YOLO(You look only once)

> 不同于以往的two－stage算法，yolo算法的实现正如它的名字一样（你只需要看一次），yolo算法是属于one-stage，只需要一次CNN特征提取就可以完成分类、识别和定位三种工作，并且能够保持较高的准确率，这也使得yolo系列算法成为目标检测领域的主流。

1. yolov1

yolov1论文

YOLOv1幻灯片

参考一

参考二

参考三

1.1 算法描述

yolov1是yolo系列算法的开端，相比与以往基于滑动窗口和候选区域的算法，yolo对原始图片采用直接学习图像的全局信息，只需要一次便可以完成对检测目标的识别、定位等。系统的整体流程如下图：

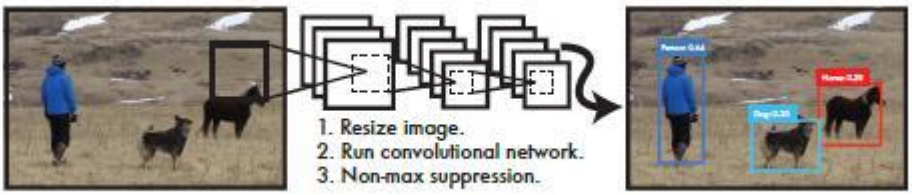
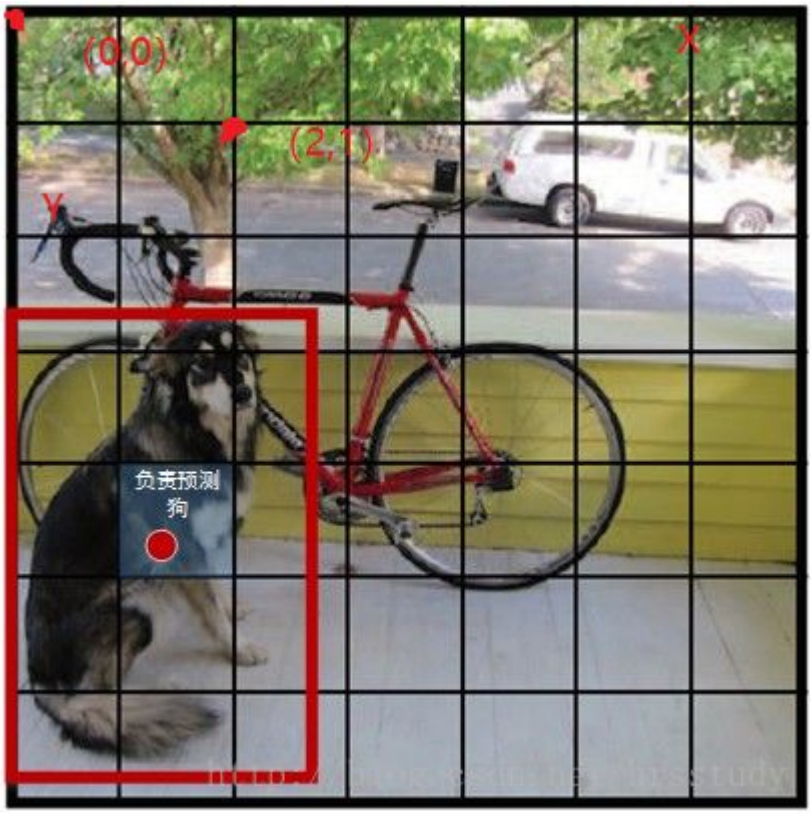


Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448 × 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

yolov1算法在执行时，首先会把图片划分成 S × S 的网格，如下图:



而对于每一个小的网格来说，只要图片中的目标对象的中心点落在该格子内，则该格子就要负责预测该对象。这时每一个格子中都会预测B个bounding boxs，每一个bounding boxs会有 5 个预测值，包括*x*、*y*、*w*、*h*和一个置信度*confidence*，此时的x,y是相对与所在网格左上角的偏移值，w,h 是相对整张图片归一化后的值，b-boxs的confidence是通过如下公式（$Pr(object)$*表示网格内的是否存在检测的目标对象，通常以 1 表示存在，以 0 表示不能存在*）：

