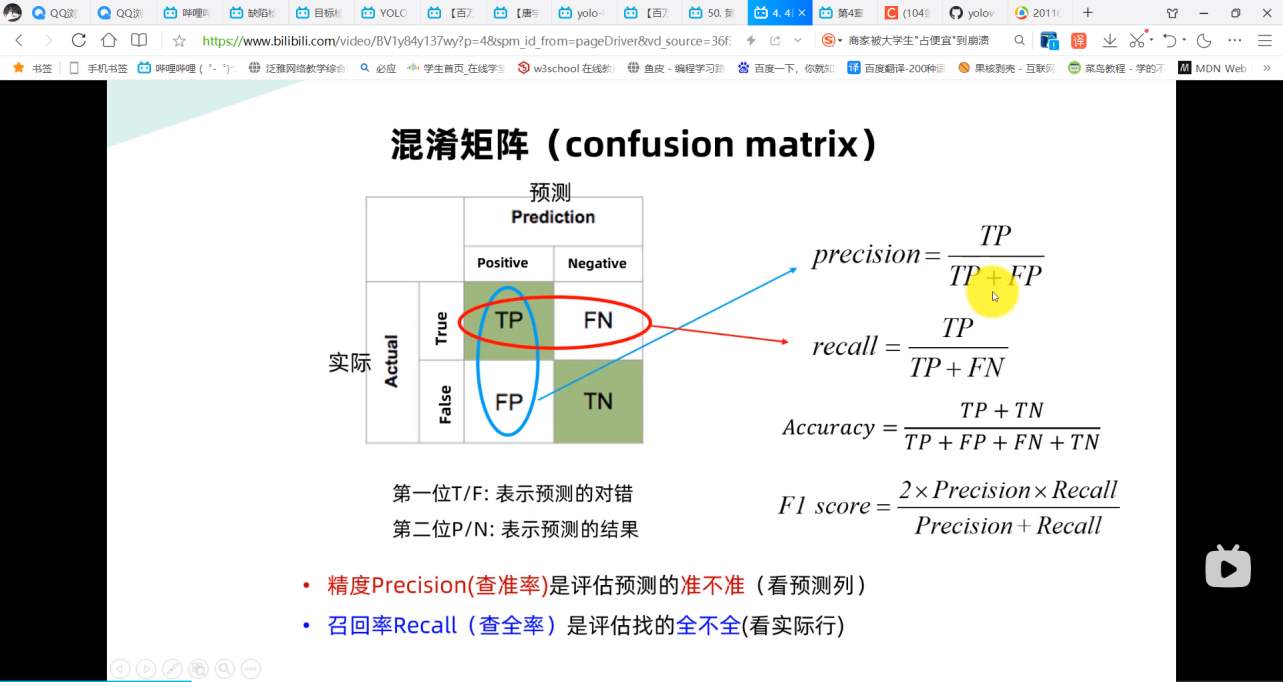
# 性能指标

## 检测精度

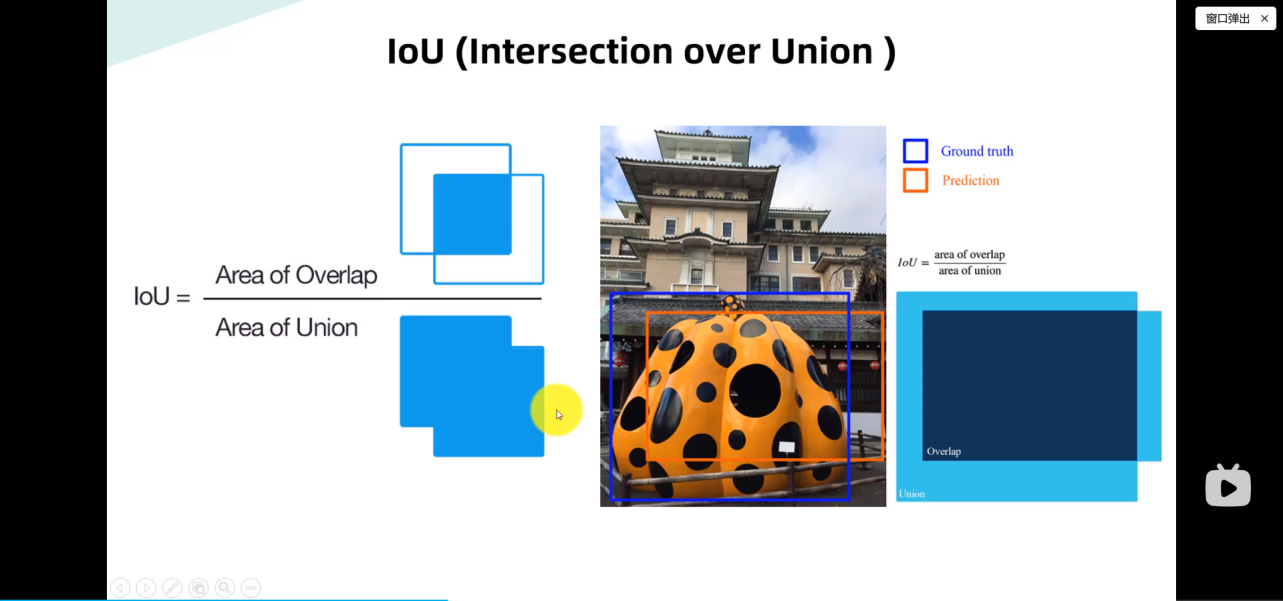
### 混淆矩阵

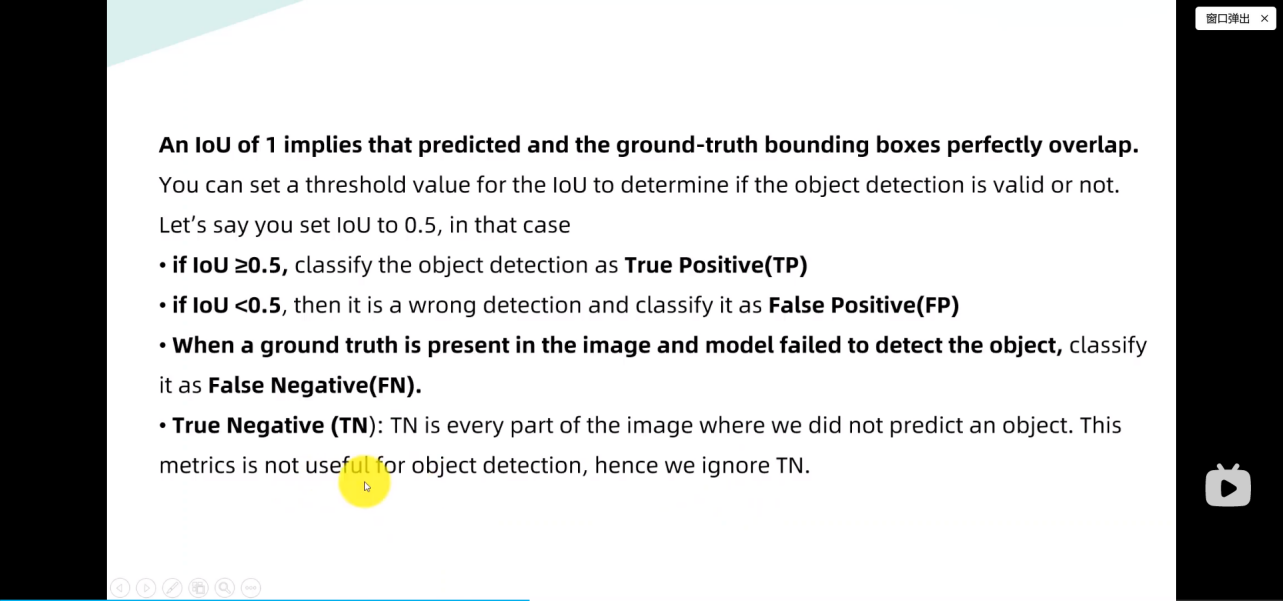


precision精度：所有预测为正样本中有多少真实确实是正样本

recall召回率：所有实际为正样本的有多少被检测出来了

### IoU

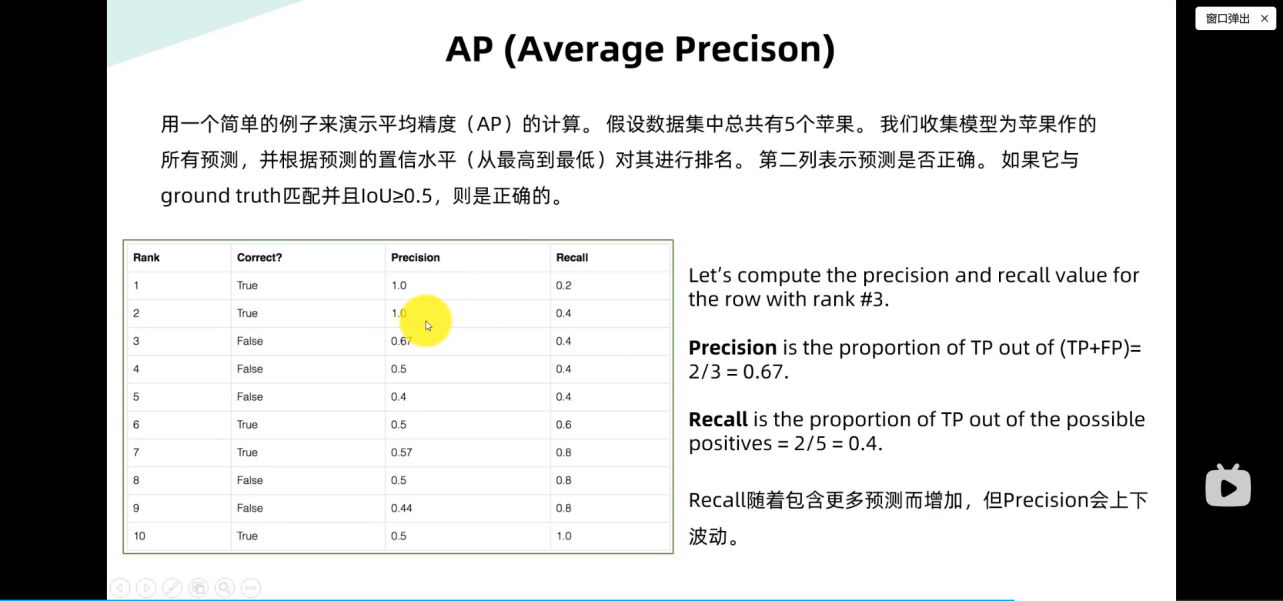




