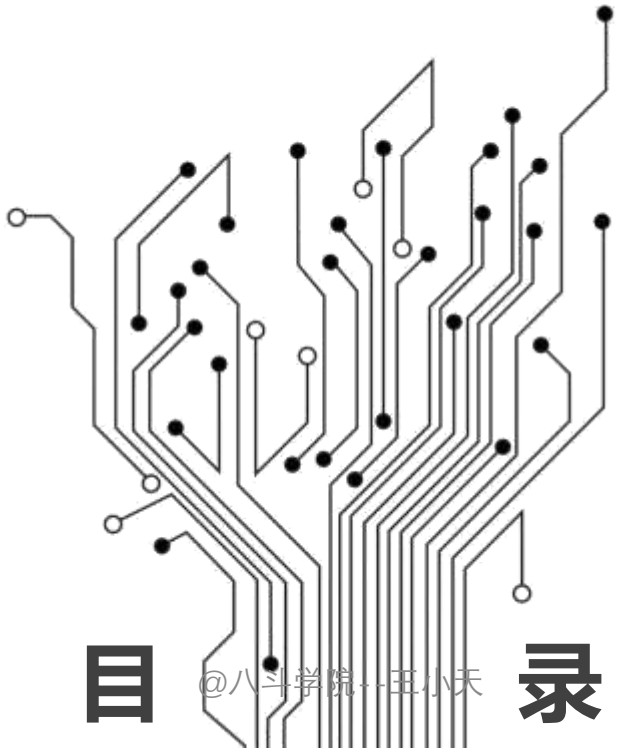
目标检测

@八斗学院--王小天(Michael)

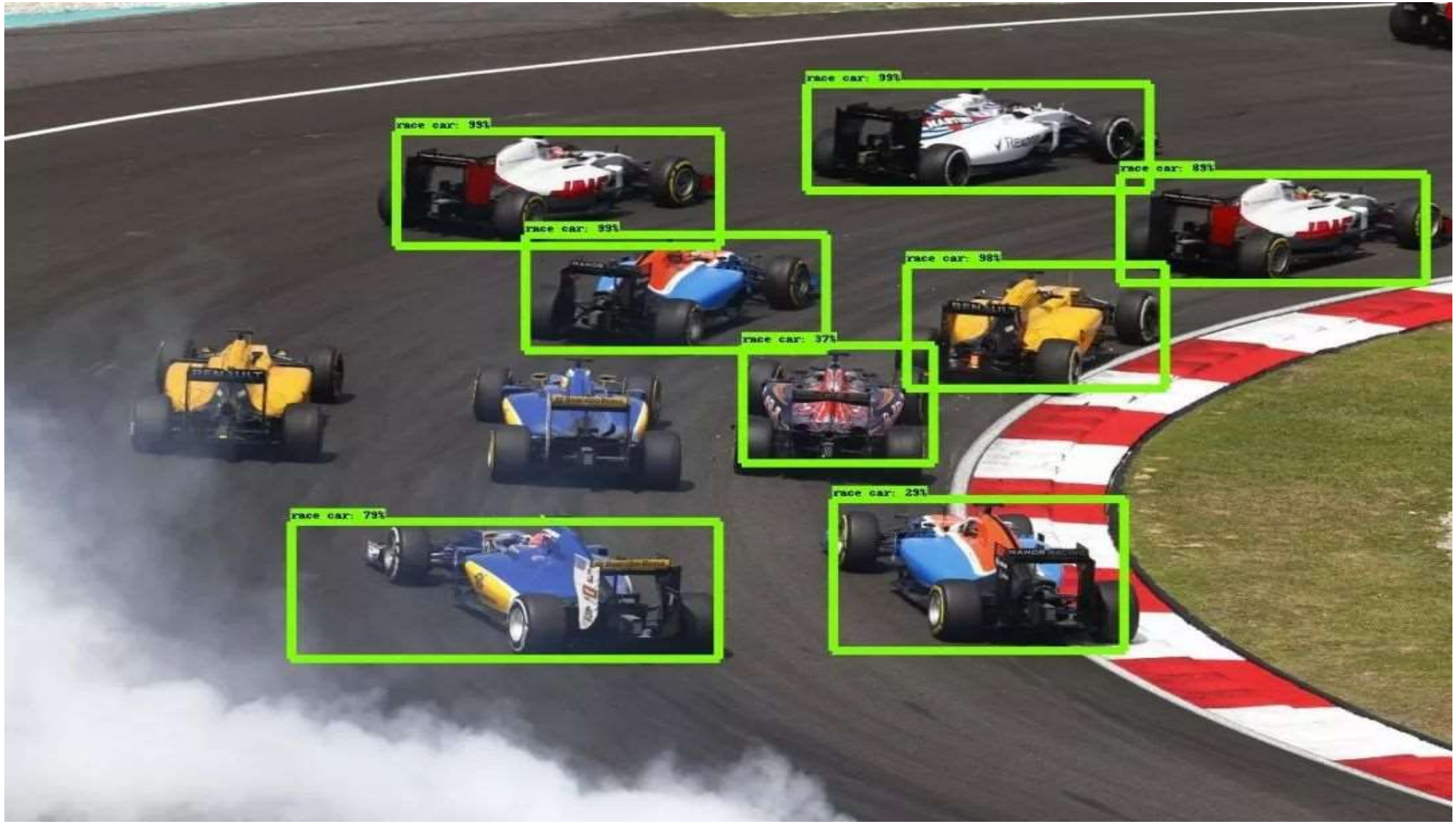
2024/07/14



目　录

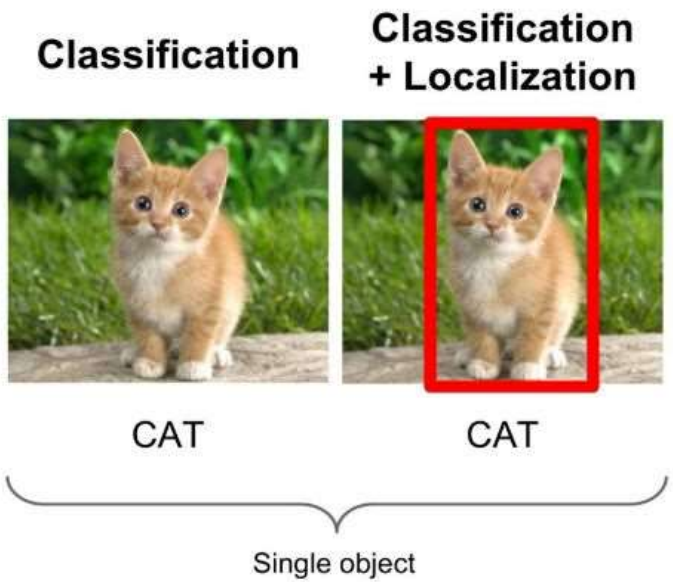
计算机视觉的五大应用

目标检测



目标检测

- 物体识别是要分辨出图片中有什么物体，输入是图片，输出是类别标签和概率。物体检测算法不仅要检测图片中有什么物体，还要输出物体的外框（x, y, width, height）来定位物体的位置。
- **object detection，**就是在给定的图片中精确找到物体所在位置，并标注出物体的类别。
- object detection要解决的问题就是物体在哪里以及是什么的整个流程问题。
- 然而，这个问题可不是那么容易解决的，物体的尺寸变化范围很大，摆放物体的角度，姿态不定，而且可以出现在图片的任何地方，更何况物体还可以是多个类别。



目标检测

目前学术和工业界出现的目标检测算法分成3类：

1. 传统的目标检测算法：Cascade + HOG/DPM + Haar/SVM以及上述方法的诸多改进、优化；

2. 候选区域/框 + 深度学习分类：通过提取候选区域，并对相应区域进行以深度学习方法为主的分类的方案，如：

- R-CNN（Selective Search + CNN + SVM）
- SPP-net（ROI Pooling）
- Fast R-CNN（Selective Search + CNN + ROI）
- Faster R-CNN（RPN + CNN + ROI）

3. 基于深度学习的回归方法：YOLO/SSD 等方法

IOU

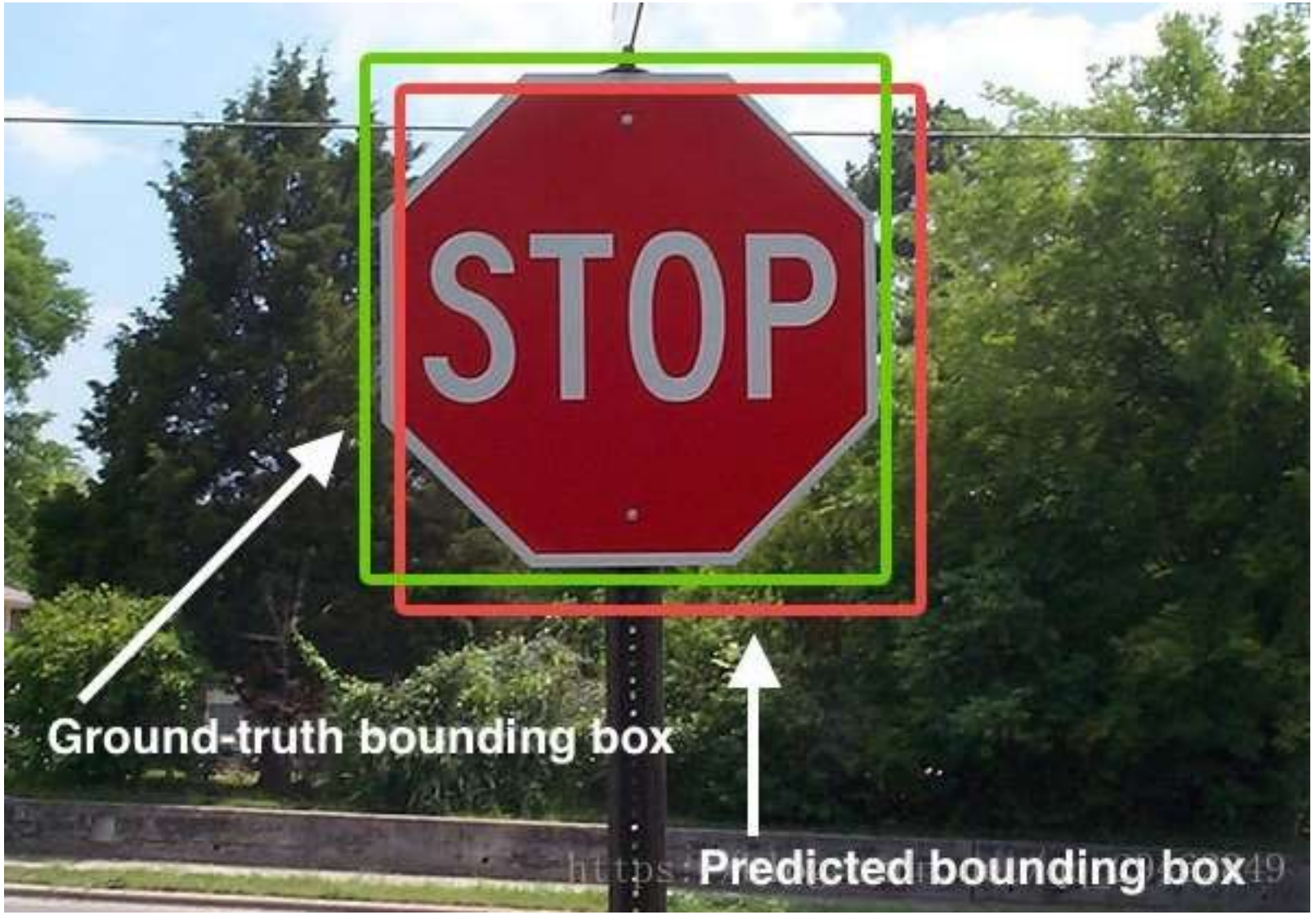
Intersection over Union是一种测量在特定数据集中检测相应物体准确度的一个标准。

IoU是一个简单的测量标准，只要是在输出中得出一个预测范围(bounding boxex)的任务都可以用IoU来进行测量。

为了可以使IoU用于测量任意大小形状的物体检测，我们需要：
1、 ground-truth bounding boxes（人为在训练集图像中标出要检测物体的大概范围）；
2、我们的算法得出的结果范围。

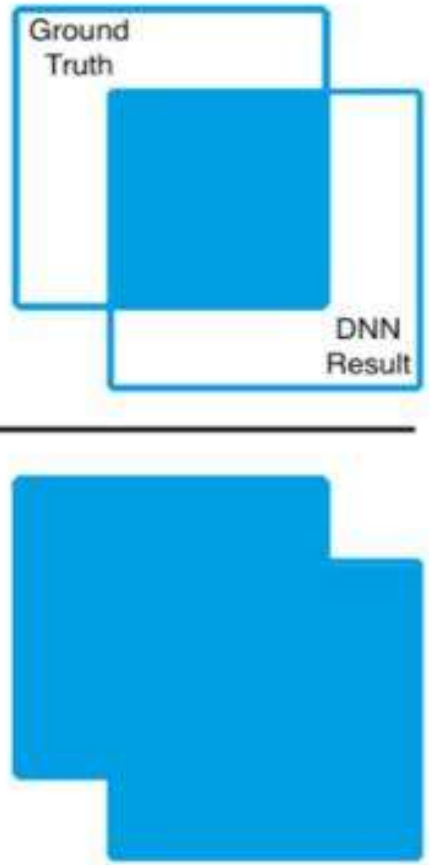
也就是说，这个标准用于测量真实和预测之间的相关度，相关度越高，该值越高。

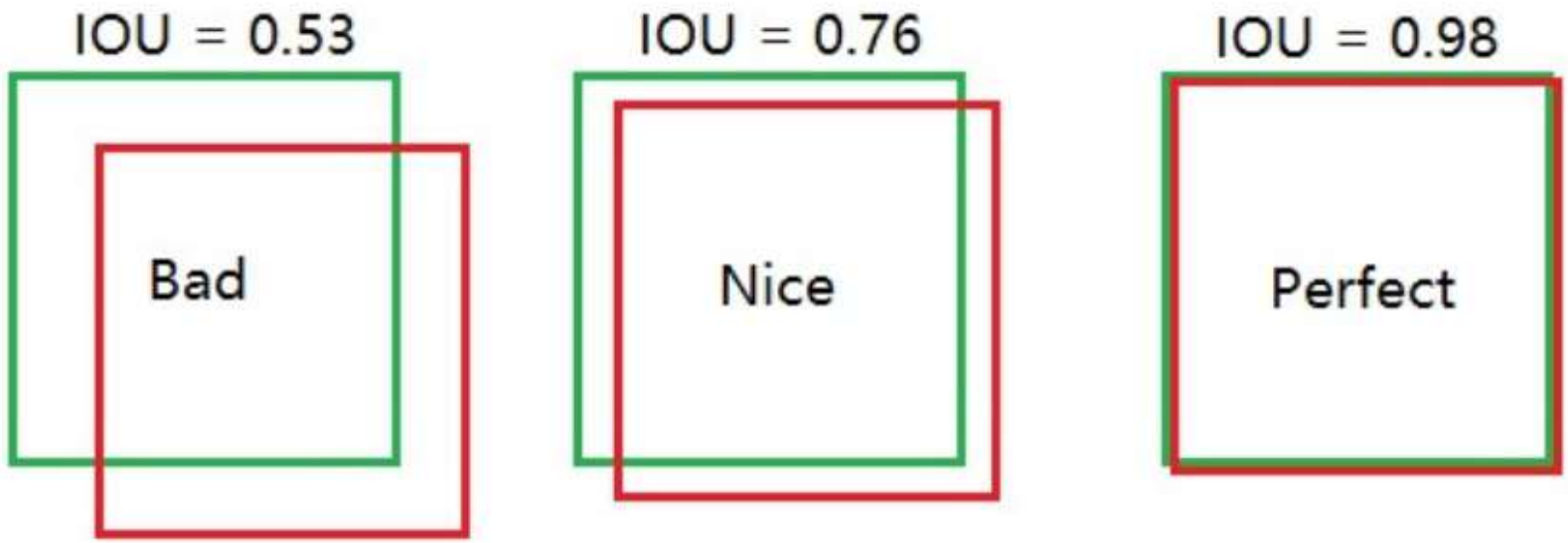
IOU



$$IOU = \frac{S_{交}}{S_{并}}$$

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$





TP TN FP FN

TP TN FP FN里面一共出现了4个字母，分别是T F P N。
T是True；
F是False；
P是Positive；
N是Negative。

T或者F代表的是该样本 是否被正确分类。
P或者N代表的是该样本 原本是正样本还是负样本。

TP（True Positives）意思就是被分为了正样本，而且分对了。
TN（True Negatives）意思就是被分为了负样本，而且分对了，
FP（False Positives）意思就是被分为了正样本，但是分错了（事实上这个样本是负样本）。
FN（False Negatives）意思就是被分为了负样本，但是分错了（事实上这个样本是正样本）。

在mAP计算的过程中主要用到了，TP、FP、FN这三个概念。

precision（精确度）和recall（召回率）

$$Precision = \frac{TP}{TP + FP}$$

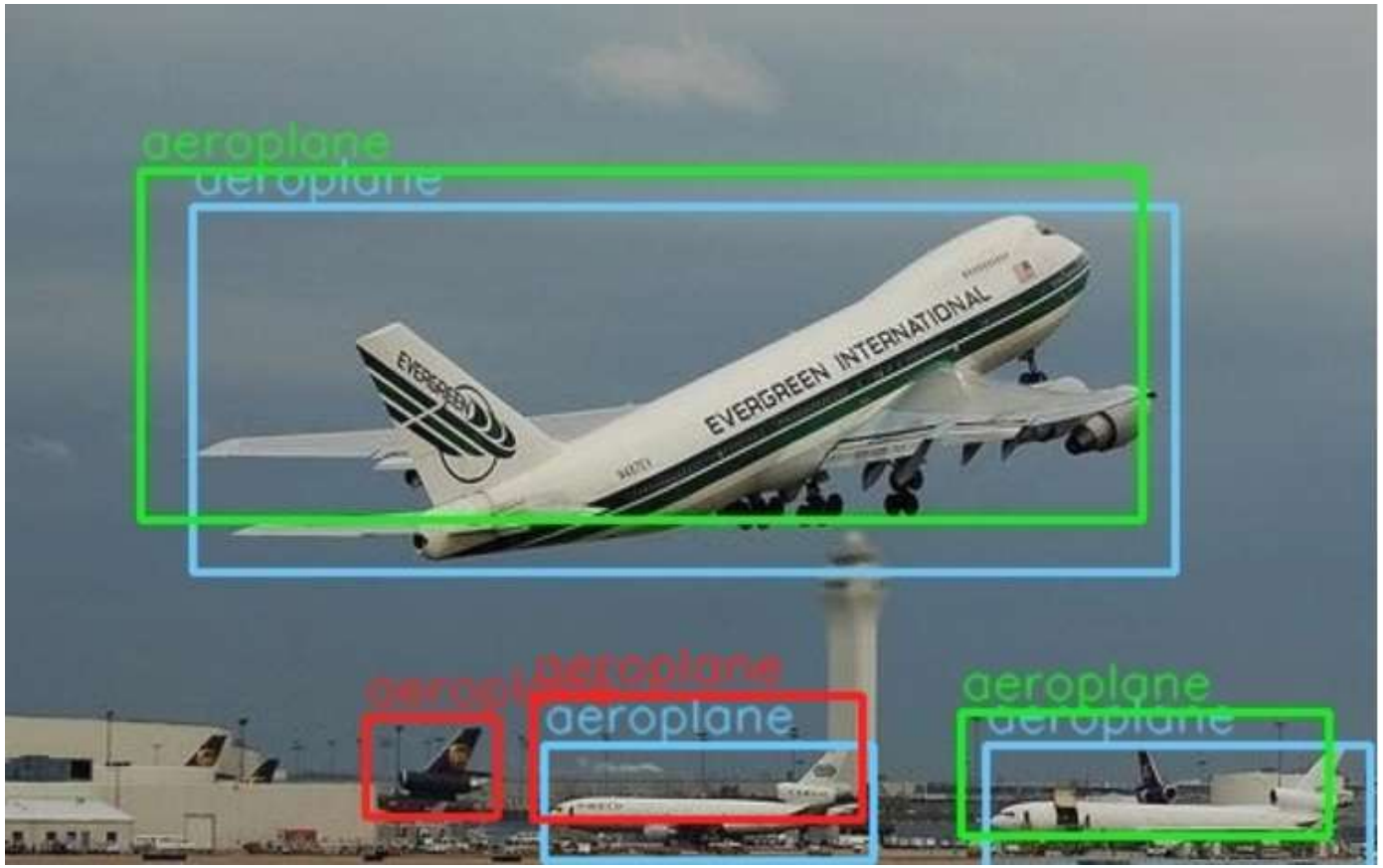
TP是分类器认为是正样本而且确实是正样本的例子，FP是分类器认为是正样本但实际上不是正样本的例子，Precision翻译成中文就是“分类器认为是正类并且确实是正类的部分 占 所有分类器认为是正类的比例”。

$$Recall = \frac{TP}{TP + FN}$$

TP是分类器认为是正样本而且确实是正样本的例子，FN是分类器认为是负样本但实际上不是负样本的例子，Recall翻译成中文就是“分类器认为是正类并且确实是正类的部分 占 所有确实是正类的比例”。

