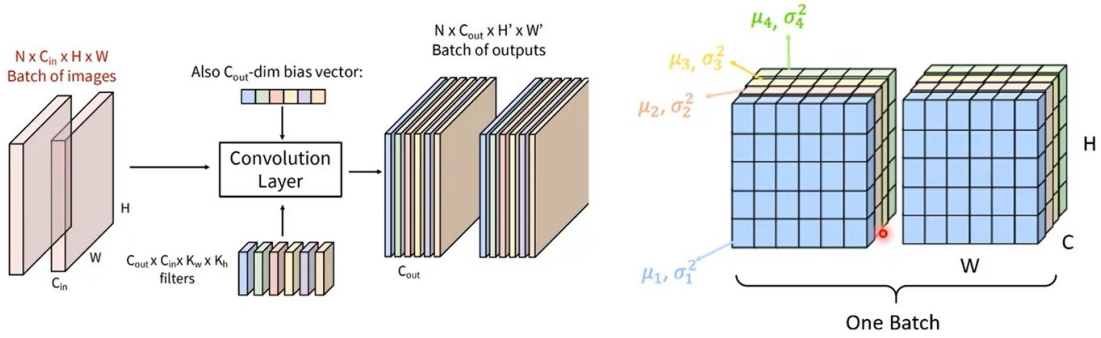
YOLO v2

1. Batch Normalization 批量归一化

YOLO v1 使用 dropout Layer 来避免过拟合，增加了训练时间，影响收敛速度。YOLO v2 提出了 Batch Normalization，可以改善上述问题。

概念：
在训练过程中，对当前 batch 的输入经过每一卷积层（激活函数之前）输出的特征图进行标准化处理。



步骤：
1. 当前 batch 中，对于每个 channel，计算特征的均值μB 和方差σ^2B。
2. 对每个特征进行归一化，使其具有零均值和单位方差。 $\hat{x}^{(i)} = \frac{x^{(i)} - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$ （分母为什么要引入这两个参数？是为了防止分母为 0，特征值变得无意义）
3. 之后，通常会引入两个可学习的参数（缩放因子γ和偏移因子β），允许模型恢复表达能力。 $y^{(i)} = \gamma \hat{x}^{(i)} + \beta$

缩放因子γ：调整标准化后特征的尺度，放大或缩小标准化后的特征
偏移因子β：调整标准化后特征的偏移（位置）

好处：
1. 稳定的激活分布使得模型更容易学习到有效的特征，从而提升检测准确率
2. 无需 Dropout ，仍保持泛化能力
3. 标准化后的梯度更稳定，可提高学习率，加快收敛速度

2. Higer Resolution Classifier 高分辨率分类器
指的是 **pre-training** 阶段。分两步：先采用 224x224 的 ImageNet 图像数据集进行约 160 个 epoch 的训练，快速训练学习基本特征；然后再将输入图像的尺寸更改为 448x448 再训练 10 个 epoch，学习更复杂的特征

而 YOLO v1 在 pre-training 时采用 224×224, 在 fine tuning 使用 YOLO 模型做目标检测的时候才将输入变成 448×448

好处：

1. 与 YOLO v1 相比，后 10 个 epoch 的 input size 改为 448x448，有利于模型学到更复杂的特征
2. 在下一步 fine tuning 时可以快速适用高分辨率的输入

Q：既然 448x448 有利于模型学到更复杂的特征，为什么不全部 448？

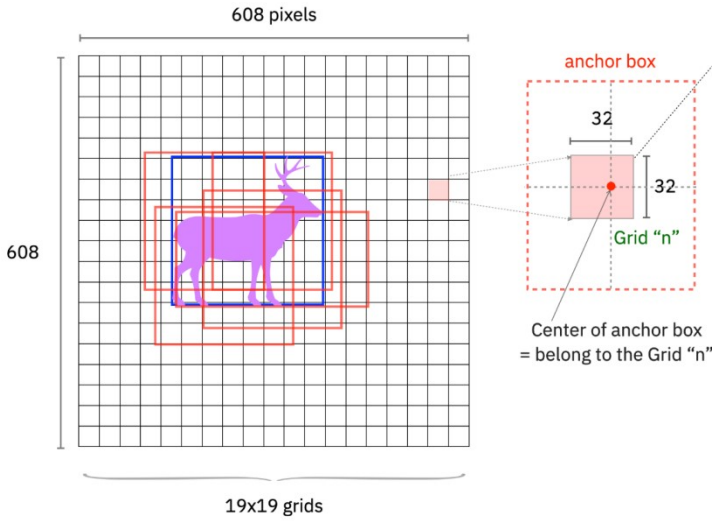
因为 input size 变大了，batch size 就会缩小，计算消耗太大，收敛太慢。
pre-training 阶段的训练是从 0-1，需要消耗很多计算资源，用较小的尺寸可以让训练时间缩小，收敛速度加快。而且这个阶段学到的是基础特征，不需要那么详细的特征。
fine tuning 用 416 * 416，因为这个阶段需要学位置信息，需要更详尽的特征）