$$Pr(Object) * IOU(truth/pred)$$

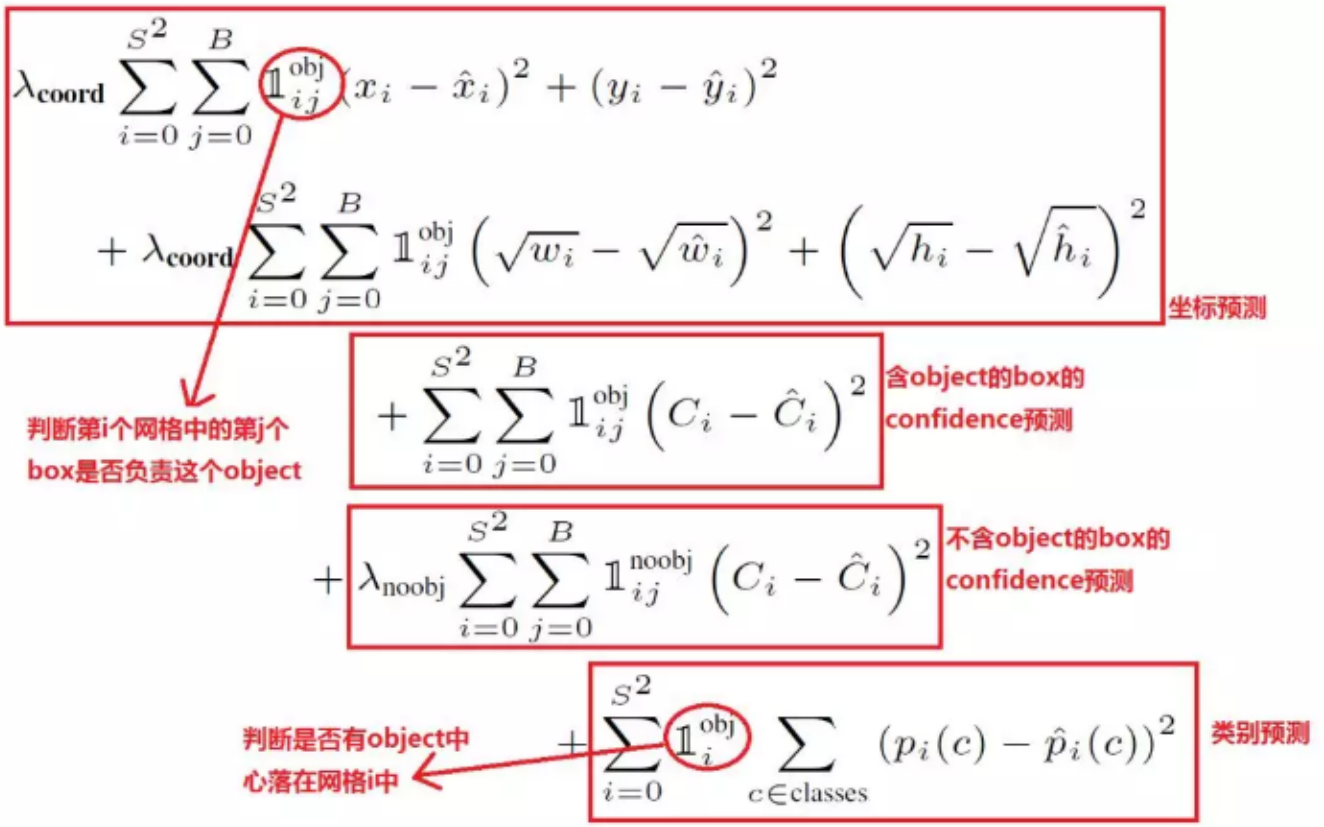
除此之外，每一个预测的格子都会有一个 C 个类别对象类别预测值（注意，此时每一个格子虽然会有 B 的b-boxs，但是真正能在最后够预测是的b-boxs只能有一个，并且每一个格子只能输出一种类别信息）,其中分类置信度通过如下公式：

$$Pr(classi|object) * Pr(object) * IOU(truth/pred) = Pr(classi) * IOU(truth/pred)$$

这一个乘积不仅预测了b-boxs属于某一类的概率,也有b-boxs准确度的信息。然后，通过得到每个b-boxs 的 class-specific confidence score ，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（注意：NMS 执行时是对所有的预测框一视同仁）。

