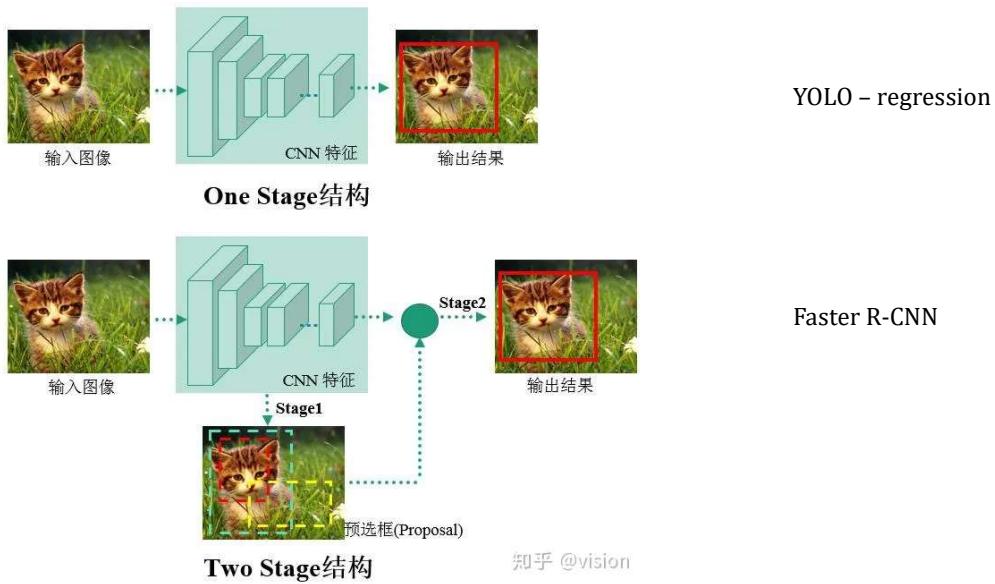


YOLO

YOLO



YOLO是一种基于深度学习的One-Stage目标检测算法，主要思路是直接通过卷积神经网络提取特征，预测目标分类与定位。

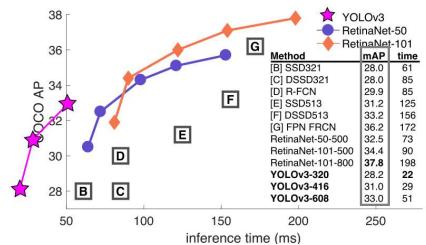
我们给一张输入图像，最后输出一个框，实际上我们只需要左上角的 x_1 y_1 左下角 x_2 y_2 坐标，相对来说我得到了四个预测结果，这就类似于一个回归问题。

在one-stage网络里用一个CNN网络去获得特征然后做回归就够了。

它比two-stage少了一个rpn也就是区域建议网络。就是说faster rcnn需要检测任务去得到一些坐标值，但是多做了一个预选操作，再从预选的结果中得到最终的结果。

指标分析

指标分析



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

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

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}$$

$$\text{F1 Score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2TP}{2TP + FP + FN}$$

mAP(mean Average Precision)

mAP -> AP -> Precision - Recall

		实际值	
		实际为正	实际为负
预测值	预测为正	True Positive	False Positive Type I Error
	预测为负	False Negative Type II Error	True Negative

Yolo系列的论文里用图件表明该version对上一个version有怎样的提高，这个指标叫mAP平均精度均值，很显然它于AP有关，那么它是怎么计算的？

预测可以分为四种情况，根据实际值和预测值有TP FP FN TN四种情况。

精度，表示预测为正中实际为正的比例，模型预测的所有目标中预测正确的比例。

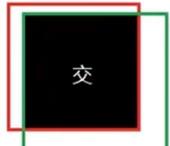
召回率，表示实际为正中预测为正的比例，真实目标中模型预测正确的比例。

如果从猫咖（人+猫）里检测猫，我检测出的这堆猫中真的是猫的比例；召回率表示猫咖里的猫有多少被我成功检测出来

AP，P-R曲线下的面积

mAP，各类别AP的均值

指标分析

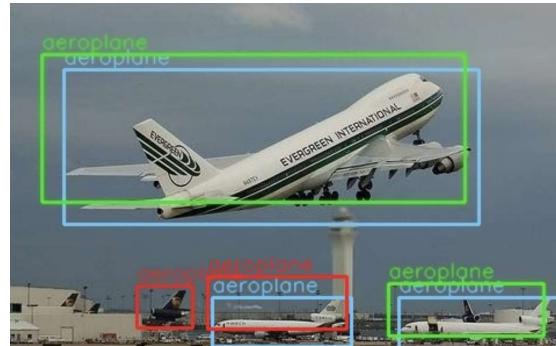


$$IOU = \frac{S_{\text{交}}}{S_{\text{并}}} > 0.5$$



$$\begin{aligned} \text{Precision} &= 2 / (2 + 2) = 1 / 2 \\ \text{Recall} &= 2 / (2 + 1) = 2 / 3 \end{aligned}$$

Confidence ?



真实框 预测框
正样本 负样本

ground truth

prediction

它会跟IOU有关，IOU表示预测框与实际框之间的交并比，比如红框为预测框绿框为真实框。

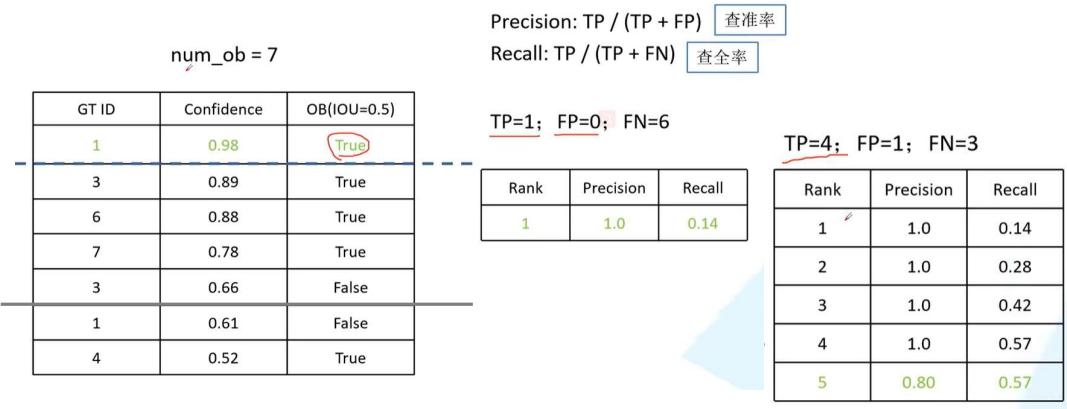
它用来判断预测框是正样本还是负样本。

精度，模型预测的所有目标中预测正确的比例。

召回率，真实目标中模型预测正确的比例。

Confidence – 这边界框里有物体的概率

指标分析



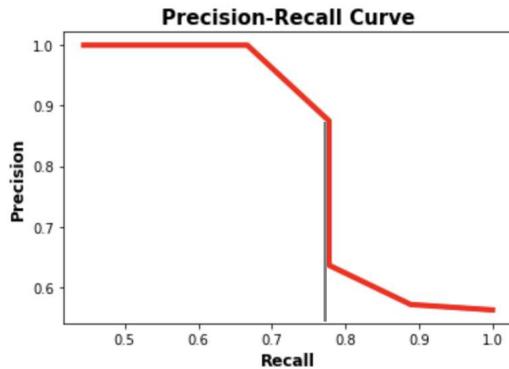
- Confidence + IOU -> positive or negative samples
- Confidence > 0.66 -> P IOU > 0.5 -> T

在目标个数为7, confidence为0.98时, TP正确检测到的目标个数, FP假阳性, 也就是误检个数, 这里只检测到一个目标所以是等于0的。总目标个数为7, 只检出1个, 漏检6个, 所以FN为6。因此可以计算P-R。

Confidence为0.66时, TP正确检测到的目标个数是4, 误检个数1, 没检出的为3, 所以相应的可以算出Precision和Recall的值。

因此可以总结得到: 当通过IOU判断是正样本的情况属于P, 当通过confidence阈值判断的正样本属于T, 二者都为正属于TP, 剩下的实际的GT但是置信度小于confidence阈值的属于FN。

指标分析



$$AP = \sum_{k=0}^{k=n-1} [Recalls(k) - Recalls(k + 1)] * Precisions(k)$$

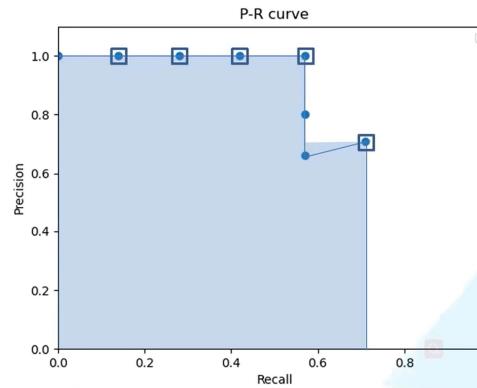
$Recalls(n) = 0, Precisions(n) = 1$
 $n = Number\ of\ thresholds.$

Precision和Recall有一个关系，就是Recall高，Precision就低，Recall低，Precision就高，他俩不是线性变化的。

拿Recall来说，它表示真实样本中有多少被正确检测出来，我们的模型肯定是有误差的，在误差不变的情况下我疯狂检测，这样一来总能检测到正样本吧，分子就上去了，但是精度就下来了，因为预测的正样本中误判的概率也上去了。

指标分析

Rank	Precision	Recall
1	1.0	0.14
2	1.0	0.28
3	1.0	0.42
4	1.0	0.57
5	0.80	0.57
6	0.66	0.57
7	0.71	0.71



$$(0.14 - 0) \times 1.0 + (0.28 - 0.14) \times 1.0 + (0.42 - 0.28) \times 1.0 + (0.57 - 0.42) \times 1.0 + (0.71 - 0.57) \times 0.71 = 0.6694$$

Precision和Recall有一个关系，就是Recall高，Precision就低，Recall低，Precision就高，他俩不是线性变化的。

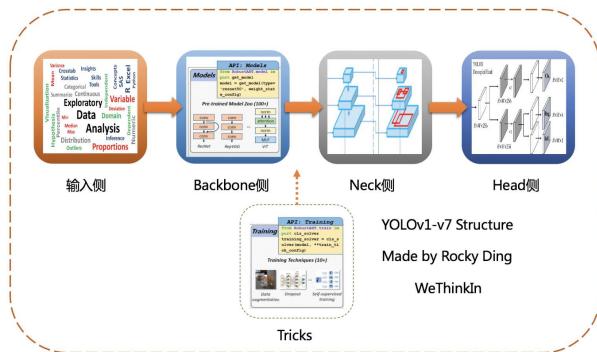
拿Recall来说，它表示真实样本中有多少被正确检测出来，我们的模型肯定是有误差的，在误差不变的情况下我疯狂检测，这样一来总能检测到正样本吧，分子就上去了，但是精度就下来了，因为预测的正样本中误判的概率也上去了。