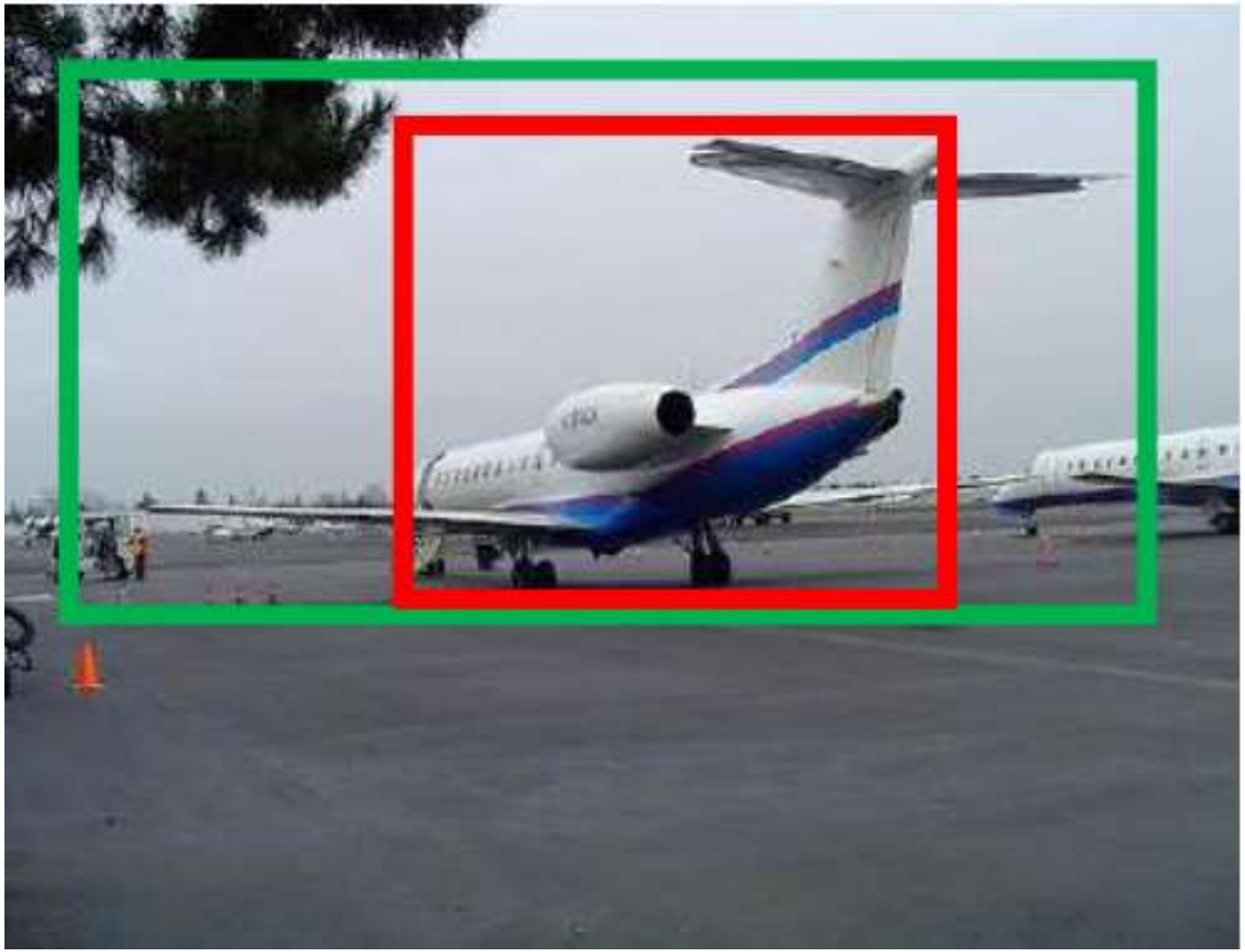
精度就是找得对，召回率就是找得全。

precision（精确度）和recall（召回率）



- 蓝色的框是真实框。绿色和红色的框是预测框，绿色的框是正样本，红色的框是负样本。
- 一般来讲，当预测框和真实框IOU>=0.5时，被认为是正样本。

边框回归Bounding-Box regression



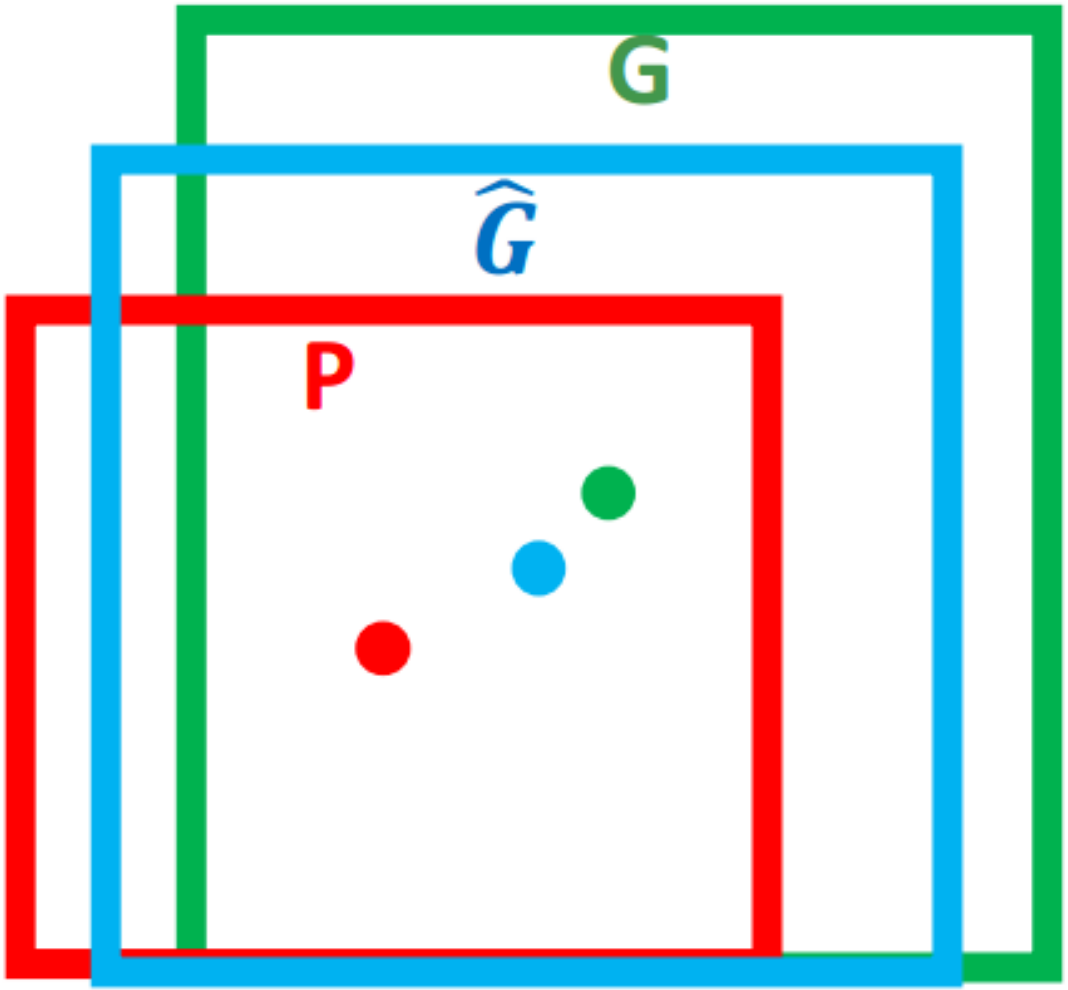
边框回归Bounding-Box regression

边框回归是什么?

- 对于窗口一般使用四维向量(x,y,w,h) 来表示， 分别表示窗口的中心点坐标和宽高。
- 红色的框 P 代表原始的Proposal,;
- 绿色的框 G 代表目标的 Ground Truth;

我们的目标是寻找一种关系使得输入原始的窗口 P 经过映射得到一个跟真实窗口 G 更接近的回归窗口G^。

所以，边框回归的目的即是:
给定(Px,Py,Pw,Ph)寻找一种映射f， 使得：
f(Px,Py,Pw,Ph)=(Gx^,Gy^,Gw^,Gh^)
并且 (Gx^,Gy^,Gw^,Gh^)≈(Gx,Gy,Gw,Gh)



边框回归Bounding-Box regression

边框回归怎么做?

比较简单的思路就是: 平移+尺度缩放

1. 先做平移$(\Delta x, \Delta y)$， $\Delta x = P_w d_x(P), \Delta y = P_h d_y(P)$ 这是R-CNN论文的:

$$\hat{G}_x = P_w d_x(P) + P_x, (1)$$

$$\hat{G}_y = P_h d_y(P) + P_y, (2)$$

2. 然后再做尺度缩放(S_w, S_h), $S_w = exp(d_w(P)), S_h = exp(d_h(P))$, 对应论文中:

$$\hat{G}_w = P_w exp(d_w(P)), (3)$$

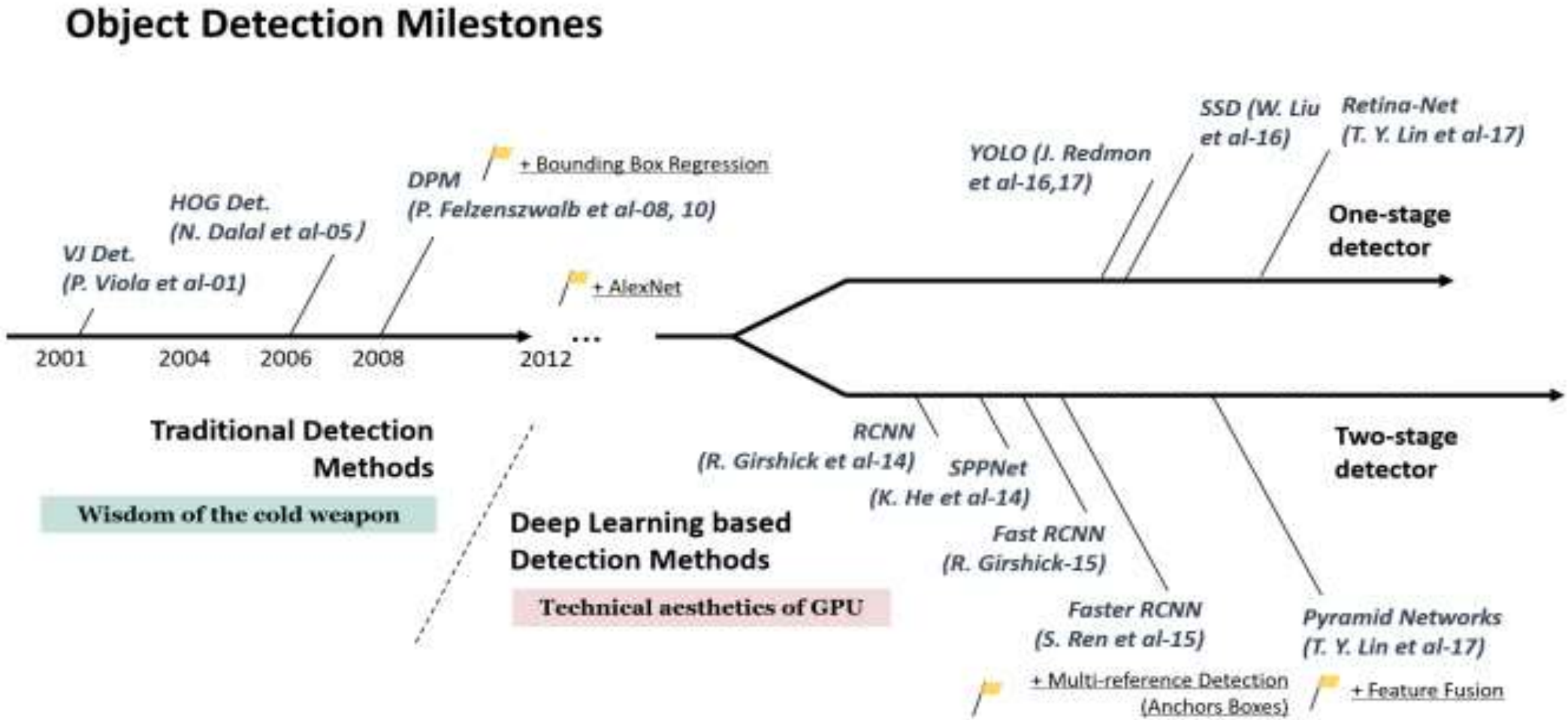
$$\hat{G}_h = P_h exp(d_h(P)), (4)$$

边框回归Bounding-Box regression

Input:
P=(Px,Py,Pw,Ph)
(注：训练阶段输入还包括 Ground Truth)

Output:
需要进行的平移变换和尺度缩放 dx,dy,dw,dh， 或者说是Δx,Δy,Sw,Sh 。

有了这四个变换我们就可以直接得到 Ground Truth。


Object Detection Milestones
VJ Det.
(P. Viola et al-01)
HOG Det.
(N. Dalal et al-05)
DPM
(P. Felzenszwalb et al-08, 10)
+ Bounding Box Regression
+ AlexNet
2001
2004
2006
2008
2012
...
Traditional Detection Methods
Wisdom of the cold weapon
Deep Learning based Detection Methods
Technical aesthetics of GPU
YOLO (J. Redmon et al-16,17)
SSD (W. Liu et al-16)
Retina-Net (T. Y. Lin et al-17)
One-stage detector
RCNN
(R. Girshick et al-14)
SPPNet
(K. He et al-14)
Fast RCNN
(R. Girshick-15)
Faster RCNN
(S. Ren et al-15)
Pyramid Networks
(T. Y. Lin et al-17)
+ Multi-reference Detection (Anchors Boxes)
+ Feature Fusion
Two-stage detector

Two stage

候选区域/框 + 深度学习分类：通过提取候选区域，并对相应区域进行以深度学习方法为主的分类的方案，如：

- R-CNN（Selective Search + CNN + SVM）
- SPP-net（ROI Pooling）
- Fast R-CNN（Selective Search + CNN + ROI）
- Faster R-CNN（RPN + CNN + ROI）

拓展--Selective Search

Selective Search 通过颜色、纹理、大小等特征的相似度把图像分成许多个不同的区域。目标检测算法可以从这些区域中检测对象，加快检测速度。

拓展--Selective Search

第一步，使用 **Efficient graph-based image segmentation 算法（基于图的图像分割）**生成初始区域集R，同时设置区域相似S为空集。
第二步，对于相邻的区域（ri，rj），计算他们的相似度s（ri，rj），并添加到相似集S中
第三步，获取S中相似度最高的两个区域s（ri，rj）
第四步，合并区域ri和rj成rt
第五步，删除与ri和rj有关的相似度
第六步，计算新区域rt与相邻区域的相似度St，并把St添加到S中，rt添加到区域集R中
第七步，如果S不为空，返回第三步。

总的来说，Selective Search 算法不断合并相似的区域。

Two stage

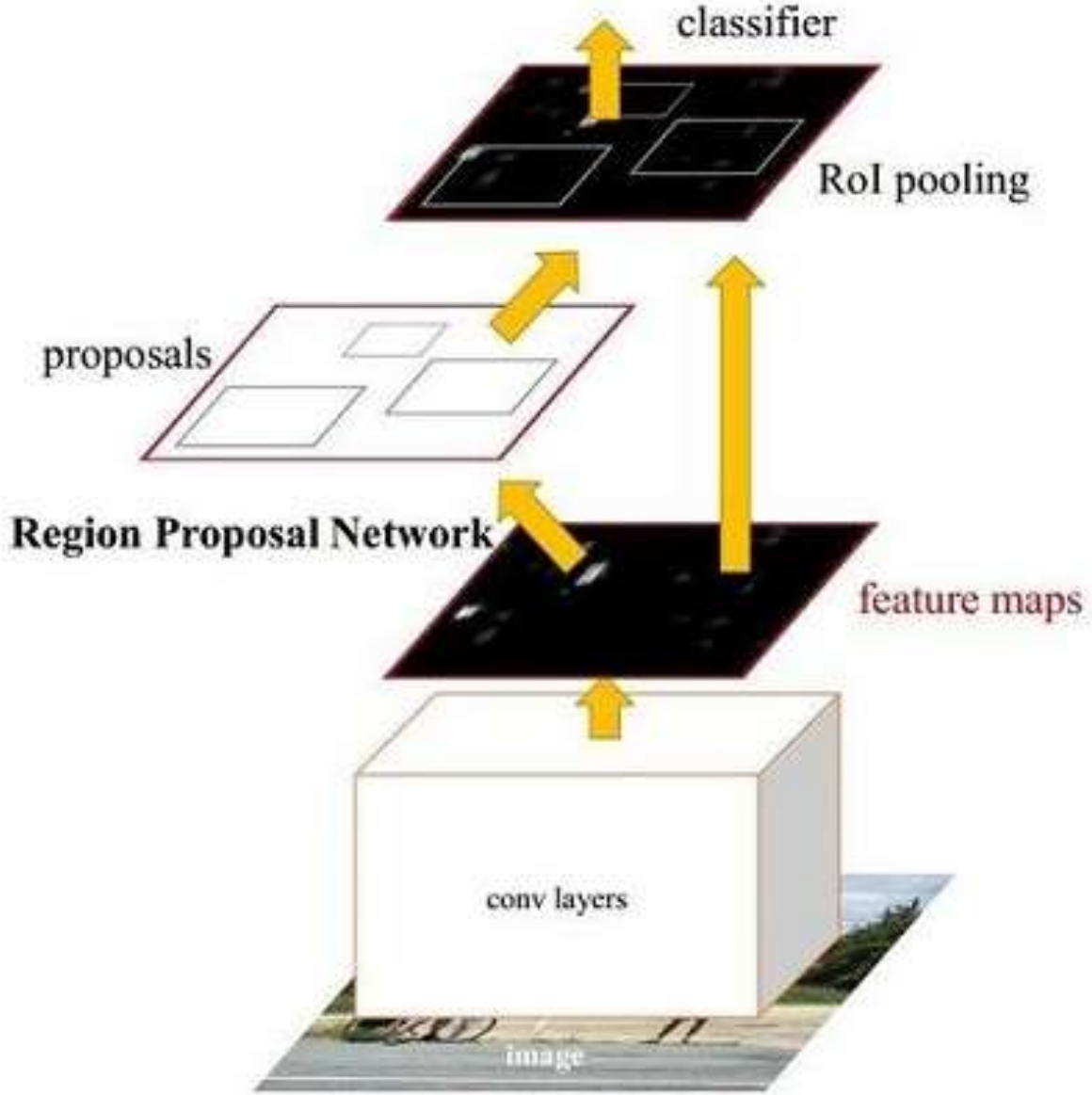
候选区域/框 + 深度学习分类：通过提取候选区域，并对相应区域进行以深度学习方法为主的分类的方案，如：

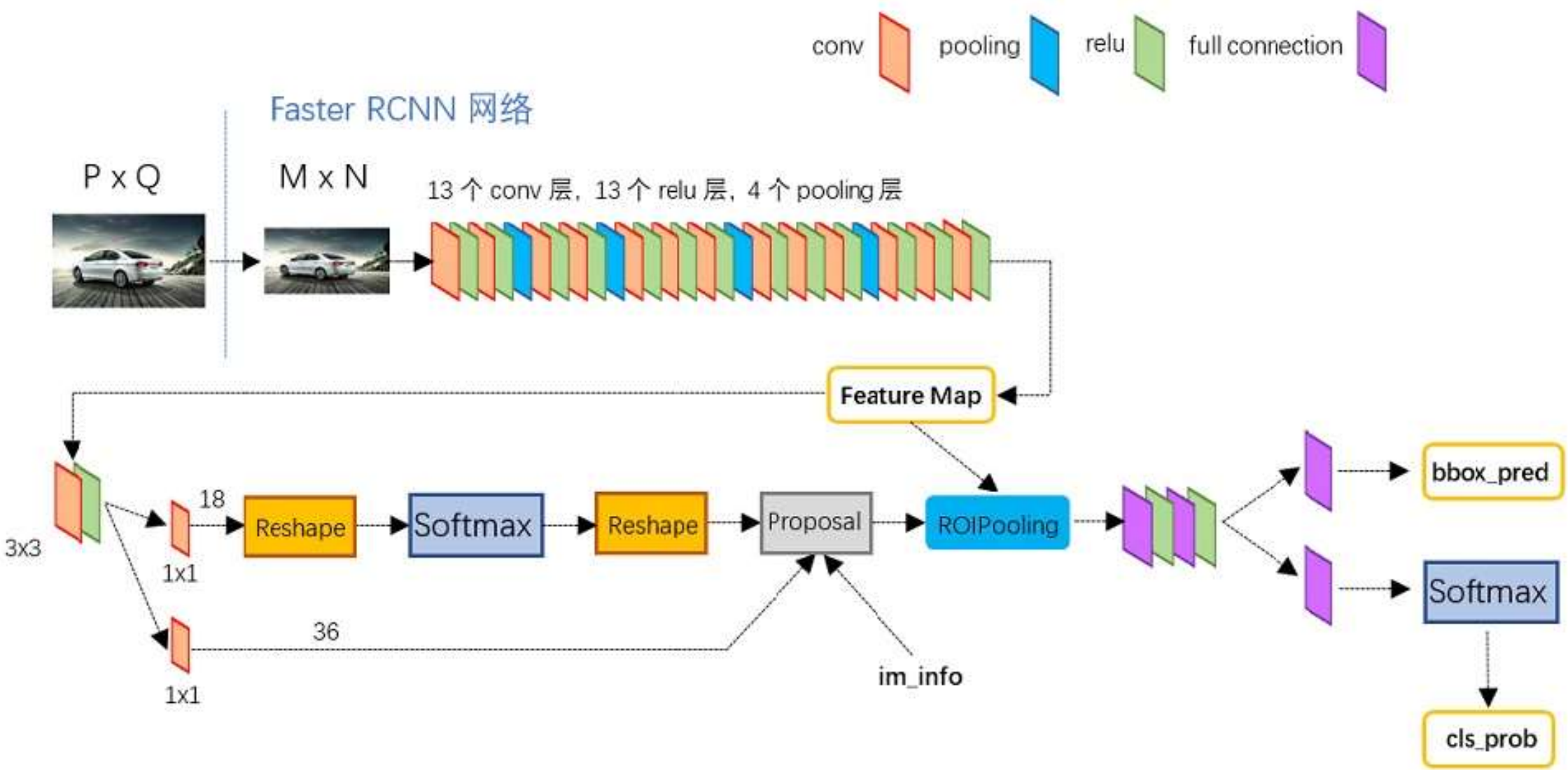
- R-CNN（Selective Search + CNN + SVM）
- SPP-net（ROI Pooling）
- Fast R-CNN（Selective Search + CNN + ROI）
- Faster R-CNN（RPN + CNN + ROI）

Faster-RCNN

Faster RCNN可以分为4个主要内容：

1. **Conv layers**：作为一种CNN网络目标检测方法，Faster RCNN首先使用一组基础的conv+relu+pooling层提取image的feature maps。该feature maps被共享用于后续RPN层和全连接层。
2. **Region Proposal Networks（RPN）**：RPN网络用于生成region proposals。通过softmax判断anchors属于positive或者negative，再利用bounding box regression修正anchors获得精确的proposals。
3. **Roi Pooling**：该层收集输入的feature maps和proposals，综合这些信息后提取proposal feature maps，送入后续全连接层判定目标类别。
4. **Classification**：利用proposal feature maps计算proposal的类别，同时再次bounding box regression获得检测框最终的精确位置。