### 置信度

框中是物体的可能性

### AP（平均精度）

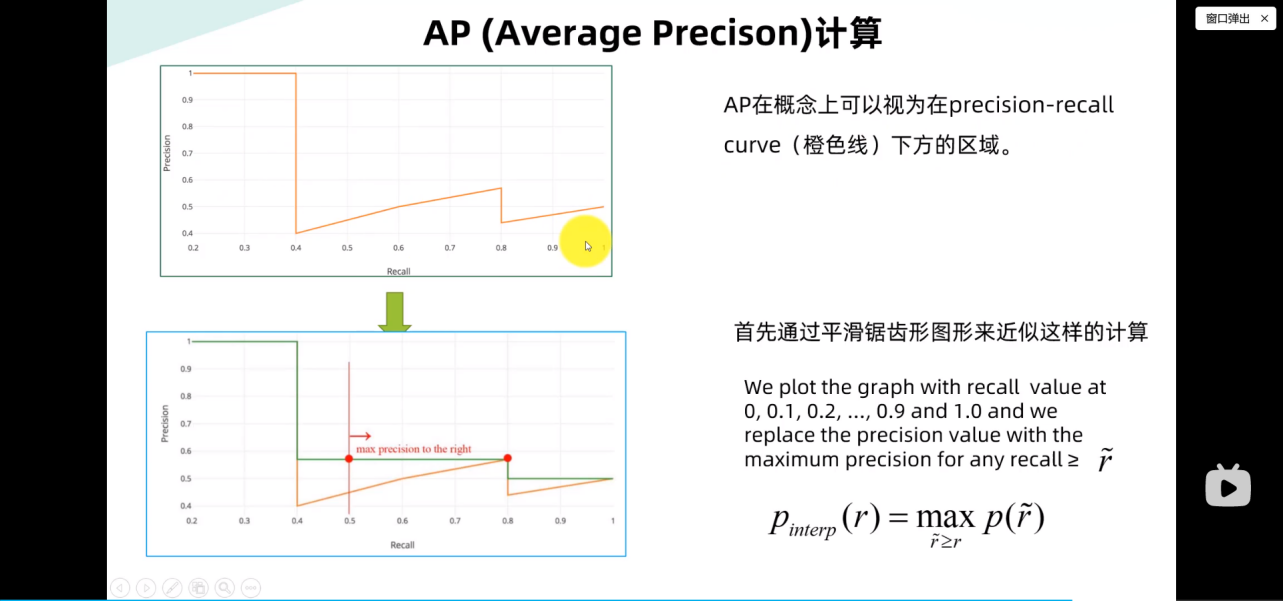


AP将精度和召回率两个指标综合起来

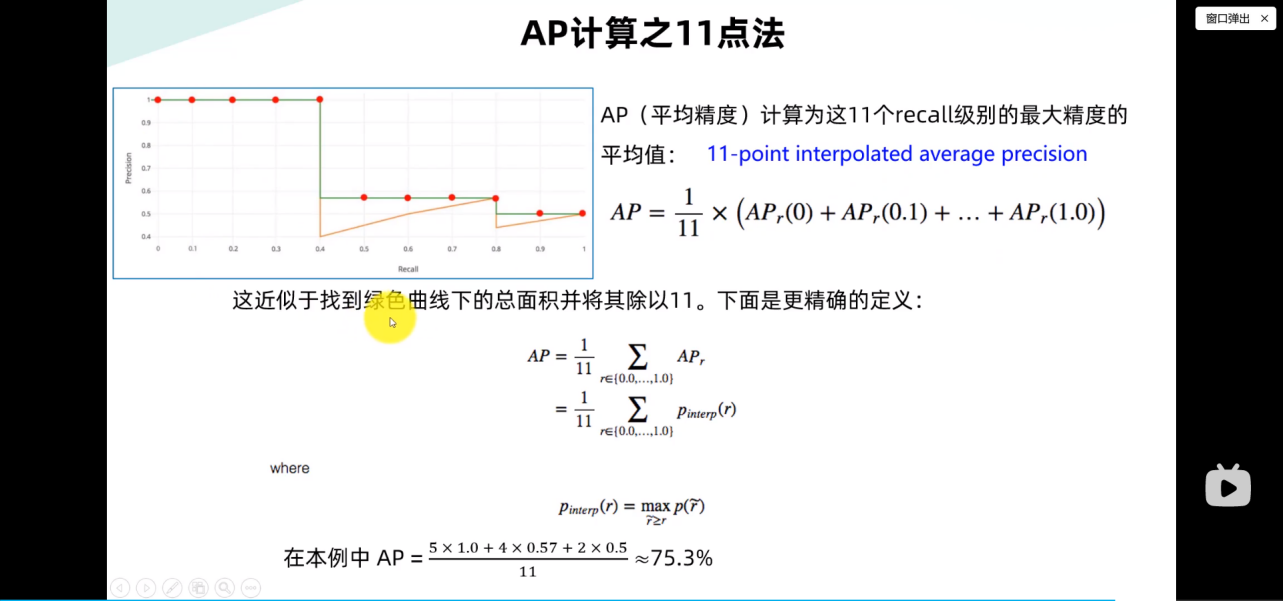
图中预测有十个正样本，而实际只有五个正样本

recall会随着预测出的样本数量而增加，直到把所有正样本全部预测出来，recall为1，但precision会上下波动

