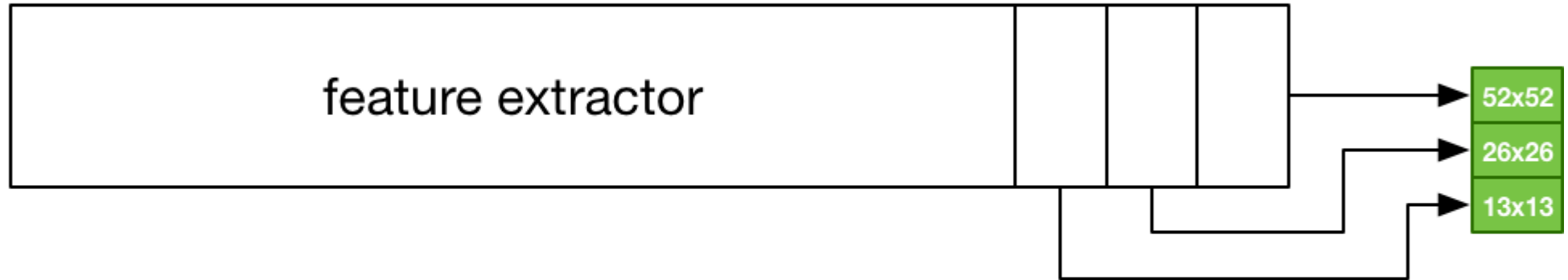
# YOLOv3算法的基本思想

- 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 $13 \times 13$ （相当于 $416 \times 416$ 图片大小），然后将输入图像分成 $13 \times 13$ 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 $13 \times 13$ ，还有一个维度（深度）是  $B \times (5+C)$ ，注：YOLO v1中是  $(B \times 5+C)$ ，其中B表示每个grid cell预测的边界框的数量（比如YOLO v1中是2个，YOLO v2中是5个，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)。

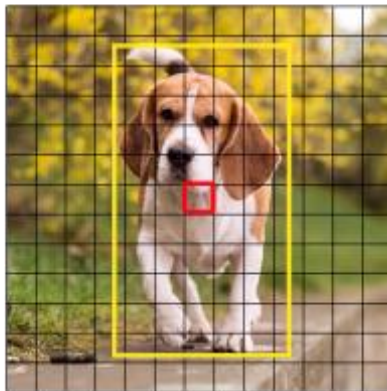
## 类别预测(Class Prediction)