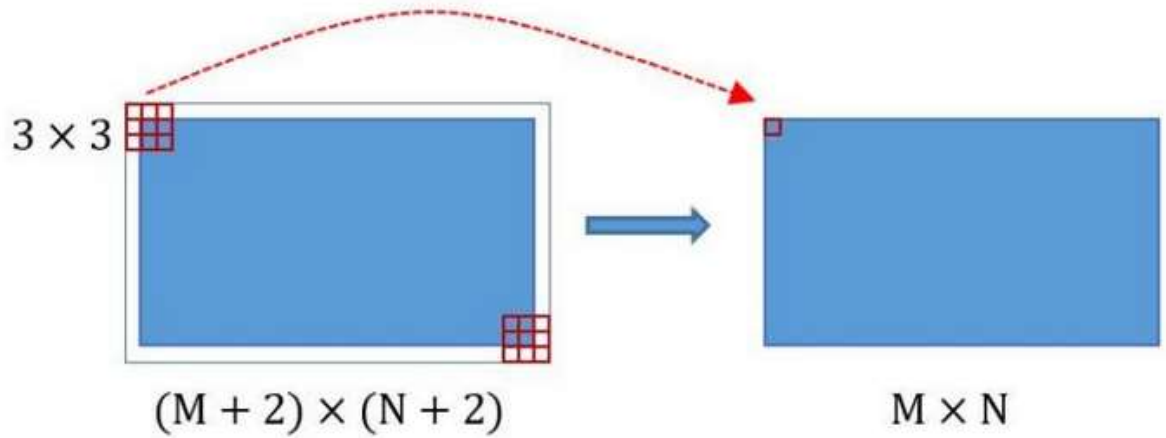

conv
pooling
relu
full connection
Faster RCNN 网络
P x Q
M x N
13 个 conv 层，13 个 relu 层，4 个 pooling 层
Feature Map
3x3
18
1x1
Reshape
Softmax
Reshape
Proposal
ROIPooling
bbox_pred
Softmax
cls_prob
36
1x1
im_info

1 Conv layers

Conv layers包含了conv，pooling，relu三种层。共有13个conv层，13个relu层，4个pooling层。

在Conv layers中：
1. 所有的conv层都是：kernel_size=3，pad=1，stride=1
2. 所有的pooling层都是：kernel_size=2，pad=1，stride=2

在Faster RCNN Conv layers中对所有的卷积都做了pad处理（pad=1，即填充一圈0），导致原图变为 (M+2)x(N+2)大小，再做3x3卷积后输出MxN 。正是这种设置，导致Conv layers中的conv层不改变输入和输出矩阵大小。



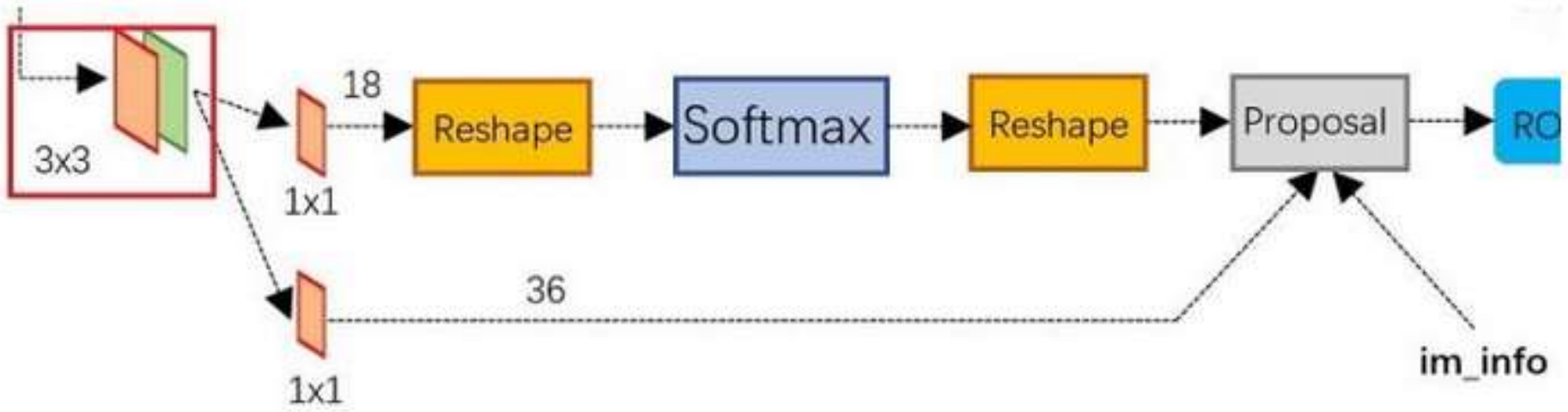
类似的是，Conv layers中的pooling层kernel_size=2，stride=2。
这样每个经过pooling层的MxN矩阵，都会变为(M/2)x(N/2)大小。

综上所述，在整个Conv layers中，conv和relu层不改变输入输出大小，只有pooling层使输出长宽都变为输入的1/2。

那么，**一个MxN大小的矩阵经过Conv layers固定变为(M/16)x(N/16)。**
这样Conv layers生成的feature map都可以和原图对应起来。

2. 区域生成网络Region Proposal Networks(RPN)

经典的检测方法生成检测框都非常耗时。直接使用RPN生成检测框，是Faster R-CNN的巨大优势，能极大提升检测框的生成速度。



- 可以看到RPN网络实际分为2条线：
  1. 上面一条通过softmax分类anchors，获得positive和negative分类；
  2. 下面一条用于计算对于anchors的bounding box regression偏移量，以获得精确的proposal。
- 而最后的Proposal层则负责综合positive anchors和对应bounding box regression偏移量获取proposals，同时剔除太小和超出边界的proposals。
- 其实整个网络到了Proposal Layer这里，就完成了相当于目标定位的功能。

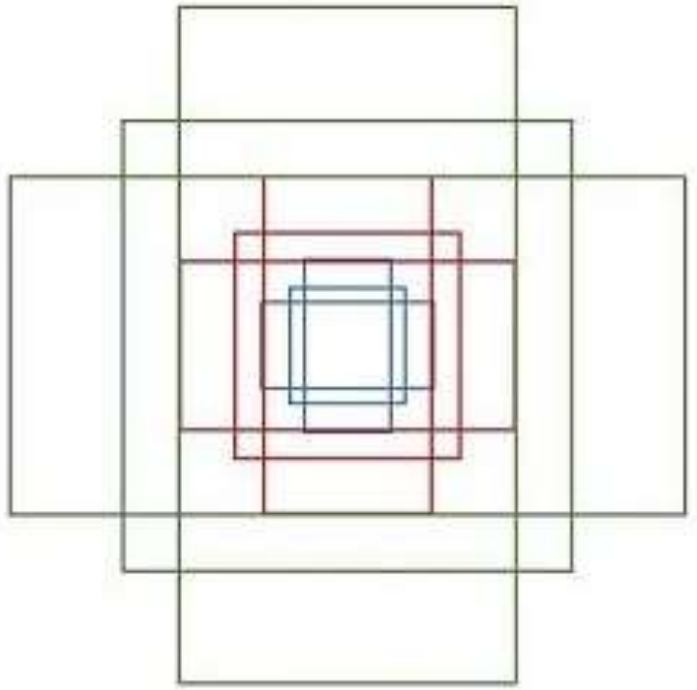
anchors

RPN网络在CNN卷积后，对每个点，上采样映射到原始图像一个区域，找到这个区域的中心位置，然后基于这个中心位置按规则选取9种anchor box。

9个矩形共有3种面积：128,256,512;
3种形状：长宽比大约为1:1, 1:2, 2:1。
(不是固定比例，可调)

每行的4个值表示矩形左上和右下角点坐标。

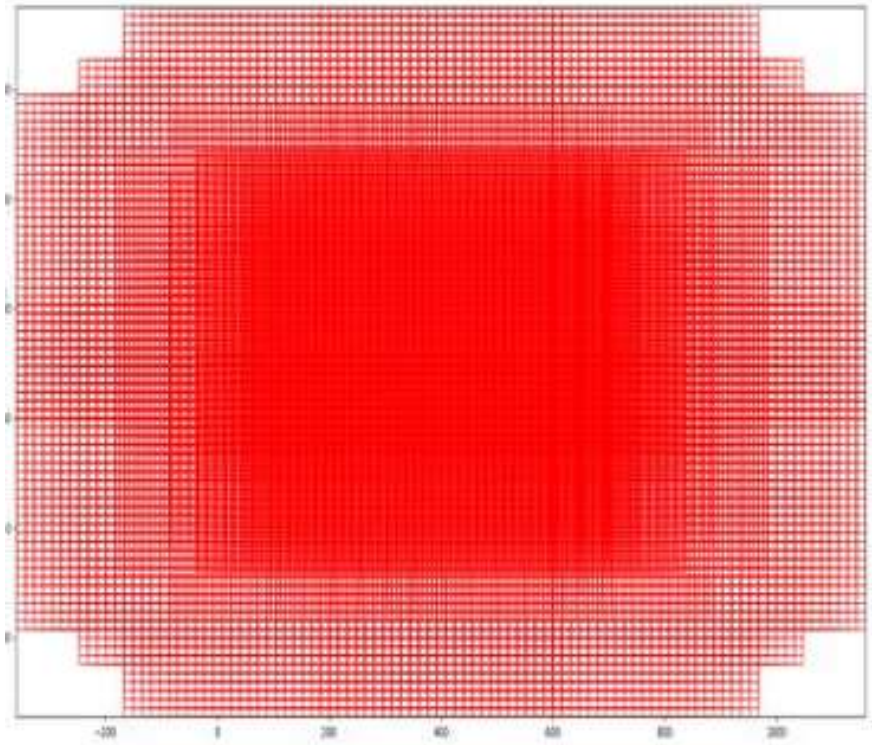
```
[[ -84.  -40.   99.   55.]
 [-176.  -88.  191.  103.]
 [-360. -184.  375.  199.]
 [ -56.  -56.   71.   71.]
 [-120. -120.  135.  135.]
 [-248. -248.  263.  263.]
 [ -36.  -80.   51.   95.]
 [ -80. -168.   95.  183.]
 [-168. -344.  183.  359.]]
```



Faster-RCNN：Region Proposal Networks(RPN)

anchors

遍历Conv layers获得的feature maps，为每一个点都配备这9种anchors作为初始的检测框。



softmax判定positive与negative

其实RPN最终就是在原图尺度上，设置了密密麻麻的候选Anchor。然后用cnn去判断哪些Anchor是里面有目标的positive anchor，哪些是没目标的negative anchor。所以，仅仅是个二分类而已。

softmax判定positive与negative

可以看到其conv的num_output=18，也就是经过该卷积的输出图像为WxHx18大小。

这也就刚好对应了feature maps每一个点都有9个anchors，同时每个anchors又有可能是positive和negative，所有这些信息都保存在WxHx(9*2)大小的矩阵。

为何这样做？后面接softmax分类获得positive anchors，也就相当于初步提取了检测目标候选区域box（一般认为目标在positive anchors中）。

softmax判定positive与negative

那么为何要在softmax前后都接一个reshape layer？其实只是为了便于softmax分类。

前面的positive/negative anchors的矩阵，其在caffe中的存储形式为[1, 18, H, W]。而在softmax分类时需要进行positive/negative二分类，所以reshape layer会将其变为[1, 2, 9xH, W]大小，即单独“腾空”出来一个维度以便softmax分类，之后再reshape回复原状。

综上所述，RPN网络中利用anchors和softmax初步提取出positive anchors作为候选区域。

Faster-RCNN：Region Proposal Networks(RPN)

对proposals进行bounding box regression

可以看到conv的 num_output=36，即经过该卷积输出图像为WxHx36。
这里相当于feature maps每个点都有9个anchors，每个anchors又都有4个用于回归的变换量：

$$[d_x(A), d_y(A), d_w(A), d_h(A)]$$

Faster-RCNN：Region Proposal Networks(RPN)

Proposal Layer

Proposal Layer负责综合所有变换量和positive anchors，计算出精准的proposal，送入后续RoI Pooling Layer。

Proposal Layer有4个输入：
1. positive vs negative anchors分类器结果rpn_cls_prob_reshape，
2. 对应的bbox reg的变换量rpn_bbox_pred，
3. im_info
4. 参数feature_stride=16

im_info：对于一副任意大小PxQ图像，传入Faster RCNN前首先reshape到固定MxN，im_info=[M, N, scale_factor]则保存了此次缩放的所有信息。
输入图像经过Conv Layers，经过4次pooling变为WxH=(M/16)x(N/16)大小，其中feature_stride=16则保存了该信息，用于计算anchor偏移量。

Faster-RCNN：Region Proposal Networks(RPN)

Proposal Layer

Proposal Layer 按照以下顺序依次处理：

1. 利用变换量对所有的positive anchors做bbox regression回归
2. 按照输入的positive softmax scores由大到小排序anchors，提取前pre_nms_topN(e.g. 6000)个anchors，即提取修正位置后的positive anchors。
3. 对剩余的positive anchors进行NMS（non-maximum suppression）。
4. 之后输出proposal。

严格意义上的检测应该到此就结束了，后续部分应该属于识别了。

Faster-RCNN：Region Proposal Networks(RPN)

RPN网络结构，总结起来：

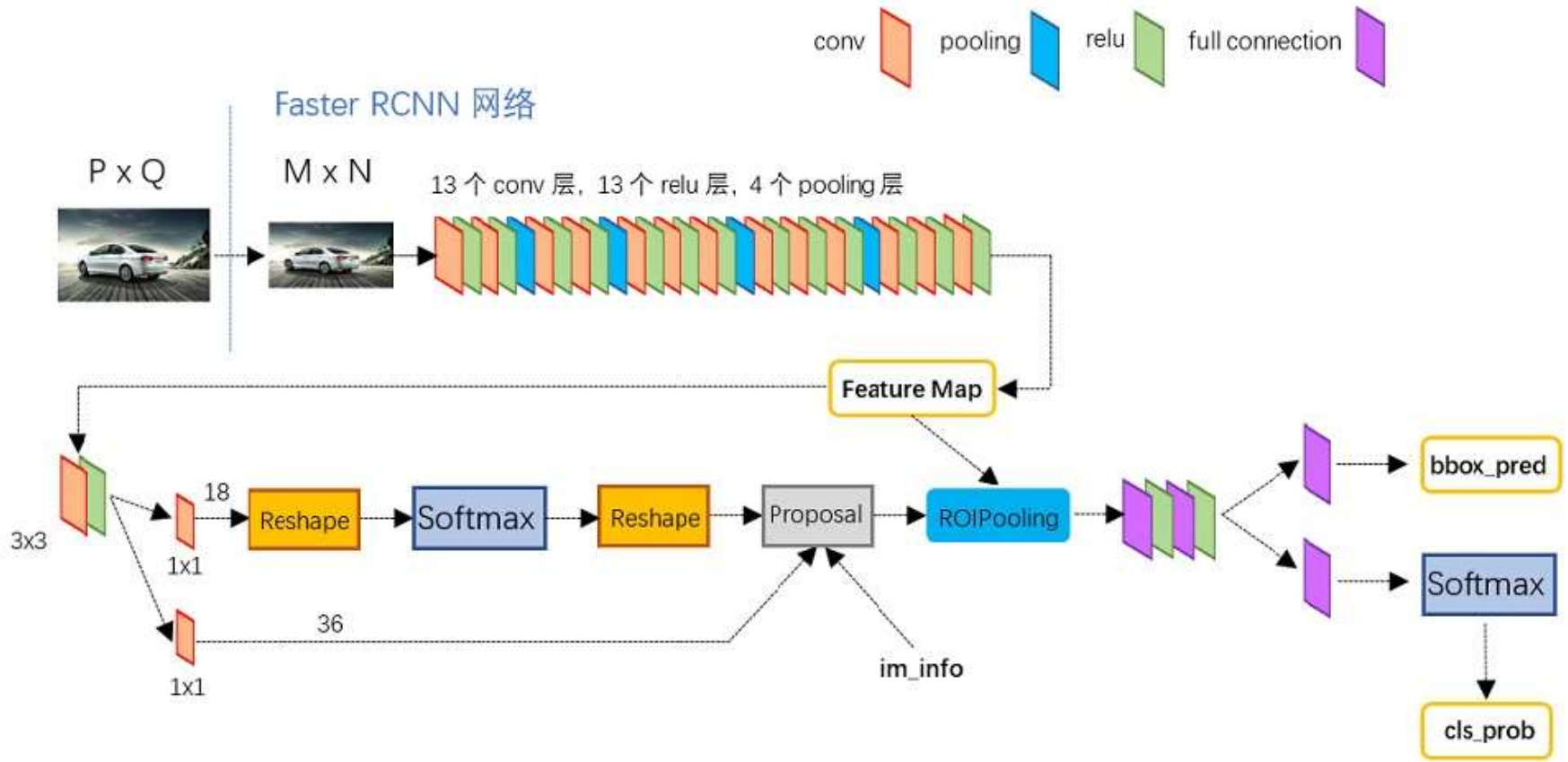
生成anchors -> softmax分类器提取positvie anchors -> bbox reg回归positive anchors -> Proposal Layer生成proposals

Faster-RCNN：Roi pooling

RoI Pooling层则负责收集proposal，并计算出proposal feature maps，送入后续网络。

RoI pooling层有2个输入：

1. 原始的feature maps
2. RPN输出的proposal boxes（**大小各不相同**）



Faster-RCNN：Roi pooling

为何需要RoI Pooling?

对于传统的CNN（如AlexNet和VGG），当网络训练好后输入的图像尺寸必须是固定值，同时网络输出也是固定大小的vector or matrix。如果输入图像大小不定，这个问题就变得比较麻烦。
有2种解决办法：

1. 从图像中crop一部分传入网络将图像（破坏了图像的完整结构）
2. warp成需要的大小后传入网络（破坏了图像原始形状信息）

crop

warp

Faster-RCNN：Roi pooling

RoI Pooling原理

新参数pooled_w、pooled_h和spatial_scale（1/16）

RoI Pooling layer forward过程：

1. 由于proposal是对应M*N**尺度**的，所以首先使用spatial_scale参数将其映射回(M/16)*(N/16)大小的feature map**尺度**；
2. 再将每个proposal对应的feature map区域水平分为pooled_w * pooled_h的网格；
3. 对网格的每一份都进行max pooling处理。

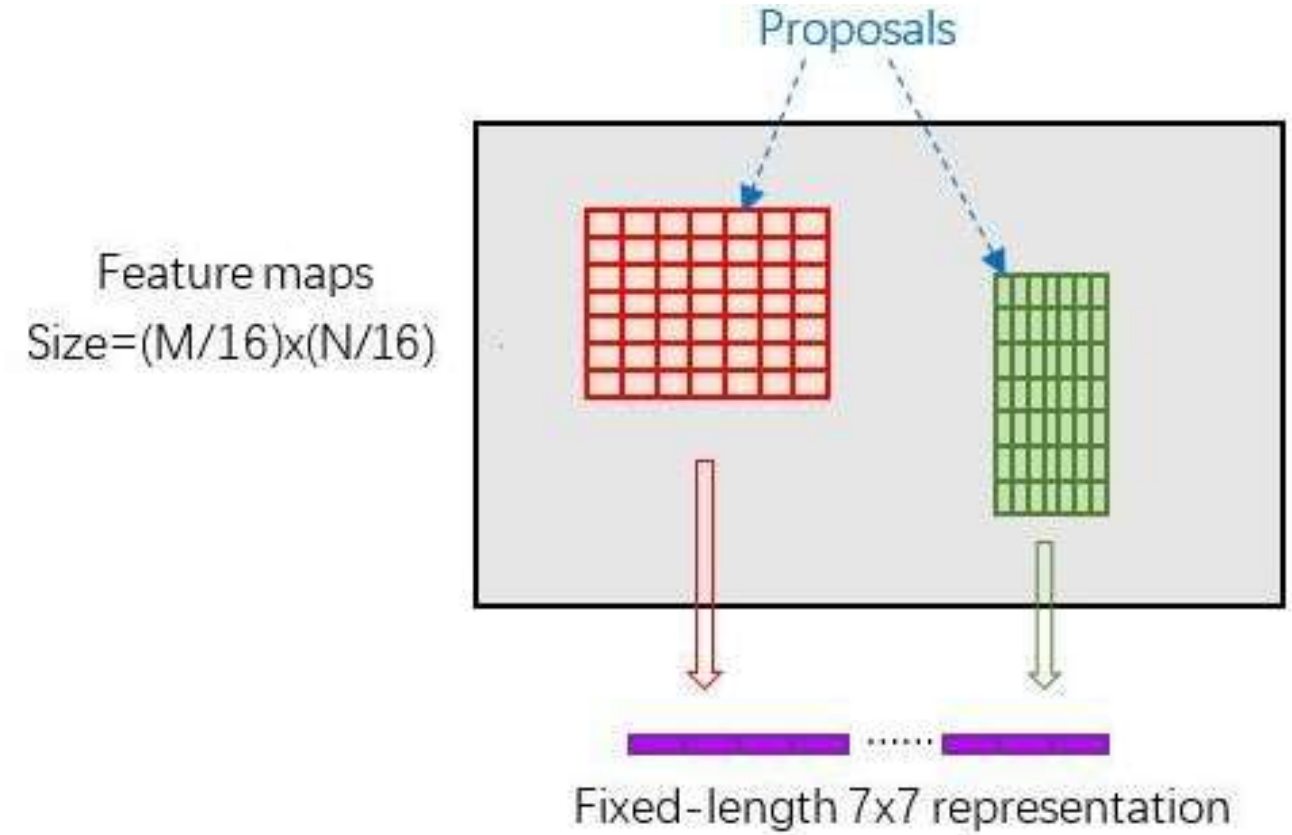
这样处理后，即使大小不同的proposal输出结果都是pooled_w * pooled_h固定大小，实现了固定长度输出。

Faster-RCNN：Roi pooling

2. 再将每个proposal对应的feature map区域水平分为pooled_w * pooled_h的网格;