1.2 损失函数：

Yolov1算法将目标检测看作是一个回归问题，所以采用的是均方差损失函数。公式如下：



损失函数在计算时，通过 调节λ_coord和 λ_noobj的值来实现对定位误差和分类误差的的权重分配，我们更加注重对于坐标误差的调整，所以设置$\lambda_crood = 5$。在不含有对象的box的confidence预测中，设置$\lambda_noobj = 0.5$,消除了无目标对象时对损失函数的影响，其它的权重值设置为 1。然后采用均方误差，其同等对待大小不同的边界框，但是实际上较小的边界框的坐标误差应该要比较大的边界框要更敏感。为了保证这一点，将网络的边界框的宽与高预测改为对其平方根的预测，即预测值变为 $(x, y, \sqrt{w}, \sqrt{h})$ 。

注意：

1. 只有当某个网络中存在object时才会对分类误差的进行惩罚。
2. 每一个网格的预测的b-boxs的选定是根据含有object的网格中，预测框与真实框的最大IOU值的哪一个框来决定的，且唯一。

1.3 Yolov1的优缺点

优点：

1. YOLO的速度非常快。在Titan X GPU上的速度是45 fps（frames per second），加速版的YOLO差不多是150fps。
2. YOLO是基于图像的全局信息进行预测的。这一点和基于sliding window以及region proposal等检测算法不一样。与Fast R-CNN相比，YOLO在误检测（将背景检测为物体）方面的错误率能降低一半多。
3. YOLO可以学到物体的generalizable representations。可以理解为泛化能力强。
4. 准确率高，有实验证明。

缺点：

1. YOLO对相互靠的很近的物体，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

    2. 对测试图像中，同一类物体出现的新的不常见的长宽比和其他情况是。泛化能力偏弱。 3. 由于损失函数的问题，定位误差是影响检测效果的主要原因。尤其是大小物体的处理上，还有待加强。

2. Yolov2

Yolov2论文

参考一

参考二

参考三

参考四

2.1 Yolov2的改进

Yolov2改进图如下：

	YOLO							YOLOv2
batch norm?		✓	✓	✓	✓	✓	✓	✓
hi-res classifier?			✓	✓	✓	✓	✓	✓
convolutional?				✓	✓	✓	✓	✓
anchor boxes?				✓	✓			
new network?					✓	✓	✓	✓
dimension priors?						✓	✓	✓
location prediction?						✓	✓	✓
passthrough?							✓	✓
multi-scale?								✓
hi-res detector?								✓
VOC2007 mAP	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	**78.6**

Table 2: The path from YOLO to YOLOv2. Most of the listed design decisions lead to significant increases in mAP. Two exceptions are switching to a fully convolutional network with anchor boxes and using the new network. Switching to the anchor box style approach increased recall without changing mAP while using the new network cut computation by 33%.

Batch Normalization

CNN在训练过程中网络中的每一层的输入分布都在一直改变，这样会使得训练过程中的难度加大，通过归一化输入的方式可以消除这种问题。BN可以提升模型的收敛速度，而且可以起到一定的正则化的效果，降低模型的过拟合问题。通BN之后的模型在mAp上有显著地提升。

High Resolution Classifer

目前的目标检测方法中，基本上都会使用ImageNet预训练过的模型（classifier）来提取特征，如果用的是AlexNet网络，那么输入图片会被resize到不足256 * 256，导致分辨率不够高，给检测带来困难。为此，新的YOLO网络把分辨率直接提升到了448 * 448，这也意味之原有的网络模型必须进行某种调整以适应新的分辨率输入。对于YOLOv2，作者首先对分类网络（自定义的darknet）进行了fine tune，分辨率改成448 * 448，在ImageNet数据集上训练10轮（10 epochs），训练后的网络就可以适应高分辨率的输入了。然后，作者对检测网络部分（也就是后半部分）也进行fine tune。这样通过提升输入的分辨率，mAP获得了4%的提升。

Anchor boxes

召回率大幅提升到88%，同时mAP轻微下降了0.2。

借鉴Faster RCNN的做法，YOLO2也尝试采用先验框（anchor）。在每个grid预先设定一组不同大小和宽高比的边框，来覆盖整个图像的不同位置和多种尺度，这些先验框作为预定义的候选区在神经网络中将检测其中是否存在对象，以及微调边框的位置。同时YOLO2移除了全连接层。另外去掉了一个池化层，使网络卷积层输出具有更高的分辨率。