系列演进

系列演进

- Network
- Forward Propagation
 - Loss Function
 - Back Propagation

- Structure
- Input
 - Backbone
 - Neck(v3 -)
 - Head

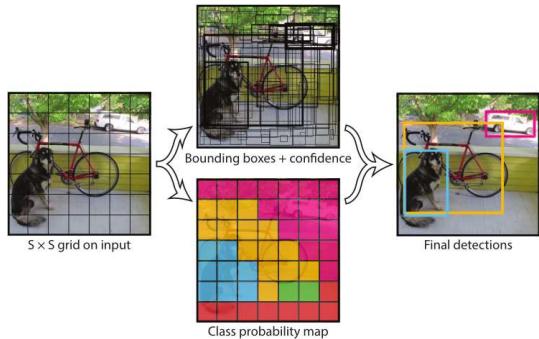


对于一个网络，前向传播、损失函数、反向传播十分重要，事实上就是在过程中分为训练阶段预测阶段（模型已经训练好之后输入未知图片，对未知图片进行预测/测试）。

结构上，目标检测的框架可以分为，neck-输出的特征进行整合，head-若干卷积层进行预测，它是在前面网络输出的特征上去进行预测，约等于是从这些信息里解耦出来图像中物体的类别和位置信息

所以归纳系列演进的时候也会从这几个方面去描述。具体来说就是以结构为基础，穿插讲一下每个阶段干了啥。

YOLOv1



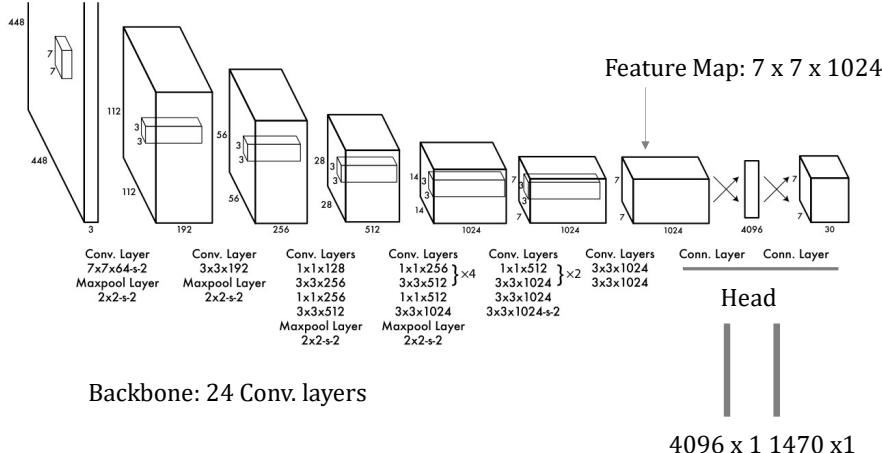
Innovation

- $S \times S$ grid cell
- grid cell containing an object center to predict that object
- one grid cell predicts B bboxes and a confidence

Yolo用于目标检测的，它就要解决对象识别和定位的问题。创新点在于它把整幅图像进行检测，使用 $s \times s$ 的grid cell，grid cell 包含某个object的中心的话就由该grid cell负责预测这个object。每个grid cell预测Bbboxes和一个confidence。

YOLOv1

Input: Resize images to 448 x 448



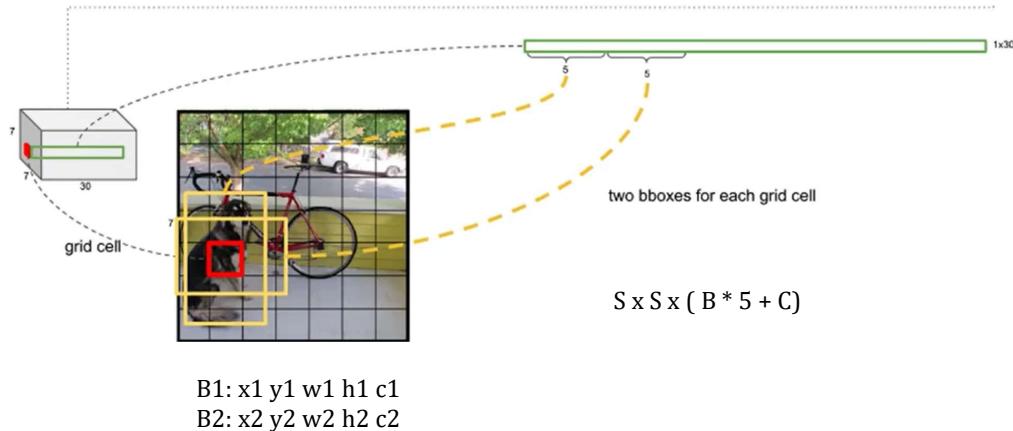
它的结构很简单，就是输入图片经过神经网络的变换得到一个输出的张量。 $448 \times 448 \times 3$ ，这里的resize是因为全连接层需要传入固定尺寸的输入。

为什么全连接层需要固定尺度的输入呢？Flatten拉平。因为权重矩阵固定好了，必须跟权重矩阵匹配不然无法相乘。

经过这些卷积层获得一个 $7 \times 7 \times 1024$ 的特征图。然后全连接层其实是有两层的，第二层的1470是为了方便reshape成 $7 \times 7 \times 30$ 的结果。

最后得到 $7 \times 7 \times 30$ 的张量，30表示啥呢？

YOLOv1



事实上对于每一个grid cell，它会预测两个B，每个B包含 $x1\ y1\ w1\ h1$ ，xy都不是真实的坐标，而是相对于grid cell的位置。

所以一个格子就有两个B，一个B包含5个值，所以就会有这样的公式。

S =图像尺寸/最大stride

YOLOv1

$S \times S \times (B * 5 + C)$

B1: $x_1 \ y_1 \ w_1 \ h_1 \ c_1$
B2: $x_2 \ y_2 \ w_2 \ h_2 \ c_2$

Have or Not 2
What 3
Where 1

$$\begin{aligned}
 & 1 \\
 & \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
 & + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \right] \\
 & 2 \\
 & + \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
 & + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
 & 3 \\
 & + \sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2
 \end{aligned}$$

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

$$\mathbf{C}_i = [c_{i1}, c_{i2}, \dots, c_{i20}]$$

$$P(\text{class}_j) = \frac{e^{c_{ij}}}{\sum_{j=1}^{20} e^{c_{ij}}}$$

$$P(\text{class}_i | \text{Object}) = \frac{e^{c_{ij}}}{\sum_{j=1}^{20} e^{c_{ij}}}$$

我们现在获得了一个输出数据，那么怎么来量化？也就是说怎么判断准不准？那么就会引出损失函数。

我们的数据可以从三个方面进行量化，一个是表达有没有物体、第二个这个物体是什么、第三个这个物体在哪。

第一个，有没有物体，Cell中，sum是有 $S \times S$ 个格子。如果格子内有物体，则 $\Pr(\text{Object})=1$ ，否则 $\Pr(\text{Object})=0$ 。但是计算机表达出来的是它有多大把握认为这个bbox里有物体。

它要输出0-1之间的数字来表达，但yolov1输出的是从 $-\infty$ 到 $+\infty$ 的任意实数，收敛很慢。

第二个，是什么，也是随机输出的，对于网格 (grid cell) i ，输出一个网格 (grid cell) 类别数据向量 C_i ，假设类别有20个，对于每个格子 i ，类别数据可以表示为， c_{ij} 表示网格 i 属于类别 j 的数据。

它这个不是真实的概率，也是在 $-\infty$ 到 $+\infty$ 取得，使用 softmax 函数进行归一化。但是只有存在物体的情况下才会使用20维数据。

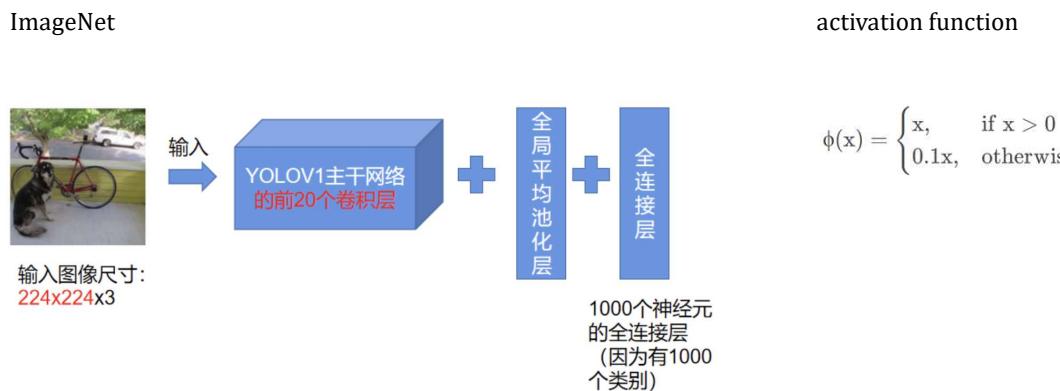
也就是说这里的 p 是条件概率： $P(\text{Class}_i | \text{Object})$ ，当前 bounding box 已经包含物体的条件下给类别的概率，20个条件概率拿出来和 bbox 的置信度相乘获得

第三个，在哪里。第一个 x_i 表示模型预测的 bounding box 的横坐标，为0-1之间的数。第二个 x_i 为标签值。

为什么这里要开根号，而不是像第一项一样直接用w呢？这是为了使得小框对于宽高的拟合损失更加敏感，就是说对同样的误差，小框产生的误差会更大，即对小框惩罚的更严重。

这边第i个gridcell是否包含物体， i_j 表示第i个gc的j个bbox负责预测物体为1否则为0.