3. 对网格的每一份都进行max pooling处理。

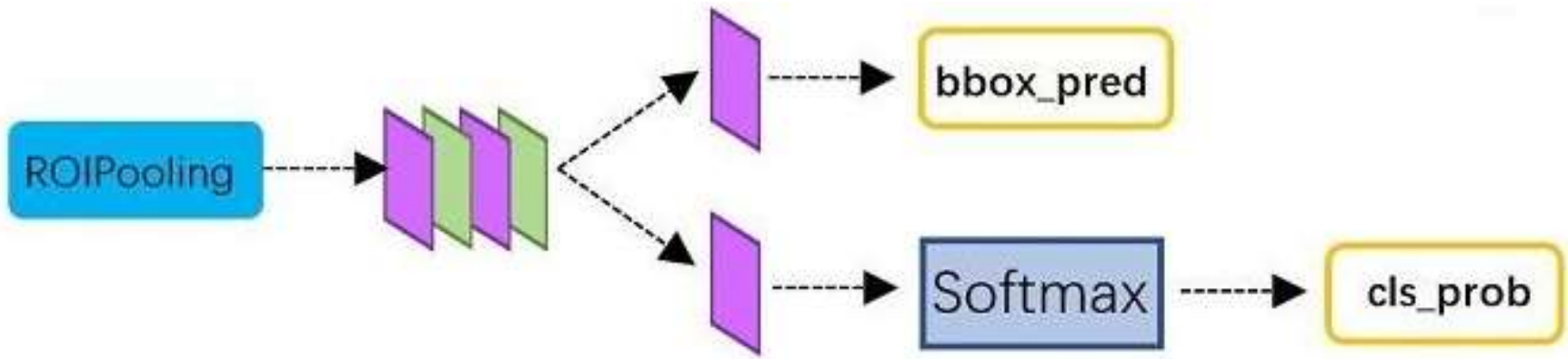
这样处理后，即使大小不同的proposal输出结果都是pooled_w * pooled_h固定大小，实现了固定长度输出。



Faster-RCNN: Classification

Classification部分利用已经获得的proposal feature maps，通过full connect层与softmax计算每个proposal具体属于那个类别（如人，车，电视等），输出cls_prob概率向量；

同时再次利用bounding box regression获得每个proposal的位置偏移量bbox_pred，用于回归更加精确的目标检测框。



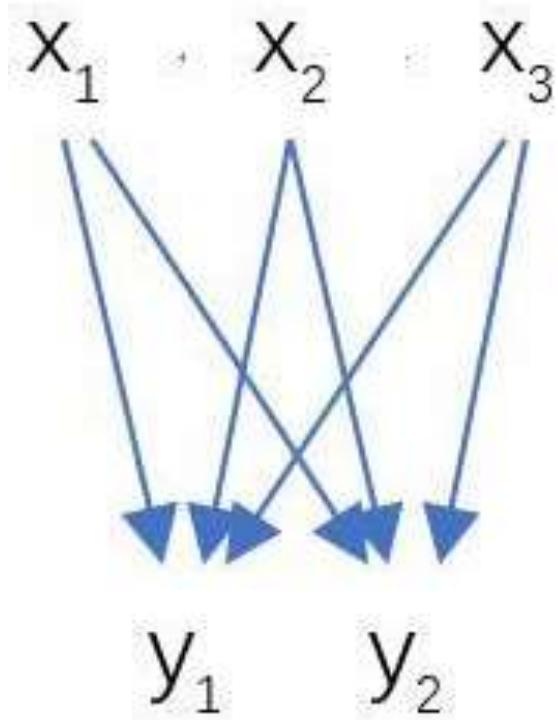
Faster-RCNN: Classification

从RoI Pooling获取到pooled_w * pooled_h大小的proposal feature maps后，送入后续网络，做了如下2件事：

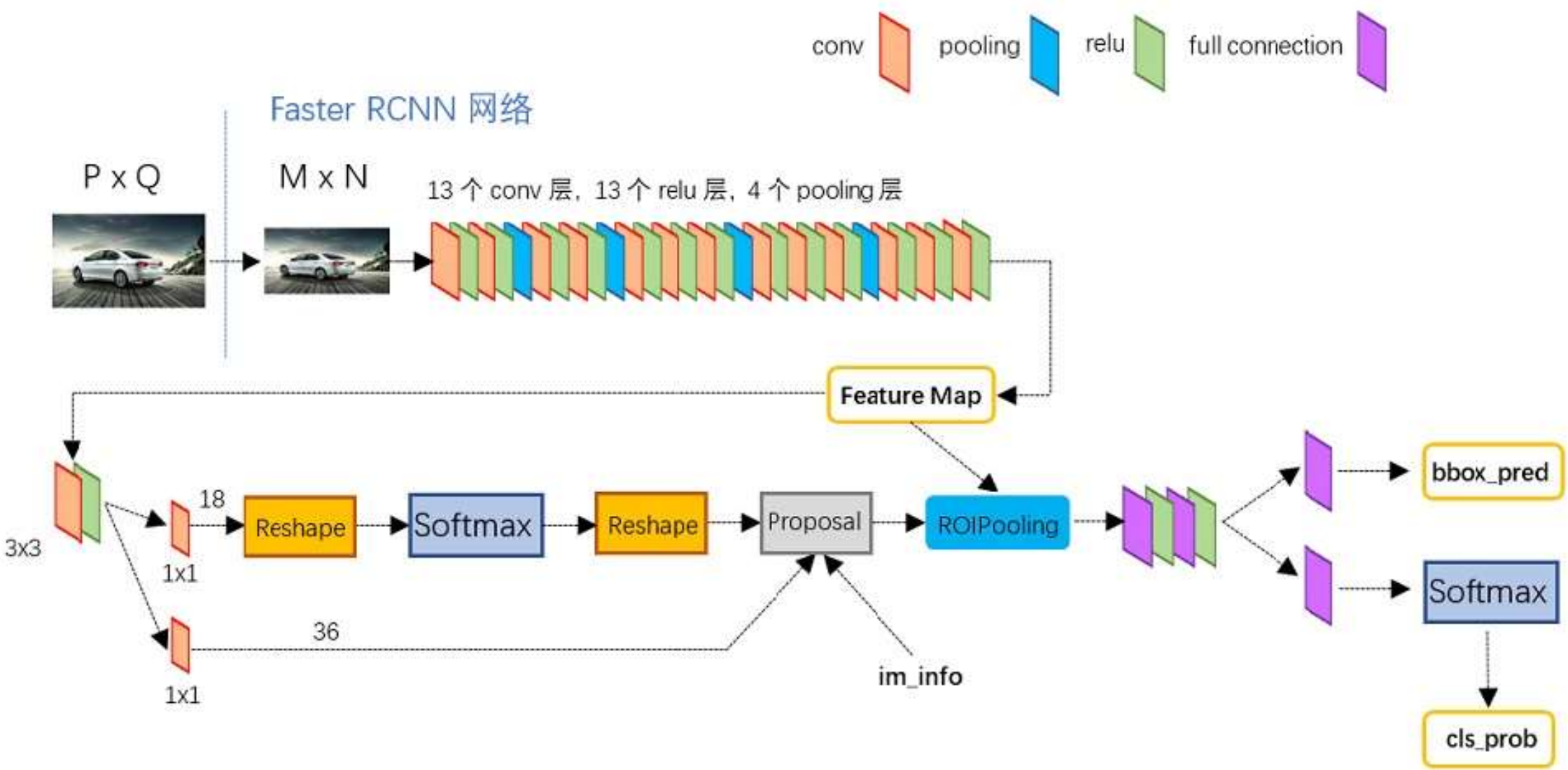
1. 通过全连接和softmax对proposals进行分类，这实际上已经是识别的范畴了
2. 再次对proposals进行bounding box regression，获取更高精度的预测框

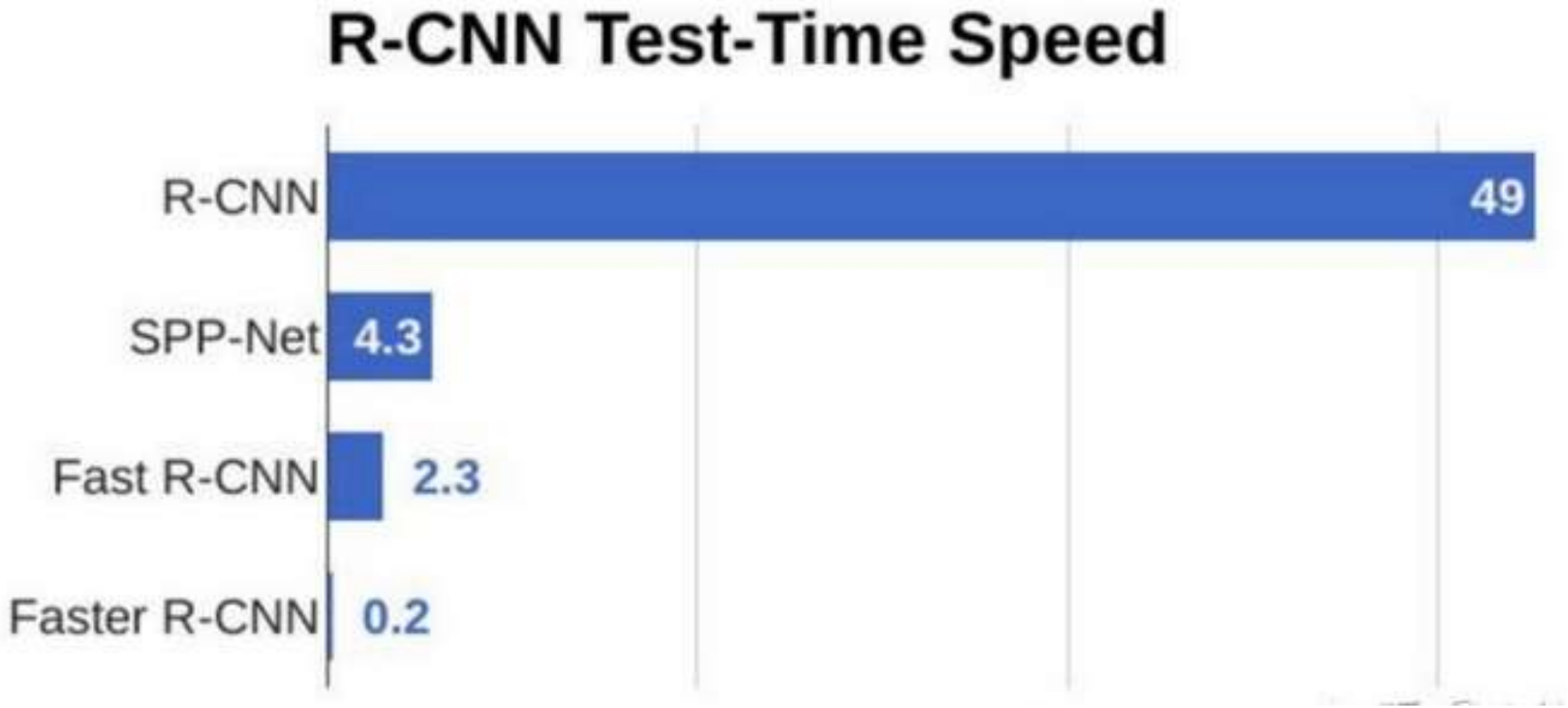
全连接层InnerProduct layers:

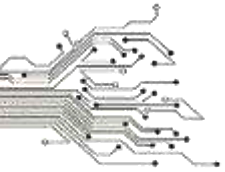
$$(x_1 \quad x_2 \quad x_3)\begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{pmatrix} + (b_1 \quad b_2) = (y_1 \quad y_2)$$



输入X和输出Y是固定大小。所以，这也就印证了之前Roi Pooling的必要性


conv
pooling
relu
full connection
Faster RCNN 网络
P x Q
M x N
13 个 conv 层，13 个 relu 层，4 个 pooling 层
Feature Map
3x3
18
1x1
Reshape
Softmax
Reshape
Proposal
ROIPooling
bbox_pred
Softmax
cls_prob
36
1x1
im_info

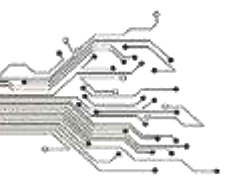

R-CNN Test-Time Speed
R-CNN
49
SPP-Net
4.3
Fast R-CNN
2.3
Faster R-CNN
0.2



two stage和one stage

two-stage：two-stage算法会先使用一个网络生成proposal，如selective search和RPN网络，RPN出现后，ss方法基本就被摒弃了。RPN网络接在图像特征提取网络backbone后，会设置RPN loss（bbox regression loss+classification loss）对RPN网络进行训练，RPN生成的proposal再送到后面的网络中进行更精细的bbox regression和classification。

one-stage：One-stage追求速度舍弃了two-stage架构，即不再设置单独网络生成proposal，而是直接在feature map上进行密集抽样，产生大量的先验框，如YOLO的网格方法。这些先验框没有经过两步处理，且框的尺寸往往是人为规定。

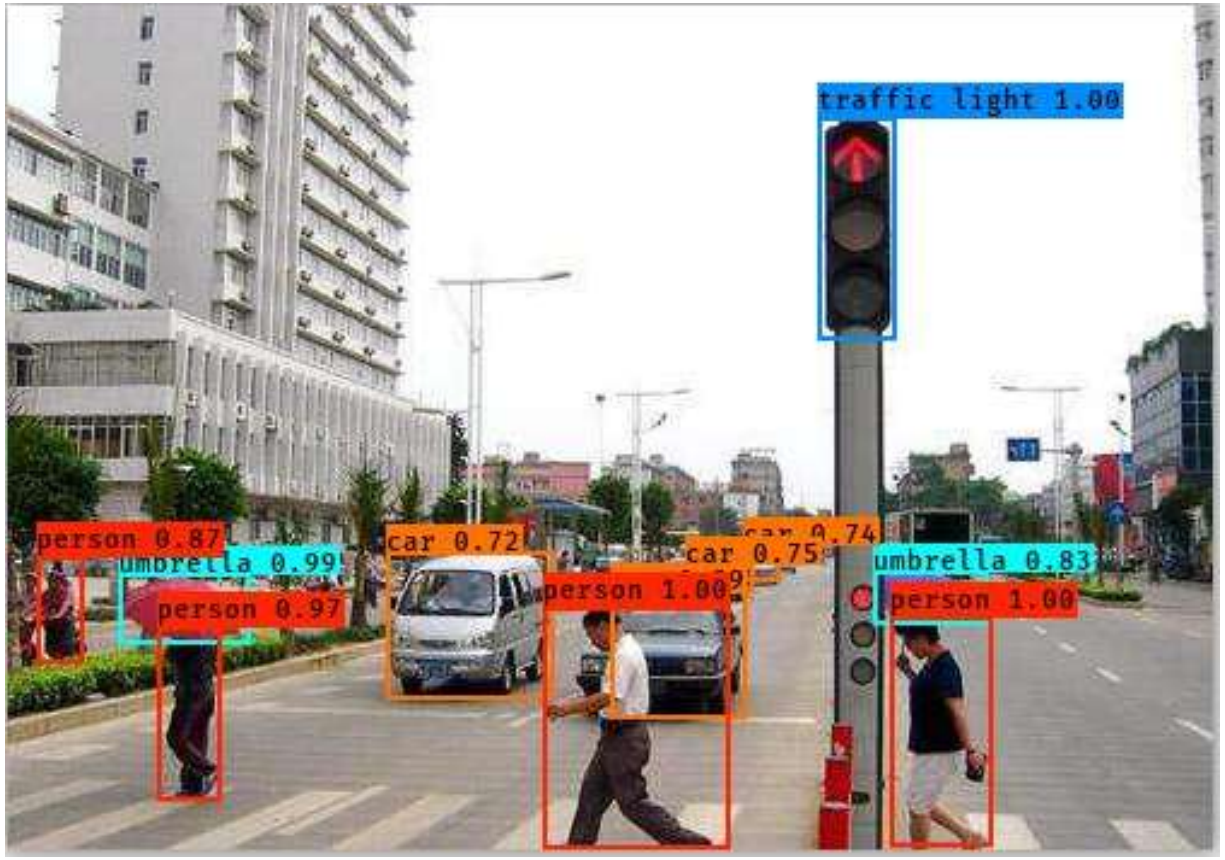


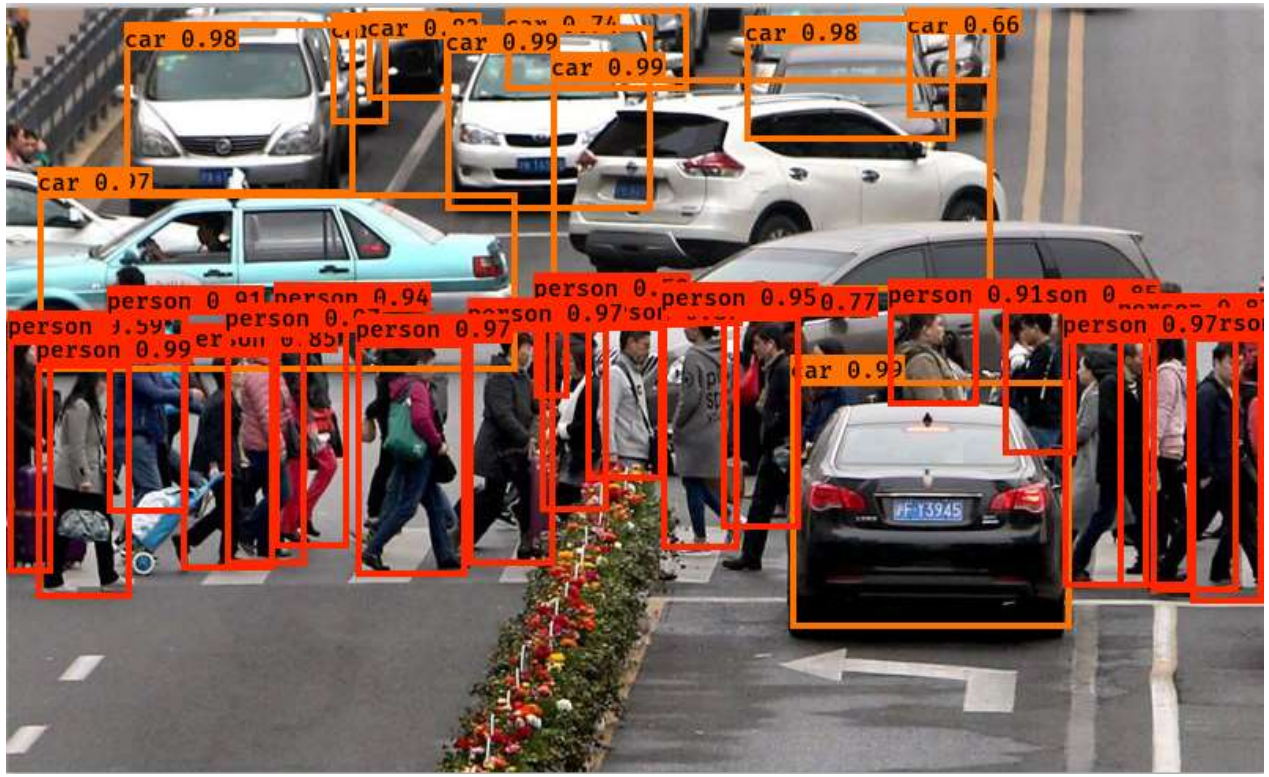
two stage和one stage

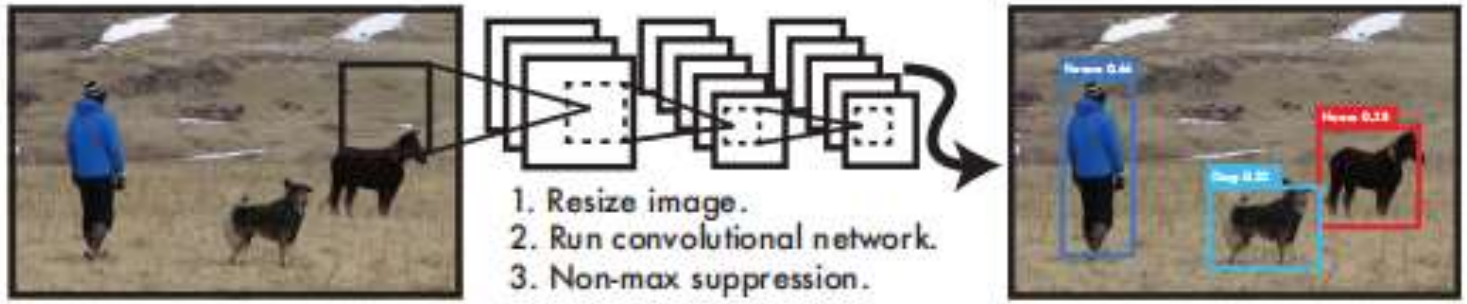
two-stage算法主要是RCNN系列，包括**RCNN**, **Fast-RCNN**，**Faster-RCNN**。之后的**Mask-RCNN**融合了Faster-RCNN架构、ResNet和FPN（Feature Pyramid Networks）backbone，以及FCN里的segmentation方法，在完成了segmentation的同时也提高了detection的精度。

one-stage算法最典型的是**YOLO**，该算法速度极快。

行人检测-Yolo3


traffic light 1.00
person 0.87
umbrella 0.99
person 0.97
car 0.72
car 0.75
person 1.00
umbrella 0.83
person 1.00

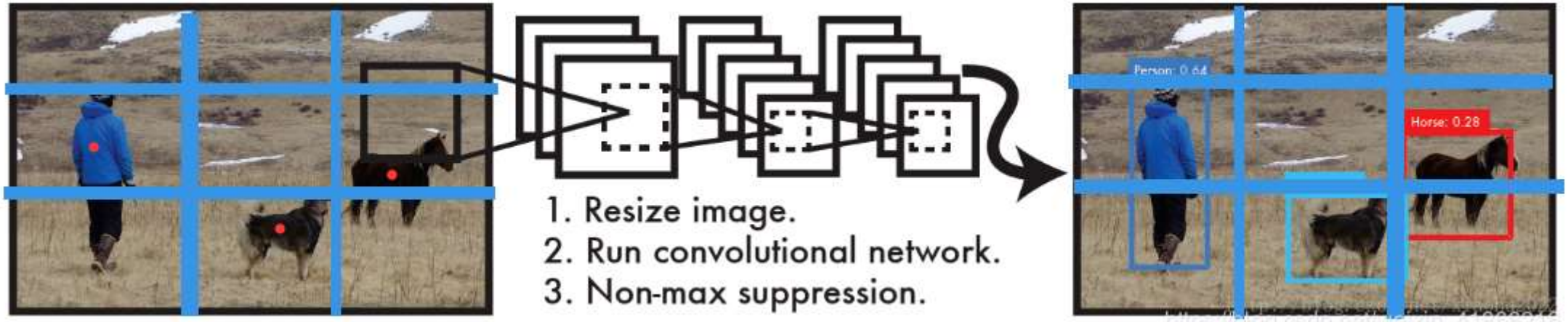

car 0.98
car 0.99
car 0.99
car 0.98
car 0.66
person 0.94
person 0.97
person 0.95
person 0.91
person 0.97
person 0.99