- 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。

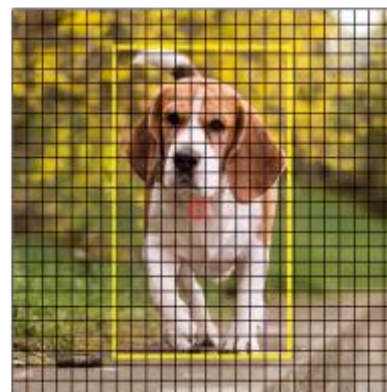


**YOLO v2****YOLO v3**

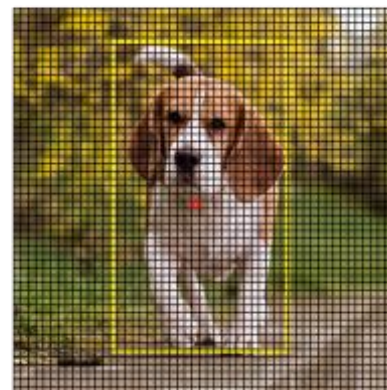





$13 \times 13$



$26 \times 26$



$52 \times 52$



# YOLO v3 Structure

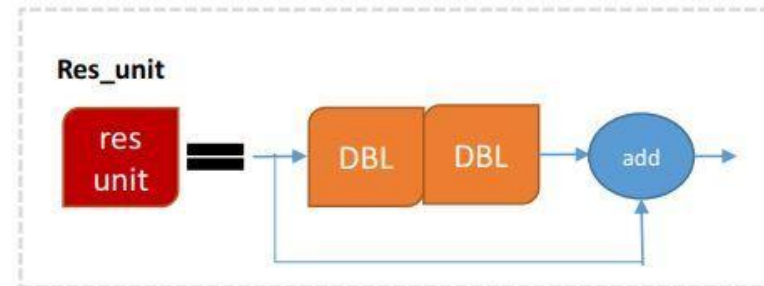
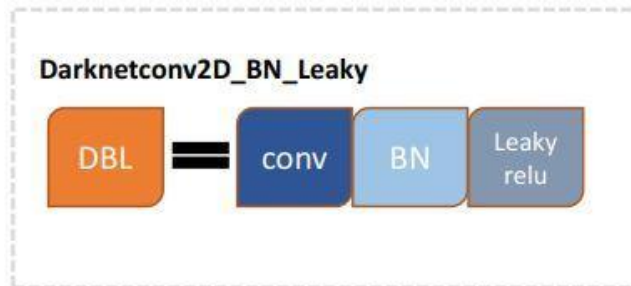
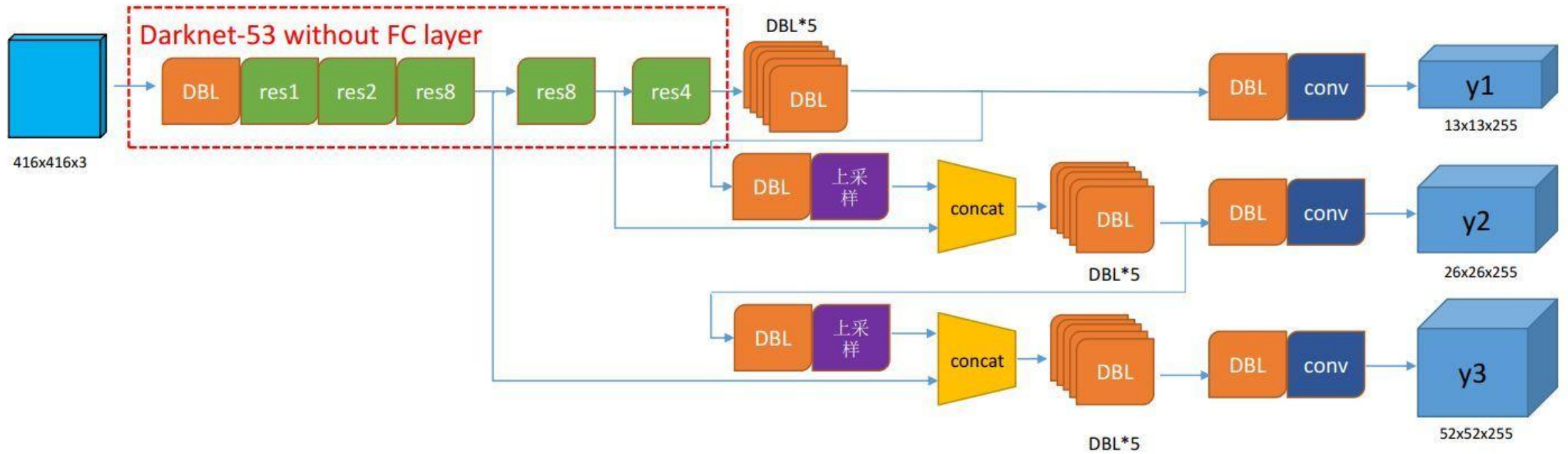


Image Credit: Levio

## 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 $3 \times (5 + 80) = 255$ 。

- 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 $416 \times 416$ 为例： $(13 \times 13 + 26 \times 26 + 52 \times 52) \times 3 = 10647$ 比 $13 \times 13 \times 5$ 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  $(10 \times 13)$ ， $(16 \times 30)$ ， $(33 \times 23)$ ， $(30 \times 61)$ ， $(62 \times 45)$ ， $(59 \times 119)$ ， $(116 \times 90)$ ， $(156 \times 198)$ ， $(373 \times 326)$ 。这应该是按照输入图像的尺寸是 $416 \times 416$ 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