YOLOv1-training



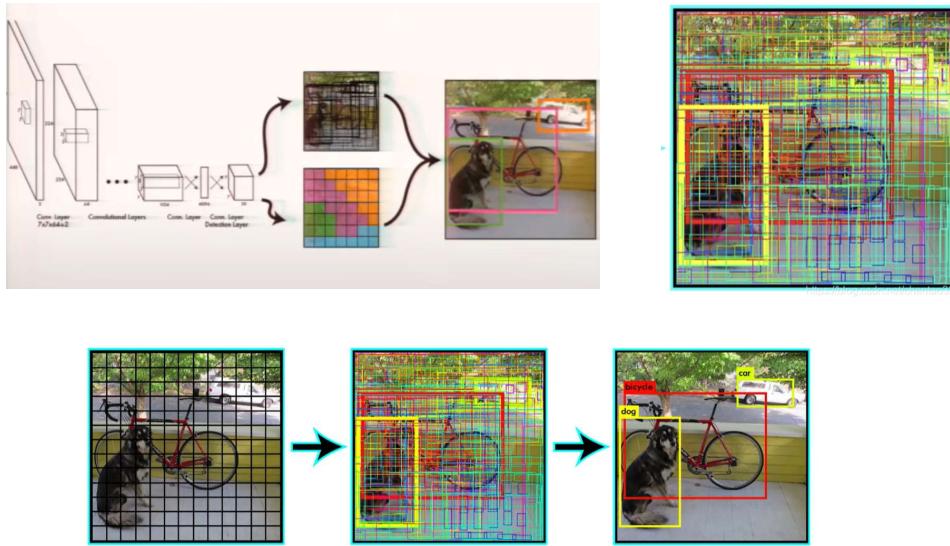
具体的训练过程首先拿出了网络结构的前20个卷积层，在其后面加上全局平均池化层和全连接层，在Image-Net1000个类别的**图像分类**数据集上对主干结构进行**预训练**，并且输入图像的尺寸是224x224。

为啥这里可以用224 x 224去预训练？因为加入了全局平均池化层，这样不论输入尺寸是多少，在和最后的全连接层连接时都可以保证相同的神经元数目。

经过上一步的预训练，就已经把主干网络的前20个卷积层给训练好了，前20层的参数已经学到了图片的特征。接下来的步骤本质就是迁移学习，在训练好的前20层卷积层后加上4层卷积层和2层全连接层，然后在**目标检测**的任务上进行迁移学习。

除最后一层采用ReLU函数外，leaky ReLU相对于ReLU函数可以解决在输入为负值时的零梯度问题。

YOLOv1-predicting

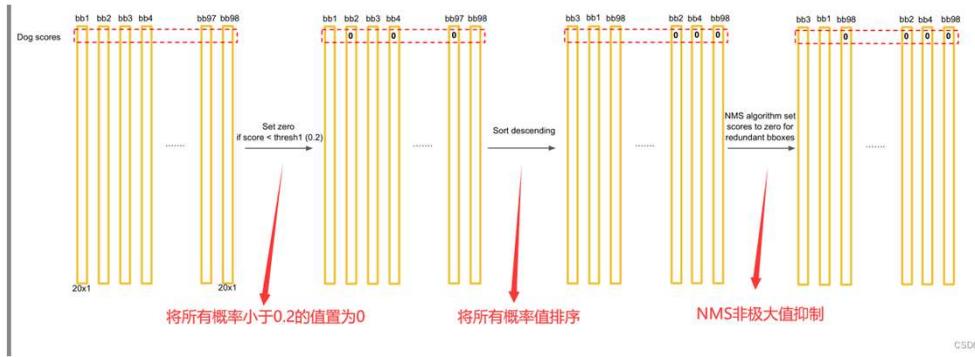


预测阶段，模型训练好之后要对未知图片进行预测。可以简化为前向推断问题，把流程过一遍得到 $7 \times 7 \times 30$ 的输入框

然后进入后处理，就是把复杂的预测框筛选过滤只保留一个，过滤掉低置信度和重复的框。（NMS）

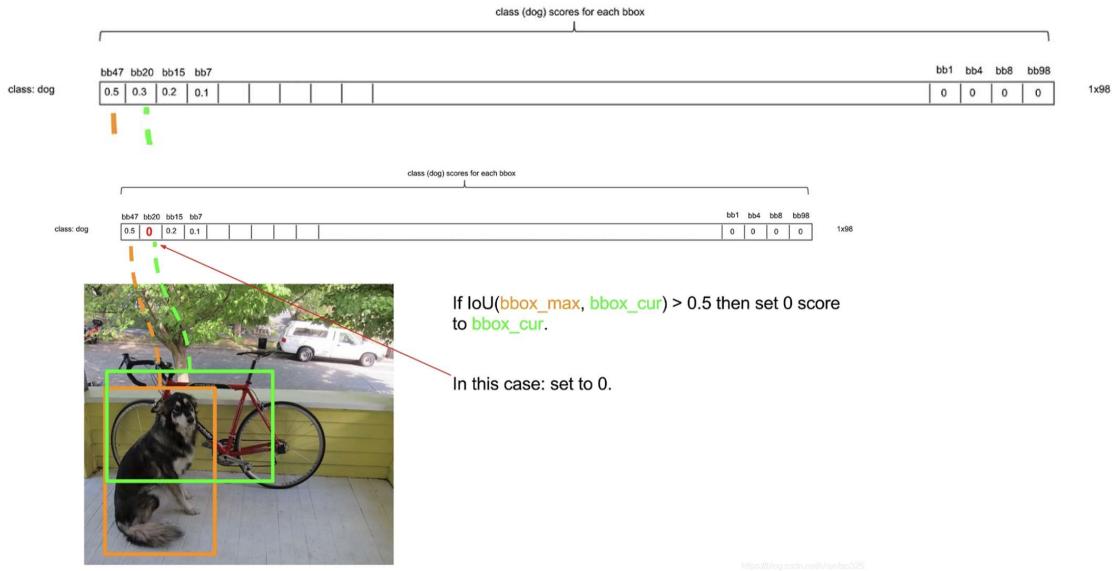
每一个预测框有一个confidence属性表征其包含物体的概率。但具体包含的是哪个物体，就需要计算出这个框包含每一类物体的概率。在YOLOv1中有20个类别的物体，所以输出结果的后20个值就表示每一个小网格（grid cell）对应每一类物体的概率。一张图片一共生产 $7 \times 7 \times 2$ 个框，但只有一部分框我们认为是真正找到物体的，所以我们还需要使用一些后处理方法，去掉无用的框。

YOLOv1-predicting



将二者相乘就是每个框是每个类别物体的概率。我们通过这个概率即可去掉无用的框，选出真正预测到物体的框。我们将所有框预测到的概率值按照每一行是一个概率列出来，并逐行处理。例如我们处理第一行——所有框关于类别狗的预测概率。我们先将所有小于0.2的预测概率置为0，然后将概率值按照降序进行排列，对排序后的概率采用**非极大值抑制（NMS）**。

YOLOv1-predicting



- 设置阈值

- 排序

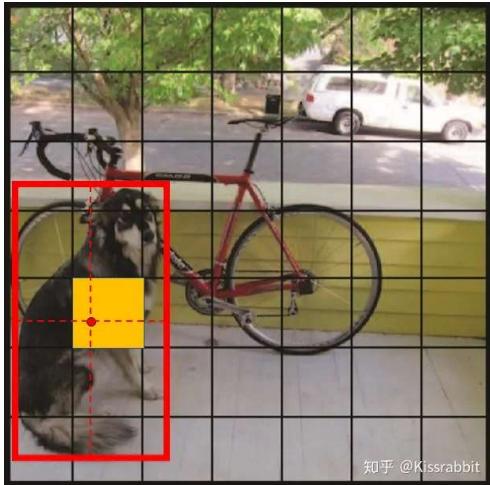
- NMS

先把第一个和第二个比，把他们的IOU>阈值，则把较低的过滤，没超过阈值就留着。

重复以上步骤，第一个和他后面的比，超过阈值的抹去，未超的保留，全部比完后转至第一个之后还有值的概率开始比较，重复以上步骤。

全部比完之后剩余的非 0 class score 的就是第一个类别的框了。

YOLOv1



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

知乎 @Kissrabbit

- 补充一下测试阶段怎么区分正样本，对于黄框，这一网格的objectness的概率为1。
 $\Pr(\text{objectness})=1$ 就是指正样本候选区域，只和label有关，因为gt box的中心点落在哪个grid，哪个grid就是正样本候选区域，也就是 $\Pr(\text{objectness})=1$ 。正样本候选区域的作用就是告知我们：该标签的正样本只会来源于此。

YOLO一共会输出三个预测，是否有物体的objectness预测、class预测和bbox预测。

首先，我们计算 $\text{score} = \text{objectness} * \text{class}$ 获得边界框的置信度confidence。

YOLOv1中给出的测试阶段bbox的置信度计算公式如图

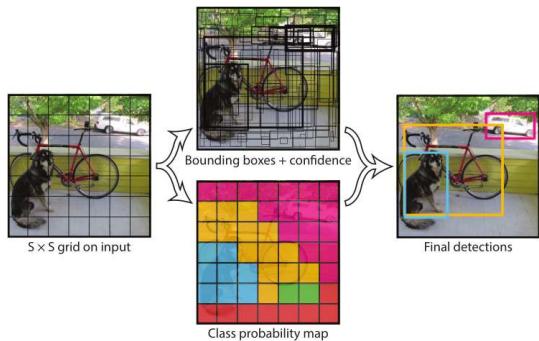
但是，测试阶段根本就没有真实框。所以问题就来源于这个IOU。

所以这里就是指YOLO预测的object，它的物理意义就是：这个grid cell是否有物体。

Reference-YOLOv1

- 你一定从未看过如此通俗易懂的YOLO系列(从v1到v5)模型解读 (上)
<https://zhuanlan.zhihu.com/p/183261974>
- <机器爱学习>YOLO v1深入理解
<https://zhuanlan.zhihu.com/p/46691043>
- YOLO系列全面解析 | YOLOv1 1234567X模型全系列大解析（Backbone篇）
https://blog.csdn.net/qq_38668236/article/details/127378254
- 1.1 YOLO入门教程: YOLOv1(1)-目标检测结构浅析
<https://zhuanlan.zhihu.com/p/364372881>
- 1.2 YOLO入门教程: YOLOv1(2)-浅析YOLOv1
<https://zhuanlan.zhihu.com/p/364367221>
- 全连接层输入为什么是固定维度的（拉直/压扁Flatten成为列向量）
https://blog.csdn.net/weixin_39306118/article/details/97904387
- YOLO系列----完全通透YOLOv1的思想
<https://zhuanlan.zhihu.com/p/595221376>
- 【精读AI论文】YOLO V1目标检测，看我就够了
https://www.bilibili.com/video/BV15w411Z7LG?p=6&vd_source=9ea40c1ac510f1e604555a3c8278ff94

YOLOv2

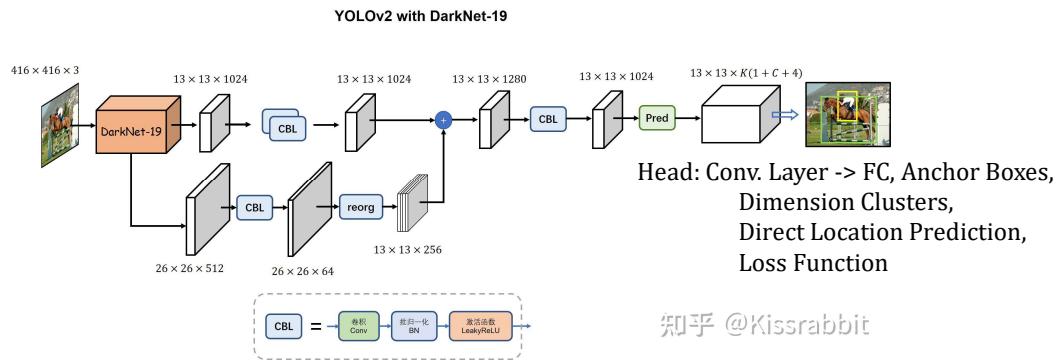


Innovation

- Batch Normalization
- High Resolution Classifier
- Convolutional With Anchor Boxes
- Dimension Clusters
- Direct location prediction
- Fine-Grained Features
- Multi-Scale Training

YOLOv2

Input: Multi-Scale Training & High Resolution Classifier



Backbone: DarkNet-19(+ BN & passthrough)