YOLO算法采用一个单独的CNN模型实现**end-to-end**的目标检测：

1. **Resize**成448*448，图片分割得到7*7网格(cell)
2. **CNN提取特征和预测**：卷积部分负责提取特征，全连接部分负责预测。
3. **过滤**bbox（通过nms）

Yolo-You Only Look Once



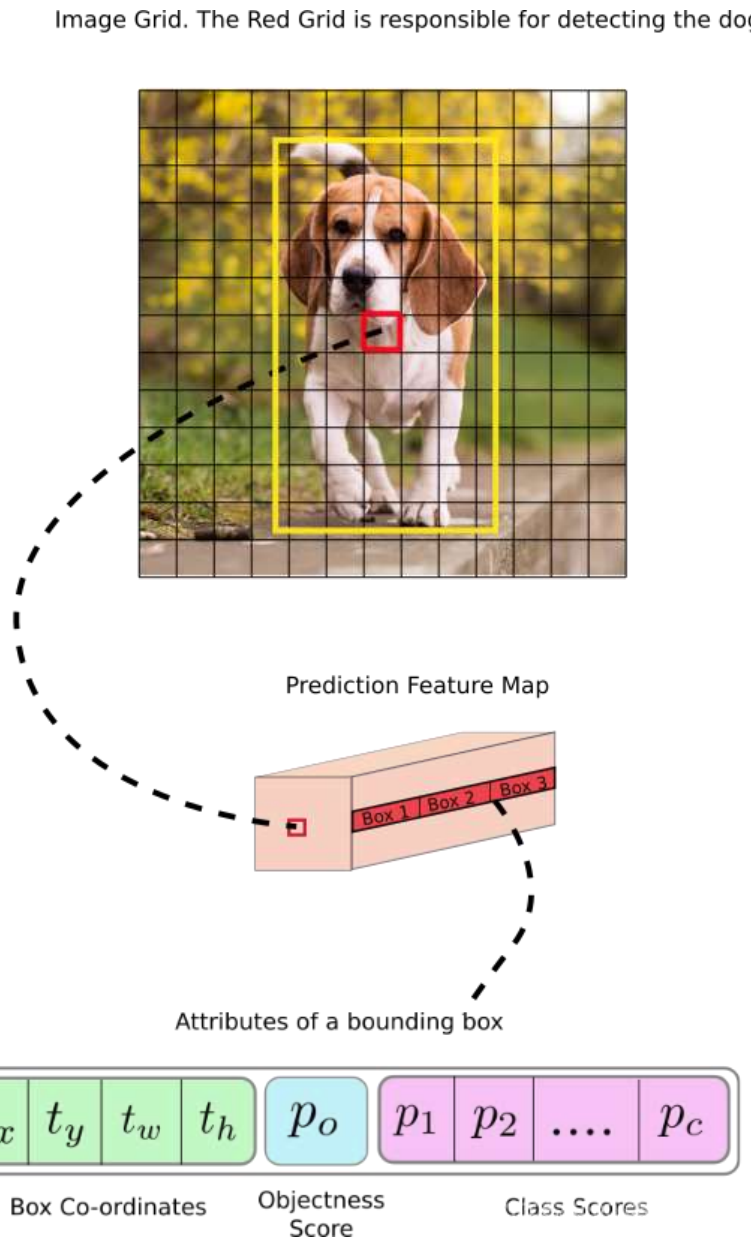
- YOLO算法整体来说就是把输入的图片划分为S*S格子，这里是3*3个格子。
- 当被检测的目标的中心点落入这个格子时，这个格子负责检测这个目标，如图中的人。
- 我们把这个图片输入到网络中，最后输出的尺寸也是S*S*n（n是通道数），这个输出的S*S与原输入图片S*S相对应（都是3*3）。
- 假如我们网络一共能检测20个类别的目标，那么输出的通道数n=2*（4+1）+20=30。这里的2指的是每个格子有两个标定框（论文指出的），4代表标定框的坐标信息， 1代表标定框的置信度，20是检测目标的类别数。
- 所以网络最后输出结果的尺寸是S*S*n=3*3*30。

Yolo-You Only Look Once

关于标定框

- 网络的输出是S x S x (5*B+C) 的一个 tensor(S-尺寸，B-标定框个数，C-检测类别数，5-标定框的信息)。

- 5分为4+1:
- 4代表标定框的位置信息。框的中心点(x，y)，框的高宽h，w。
- 1表示每个标定框的置信度以及标定框的准确度信息。



Yolo-You Only Look Once

一般情况下，YOLO 不会预测边界框中心的确切坐标。它预测：

- 与预测目标的网格单元左上角相关的偏移；
- 使用特征图单元的维度进行归一化的偏移。

例如：
以上图为例，如果中心的预测是 (0.4, 0.7)，则中心在 13 x 13 特征图上的坐标是 (6.4, 6.7)（红色单元的左上角坐标是 (6,6)）。

但是，如果预测到的 x,y 坐标大于 1，比如 (1.2, 0.7)。那么预测的中心坐标是 (7.2, 6.7)。注意该中心在红色单元右侧的单元中。这打破了 YOLO 背后的理论，因为如果我们假设红色框负责预测目标狗，那么狗的中心必须在红色单元中，不应该在它旁边的网格单元中。
因此，为了解决这个问题，我们对输出执行 sigmoid 函数，将输出压缩到区间 0 到 1 之间，有效确保中心处于执行预测的网格单元中。

每个标定框的置信度以及标定框的准确度信息：

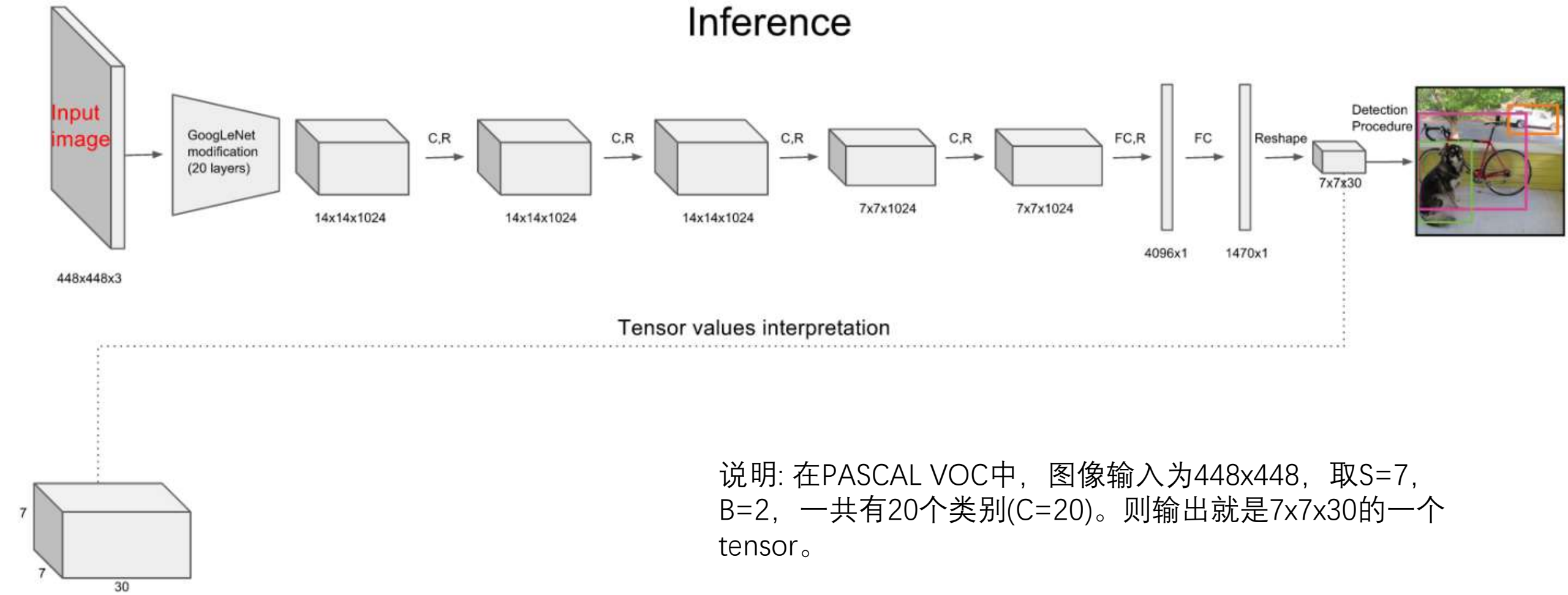
左边代表包含这个标定框的格子里是否有目标。有=1没有=0。
右边代表标定框的准确程度， 右边的部分是把两个标定框（一个是Ground truth，一个是预测的标定框）进行一个IOU操作，即两个标定框的交集比并集，数值越大，即标定框重合越多，越准确。

$$\Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}}$$

我们可以计算出各个标定框的**类别置信度（class-specific confidence scores/ class scores）**：
表达的是该标定框中目标属于各个类别的可能性大小 以及 标定框匹配目标的好坏。

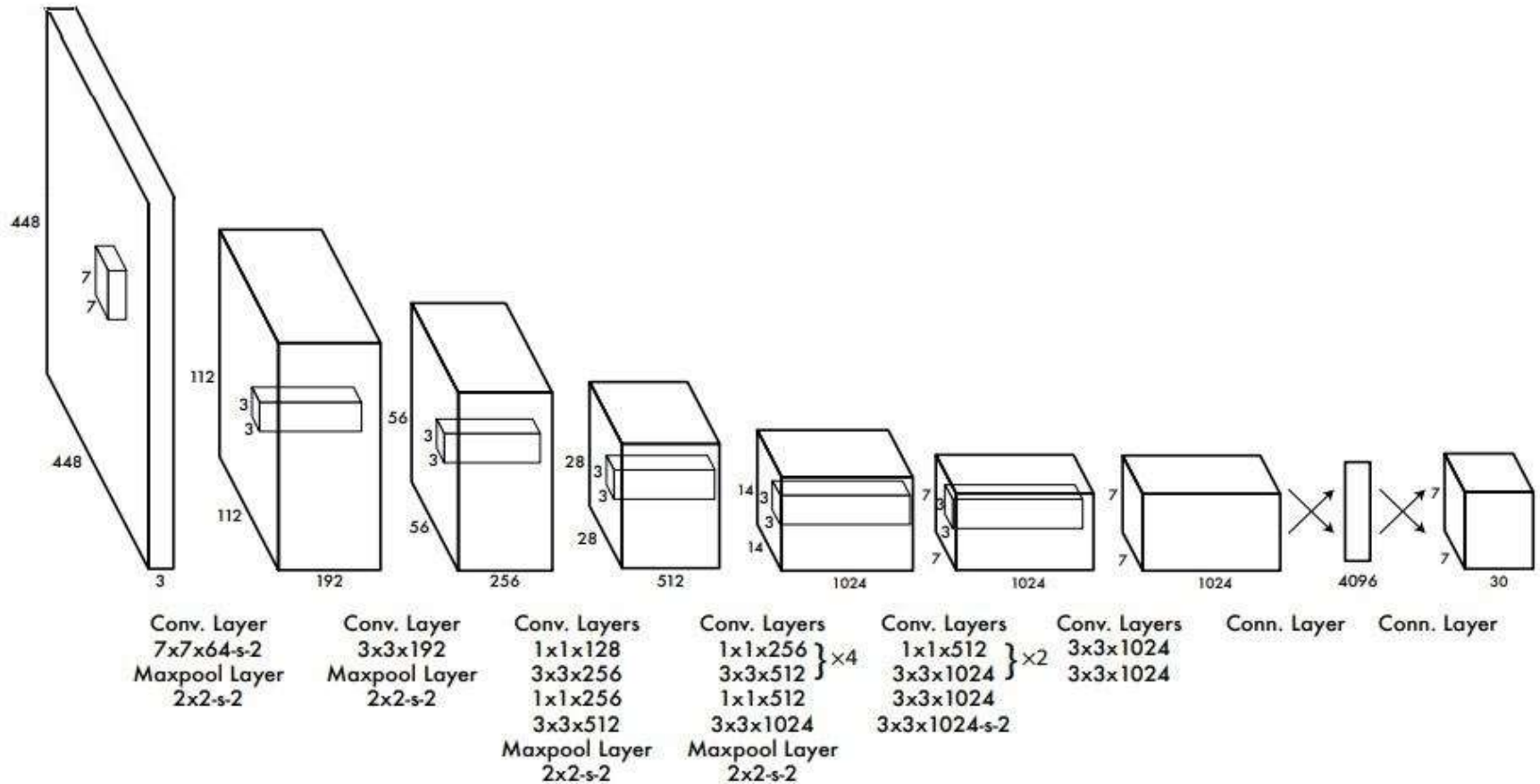
每个网格预测的class信息和bounding box预测的confidence信息相乘，就得到每个bounding box的class-specific confidence score。

$$\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}} = \Pr(\text{Class}_i) * \text{IOU}_{\text{pred}}^{\text{truth}}$$

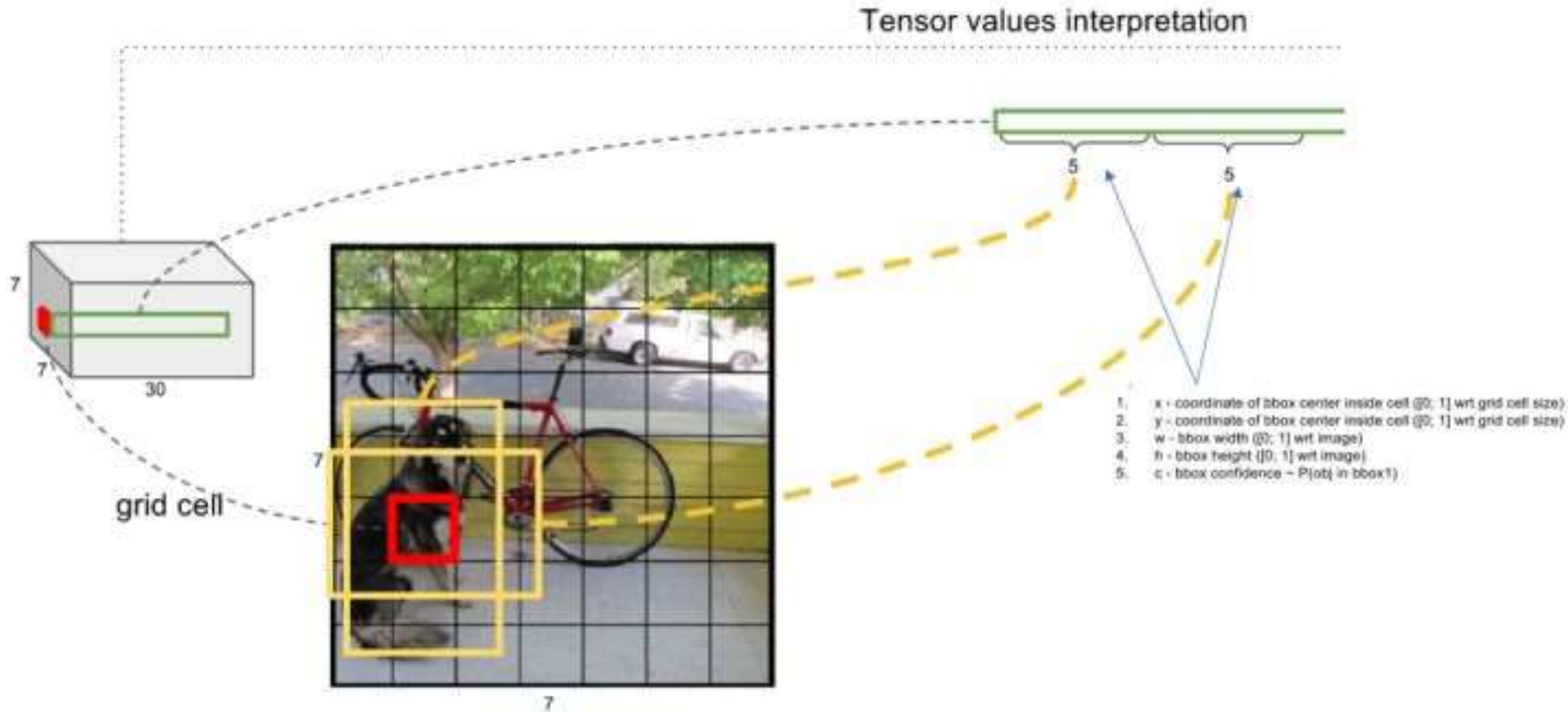


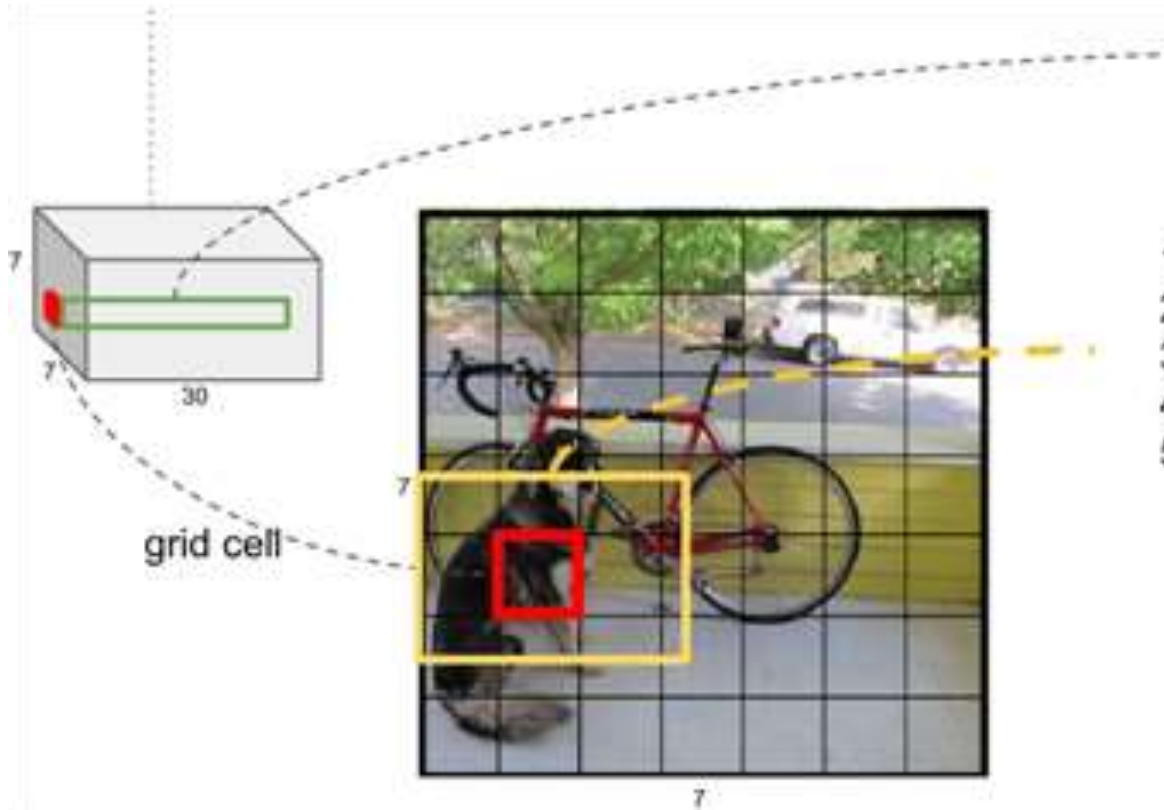
说明: 在PASCAL VOC中，图像输入为448x448，取S=7，B=2，一共有20个类别(C=20)。则输出就是7x7x30的一个tensor。

Yolo-You Only Look Once



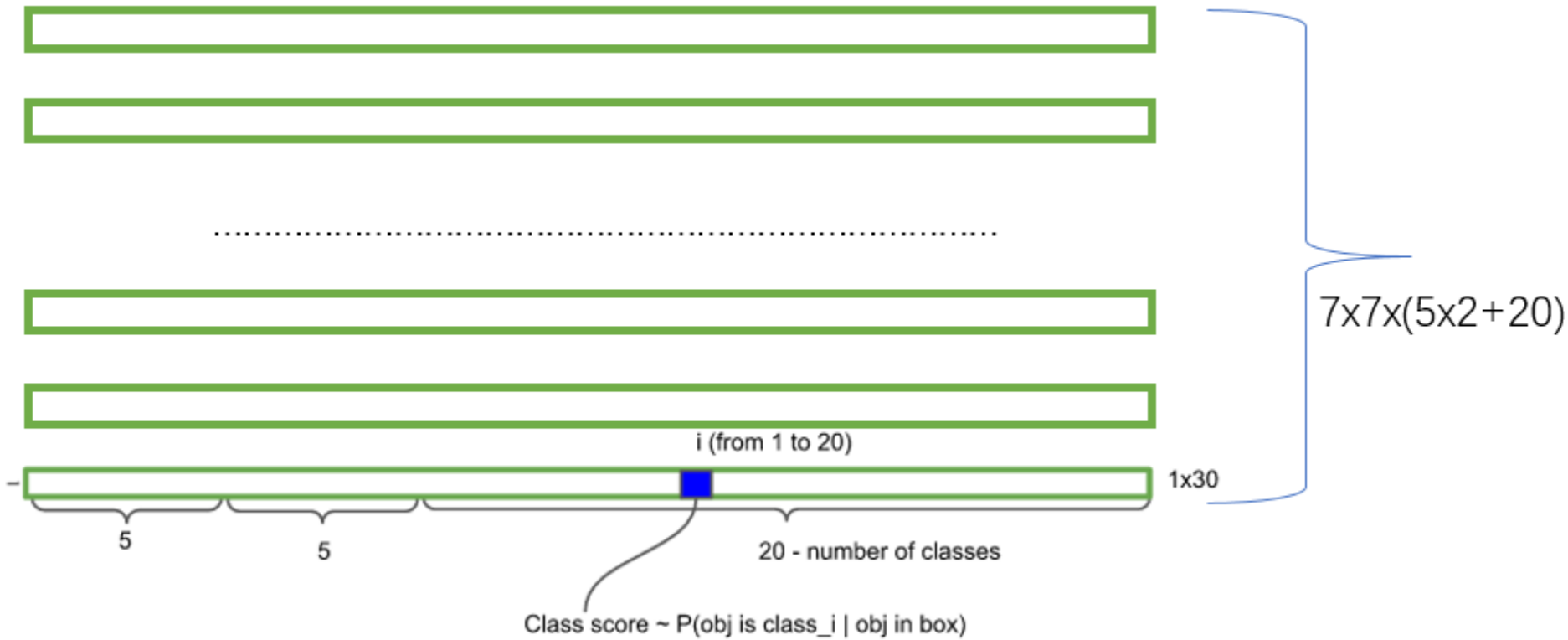
- 其进行了二十多次卷积还有四次最大池化。其中3x3卷积用于提取特征，1x1卷积用于压缩特征，最后将图像压缩到7x7xfilter的大小，相当于将整个图像划分为7x7的网格，每个网格负责自己这一块区域的目标检测。
- 整个网络最后利用全连接层使其结果的size为(7x7x30)，其中7x7代表的是7x7的网格，30前20个代表的是预测的种类，后10代表两个预测框及其置信度(5x2)。