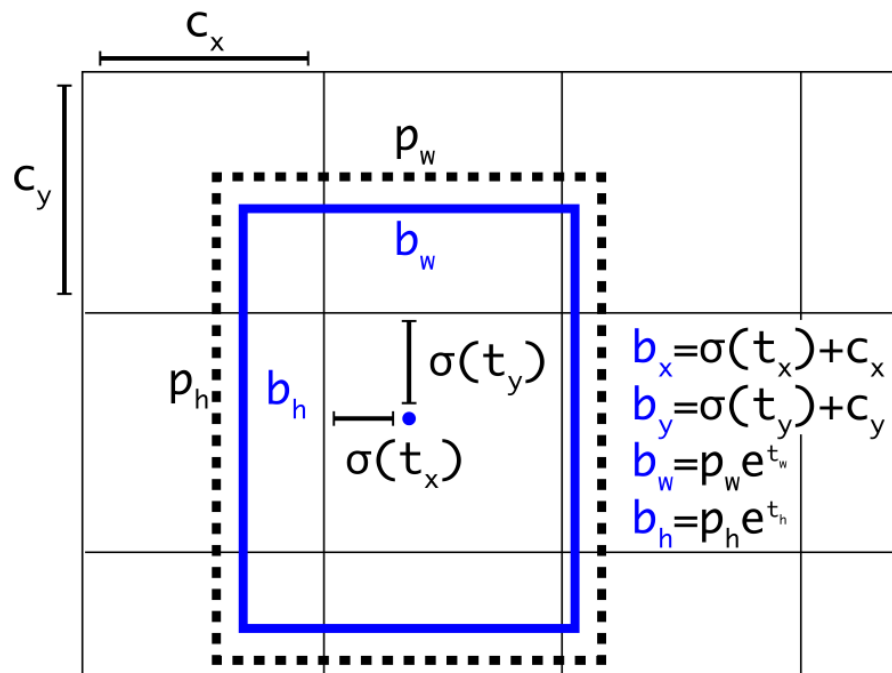
	Type	Filters	Size	Output
	Convolutional	32	$3 \times 3$	$256 \times 256$
	Convolutional	64	$3 \times 3 / 2$	$128 \times 128$
1x	Convolutional	32	$1 \times 1$	$128 \times 128$
	Convolutional	64	$3 \times 3$	
	Residual			
	Convolutional	128	$3 \times 3 / 2$	
2x	Convolutional	64	$1 \times 1$	$64 \times 64$
	Convolutional	128	$3 \times 3$	
	Residual			
	Convolutional	256	$3 \times 3 / 2$	
8x	Convolutional	128	$1 \times 1$	$32 \times 32$
	Convolutional	256	$3 \times 3$	
	Residual			
	Convolutional	512	$3 \times 3 / 2$	
8x	Convolutional	256	$1 \times 1$	$16 \times 16$
	Convolutional	512	$3 \times 3$	
	Residual			
	Convolutional	1024	$3 \times 3 / 2$	
4x	Convolutional	512	$1 \times 1$	$8 \times 8$
	Convolutional	1024	$3 \times 3$	
	Residual			
	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 $3 \times 3$ 和 $1 \times 1$ 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。



- 网络在特征图 ( $13 \times 13 = 169$ ) 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 $t_x, t_y$ 作激活(sigma)处理,不对 $t_w, t_h$ 作激活处理。

使用Dimension Clusters和Direct Location Prediction这两项Anchor Boxes改进方法，mAP获得了5%的提升。



# 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。



Multiple Grids may detect the same object  
NMS is used to remove multiple detections

NMS (Non-Maximum Suppression)

# 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 <sub>50</sub>	AP <sub>75</sub>	AP <sub>S</sub>	AP <sub>M</sub>	AP <sub>L</sub>
<i>Two-stage methods</i>							
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	<b>52.1</b>
<i>One-stage methods</i>							
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	<b>40.8</b>	<b>61.1</b>	<b>44.1</b>	<b>24.1</b>	<b>44.2</b>	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

# Darknet

一个轻型的开源深度学习框架

功能：CNN底层实现；YOLO目标检测；RNN；图像实例分割等

特点：

- 由C语言实现
- 没有依赖项（摄像头和视频处理需要OpenCV）
- 容易安装
- 移植性好
- 支持CPU与GPU(CUDA)两种计算方式

开发者：美国华盛顿大学 Joseph Redmon

Darknet 官网：<https://pjreddie.com/darknet/>