输入端做的改进是多尺度训练和高分辨率分类器，

Multi-Scale Training每隔几次迭代就改变网络，而不是固定输入图像的大小。每10批次我们的网络随机选择一个新的图像尺寸大小。这种机制迫使网络学会在各种输入维度上进行良好的预测。

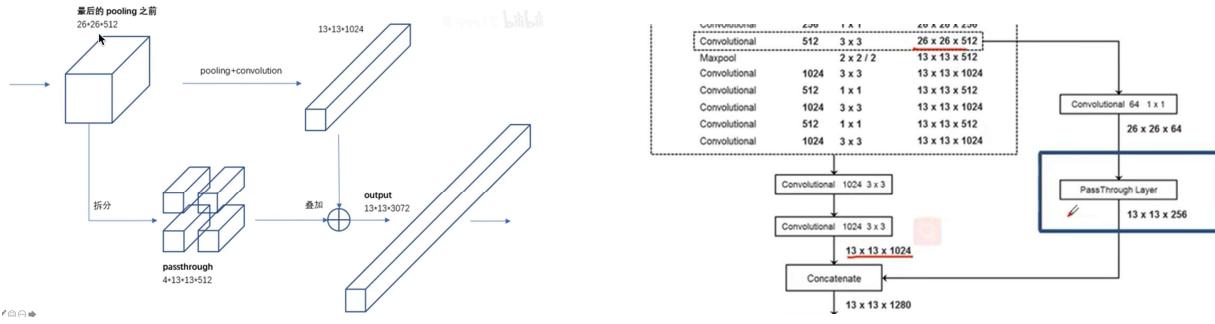
高分辨率分类器是因为v1的时候是 224×224 在imageNet上训练，检测的时候用的 448×448 ，那么网络就得切换分辨率。V2先 224 训练一会儿，再在 448 的图像分类数据上训练10个epoch，再在目标检测数据集上进行

BN，-均值/标准差，把它变为以0为均值标准差为1的分布。每个batch都会求均值和标准差，最后总的均值就用训练阶段很多batch的均值的期望。+是为了防止这个分母为0，

一般是在线性层后面激活函数的前面，加快收敛，**正则化**是通过对学习算法的修改，如在原约束函数上添加额外的约束和惩罚，改善模型在测试集上的表现，达到减少泛化误差、提高模型泛化能力的目的的技术。

那么这个passthrough是啥呢，简单来说它是将多张特征图在通道维度进行拼接，目的是使用更高分辨率的特征

YOLOv2

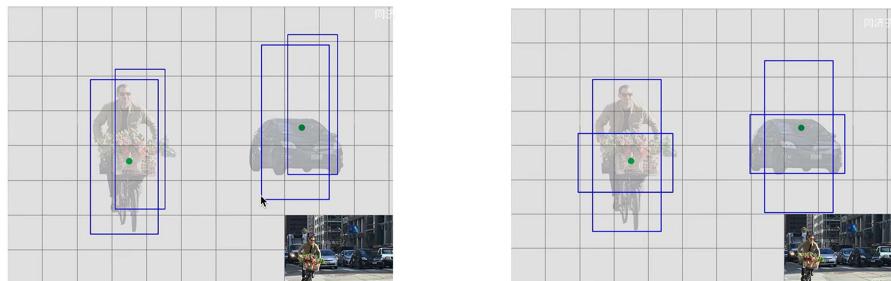


具体来说就是一路把浅层的网络拆分成4份，一路就进行正常的下采样也好卷积也好，把这2个结果在通道维度上面进行拼接。

最后这个结果就包含底层的细粒度信息和高层的信息。

YOLOv2

Anchor



YOLO V1的时候它并没有对bounding Box进行长宽限制，无论是这个高瘦的人还是这个矮胖的车都是由这2个bounding Box去预测的。

yolov2给他一个先验的参考框，这样可以避免大的变化，只学习若干偏移量。因此anchor box也叫先验框。

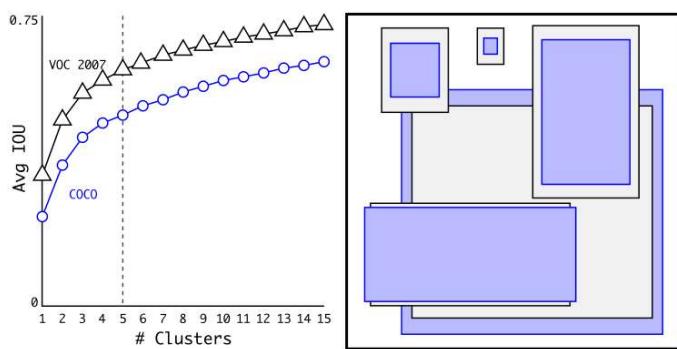
每一个预测框只需要知道它相对于这个anchor的偏移量就可以了。

v2中把一张图片分成 13×13 个grid cell每一个cell有5个anchor，每个anchor对应一个预测框，预测框只需要预测出它相对于anchor的偏移量即可。

他还是一个One stage的模型，因为它输入图片输出张量，目标检测结果都包含在张量里面。

YOLOv2

Anchor



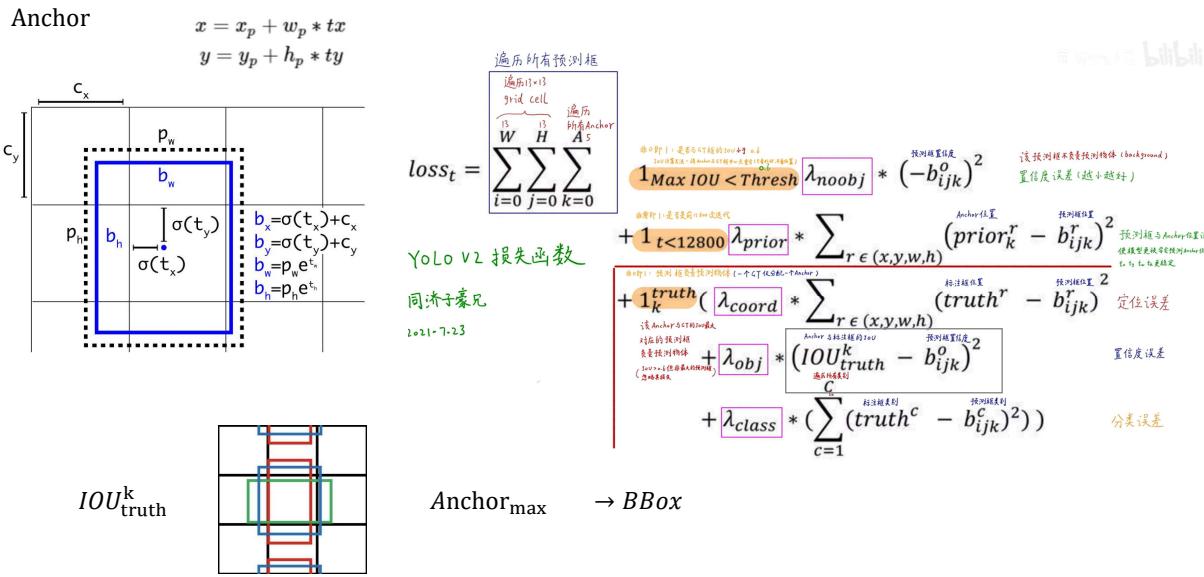
Dimension Cluster

那么这边又会有一个问题为什么anchor的个数是5?
Anchor的长宽比怎么指定?

手工选定的方法毕竟比较耗时耗力，
作者对voc和Coco数据集进行了长宽比的聚类，
不同的聚类中心对应着不同的IOU，聚类中心越
多anchor能覆盖的IOU就越大，但是anchor越多
模型就会变得越复杂，所以作者为了折中选择了
五。

基于训练集中所有目标的边界框用k-means聚类
的方法获得相应的先验框。

YOLOv2



在换上了新的先验框后，作者又对边界框的预测方法做了相应的调整。

首先，对每一个边界框，YOLO仍旧去学习中心点偏移量tx和ty。

在YOLOv1时，直接使用线性函数输出，这显然是有问题的，在训练初期，模型很有可能会输出数

值极大的中心点偏移量，导致训练不稳定甚至发散。作者使用sigmoid函数使得网络对偏移量的预测是处在01范围内。

其次，对每一个边界框，由于有了边界框的尺寸先验信息，故网络不必再去学习整个边界框的宽高了。

所以这个pw ph是先验框的宽高。Cx cy表示grid cell左上角相对于feature map左上角的距离。

相应的，损失函数也变成了这样。首先遍历所有的预测框，遍历 13×13 的grid cell，遍历了所有的anchor。

第一部分是计算background的置信度误差，但是用哪些预测框来预测背景呢？

这里需要计算各个预测框和所有的ground truth之间的IOU值，并且取最大值记作MaxIOU，如果该值小于一定的阈值，0.6，那么这

一个预测框就标记为background。

第一项是这个预测框是否与GT的IOU小于0.6，这边的IOU是将anchor与GT框中心点重合就是只看形状不看位置，（0-预测框置信度），0就是它的标签，

第二项是模型训练早期，要让anchor的坐标和预测框的坐标尽可能接近。为了让tx ty tw th更稳定。

第三项是对于预测框负责预测物体，一个GT分配一个anchor，该anchor与GT的iou最大对应的预测框负责预测物体。

让标注框与预测框位置尽可能重合，第二个，Anchor与gt的IOU就是标签，，第三项是标注框的类别和预测框的类别，都是20维向量，逐元素做差平方再求和。

这里补充一下正负样本的理解，这也是yolov1和v2的区别所在，

对于正样本：v1是选择两个预测框与真实框IOU最大的那个，

由于v2引入了anchor，做法是：计算当前cell中的所有Anchor 与 当前标注框的IOU，选择最大的IOU的Anchor 对应的 预测框 作为正样

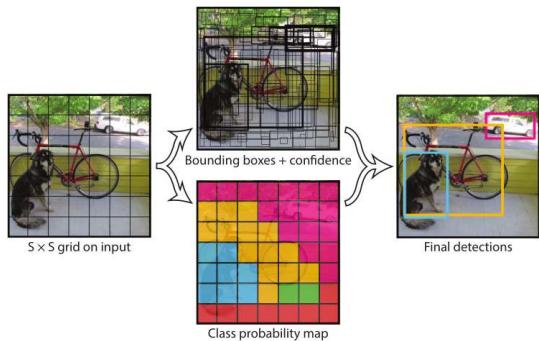
本，并且计算该IOU，
对于负样本：

v2 先计算当前cell 中 每个 预测框
与 所有 标注框 的 最大IOU， 若每
个最大IOU小于等于0.6， 那么设
置为负样本。

Reference-YOLOv2

- 3.1 YOLO系列理论合集(YOLOv1~v3)
https://www.bilibili.com/video/BV1yi4y1g7ro?p=2&vd_source=9ea40c1ac510f1e604555a3c8278ff94
- 【精读AI论文】YOLO V2目标检测算法
https://www.bilibili.com/video/BV1Q64y1s74K/?spm_id_from=333.999.0.0&vd_source=9ea40c1ac510f1e604555a3c8278ff94
- 【目标检测-YOLO】YOLO v2总结
<https://blog.csdn.net/hymn1993/article/details/123040206>