3. Convolutional With Anchor Boxes

引入 **Anchor Box**, 输出 Anchor Boxes 的偏移量和置信度, 而不是直接输出 bounding box 的坐标和置信度

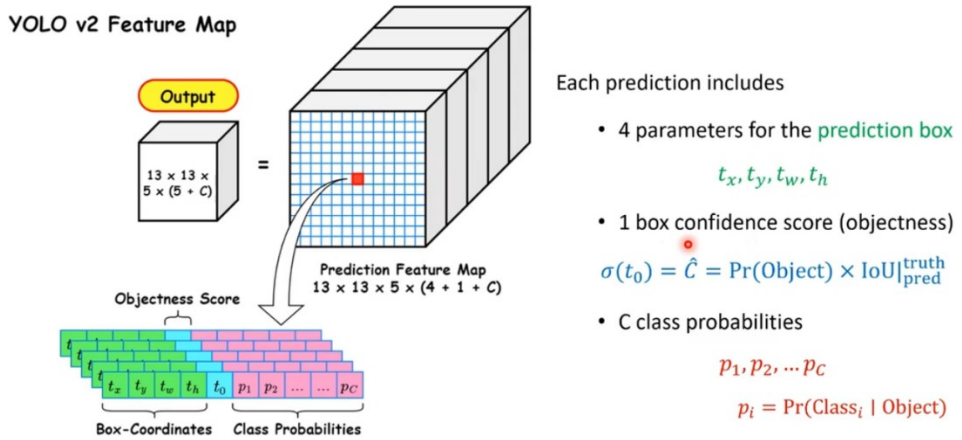
Anchor Box 概念：

以特征图上的 grid cell **在原始图像**上的映射点为中心，生成的不同大小和形状的边框，这样的边框叫 anchor box



引入 anchor box，input 和 output 的变化：

Input: 416 * 416 * 3 的图像
Output: 13 * 13 * 125 的张量



Q: 13 * 13 * 125 是什么意思?

下采样总步长为 32，416 / 32 = 13，最终得到的特征图尺寸为 13×13

125 = 5 * (4 + 1 + 20)

每个 grid cell 有 5 个 anchor box，每个 anchor box 输出 25 个值：4 个位置信息（x, y, w, h）+ 1 个目标置信度 + 20 个分类的概率值

YOLO v1:　　30 = 2 * (4 + 1) + 20

YOLO v2:　　125 = 5 * (4 + 1 + 20)

Q: 括号为什么移出来了?

之前是每个网格对应（2 个 bbox，5 个数值，20 个类）

现在是每个网格对应 5 个 anchor box，每个 anchor box 对应（5 个数值，20 个类）

好处：

1. YOLO v1 是一个 grid cell 检测一个 class，v2 是一个 anchor box 检测一个 class，不容易漏检。recall increase from 81% to 88%
2. 输出 Anchor Boxes 的偏移量比输出 bbox 要容易，有先验知识，加快模型收敛速度

Input 为什么要从 448 * 448 变成 416 * 416?

为了使网路输出的特征图有奇数大小的宽和高，这样每个特征图只有一个中心单元格

物体倾向出现在照片中央，物体中心点往往落入图片中心位置，因此使用这个中心单元格去获得这些物体的边界框，相对容易

4. Dimension Clusters

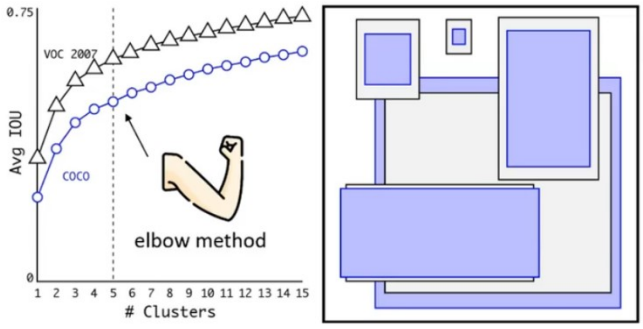
在 Faster R-CNN 中，anchor box 的尺寸是人工选择的，具有主观性

维度聚类的目的是：自动找到更好的 anchor box 尺寸

使用 k-means 聚类算法

Distance metric: $d(\text{box}, \text{centroid}) = 1 - \text{IOU}(\text{box}, \text{centroid})$