之前YOLO1并没有采用先验框，并且每个grid只预测两个bounding box，整个图像98个。YOLO2如果每个grid采用9个先验框，总共有13*13*9=1521个先验框。所以，相对YOLO1的81%的召回率，YOLO2的召回率大幅提升到88%。同时mAP有0.2%的轻微下降

Dimension priors

引入了anchor机制之后，为了能够更好的利用anchor，YOLOv2中引入了聚类机制，利用k-means算法对训练集中的目标框尺寸做聚类，找到更加适应于当前数据集的先验anchor尺寸。 需要注意的是，对目标框进行聚类操作时由于k-means算法的特性需要计算目标框和Anchor框之间的距离，如果直接采用欧式距离就会导致较大的目标框会比较小的目标框产生更大的训练误差，最终使得训练效果不理想。因此针对当前样本空间的距离，YOLOv2中使用如下公式中的距离定义方式(其中，centroid是聚类是被选作中心的边框，box 就是其它的边框，D就是两者之间的"距离"。IOU越大，相应的"距离"值就越近。)

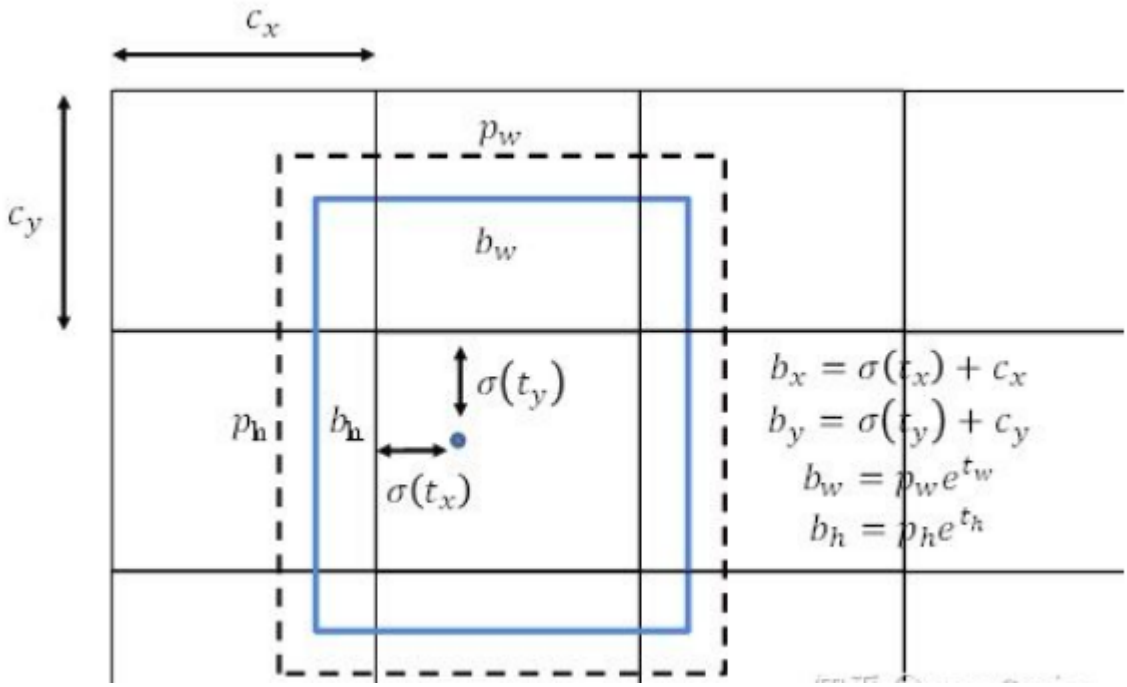
$$D(box, centroid) = 1 - IOU(box, cemtroid)$$

location prediction

Yolov2调整了先前的位置预测公式，公式如下：

$$\begin{aligned} b_x &= \sigma(t_x) + c_x \\ b_y &= \sigma(t_y) + c_y \\ b_w &= p_w e^{t_w} \\ b_h &= p_h e^{t_h} \\ Pr(object) * IOU(b, object) &= \sigma(t_o) \end{aligned}$$