YOLOv3



Innovation
- Network Structure
- Multi-Scale

V1和V2对小目标检测性能较差
V3-spp版

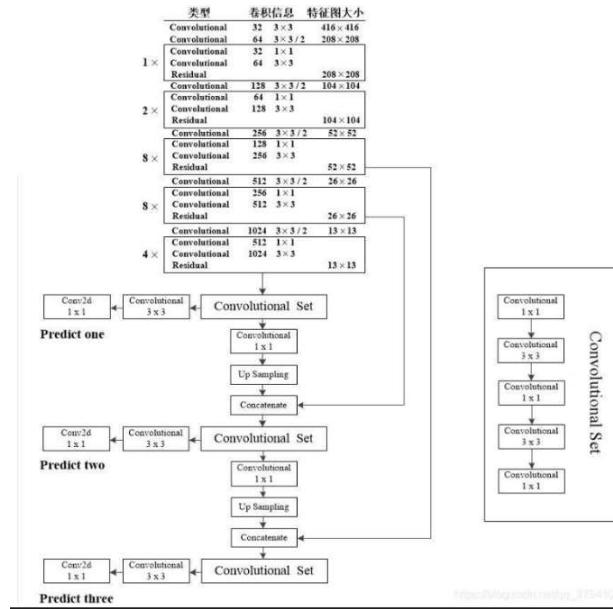
YOLOv3

Input: Data Augmentation

Backbone: DarkNet-53

Neck: FPN + SPP

Head: Logistic Classifiers,
Loss Function



Input方面spp版是使用了Mosaic数据增强，使用多张图片拼接进行增强。增加数据的多样性

Backbone方面使用了**DarkNet-53**，53层卷积，把池化层也给删掉了，添加了残差网络中的残差连结结构，以提升网络的性能。

Neck: FPN的思想，增加较少计算量的前提下融合低分辨率，语义信息较强的特征图，和高分辨率，语义信息较弱但空间信息丰富的特征图。

以支持后面的Head侧采用多尺度来对不同size的目标进行检测。

SPP的话是我看yolov3-spp版的里面在Darknet-53和预测特征图之后加了一个spp模块，

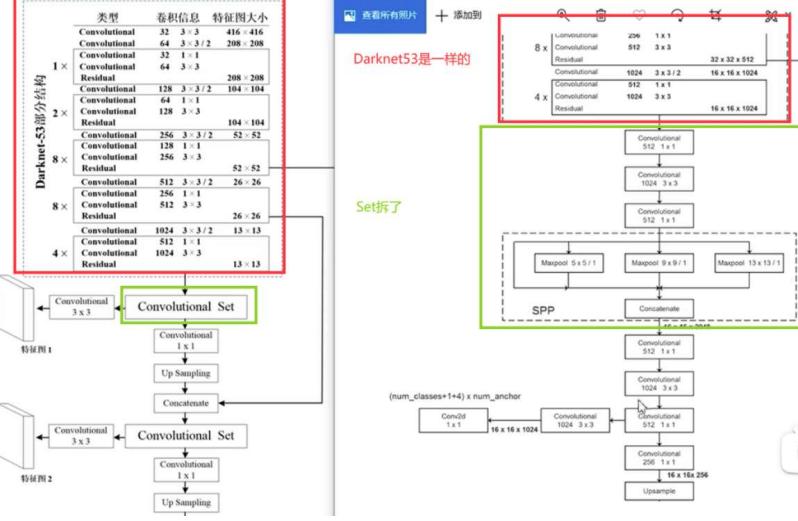
输出会输出三个尺度的Feature Map $13 \times 13 \times 255$ $26 \times 26 \times 255$ $52 \times 52 \times 255$

Head上引入了多尺度检测逻辑和多标签分类思想，优化了损失函数。

Head侧将用于单标签分类的Softmax分类器改成多个独立的用于多标签分类的Logistic分类器，取消了类别之间的互斥，可以使网络更加灵活。

Logistic分类器主要用到Sigmoid函数，可以将输入约束在0到1的范围内，当一张图像经过特征提取后的某一检测框类别置信度经过sigmoid函数约束后如果大于设定的阈值，就表示该检测框负责的物体属于该类别。

YOLOv3

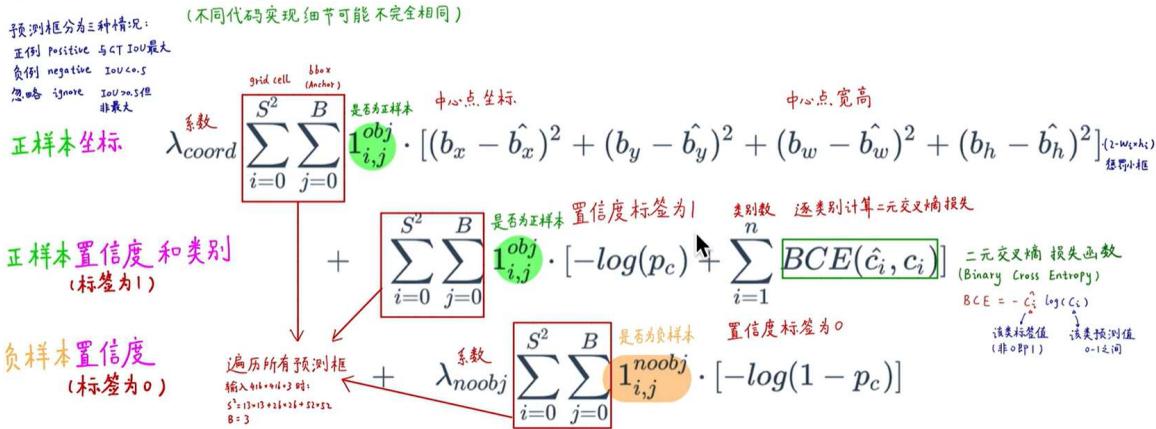


concatenate的是四个输出，一个是直接从输入接到输出，另外三个是经过最大池化下采样得到的。

因此深度 $\times 4$ ，变为 $16 \times 16 \times 2048$ 。问题：为什么只在第一个预测特征层之前接入 SPP 结构呢？因为只添加一个就够了，加很多会提高推理速度。

YOLOv3

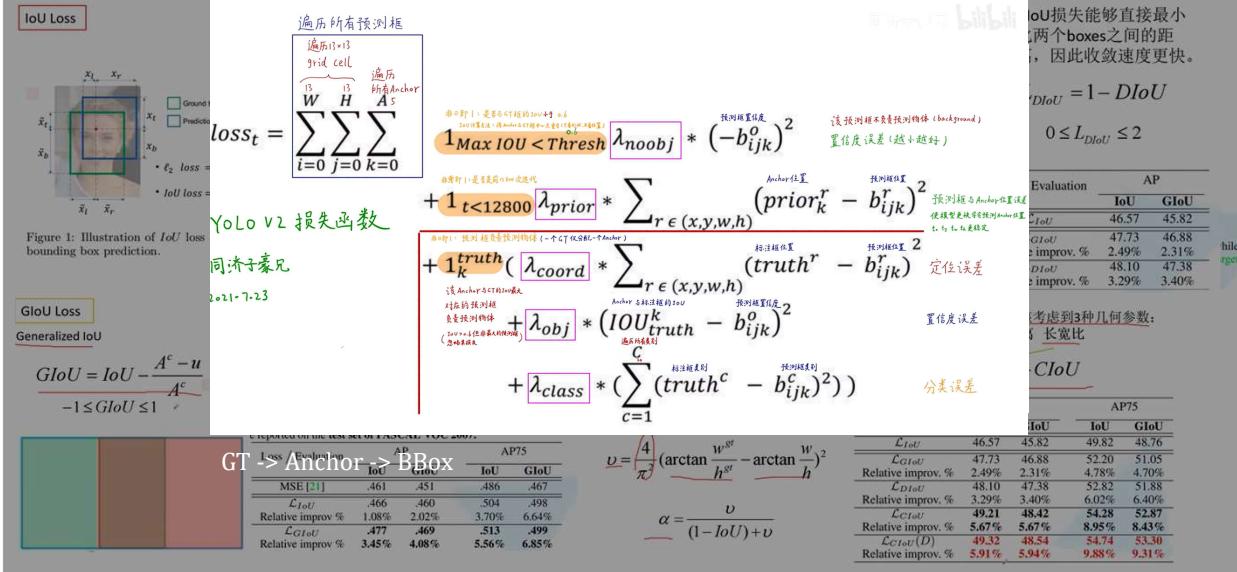
YOLO v3 目标检测 损失函数 同济子豪兄



损失函数，第一项正样本坐标，老样子，后面那一项是为了惩罚小框
 第二项，正样本置信度和类别，这个时候正样本置信度标签就变成了1，v1v2都是IOU，这里就是1，
 然后是正样本的类别，这里它不再用回归问题的损失函数了，这里用的是二元交叉熵损失函数
 第三项负样本的置信度，负样本置信度标签为0，和v1v2都一样。

YOLOv3

IoU Loss -> GIoU Loss -> DIoU Loss -> CIoU Loss



Spp版里对定位损失做了一些改进，之前就是插值平方，但是对于这三个预测框，明显最后一个效果更好，但是他们的I2损失都是一样的。所以引入了IoU loss，IoU Loss有个问题，在真实框与预测框不重合的时候分子=0 loss=0

绿色-真实目标边界框 红色-预测目标边界框 蓝色-\$A^c\$-\$u\$-绿红并集。

然后作者又发现前面两个的缺点：收敛慢、回归不精确。第一行：GIoU训练网络，让预测（蓝色框）尽可能回归到真实目标边界框（绿框）那里去，黑色代表anchor，直到迭代400步才勉强重合，DIoU120步就够了。右边这个图，IOU和GIoUloss都一样，所以也不能很好地表达关系。

于是就提出了DIoU，p表示真实框（绿框）和预测框（黑框）之间的欧氏距离。B表示中心坐标。d-两个中心的距离，c-最小外接矩形的对角线长度。

重合在一起d=0，就等于IOU，等于1，无穷远的时候趋近于1所以0-1=-1

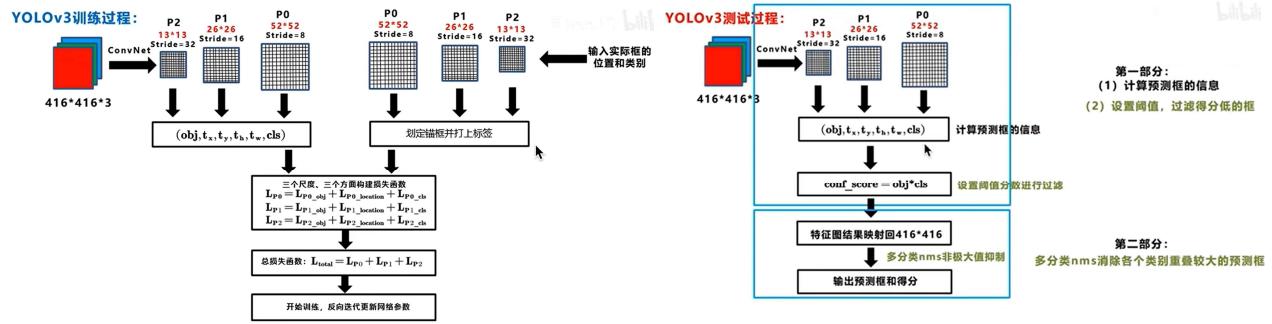
然后作者又认为一个优秀的回归定位损失要考虑三个方面，DIoU没考虑长宽比，于是又加了一个因子。