从而我们可以画出precision和recall的曲线

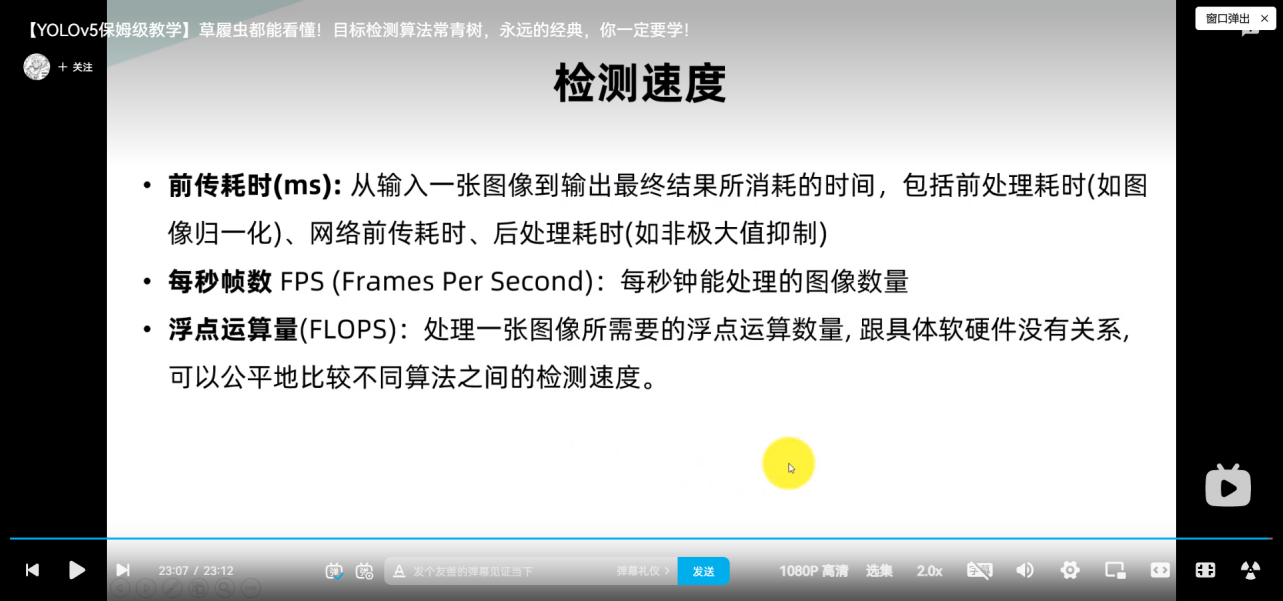


把曲线拉直



从0到1每隔0.1画一个点，找到11个点，求这11个precision的平均值为AP，mAP是所有类别的AP的平均值

## 检测速度



# Yolov1算法原理

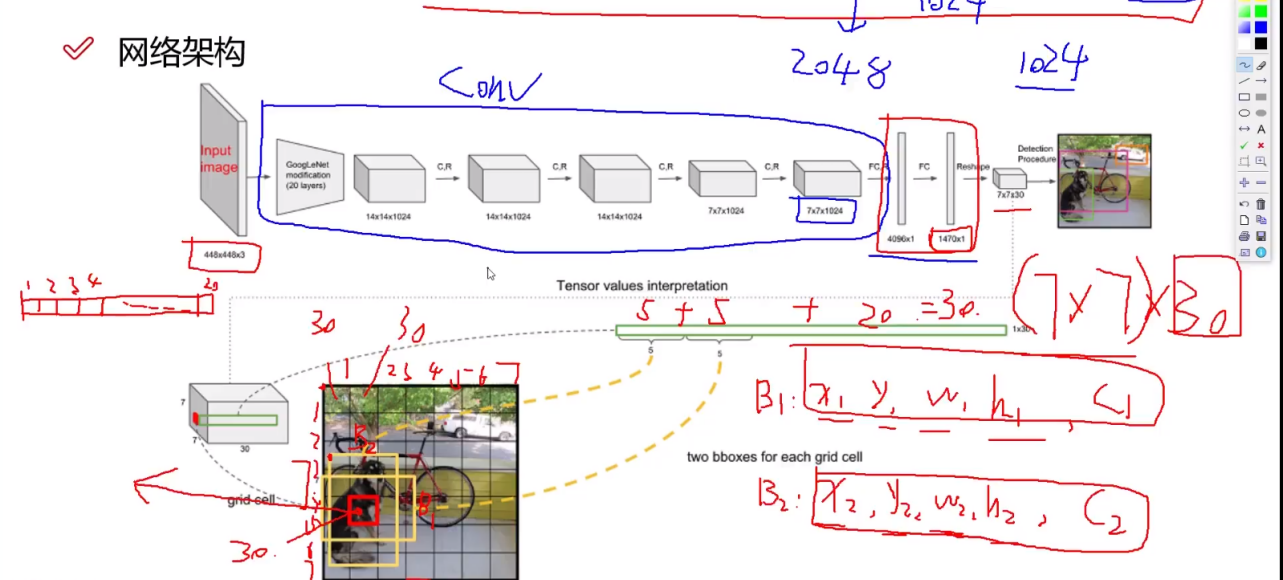
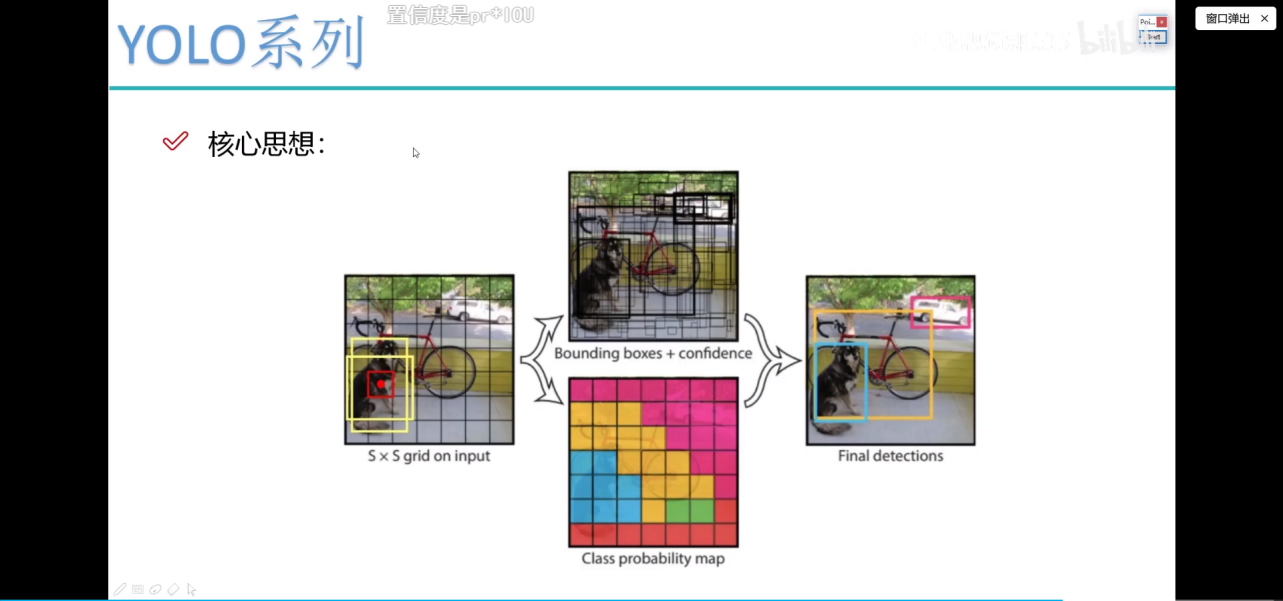
## 网络架构