其中，c_x, c_y表示该值为当前cell 相对于图像左上角的距离，t_x, t_y, t_w, t_h, t_o分别表示网络中预测的每一个anchor的位置偏移值和置信度，b_x, b_y, b_w, b_h表示最终的bounding box通过基于cell位置的预测结果，所有值的位置反应如下图：



2.2 网络模型

```
yolov2
layer     filters    size              input                output
    0 conv     32  3 x 3 / 1   608 x 608 x   3   ->   608 x 608 x  32  0.639 BFLOPs
    1 max          2 x 2 / 2   608 x 608 x  32   ->   304 x 304 x  32
    2 conv     64  3 x 3 / 1   304 x 304 x  32   ->   304 x 304 x  64  3.407 BFLOPs
    3 max          2 x 2 / 2   304 x 304 x  64   ->   152 x 152 x  64
    4 conv    128  3 x 3 / 1   152 x 152 x  64   ->   152 x 152 x 128  3.407 BFLOPs
    5 conv     64  1 x 1 / 1   152 x 152 x 128   ->   152 x 152 x  64  0.379 BFLOPs
    6 conv    128  3 x 3 / 1   152 x 152 x  64   ->   152 x 152 x 128  3.407 BFLOPs
    7 max          2 x 2 / 2   152 x 152 x 128   ->    76 x  76 x 128
    8 conv    256  3 x 3 / 1    76 x  76 x 128   ->    76 x  76 x 256  3.407 BFLOPs
    9 conv    128  1 x 1 / 1    76 x  76 x 256   ->    76 x  76 x 128  0.379 BFLOPs
   10 conv    256  3 x 3 / 1    76 x  76 x 128   ->    76 x  76 x 256  3.407 BFLOPs
   11 max          2 x 2 / 2    76 x  76 x 256   ->    38 x  38 x 256
   12 conv    512  3 x 3 / 1    38 x  38 x 256   ->    38 x  38 x 512  3.407 BFLOPs
   13 conv    256  1 x 1 / 1    38 x  38 x 512   ->    38 x  38 x 256  0.379 BFLOPs
   14 conv    512  3 x 3 / 1    38 x  38 x 256   ->    38 x  38 x 512  3.407 BFLOPs
   15 conv    256  1 x 1 / 1    38 x  38 x 512   ->    38 x  38 x 256  0.379 BFLOPs
   16 conv    512  3 x 3 / 1    38 x  38 x 256   ->    38 x  38 x 512  3.407 BFLOPs
   17 max          2 x 2 / 2    38 x  38 x 512   ->    19 x  19 x 512
   18 conv   1024  3 x 3 / 1    19 x  19 x 512   ->    19 x  19 x1024  3.407 BFLOPs
   19 conv    512  1 x 1 / 1    19 x  19 x1024   ->    19 x  19 x 512  0.379 BFLOPs
   20 conv   1024  3 x 3 / 1    19 x  19 x 512   ->    19 x  19 x1024  3.407 BFLOPs
   21 conv    512  1 x 1 / 1    19 x  19 x1024   ->    19 x  19 x 512  0.379 BFLOPs
   22 conv   1024  3 x 3 / 1    19 x  19 x 512   ->    19 x  19 x1024  3.407 BFLOPs
   23 conv   1024  3 x 3 / 1    19 x  19 x1024   ->    19 x  19 x1024  6.814 BFLOPs
   24 conv   1024  3 x 3 / 1    19 x  19 x1024   ->    19 x  19 x1024  6.814 BFLOPs
   25 route  16
   26 conv     64  1 x 1 / 1    38 x  38 x 512   ->    38 x  38 x  64  0.095 BFLOPs
   27 reorg              / 2    38 x  38 x  64   ->    19 x  19 x 256
   28 route  27 24
   29 conv   1024  3 x 3 / 1    19 x  19 x1280   ->    19 x  19 x1024  8.517 BFLOPs
   30 conv    425  1 x 1 / 1    19 x  19 x1024   ->    19 x  19 x 425  0.314 BFLOPs
   31 detection
```

网络在25层使用路由层的，并且特征图先使用１×１的卷积从26＊２６＊２６＊５１２降维到２６＊２６＊６４,在做reorg,将特征图一拆四，经过串联片便可以得到１９＊１９＊２５６的特征图，融合了粗细粒度的特征。