Spp版里还又focal loss的部分，本身是为了解决正负样本不均衡的问题，但是在v3里效果不大。可以看作是一个损失函数，它使容易分类的样本权重降低，而对难分类的样本权重增加。

这是v2的损失函数，v3与v2的思想差不多，这个图片能更好地反应损失函数的设计所以拿过来用下。

一个gt只分配给一个最高iou的anchor，对于那些IOU不是最高，但高于阈值的anchor是忽略的，对于小于阈值的anchor只有**objectness loss**，这个方法筛选了很多预测框，而**objectness loss**只是很小的一部分；计算loss的时候，统计的是那些已经和gt匹配上的loss，其实这就是解决正负样本不均衡的一个办法。

YOLOv3

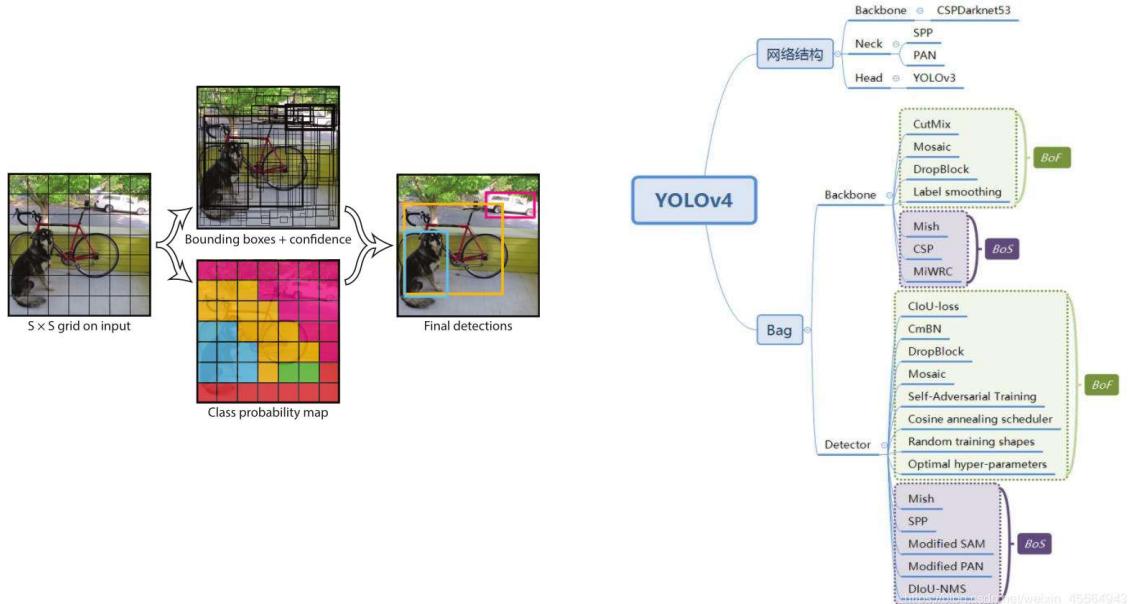


训练过程中会输出3个尺度的10647个预测框，分别对应10647个标签。那么在训练的时候就让每一个框的85个数和标签进行拟合，用损失函数进行训练，迭代更新卷积层中的权重。
测试过程输入，获取三个尺度的特征，解析，进行置信度阈值和非极大值抑制。

Reference-YOLOv3

- 3.1 YOLO系列理论合集(YOLOv1~v3)
https://www.bilibili.com/video/BV1yi4y1g7ro?p=2&vd_source=9ea40c1ac510f1e604555a3c8278ff94
- 【精读AI论文】YOLO V3目标检测（附YOLOV3代码复现）
https://www.bilibili.com/video/BV1Vg411V7bj/?spm_id_from=333.999.0.0&vd_source=9ea40c1ac510f1e604555a3c8278ff94
- YOLO算法全家桶！同济博士带你把YOLOV1/V2/V3/V4/V5目标检测算法
https://www.bilibili.com/video/BV1Fv4y1c7k7?p=19&vd_source=9ea40c1ac510f1e604555a3c8278ff94
- YOLO v3采用focal_loss 效果看法
https://blog.csdn.net/qq_34795071/article/details/89536836

YOLOv4



v4, 很乱

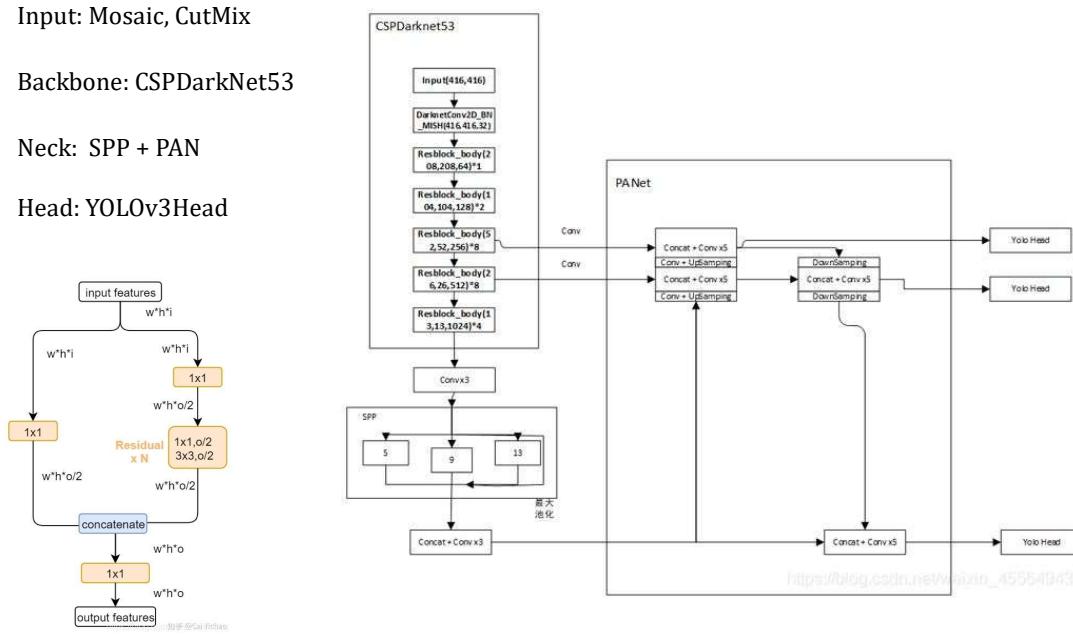
YOLOv4

Input: Mosaic, CutMix

Backbone: CSPDarkNet53

Neck: SPP + PAN

Head: YOLOv3Head



Input: Mosaic利用四张图片，分别对四张图片进行翻转、缩放、色域变化等，CutMix. CutMix则是在Cutout的基础上对置0的像素区域随机填充其他图像的部分像素值，label则按同等比例进行分配。

Backbone: 加了CSP，原输入分成两个分支，分别进行卷积操作，学习到更多特征。五层残差网络resblock_body组成，其主要功能是提取图像数据的特征信息。

Neck: SPP+PAN，SPP代替了卷积层后的常规池化层，能获取多尺度特征。

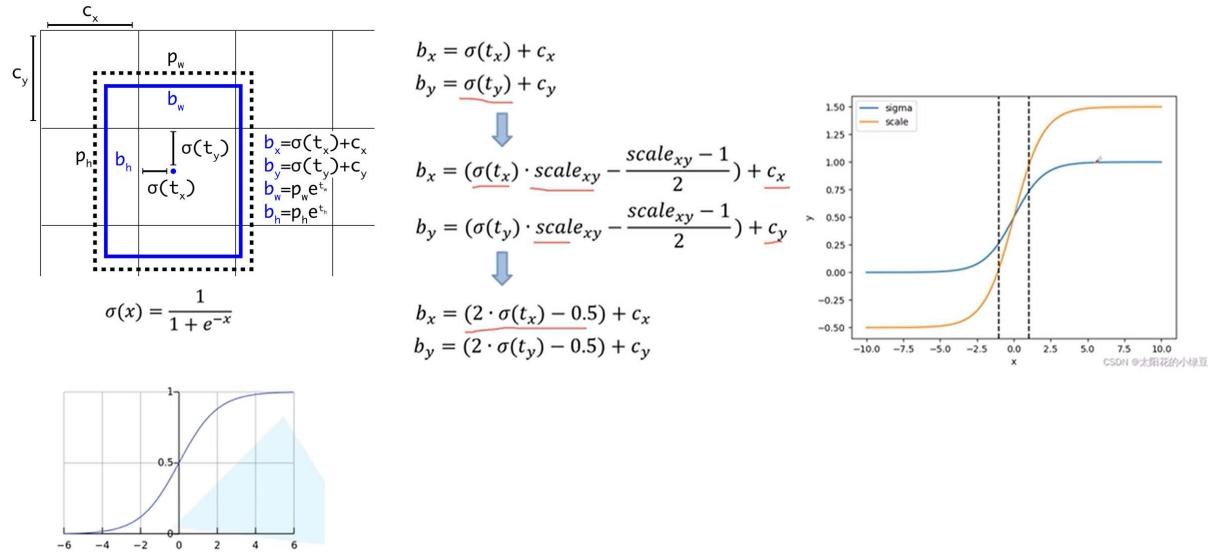
这边有一个PAN结构，把高层的语义信息往低层进行融合，然后又把低层的语义信息往高层进行融合。

Head:侧还是延续了yolov3的架构，得到输出之后，与真实数据标注相比较，计算出损失函数。引入了CIOU Loss

这边有一些优化策略，

YOLOv4

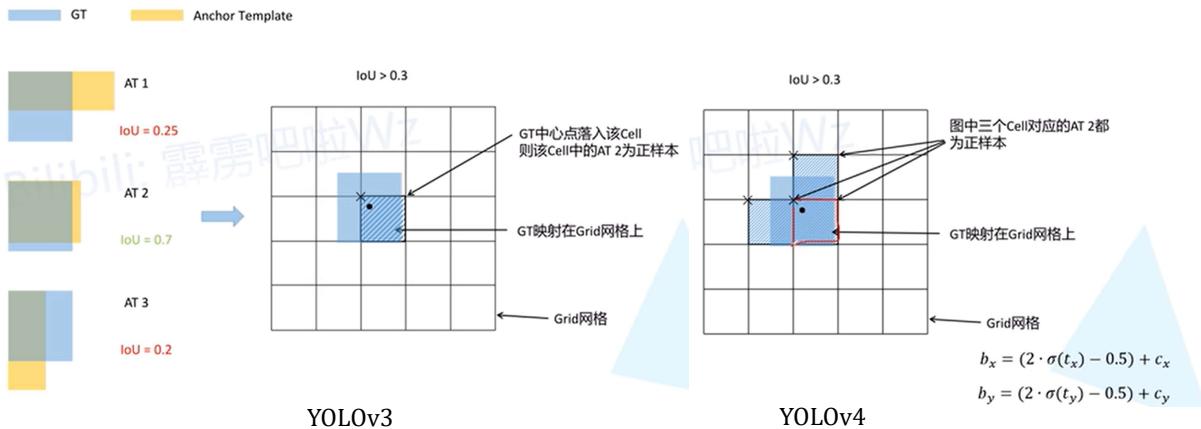
Eliminate grid sensitivity



在YOLOv3中，边界框的中心点回归公式如图，但是对于极端情况：
如果物体的中心点恰好落在网格的右边界时，sigma输出为1，但是输入正无穷才可以。
所以为了解决这个问题引入了一个大于1的系数，scale_xy，