输入图片-》卷积层-》全连接层-》结果

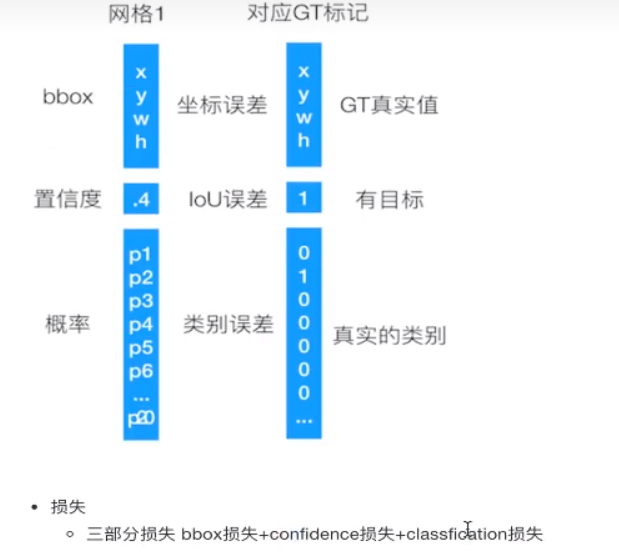
最终要得到的结果：将一张图片分为7\*7个单元格，每一个单元格有30个值：两个bbox（候选框），每个bbox有4个坐标值，和1个置信度，和属于每个类别的概率（假设一共有20个类别，就有20个概率值）=》7×7×30