Tensor values interpretation
5
5
7
7
30
grid cell
7
7
1. x - coordinate of bbox center inside cell ([0; 1] wrt grid cell size)
2. y - coordinate of bbox center inside cell ([0; 1] wrt grid cell size)
3. w - bbox width ([0; 1] wrt image)
4. h - bbox height ([0; 1] wrt image)
5. c - bbox confidence – P(obj in bbox1)

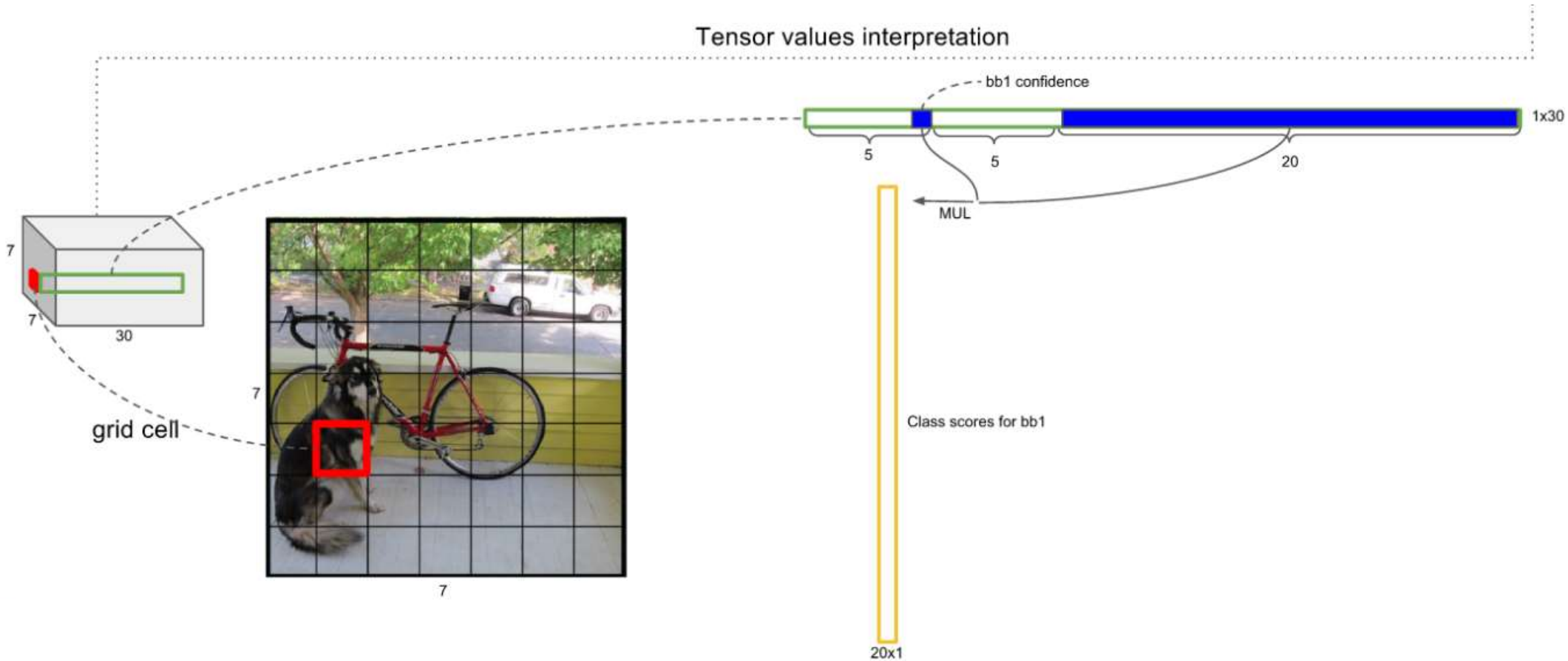


1. x - coordinate of bbox center inside cell ([0; 1] wrt grid cell size)
2. y - coordinate of bbox center inside cell ([0; 1] wrt grid cell size)
3. w - bbox width ([0; 1] wrt image)
4. h - bbox height ([0; 1] wrt image)
5. c - bbox confidence ~ P(obj in bbox2)

Yolo-You Only Look Once

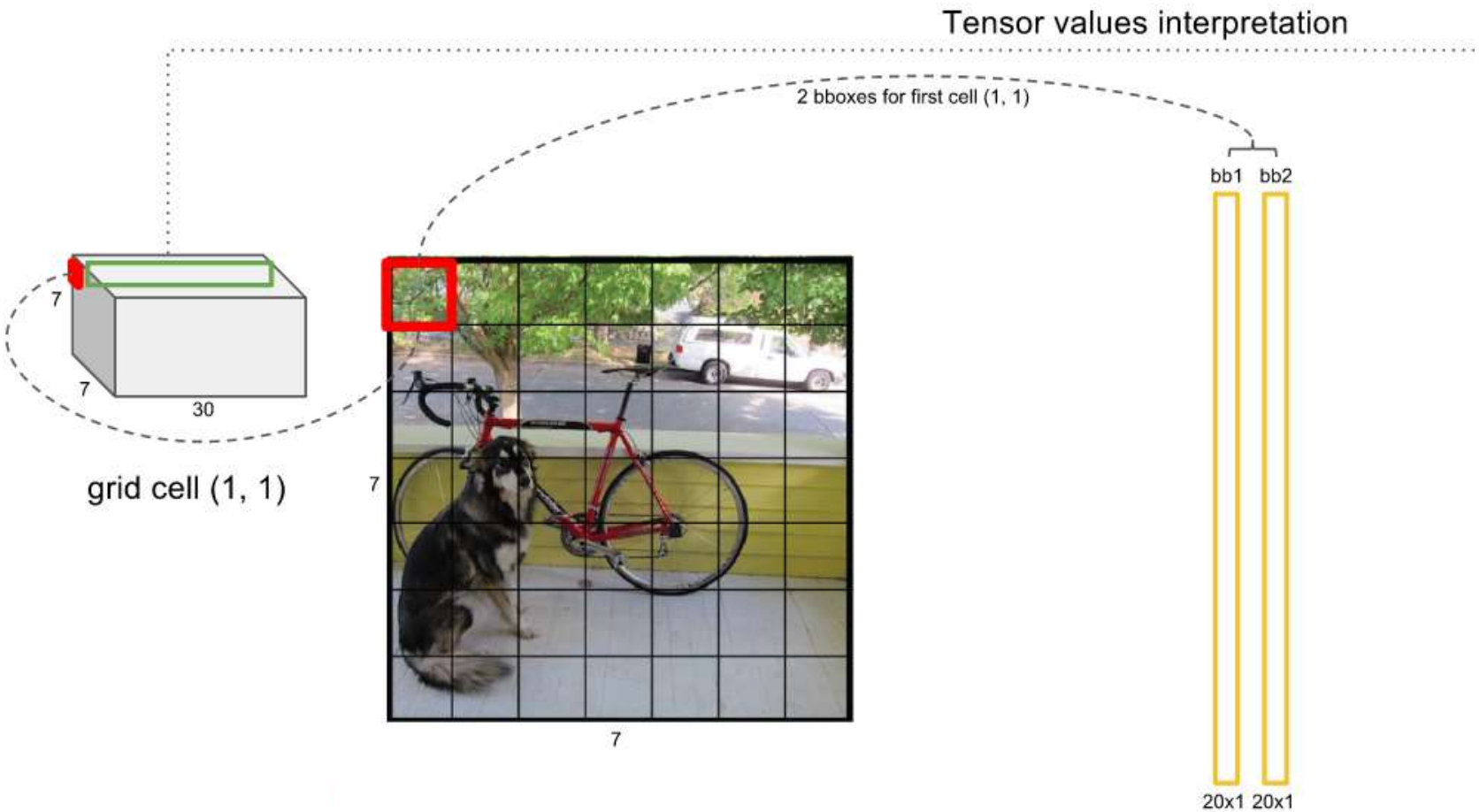


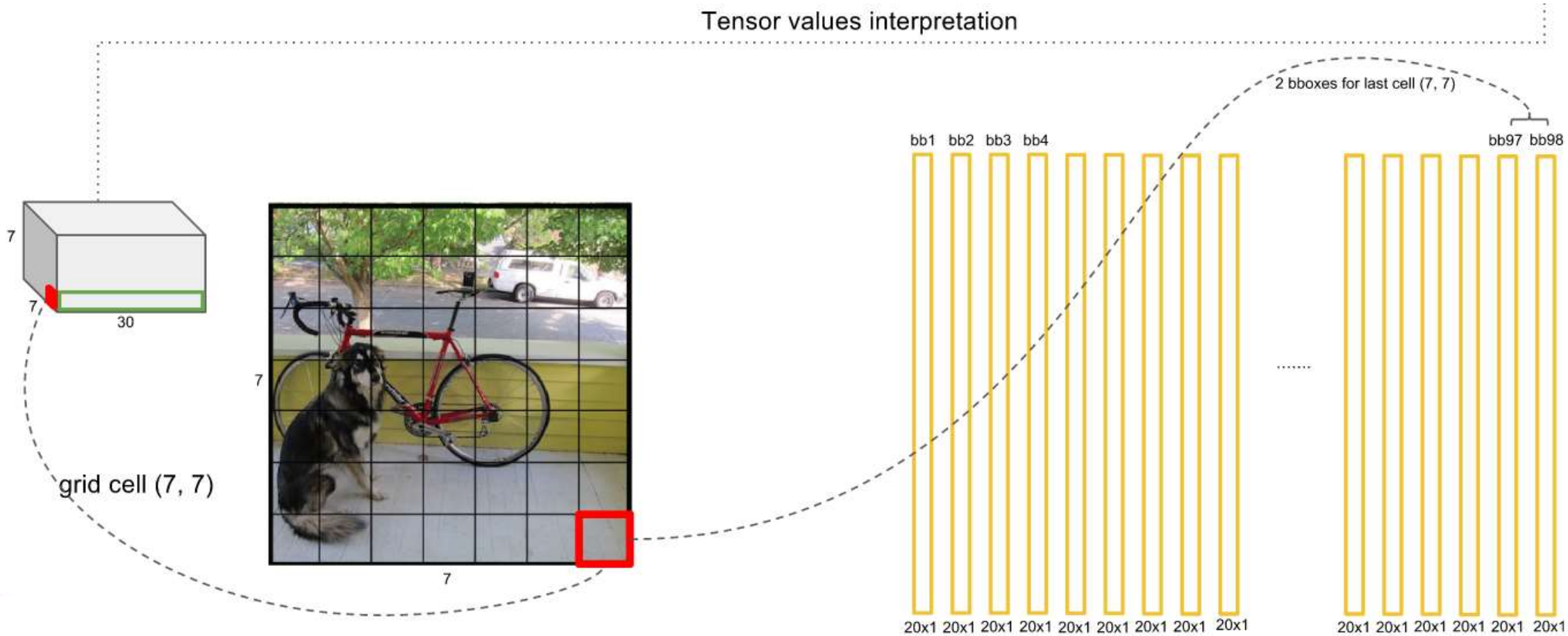
$$\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}} = \Pr(\text{Class}_i) * \text{IOU}_{\text{pred}}^{\text{truth}}$$



Yolo-You Only Look Once

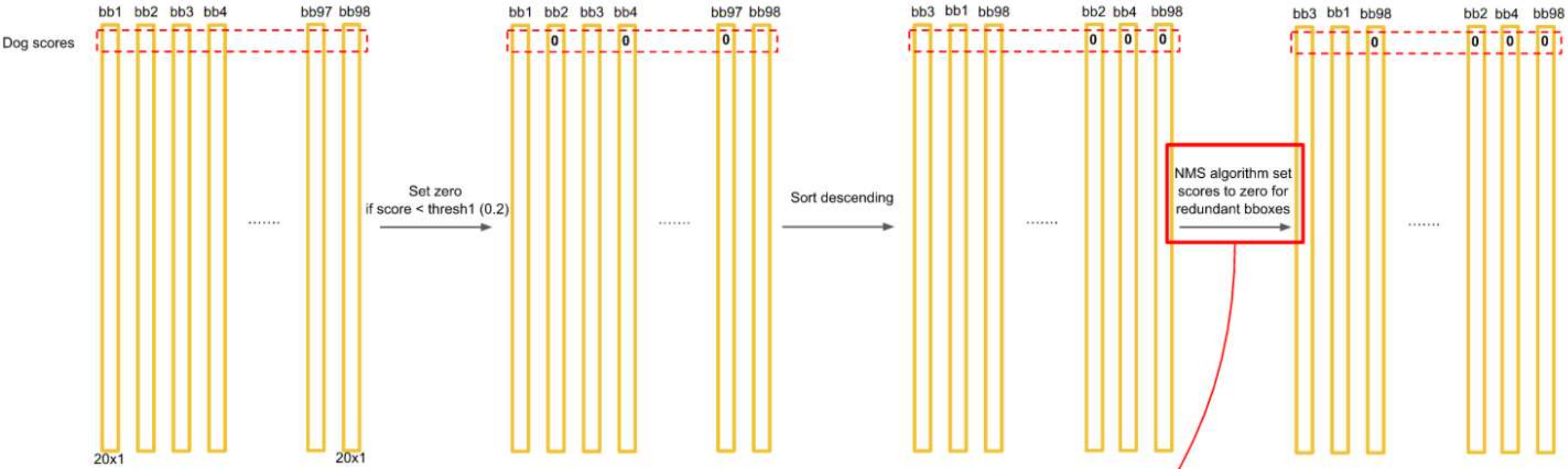
对每一个网格的每一个bbox执行同样操作： 7x7x2 = 98 bbox （每个bbox既有对应的class信息又有坐标信息）




Tensor values interpretation
7
7
30
grid cell (7, 7)
7
7
2 bboxes for last cell (7, 7)
bb1 bb2 bb3 bb4
bb97 bb98
20x1 20x1 20x1 20x1 20x1 20x1 20x1 20x1 20x1
20x1 20x1 20x1 20x1 20x1 20x1

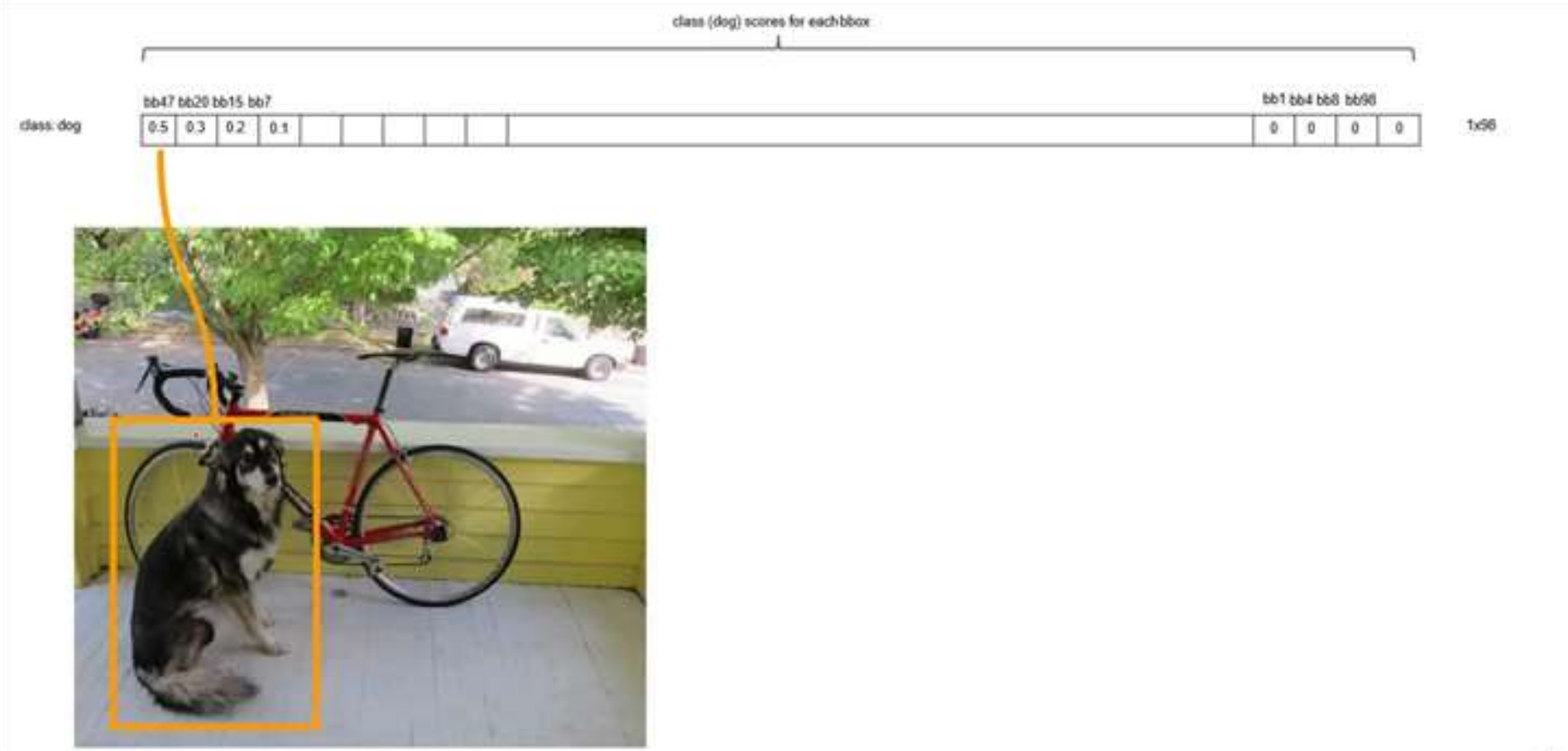
Yolo-You Only Look Once

得到每个bbox的class-specific confidence score以后，设置阈值，滤掉得分低的boxes，对保留的boxes进行NMS处理，就得到最终的检测结果。

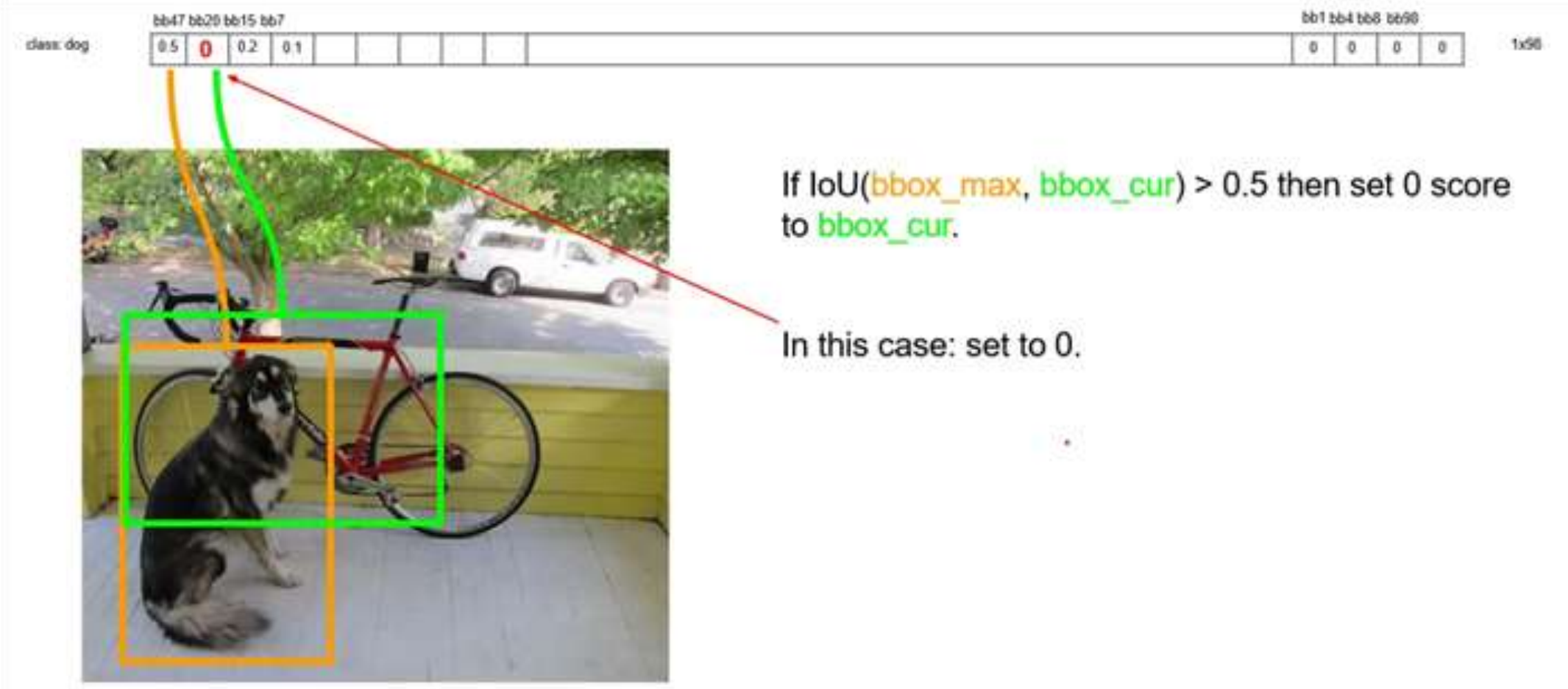


Yolo-You Only Look Once

排序后，不同位置的框内,概率不同:

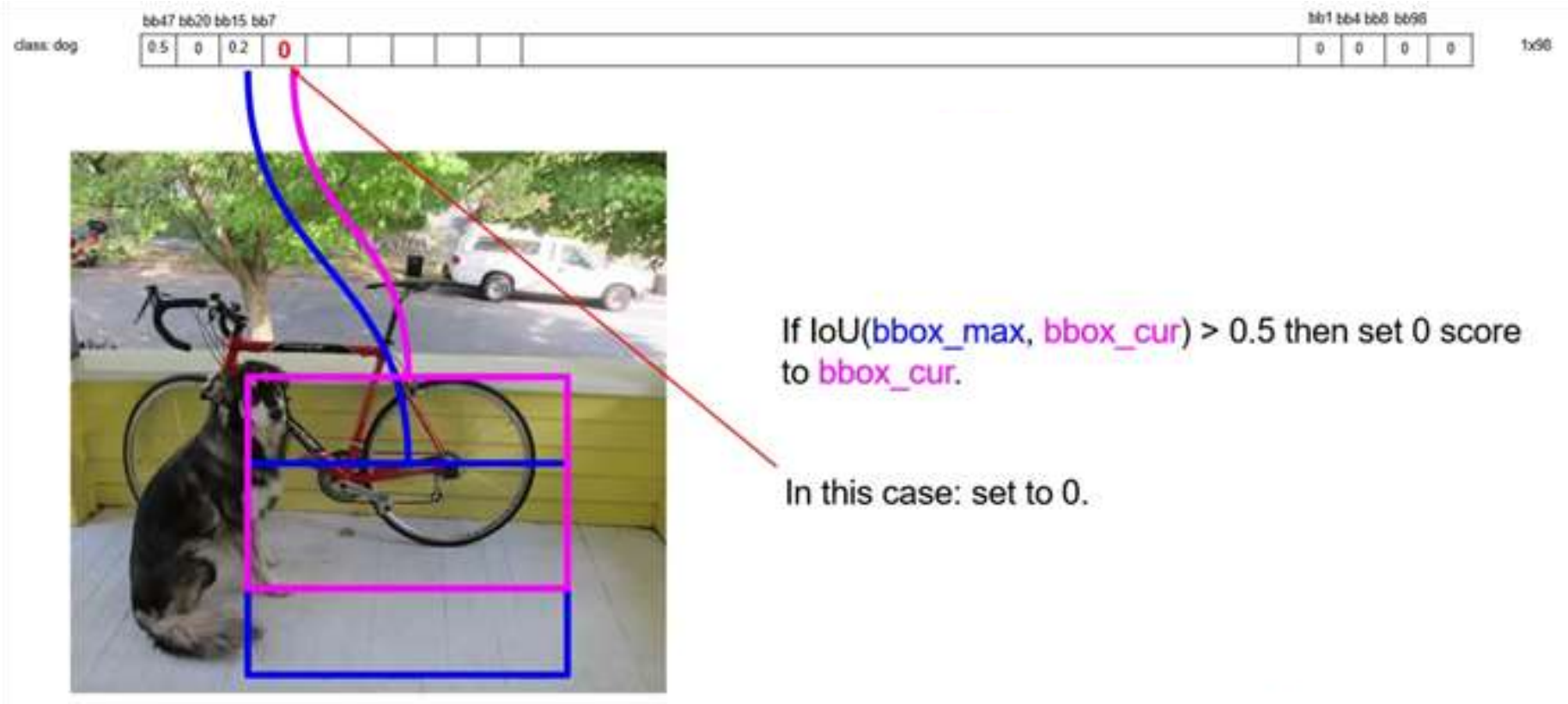


以最大值作为bbox_max，并与比它小的非0值(bbox_cur)做比较：IOU



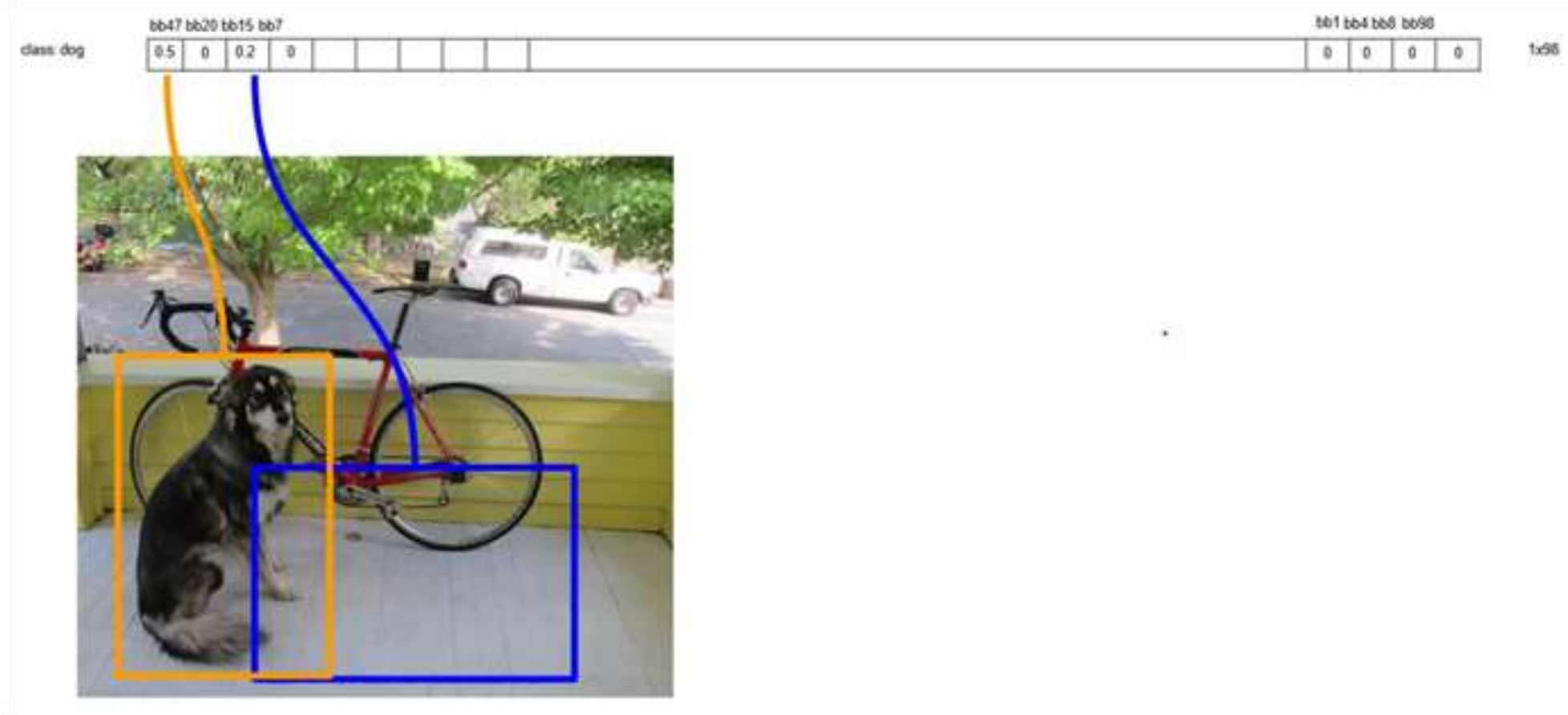
Yolo-You Only Look Once

递归，以下一个非0 bbox_cur（0.2）作为bbox_max继续比较IOU：



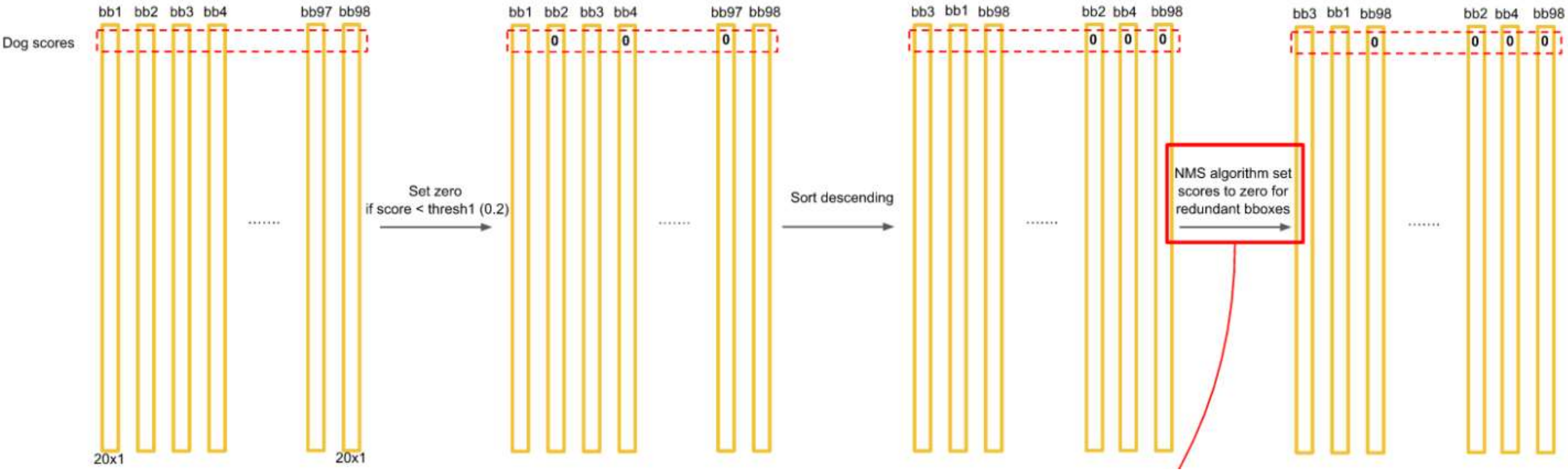
Yolo-You Only Look Once

最终，剩下n个框



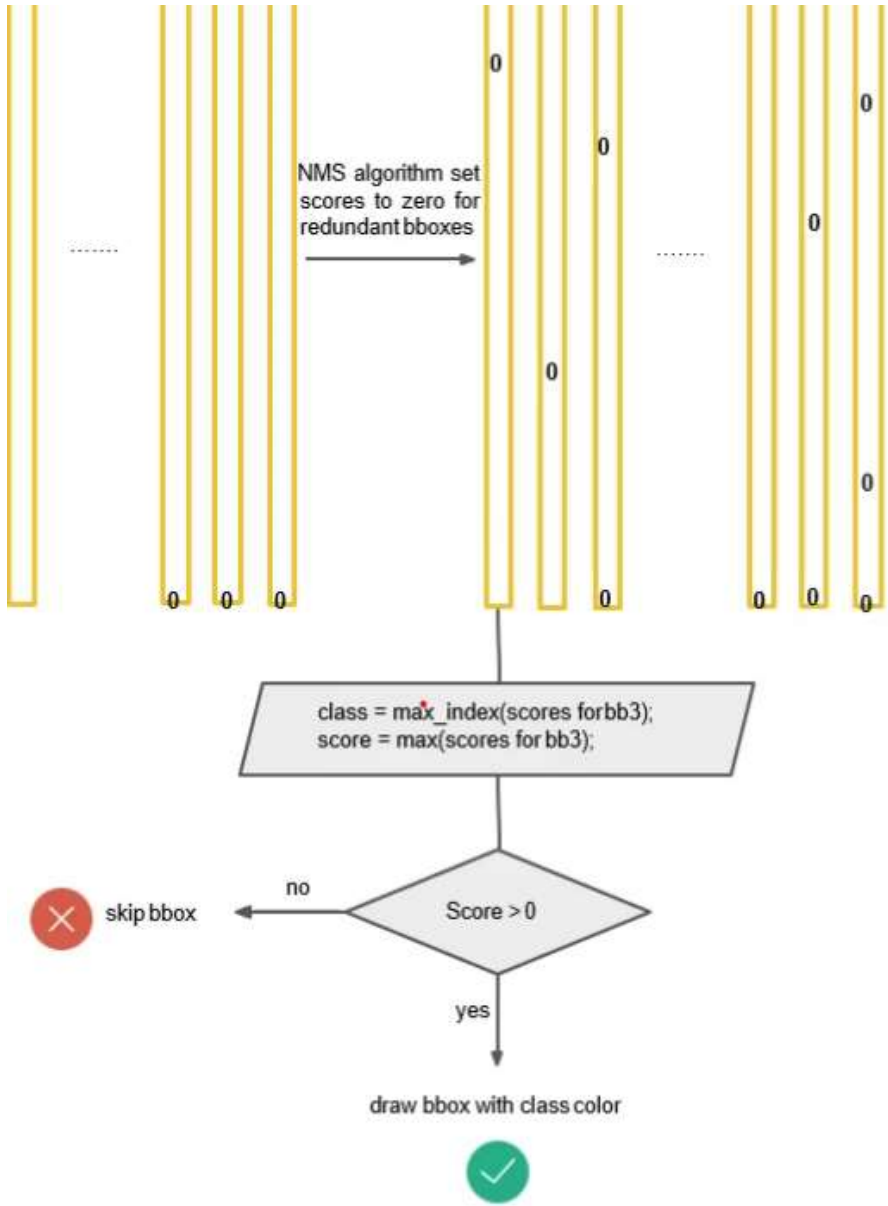
Yolo-You Only Look Once

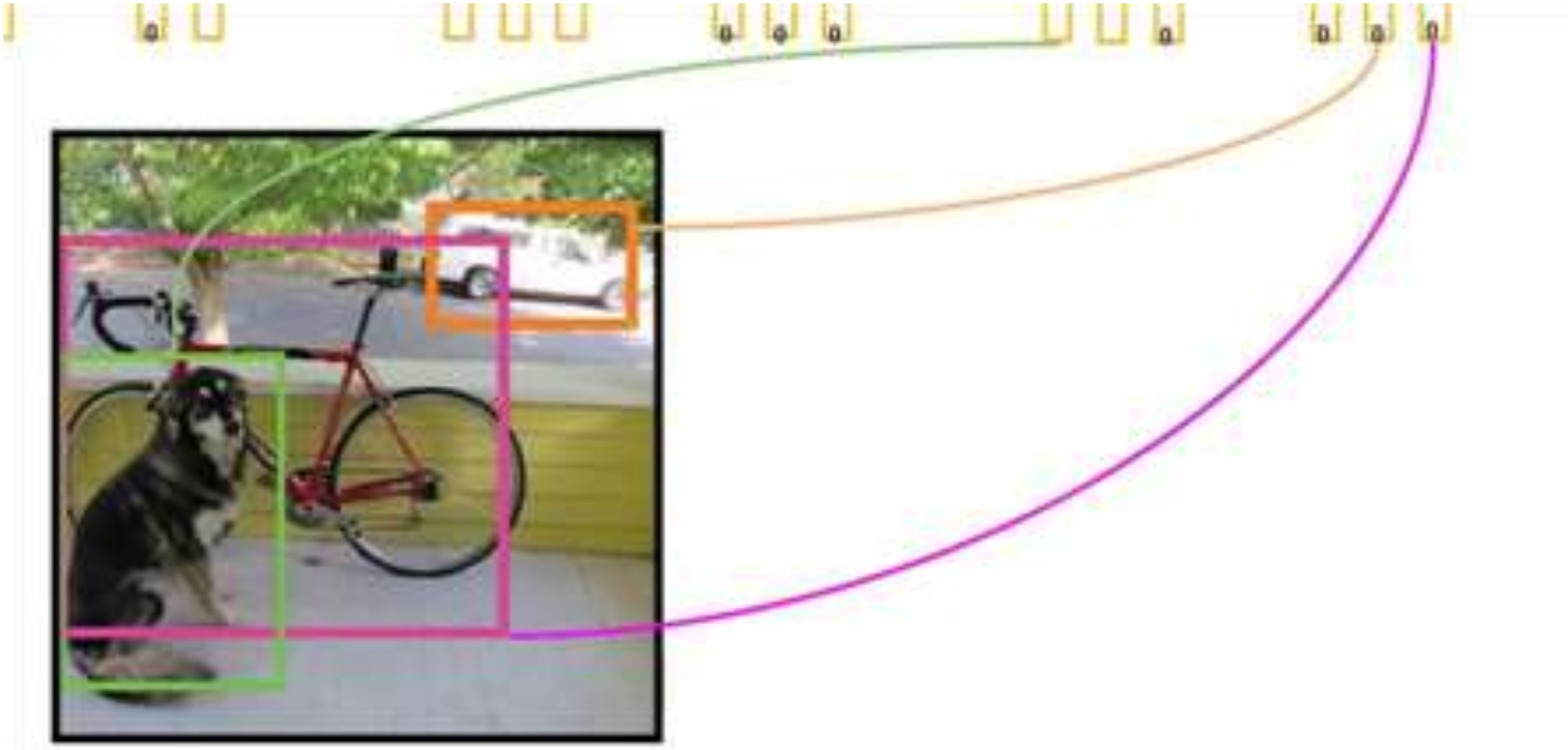
得到每个bbox的class-specific confidence score以后，设置阈值，滤掉得分低的boxes，对保留的boxes进行NMS处理，就得到最终的检测结果。



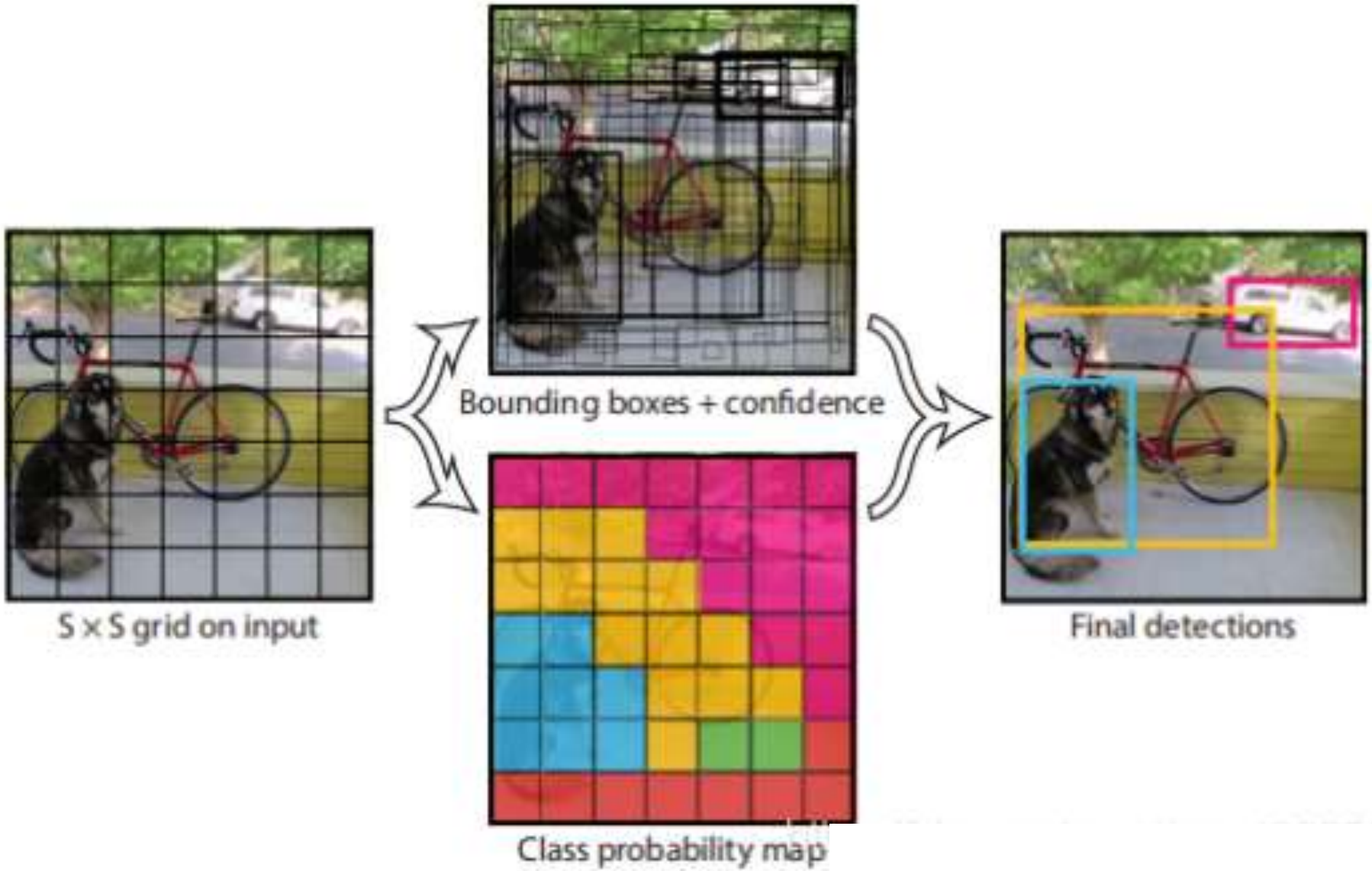
Yolo-You Only Look Once

对bb3(20×1)类别的分数，找分数对应最大类别的索引.---->class
bb3(20×1)中最大的分---->score





Yolo-You Only Look Once



Yolo-You Only Look Once

Yolo的缺点：

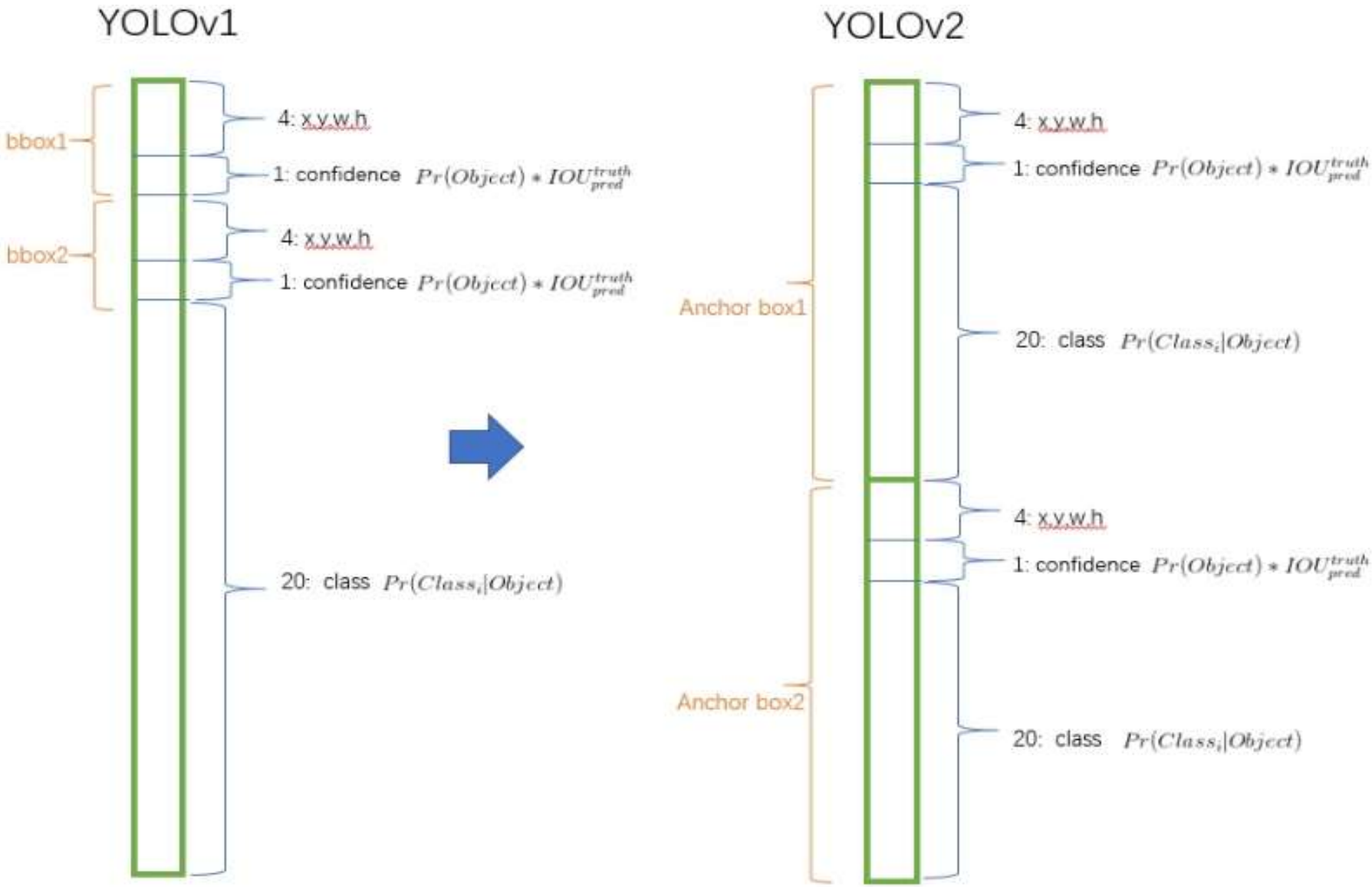
- YOLO对相互靠的很近的物体（挨在一起且中点都落在同一个格子上的情况），还有很小的群体检测效果不好，这是因为一个网格中只预测了两个框，并且只属于一类。
- 测试图像中，当同一类物体出现不常见的长宽比和其他情况时泛化能力偏弱。