YOLOv4

IoU threshold



之前匹配正样本时是用gt和每一个anchor模板进行匹配，计算IoU，yolov3中是选择IoU最大的anchor作为正样本，v4如果有多个> 阈值的就都选为正样本。

V4的话差不多，之前也是选定gt at，阈值，gt映射到grid cell。但是对于红色这个grid cell旁边两个也会被选为正样本。

原因是经过敏感度的优化，网络预测相对于左上角的偏移量相对0-1缩放到-0.5-1.5之间，所以上面这个x轴方向< 0.5 y轴方向<1.5，左边的同理。

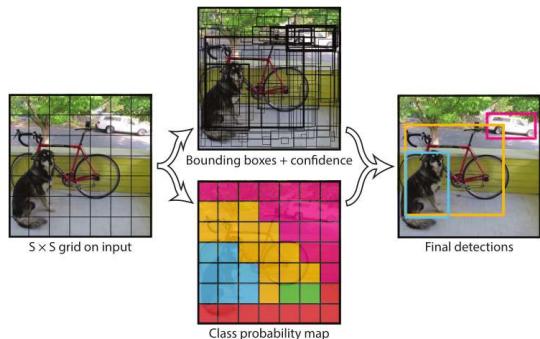
这两步就能扩充正样本。

其他都是tricks了什么bag of freebies bag of special用到再去了解吧。

Reference-YOLOv4

- YOLOv4网络详解
https://www.bilibili.com/video/BV1NF41147So/?spm_id_from=333.337.search-card.all.click
- 4.1 YOLO入门教程：YOLOv4(1)-浅析YOLOv4
<https://zhuanlan.zhihu.com/p/454472695>
- yolo系列学习笔记---yolov4（SPP原理）
<https://blog.csdn.net/YOULANSHENGMEENG/article/details/121909125>

YOLOv5

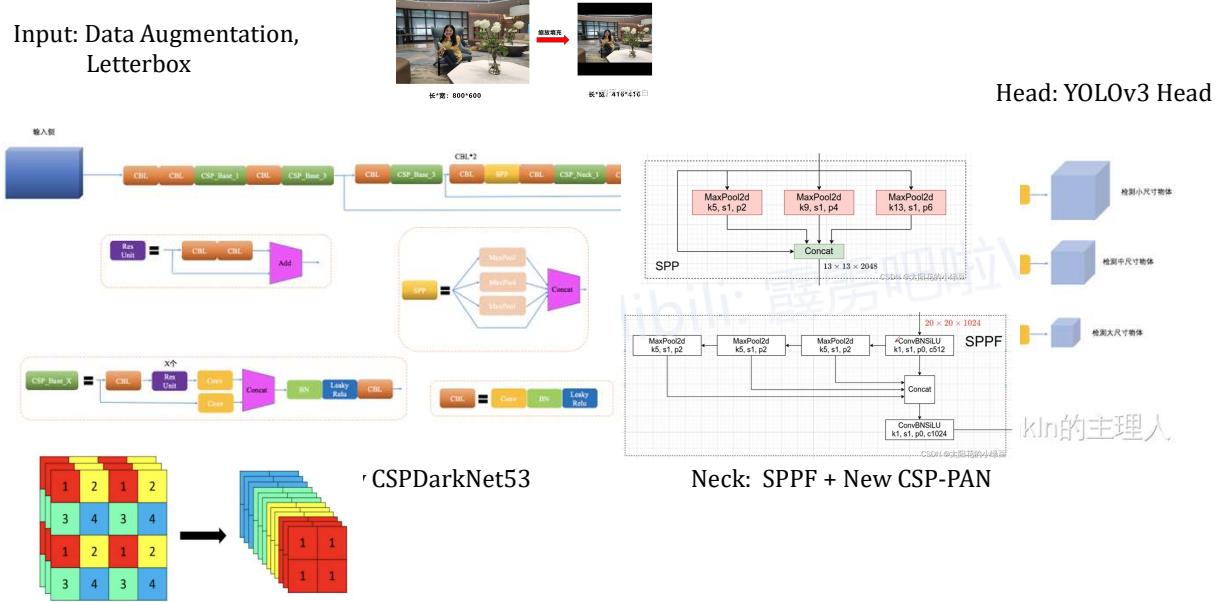


Innovation

- Network Structure
- Data Augmentation
- Optimizing Strategy
 - Loss Function
 - Eliminate Grid Sensitivity
 - Build Targets

V5, 还是从网络结构 数据增强 训练策略 其他损失计算 平衡损失 消除Grid敏感度
匹配正样本

YOLOv5



Input: Mosaic, Copy paste (不同图像中的物体扣下来复制粘贴一下) 自适应图片缩放，一开始需要对图片统一缩放到一个标准尺寸，再送入检测网络中。

缩放填充后，这个黑边如果比较多，信息冗余，影响推理速度。Yolov5是对原始图像自适应的添加最少的黑边。

步骤是先计算缩放比例、再计算缩放尺寸、计算黑边填充数值。取余数再/2，这里我没有看懂为啥会这样。

Backbone: 使用Focus结构，v5新增的，以yolov5s为例，原始的 $640 \times 640 \times 3$ ，采用切片操作，先变成 $320 \times 320 \times 12$ 的特征图，再经过一次卷积操作，最终变成 $320 \times 320 \times 32$ 的特征图。都是从计算量和参数量出发。但是在6.1版本之后给替换成 6×6 的普通卷积层了。

Neck: SPPF结构，串行地依次进过，计算量会减少。在PAN模块进行融合后，将YOLOv4中使用的常规CBL模块替换成借鉴CSPnet设计的CSP_v5结构，加强网络特征融合的能力。

Head类似

YOLOv5

YOLOv5的损失主要由三个部分组成：

- **Classes loss**, 分类损失, 采用的是BCE loss, 注意只计算正样本的分类损失。
- **Objectness loss**, obj损失, 采用的依然是BCE loss, 注意这里的obj指的是网络预测的目标边界框与GT Box的CIoU。这里计算的是所有样本的obj损失。
- **Location loss**, 定位损失, 采用的是CIoU loss, 注意只计算正样本的定位损失。

$$Loss = \lambda_1 L_{cls} + \lambda_2 L_{obj} + \lambda_3 L_{loc}$$

针对三个预测特征层 (P3, P4, P5) 上的obj损失采用不同的权重。

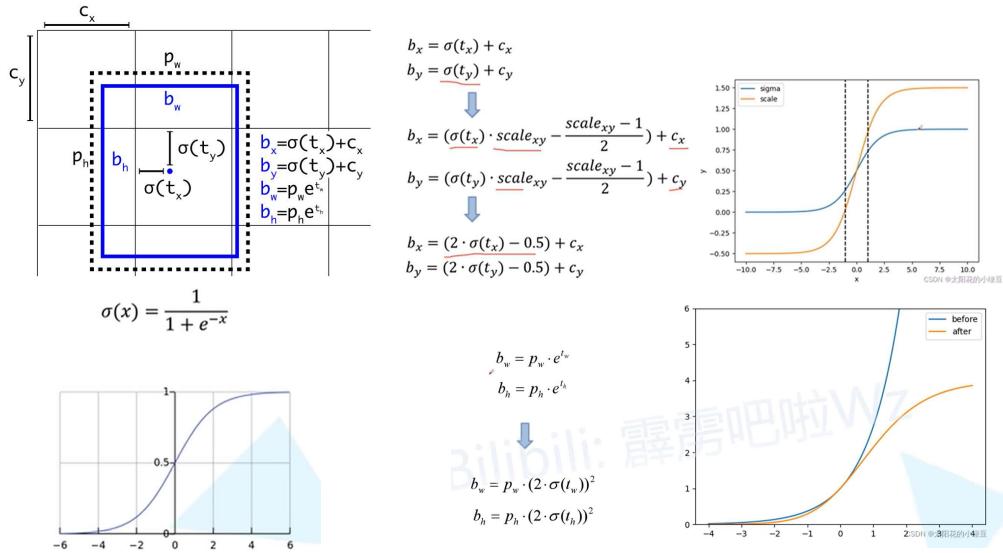
$$L_{obj} = 4.0 \cdot L_{obj}^{small} + 1.0 \cdot L_{obj}^{medium} + 0.4 \cdot L_{obj}^{large}$$

区别在于v3v4的时候Objectness loss是当前边界框中是否有目标, 有目标为1没有就为0, v5这里objectness loss也变成CIoU了。

对于obj损失, 小型目标的权重要更大一些。

YOLOv5

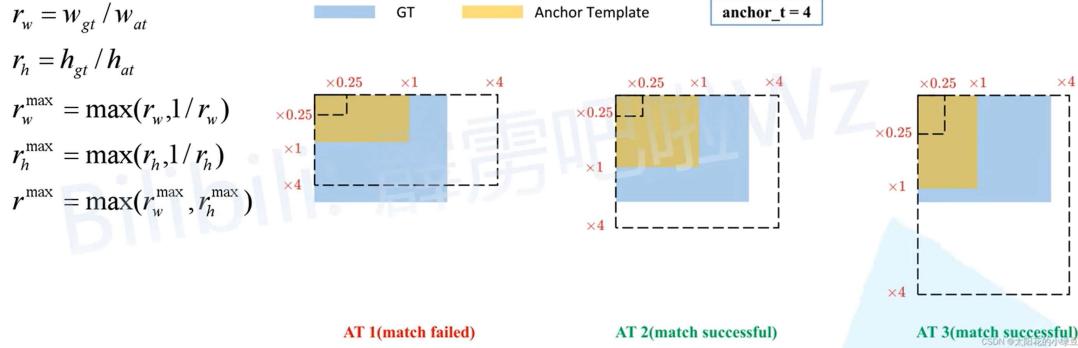
Eliminate grid sensitivity



V5中针对宽高做出了一些调整，作者说 e^x 不受限。Tw th非常大就会导致训练不稳定

YOLOv5

Match positive samples



不同的地方在于gt和at进行匹配的部分。之前是gt和at计算iou，大于阈值就将gt分配给at，

V5是先计算gt和at宽高比值，然后分别计算rw和rw分之一的最大值。

这俩max可以看作比较宽高方向的差异，如果完全重合的话max=1，不重合那肯定是越不吻合差异就越大。

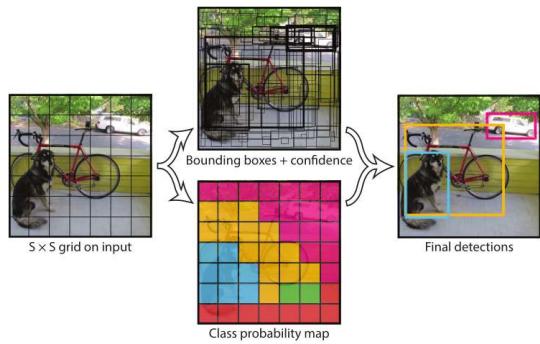
选择宽度高度方向差异最大的样本，如果 $r^{\max} < \text{anchor_t}$ 那么就是匹配成功了。

这个4是因为之前修改敏感度的那个公式，最大不会超过4.