将置信度大于阈值的bbox保留，并在两个候选框中选择IoU大的候选框的尺寸进行微调

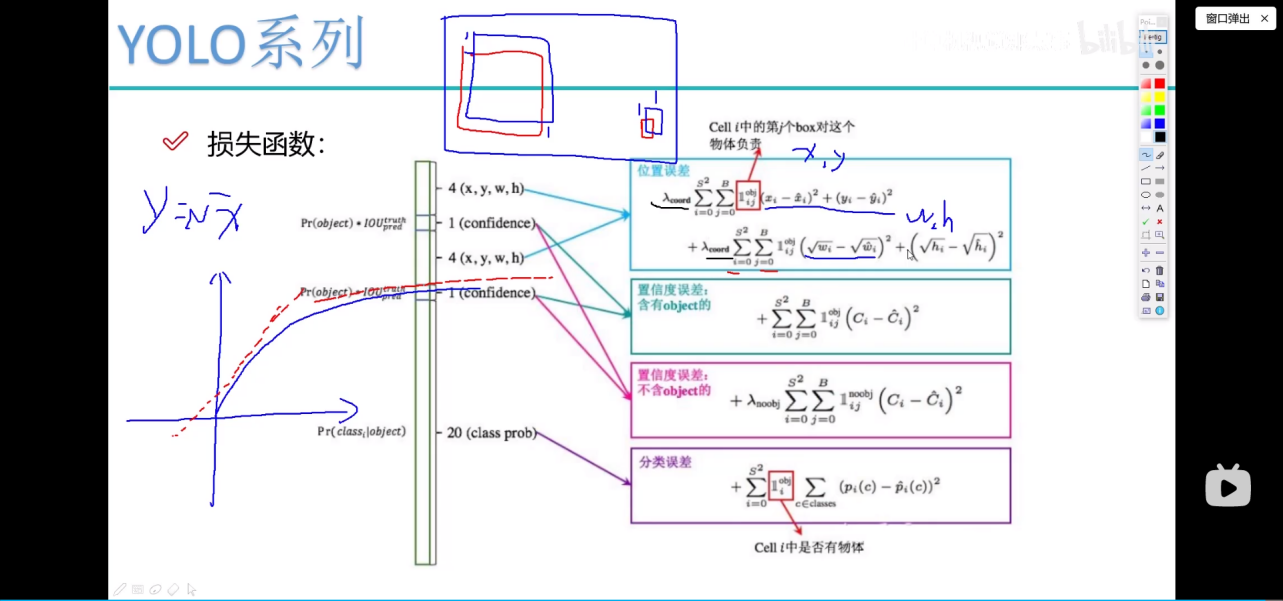
最终输出一个置信度高的bbox，和一个概率大的类别



## 损失函数



1.位置误差：



2.置信度误差：

分为前景、背景两种，前景的置信度标签值设为1，背景的置信度标签值设为0

置信度大于阈值框的就视为前景，用来算前景的置信度误差；置信度小于阈值框的就视为背景，用来算背景的置信度误差

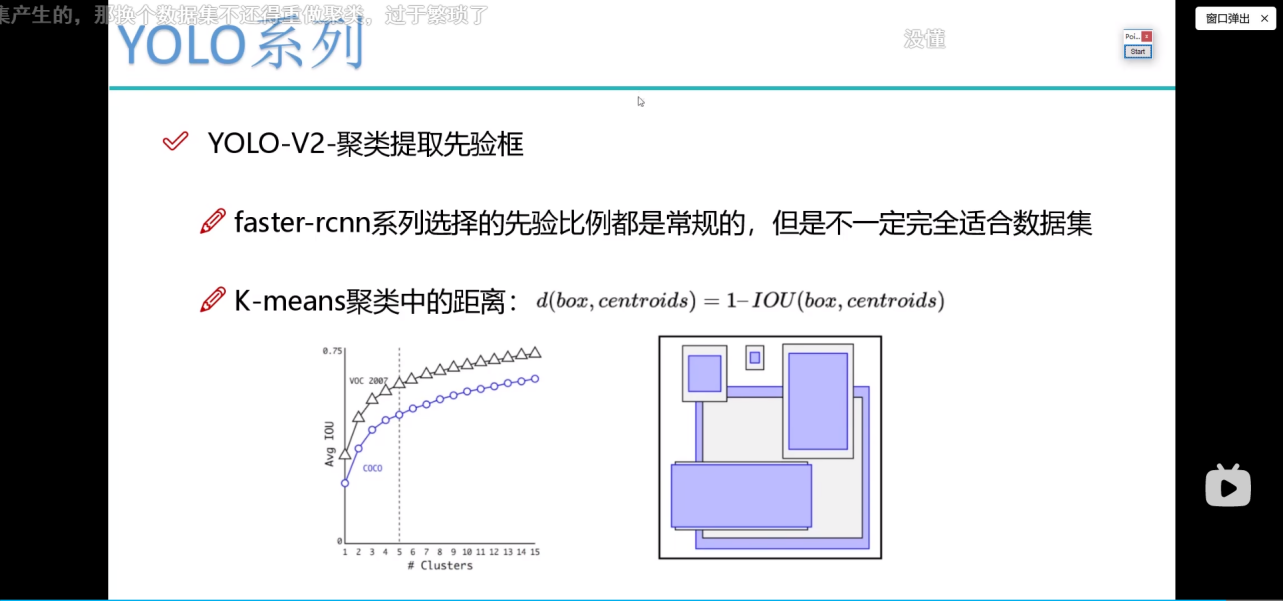
3.分类误差

预测是狗的概率和真实是狗的概率算一个交叉熵

最终的损失函数将这四块全加在一起

# v2

## 聚类提取先验框



将真实的框根据尺寸用k-means聚类分成5堆，则每个单元格产生5个候选框

计算位置误差的方式也改变了

## 感受野的概念

感受野：最后一个特征图的一个特征点相当于原始输入的多大的区域。

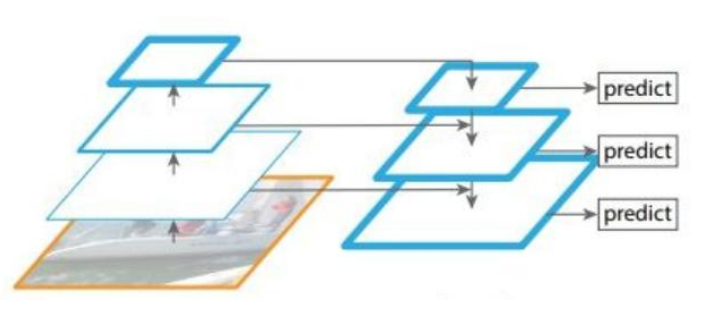
随着卷积层的增多，特征图越来越小，特征图中一个特征点的感受野越来越大，相当于能看到的原始图像的范围越大，更适合检测大目标。所以感受野小、中、大的特征图，分别适合用来检测小目标、中目标和大目标。

# V3

## 多Scale与特征融合

随着卷积层的增多，特征图越来越小，特征图中一个特征点的感受野越来越大，相当于能看到的原始图像的范围越大，更适合检测大目标。所以感受野小、中、大的特征图，分别适合用来检测小目标、中目标和大目标。三种特征图的候选框的尺寸也依次增大，每种特征图共三个候选框。

但是仅用感受野小的和中的两个特征图来对小、中目标进行检测效果可能并不好，因为他的训练的语义信息不如最后的特征图丰富。解决办法是：一方面感受野大的特征图通过上采样的方法与感受野小的特征图进行特征融合，像老者给年轻人传授经验



## 残差连接

当网络层数越多时，训练效果不一定会更好，反而会产生过拟合。但是并不是后面加的所有层都不好。借鉴resnet的思想，使用残差连接的方法：给出两条路，一条路正常往下继续走卷积层，另一条路则跳过不走下面卷积层，然后比较两个结果，来决定是否使用下面的卷积层