（box:其他框，centroid：聚类中心框）

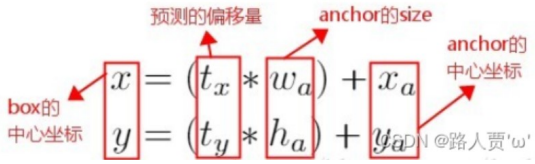


为什么 k 设置为 5? 权衡了模型复杂度和高召回率，k 越多，模型越复杂
聚类中心点与手工挑选的锚框有明显不同。多为瘦高型。

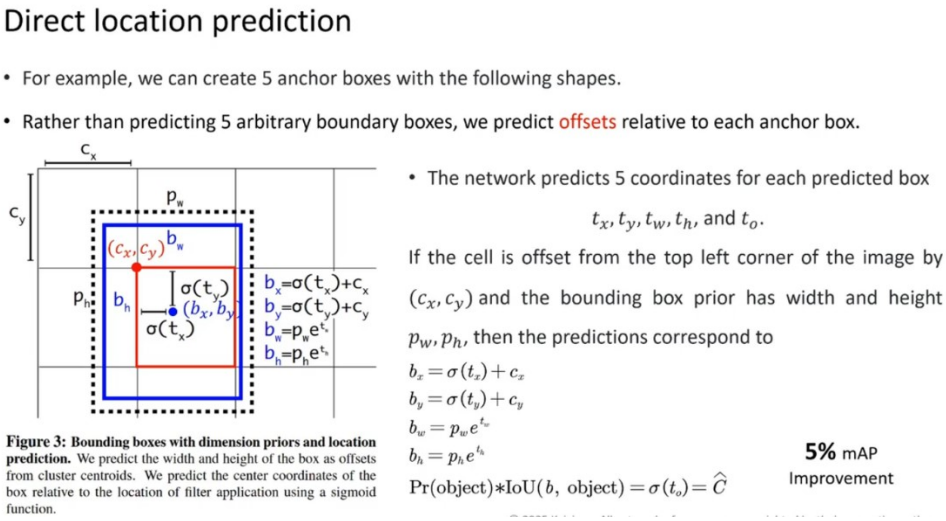
5. Direct location prediction

在 RPN 中，输出的 x y w h 是相对于 Anchor Box 的坐标/高宽的偏移，这个偏移值没有约束，使得计算出来的检测框可以在图像上的任何位置，模型需要长时间才能得稳定地输出出合理的偏移量

计算公式： 预测框中心坐标= 输出的偏移量×Anchor宽高+Anchor中心坐标



而在 YOLO v2 中对 x y 做了约束。这样 anchor box 只在附近偏移，不会偏移到其他 grid cell 负责的范围

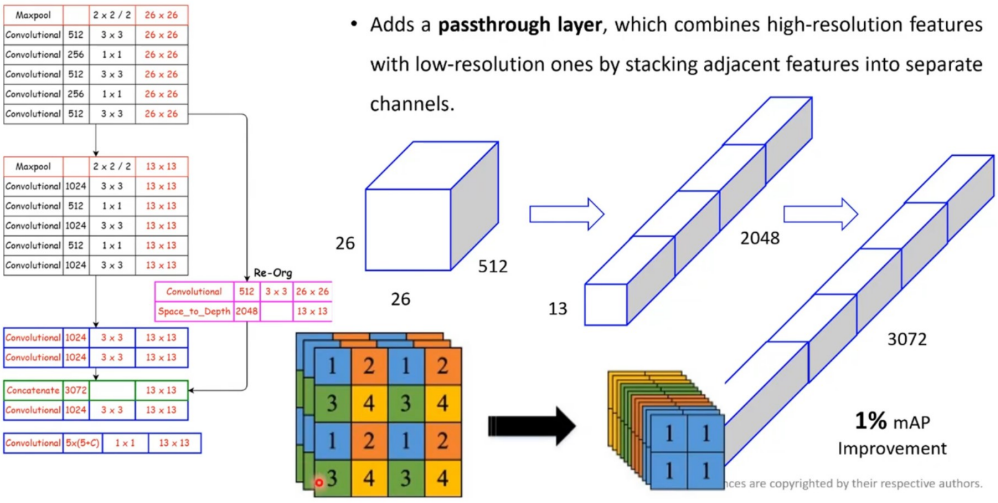


Q: 为什么不对 w h 做约束?
因为目标可大可小，无法进行约束

Q: 太小的目标可能漏检，太大的目标不会漏检，为什么?
小目标的特征在多次 pooling 中可能丢失

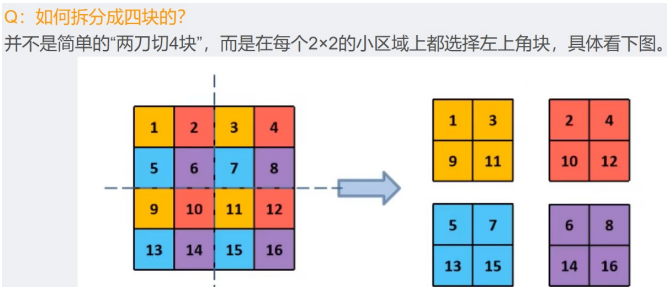
Q: 为什么可以检测到不完整的物体?
因为 w h 没有做约束。实际中，模型检测出来的框可能超出画面，此时坐标为负