上面说了最后输出参数是19×19×425, 是因为一共有19×19个格子，每个格子预测出5个bounding box，每个bounding box预测85个数，其中80个是class的probability，其余5个参数中有四个表示stx、sty、tw、th，这4个来识别边框的位置和大小，还有1个数是confidence，表示边框预测里有真正的对象的概率，所以一共是19×19×425个数。

在yolov1中在网络层的最后的是全连接下的网络，这使得在检测时，我们只能对于特定大小的图片进行检测，yolov2改进了网络模型，去除了全连接层和dropout层，改为卷积链接的网络结构，这样使得的我们可以对任意输入大小的图片进行检测。yolov2在训练时，也采用多尺度训练，训练时使用的最小图像尺寸为320×320,最大图像尺寸为608×608，这使得网络可以适应多种不同的尺度。

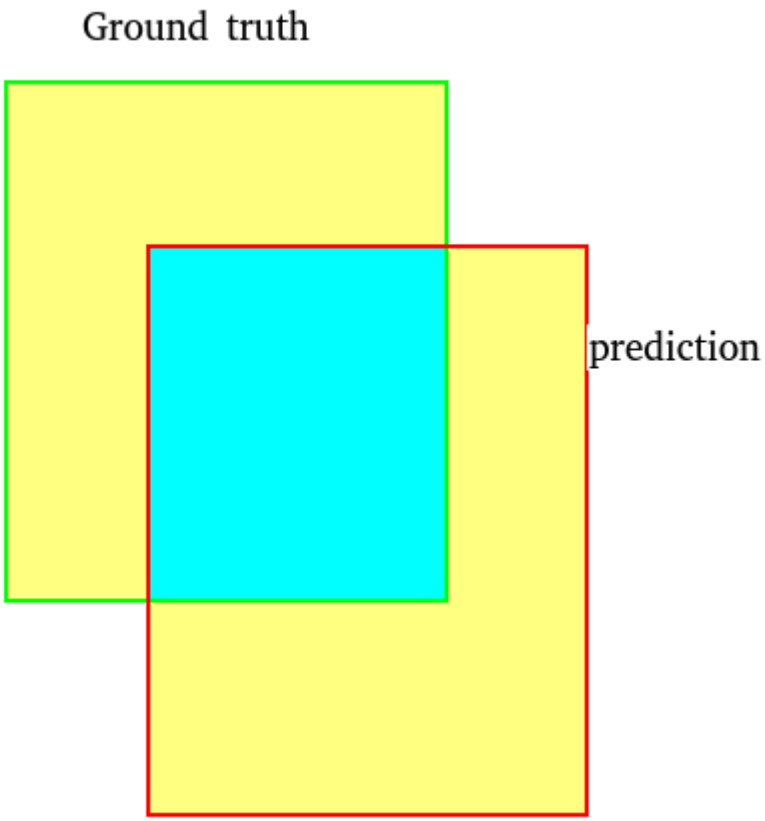
3.总结

yolov1的提出极大的推进了目标检测领域的发展，yolov2通过对以往模型进行改动，使得yolo模型更快、更健壮，这使得yolo一直成为目标检测的宠儿。自己在学习yolo模型的过程中对一些经典的检测算法整理，对一些暂时不能理解的问题，也进行了汇总，希望后续的学习中能够一一解决这些问题。

知识点汇总

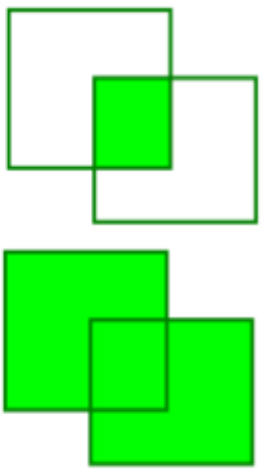
1). IOU(Intersection over Union):

IOU是用来计算预测框与真实框的相关度，IOU 的值越接近１，则表示他们的相关度越高，即表示预测窗口的效果越好，如下图（其中，绿色边框表示真实的对象边框，红色的边框表示预测的边框）：