# v5

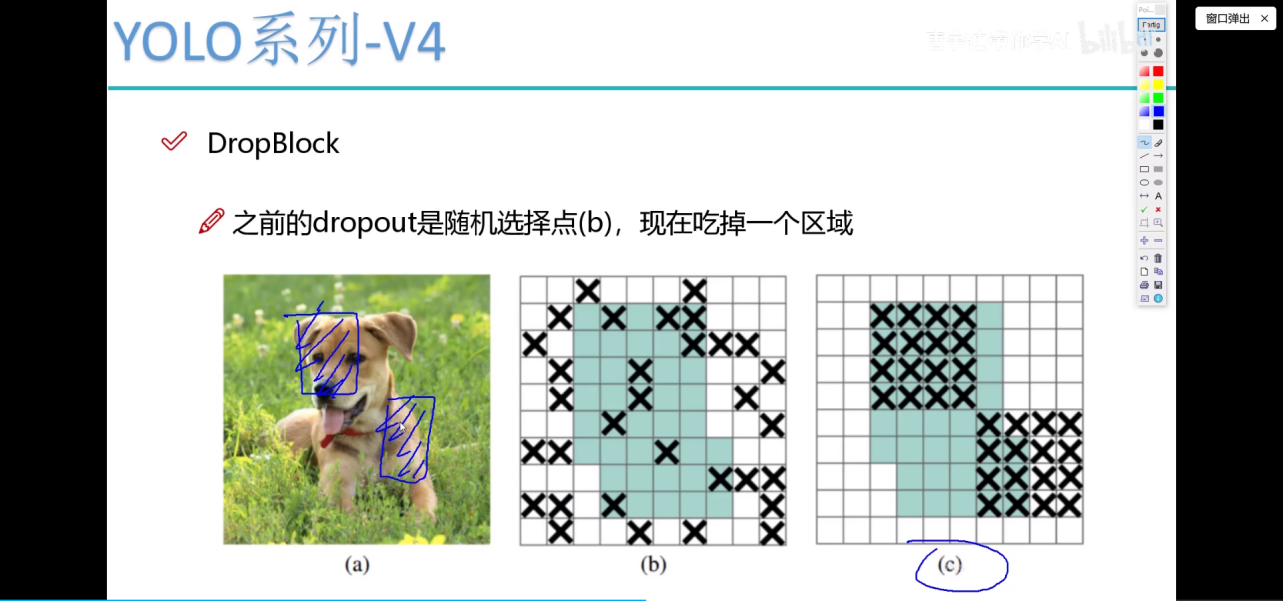
# 数据方面的改进

## 马赛克数据增强



参考CutMix，使用马赛克数据增强，将四张图像拼接成一张进行训练，间接增加了batchsize的大小

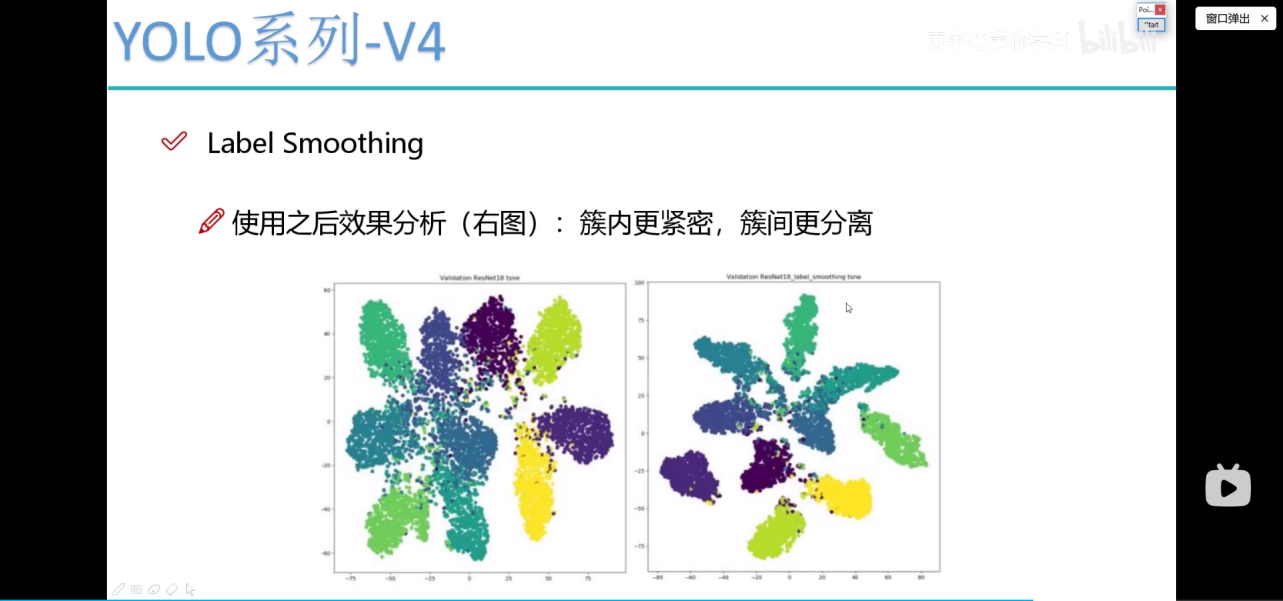
## DropBlock



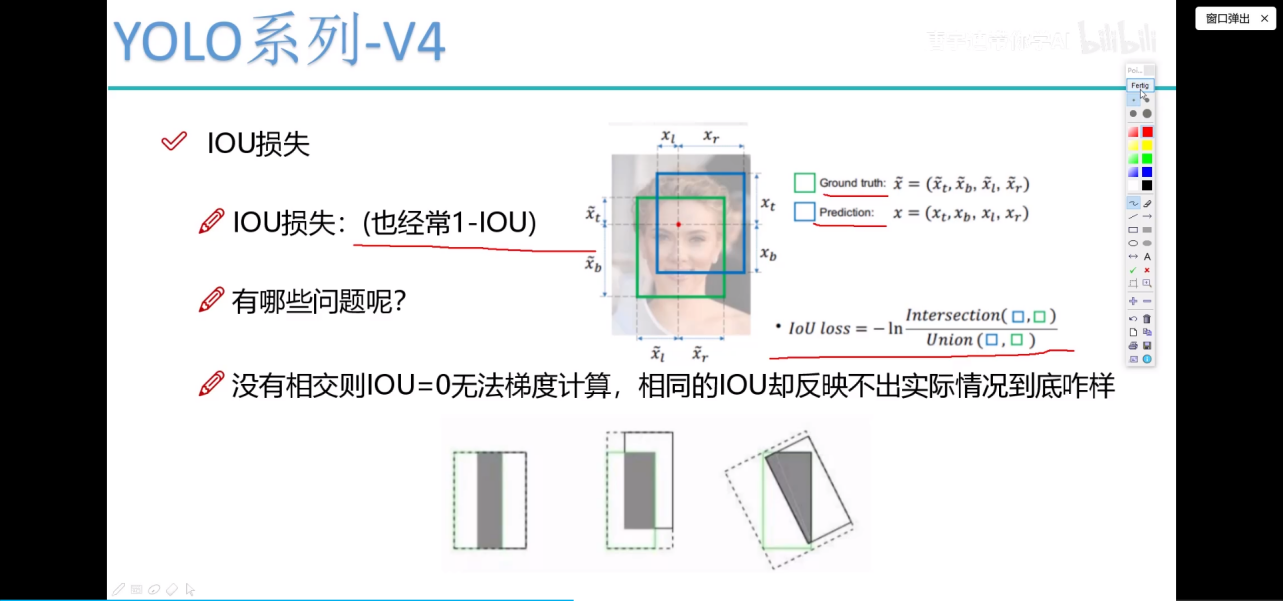
抛弃传统的随机失活方法，采用DropBlock，