Yolo2

1. Yolo2使用了一个新的分类网络作为特征提取部分。
2. 网络使用了较多的3 x 3卷积核，在每一次池化操作后把通道数翻倍。
3. 把1 x 1的卷积核置于3 x 3的卷积核之间，用来压缩特征。
4. 使用batch normalization稳定模型训练，加速收敛。
5. 保留了一个shortcut用于存储之前的特征。
6. **yolo2相比于yolo1加入了先验框部分**，最后输出的conv_dec的shape为(13,13,425):
   - 13x13是把整个图分为13x13的网格用于预测。
   - 425可以分解为(85x5)。在85中，由于yolo2常用的是coco数据集，其中具有80个类；剩余的5指的是x、y、w、h和其置信度。x5意味着预测结果包含5个框，分别对应5个先验框。

Type	Filter	size/stride	Output			
conv1	32	3x3	416, 416, 32			
pool1		2x2/2	208, 208, 32			
conv2	64	3x3	208, 208, 64			
pool2		2x2/2	104, 104, 64			
conv3_1	128	3x3	104, 104, 128			
conv3_2	64	1x1	104, 104, 64			
conv3_3	128	3x3	104, 104, 128			
pool3		2x2/2	52, 52, 128			
conv4_1	256	3x3	52, 52, 256			
conv4_2	128	1x1	52, 52, 128			
conv4_3	256	3x3	52, 52, 256			
pool4		2x2/2	26, 26, 256			
conv5_1	512	3x3	26, 26, 512			
conv5_2	256	1x1	26, 26, 256			
conv5_3	512	3x3	26, 26, 512			
conv5_4	256	1x1	26, 26, 256			
conv5_5	512	3x3	26, 26, 512	➡	shortcut	26, 26, 512
pool5		2x2/2	13, 13, 512		⬇	
conv6_1	1024	3x3	13, 13, 1024			
conv6_2	512	1x1	13, 13, 512			
conv6_3	1024	3x3	13, 13, 1024		filter为64的3x3卷积	26, 26, 64
conv6_4	512	1x1	13, 13, 512		⬇	
conv6_5	1024	3x3	13, 13, 1024			
conv7_1	1024	3x3	13, 13, 1024			
conv7_2	1024	3x3	13, 13, 1024		passthrough变小变长	13, 13, 256
合并shortcut和conv7_2得到(13, 13, 1024+256)						
conv8	1024	3x3	13, 13, 1024			
conv_dec	425	3x3	13, 13, 425			

Yolo2 -- 采用anchor boxes



Yolo2 -- Dimension Clusters（维度聚类）

使用kmeans聚类获取先验框的信息：

之前先验框都是手工设定的，YOLO2尝试统计出更符合样本中对象尺寸的先验框，这样就可以减少网络微调先验框到实际位置的难度。YOLO2的做法是对训练集中标注的边框进行聚类分析，以寻找尽可能匹配样本的边框尺寸。

聚类算法最重要的是选择如何计算两个边框之间的“距离”，对于常用的欧式距离，大边框会产生更大的误差，但我们关心的是边框的IOU。所以，YOLO2在聚类时采用以下公式来计算两个边框之间的“距离”。

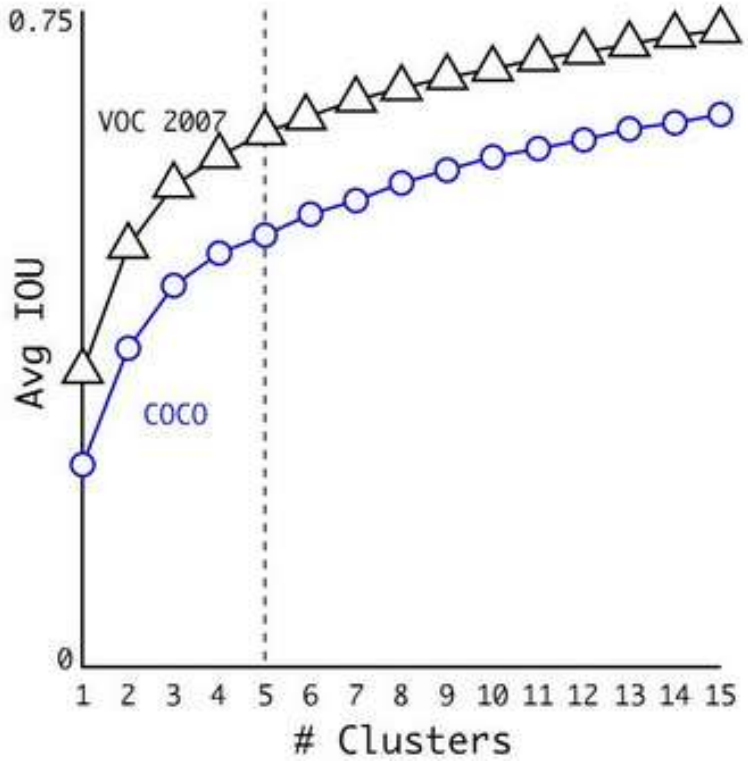
$$d(box, centroid) = 1 - IOU(box, centroid)$$

Yolo2 -- Dimension Clusters（维度聚类）

在选择不同的聚类k值情况下，得到的k个centroid边框，计算样本中标注的边框与各centroid的Avg IOU。

显然，边框数k越多，Avg IOU越大。

YOLO2选择k=5作为边框数量与IOU的折中。对比手工选择的先验框，使用5个聚类框即可达到61 Avg IOU，相当于9个手工设置的先验框60.9 Avg IOU



作者最终选取5个聚类中心作为先验框。对于两个数据集，5个先验框的width和height如下：

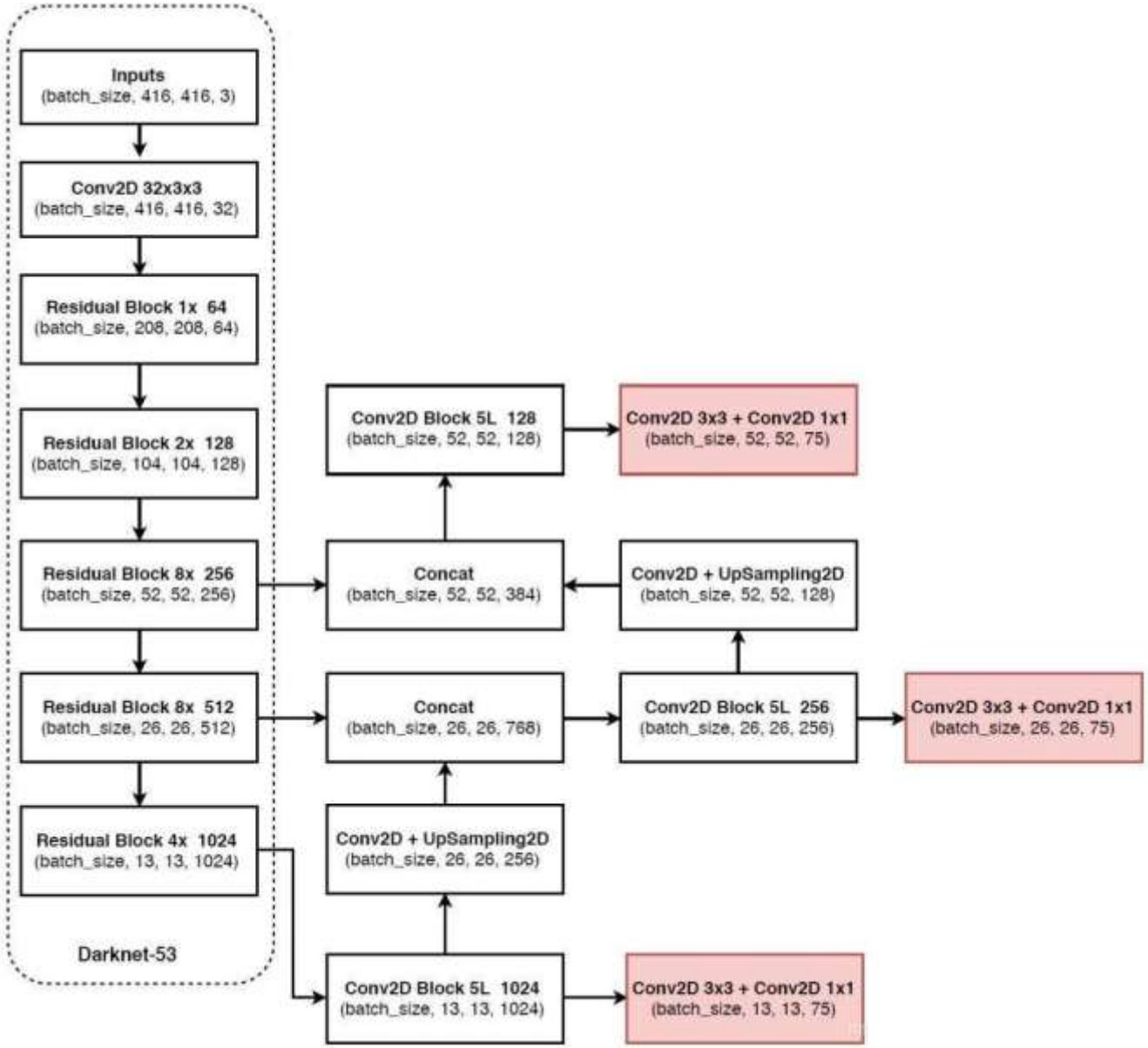
COCO: (0.57273, 0.677385), (1.87446, 2.06253), (3.33843, 5.47434), (7.88282, 3.52778), (9.77052, 9.16828)

VOC: (1.3221, 1.73145), (3.19275, 4.00944), (5.05587, 8.09892), (9.47112, 4.84053), (11.2364, 10.0071)

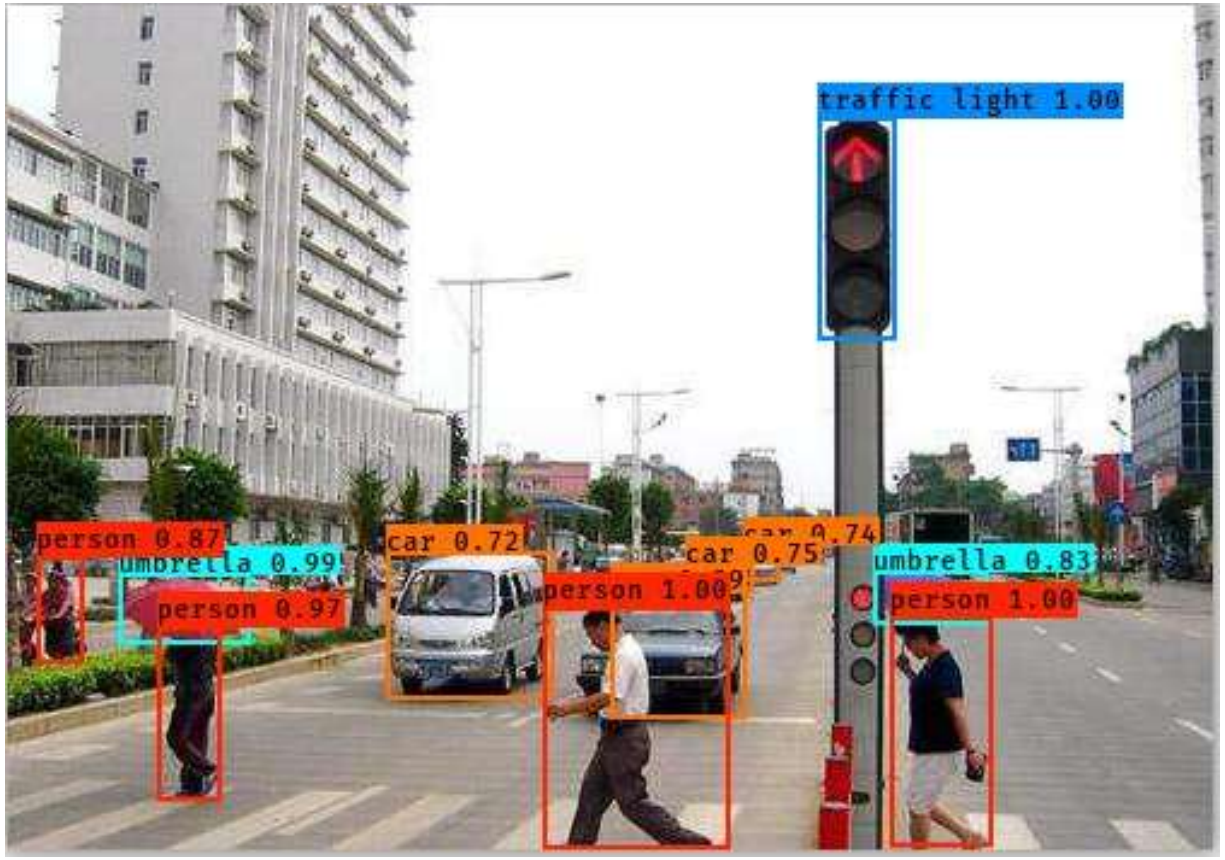
Yolo3

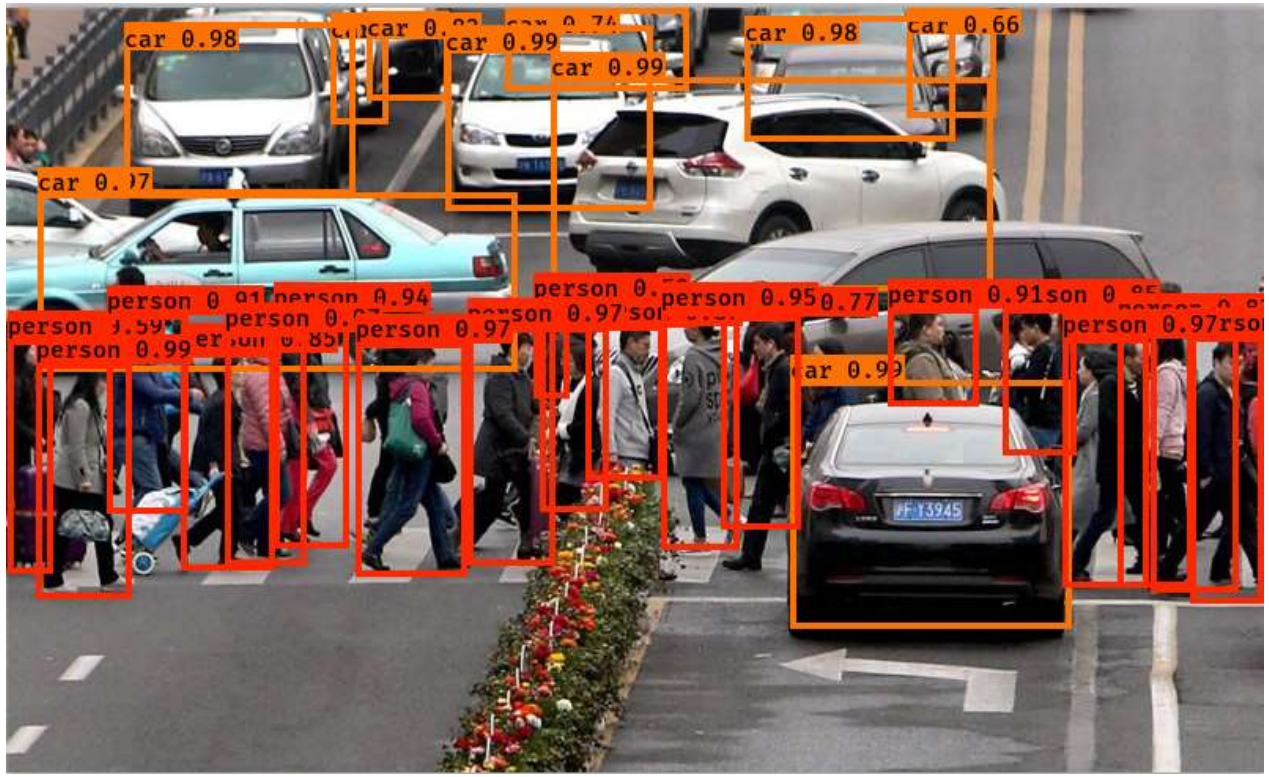
YOLOv3相比于之前的yolo1和yolo2，改进较大，主要改进方向有：

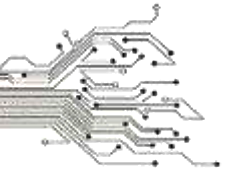
1. 使用了残差网络Residual
2. 提取多特征层进行目标检测，一共提取三个特征层，它的shape分别为(13,13,75)，(26,26,75)，(52,52,75)。最后一个维度为75是因为该图是基于voc数据集的，它的类为20种。yolo3针对每一个特征层存在3个先验框，所以最后维度为3x25。
3. 其采用了UpSampling2d设计



行人检测-Yolo3

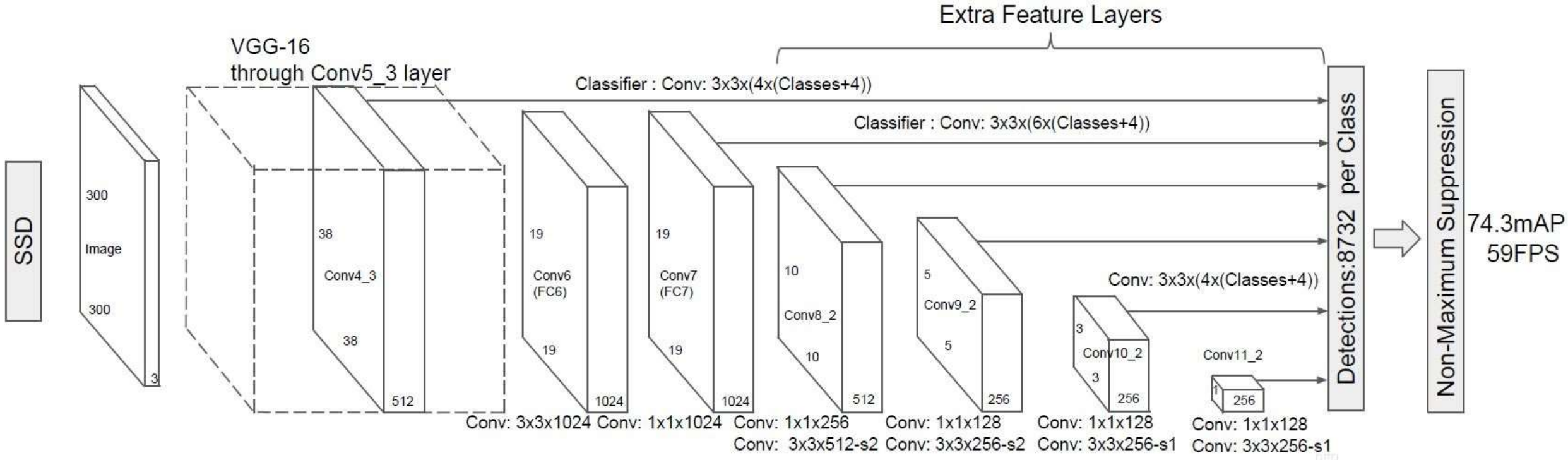

traffic light 1.00
person 0.87
umbrella 0.99
person 0.97
car 0.72
car 0.75
person 1.00
umbrella 0.83
person 1.00


car 0.98
car 0.99
car 0.99
car 0.98
car 0.66
person 0.94
person 0.97
person 0.95
person 0.91
person 0.97
person 0.99



- 涉及FPN等结构，我们后续再看

拓展-SSD



SSD也是一个多特征层网络，其一共具有11层，前半部分结构是VGG16：

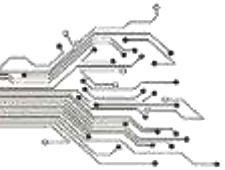
1、首先通过了多个3X3卷积层、5次步长为2的最大池化取出特征，形成了5个Block，其中第四个Block用于提取小目标（多次卷积后大目标的特征保存的更好，小目标特征会消失，需要在比较靠前的层提取小目标特征）。

2、进行一次卷积核膨胀dilate。

3、读取第七个Block7的特征。

4、分别利用1x1和3x3卷积提取特征，在3x3卷积的时候使用步长2，缩小特征数，获取第八个Block8的特征。

5、重复步骤4，获得9、10、11卷积层的特征。



谢谢！