Reference-YOLOv5

- YOLOv5网络详解
https://www.bilibili.com/video/BV1T3411p7zR/?spm_id_from=333.337.search-card.all.click&vd_source=9ea40c1ac510f1e604555a3c8278ff94
- 深入浅出Yolo系列之Yolov5核心基础知识完整讲解
<https://zhuanlan.zhihu.com/p/172121380>

YOLOX

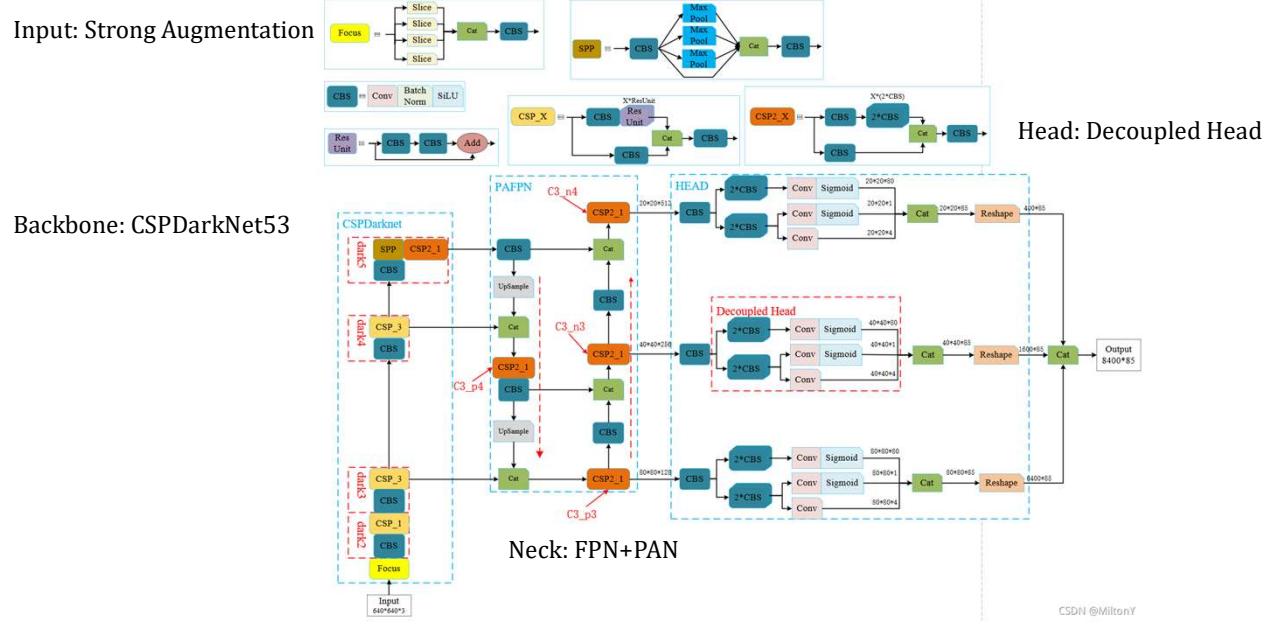


Innovation

- Anchor-Free
- Decoupled Detection Head
- Advanced Label Assigning Strategy(SimOTA)

X, 主要是引入了anchor-free 解耦头 SimOTA

YOLOX



Input: Strong Augmentation Mosaic MixUp, 用了之后发现预训练没啥效果就把预训练删了

Backbone: 没变化 v5啥样它就啥样

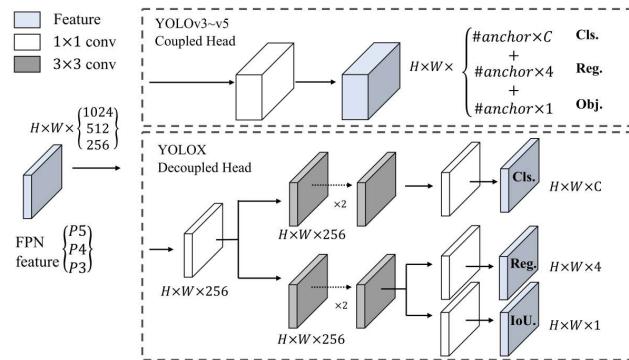
Neck: 没变化

Head: 引入了Decoupled Head, 并使用anchor-free思想和SimOTA正负样本分配策略进行损失函数的计算与优化。

我们最终会获得3个有效特征层，分别是输入进来的图片压缩三次80 80 256 四次4040512五次20 20 1024的结果。然后我们会用这3个结果进行加强特征提取网络的构建。

YOLOX

Decoupled Head



如果使用coupled head，输出channel将分类任务和回归任务放在一起，这2个任务存在冲突性。

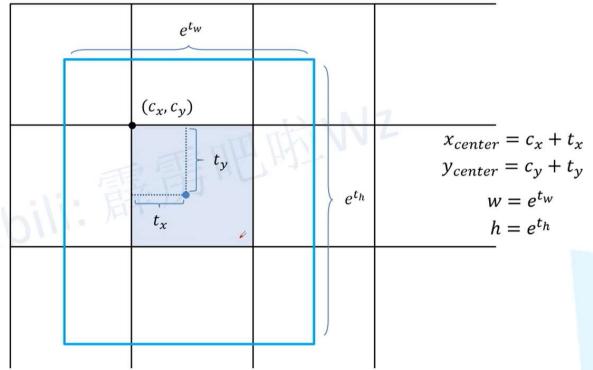
(论文中说有冲突性，但是没有理解为什么存在冲突，我考虑的是从损失函数角度存在冲突)

Concat前总共有三个分支：

1. **cls_output**: 主要对目标框的类别，预测分数。
2. **obj_output**: 主要判断目标框是前景还是背景
3. **reg_output**: 主要对目标框的坐标信息 (x, y, w, h) 进行预测

YOLOX

Anchor-Free



Xywh都是在特征图尺度上的，wh没乘参数，因为和anchor没关系

YOLOX

Loss Function & SimOTA

由于在网络的检测头中有**cls.分支**、**reg.分支**以及**IoU.分支**（其实是**obj.分支**），所以损失由 L_{cls} 、 L_{reg} 以及 L_{obj} 这三部分组成。其中 L_{cls} 和 L_{obj} 采用的都是二值交叉熵损失 (BCELoss) 而 L_{reg} 采用的是IoULoss。还要注意的是， L_{cls} 以及 L_{reg} 只计算正样本的损失，而 L_{obj} 既计算正样本也计算负样本的损失。

$$Loss = \frac{L_{cls} + \lambda L_{reg} + L_{obj}}{N_{pos}}$$

- L_{cls} 代表分类损失
- L_{reg} 代表定位损失
- L_{obj} 代表obj损失
- λ 代表定位损失的平衡系数，源码中设置是5.0
- N_{pos} 代表被分为正样的Anchor Point数

在SimOTA正负样本匹配过程中，**城市**对应的是**每个样本**（对应论文中的**anchor point**，其实就是grid网格中的每个cell），**牛奶生产基地**对应的是标注好的**GT Bbox**，那现在的**目标是怎样以最低的成本 (cost) 将GT分配给对应的样本**。根据论文中的公式1，cost的计算公式如下，其中 λ 为平衡系数，代码中设置的是3.0：

$$c_{ij} = L_{ij}^{cls} + \lambda L_{ij}^{reg}$$

最小化cost可以理解为让网络以最小的学习成本学习到有用的知识

因为正负样本匹配会影响到损失函数，SimOTA我写论文的时候看了一下，它是把标签分配问题看成一种最优传输问题，但是最优传输我没有理解，b站有很好的但是时长很长的课程，在上周的学习过程中优先级较高所以截至今我还没看完，等我看完了再做补充。

在计算损失时，yolox需要做标签分配，初筛和SimOTA精细化筛选。

anchor-free思想将感受野作为“anchor”信息

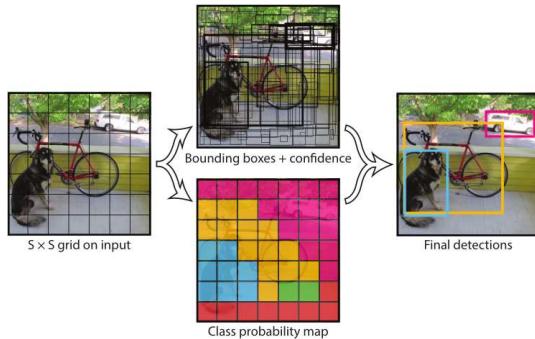
在样本初筛中，有两种方法来筛选正样本：

- 1.根据中心点判断：找到中心点落在ground truth框中的所有anchor box。
- 2.根据检测框判断：以ground truth中心点作为基准，绘制一个边长为5的正方形，找到中心点落地这个正方形中的所有anchor box。

Reference-YOLOX

- YOLOX
https://www.bilibili.com/video/BV1JW4y1k76c/?spm_id_from=333.337.search-card.all.click&vd_source=9ea40c1ac510f1e604555a3c8278ff94
- 最优传输简介
<https://zhuanlan.zhihu.com/p/458312488>

YOLO



Some Omissions:

- Activation function
- Training & Predicting
- Tricks
- Mathematical Derivation
- SimOTA

Next-Phase of Study:

- Code
- ...

一些遗漏

这个ppt诞生于上周工作的总结和理解，主要是基于结构、优化策略、损失函数、正负样本选择上做出的一些总结。

激活函数没怎么讲，很小的点，但是都有优化

Tricks，好多啊，如果后面有用到再去深入了解吧，后面可以单独做一期

数学推导，其实我一直有个疑惑，就是网络的改造中会普遍使用分支再融合的思想，有些是两支分别处理，有些是直接利用浅层特征图去获取更多信息，v4 v5CSP，分两支，分别卷积，SPP，分四支，maxpool，融合。残差网络，分两支，会解决梯度消失的问题。ResNet的延伸，通过在 3×3 卷积层前后引入 1×1 卷积层分别来减少和增加维度，让 3×3 卷积层成为“瓶颈”，减少参数和浮点运算，使得模型更加“轻量”。为啥啊，我理解从

还有就是SimOTA，没看懂，最优函数理论