## 标签平滑

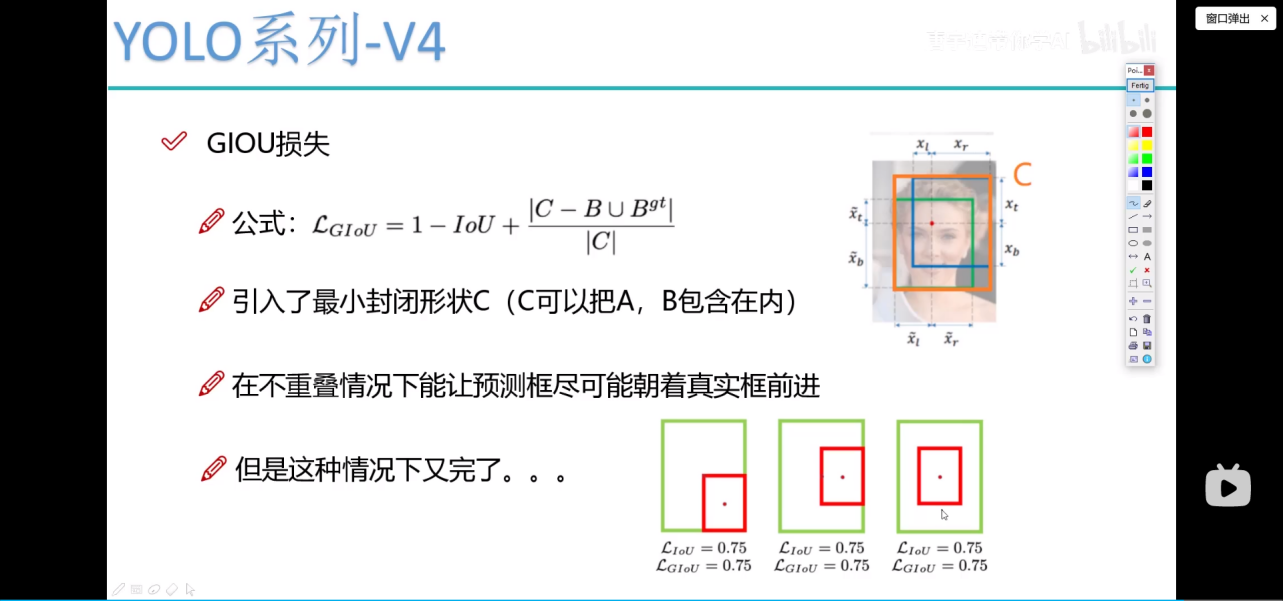




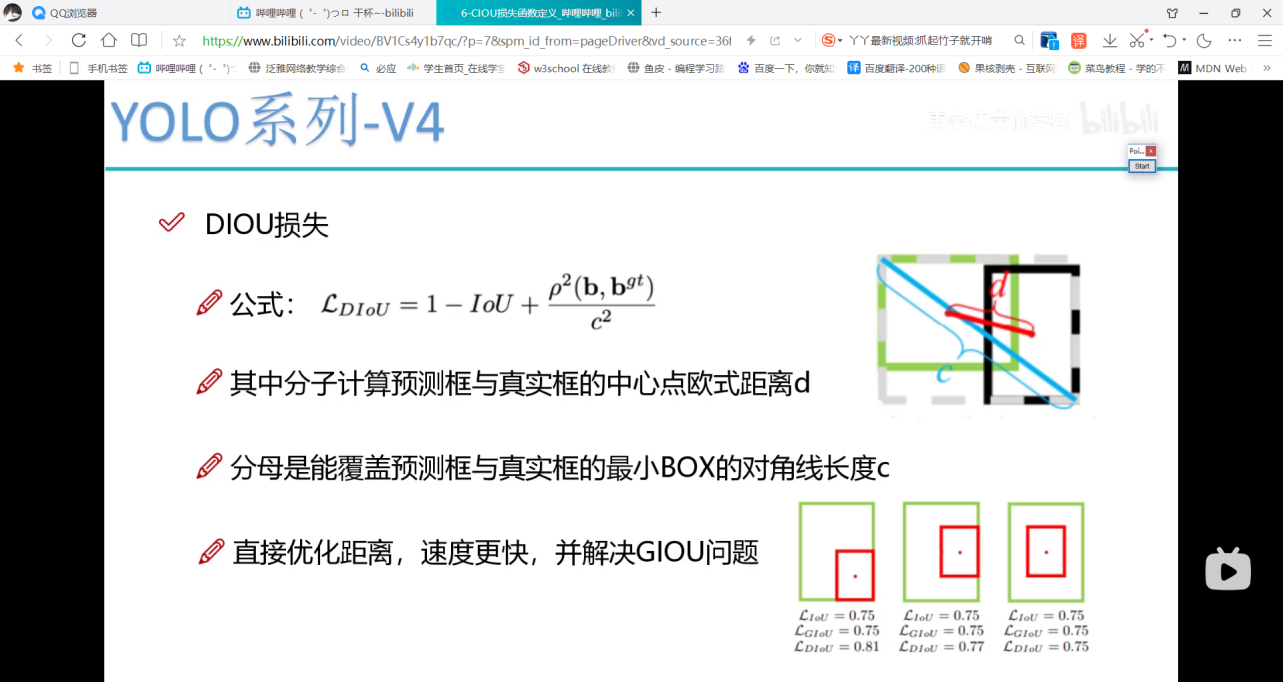
## CIOU损失



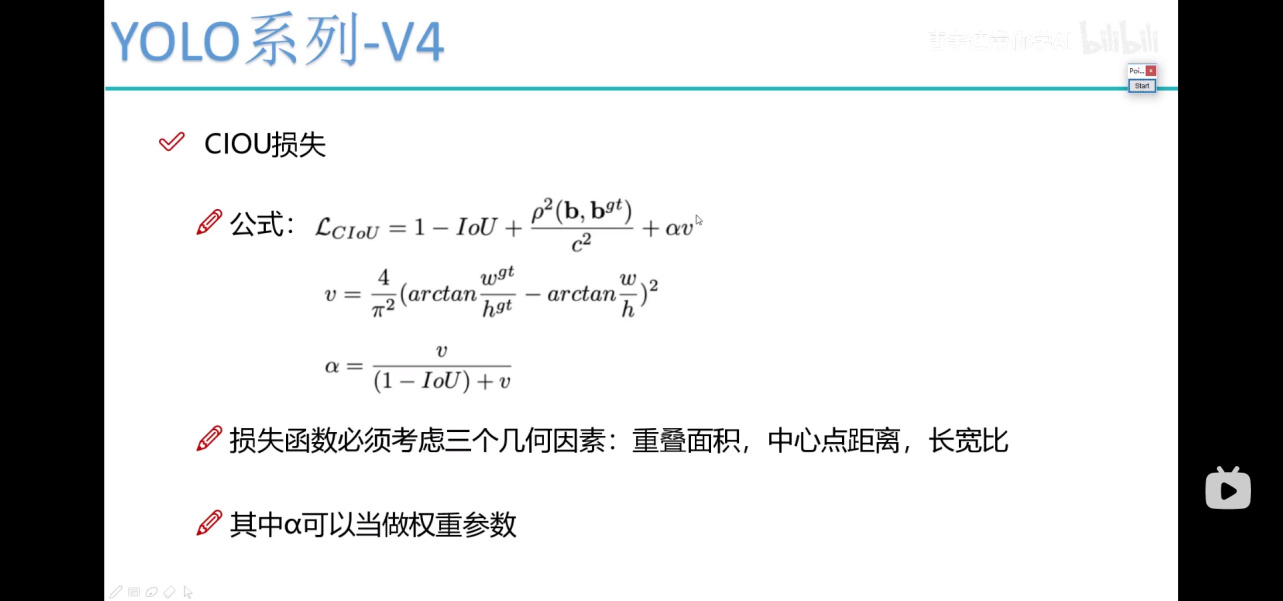
原来的IOU损失函数使用1-IOU来计算，不同的情况可能产生相同的IOU，这就可能导致相同的IOU却反应不出实际情况到底是什么样的。而且当预测框和真实框没有相交，IOU=0时无法计算梯度