6. Fine-Grained Features 细粒度特征
将浅层的细粒度特征（像素特征）和深层的粗粒度特征（语义特征）进行融合。



具体过程:
① 提取 Darknet-19 最后一个 Maxpooling 层的输入（26×26×512），分为两路，一路继续向下 pooling 和 conv，得到（13×13×1024）
② 另一路经过 **Passthrough Layer** 的处理，得到（13x13x2048）
③ 两者进行 concatenate，output 变成（13×13×3072）

Q: Passthrough Layer 如何将（26×26×512）变成（13x13×1024）？这样子会丢失位置信息吗?
抽取每个 2x2 的局部区域组成新的 channel，即大小降低 4 倍，channel 增加 4 倍



不会丢失位置信息，因为抽取规则是固定的，能够恢复之前的排列

7. Multi-Scale Training
注意是在 fine-tuning 阶段
每 10 批训练后随机选择一个新的图像尺寸大小，由于模型下采样了 32 倍，在 32 的倍数 {320,352，…，608}中选择图像维度。将网络输入调整到那个维度，并继续训练。

作用：网络可以更好地检测不同尺寸的图片，意味着同一个网络可以进行不同分辨率的检测任务，在输入 size 较大时，训练速度较慢，在输入 size 较小时，训练速度较快，而 multi-scale training 又可以提高准确率，因此算是准确率和速度都取得一个不错的平衡。

Q: 为什么可以动态调整输入大小?
只有卷积层，没有 FC 层，对输入图像尺寸没有限制。但有范围限制[320, 608]

Q: Input 时尺寸不同，如何 Resize?
等比例缩放，padding 补 0，或补灰。为什么不直接拉伸？这样会改变目标的比例

8. Training and Inferring
Training for classification: 使用 ImageNet 训练分类网络 Darknet19。前 160 epochs 使用 224x224 的 image size，后 10 epochs 微调了网络，使用 448x448 的 image size

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			

Q: 这里的 fine tune 微调，指的是什么?
原文为：After our initial training on images at 224x224, we **fine tune our network** at a larger size, 448. For this fine tuning we train with the above parameters but for only 10 epochs and starting at a learning rate of 10^{-3}.
网络无 FC 层，且用的是 Global Avgpool，输入尺寸无限制，无需调整网络结构。实际上，指的是使用之前得到的参数继续进行训练，得到新的参数。这里的 network 不是指网络结构，而是一个更大的概念，包含了网络的参数（如权重）。

Training for detection: 以 Darknet19 为 backbone，移除最后一层 Conv. Layer，添加 3 层 3x3x1024 的 Conv. Layers 以及 1 层 Passthrough Layer, 最后添加 1 层 1x1x125 的 Conv. Layer。使用 448x448 的目标检测数据集训练。

Infer 推理阶段：1. Resize image to 416x416. 2. Run CNN model. 3. Run NMS.

Q: YOLO v2 为什么要去掉全连接层?

1. 全连接层的作用是整合所有特征，方便进行分类或回归。引入了 anchor box，加入了先验知识，模型要学的内容减少了，因此不需要加这一层，也能得到准确的结果
2. 全连接层参数量很大，去掉后可以加快训练速度

Q: YOLO v1 为什么比 Fast(er) R-CNN 更快?
网络结构不同，前者是 one stage，后者是 two stage

Fast(er) R-CNN 在检测一个图像时，首先会生成大约 2000 个 region proposals，每个区域都需要进行一次 CNN，然后再对这些区域逐个分类和回归。

而 YOLO v1 则是“只看一次”图片，直接输出全图上所有目标的边界框和类别，无需生成大量候选区域，也不用对每个区域单独进行 CNN 运算。

Q: YOLO v2 为什么比 YOLO v1 更快?
引入 anchor box，在先验知识的基础上进行学习
取消了 FC 层
Batch Normalization 解决梯度消失和爆炸问题，获得更好的收敛速度

References
1506.02640
【YOLO 系列】YOLOv1 论文超详细解读（翻译 + 学习笔记） yolo 论文-CSDN 博客
1612.08242
YOLO 超详细入门 02 v2 （含代码及原文） yolov2 代码-CSDN 博客
【YOLO 系列】YOLOv2 论文超详细解读（翻译 + 学习笔记）-CSDN 博客
三万字硬核详解：yolov1、yolov2、yolov3、yolov4、yolov5、yolov7-CSDN 博客
https://medium.com/ching-i/yolo%E6%BC%94%E9%80%B2-1-33220ebc1d09