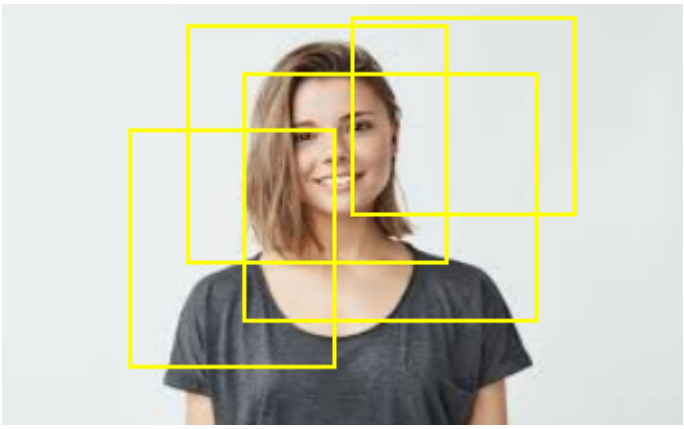
IOU的计算可以通过下面的公式进行计算：

$$IOU = \frac{\text{Area of overlap}}{\text{Area of union}}$$



2). NMS（非极大值抑制）：

在检测的过程中，由于对于同一目标对象可能存被在多个窗口框定，如下图：



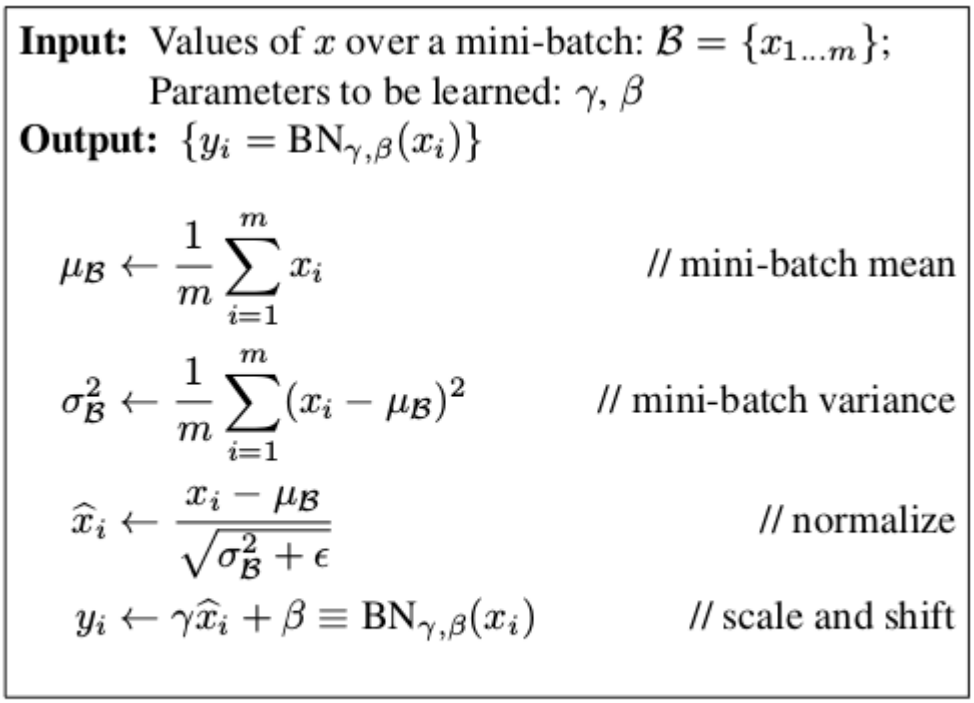
此时，为了的到一个唯一的检测窗口，需要消除冗余的边框，所以采用了NMS算法，ＮＭＳ算法的原理：首先会设定一个IOU的阀值，因为每一个检测窗口在检测时，都会有一个b-boxs 的预测值，此时，按照从高到低的顺序进行排列，随后，在其中选择一个最大值的窗口，遍历计算剩余窗口与次窗口的IOU值，如果IOU的值大于设定的阀值，则将窗口消除。然后，在剩余的窗口继续选择一个得分最高的窗口重复上述操作，直到所有的窗口都被处理。将上述图片处理后的效果如下图：

3). Batch Normalization

BN讲解１

BN见解2

计算公式如下：



Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

4**).Passthroush:**

本质其实就是特征重排，26×26×512的feature map分别按行和列隔点采样，可以得到4幅13×13×512的特征，把这4张特征按channel串联起来，就是最后的13×13×2048的feature map.还有就是，passthrough layer本身是不学习参数的，直接用前面的层的特征重排后拼接到后面的层，越在网络前面的层，感受野越小，有利于小目标的检测。