采用GIOU损失函数，在预测框a和真实框b外部引入了一个封闭形状c，将他俩包含在内，在计算损失函数时，加上a和b取并集后在c中剩余的部分



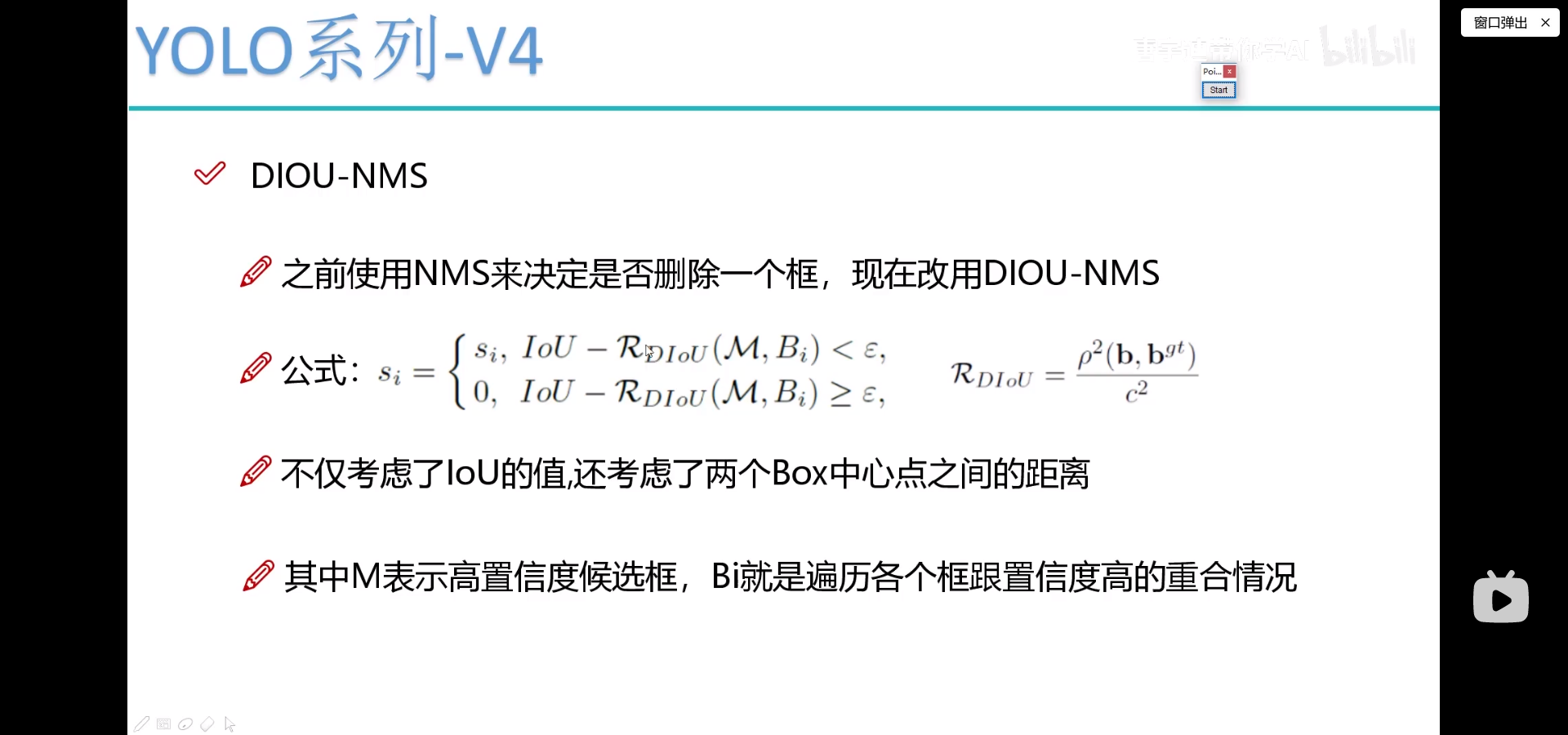
采用DIOU损失函数，在预测框a和真实框b外部引入了一个封闭形状c，将他俩包含在内，在计算损失函数时，加上“预测框和真实框中心点的欧氏距离/c框的对角线长度的平方”



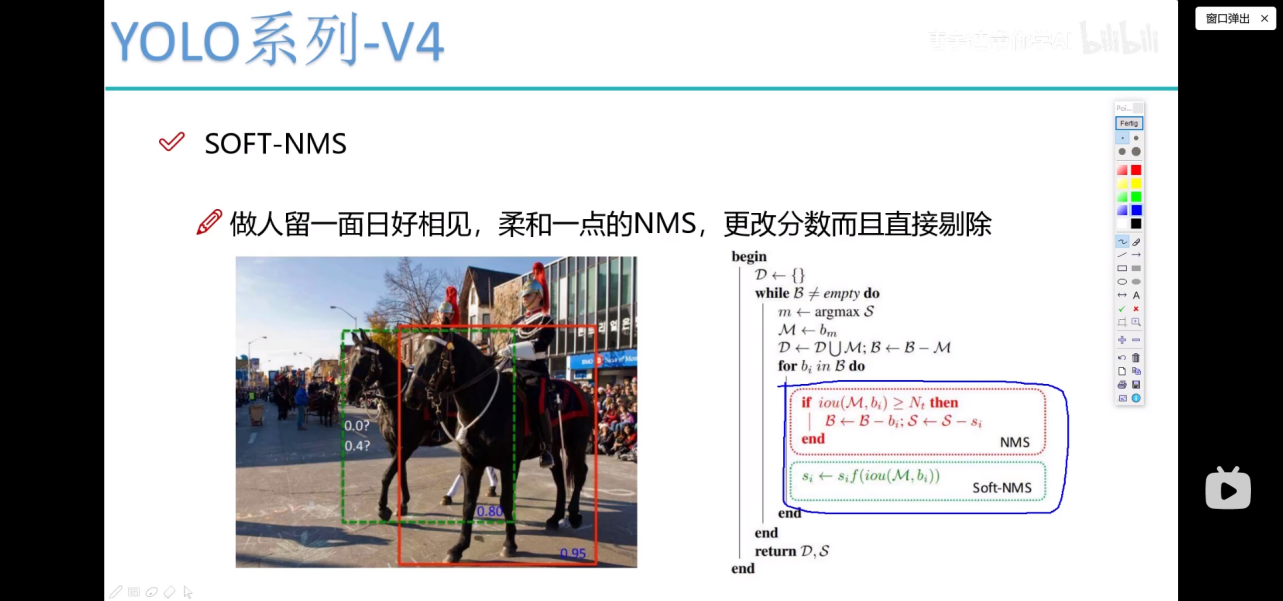
Yolov5中考虑到计算损失函数的三个几何因素：重叠面积、中心点举例、长宽比，采用CIOU损失函数，解决了上述问题。具体做法是：在预测框a和真实框b外部引入了一个封闭形状c，将他俩包含在内，在计算损失函数时，加上“预测框和真实框中心点的欧氏距离/c框的对角线长度的平方”，再加上预测框长宽比与真实框的长宽比的差值

## DIOU-NMS

之前使用NMS非极大抑制来决定是否删除一个框，现在根据DIOU考虑了一项中心点距离



## SOFT-NMS



假设有两个高度重合的预测框a和b，b框的IOU更大，传统的NMS会只保留IOU更大的那个b框。但是可能的情况是两个物体离得很近，不应该去除a框。Yolov5采用SOFT-NMS，不直接剔除a框，而是降低a框的置信度，再去判断a框的置信度是否达到了阈值，从而决定是否保留a框，它能够增大结果的召回率

# 数据预处理

train.py create\_dataloader-》

utils/dataloaders.py LoadImagesAndLabels类 -init-加载训练数据和标签 -getitem-数据增强 load\_mosaic马赛克数据增强

# 模型可视化工具

onnx