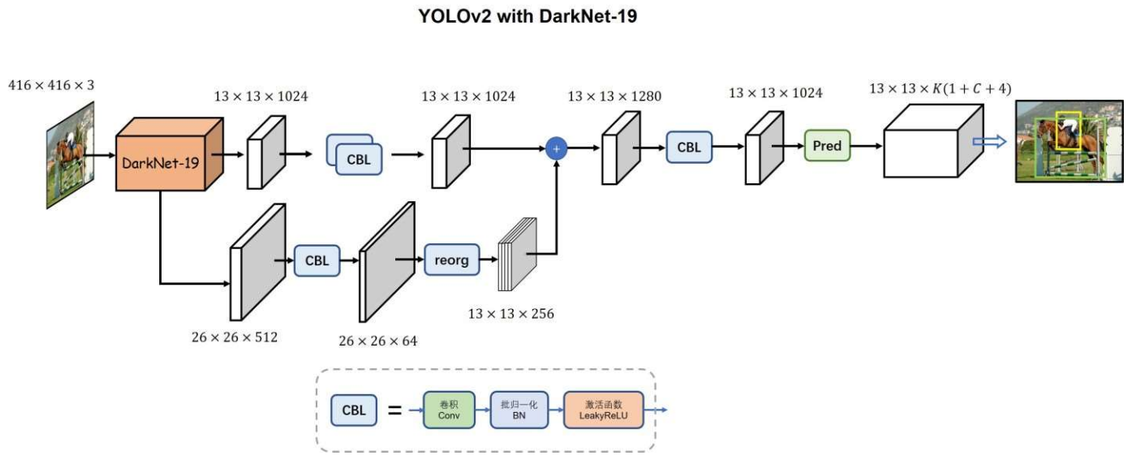


2.1 YOLO入门教程：YOLOv2(1)-解读YOLOv2

Kissrabbit
业余写手/机器人/深度学习/计算机视觉

一、序言

在整个第一章，我们学习了YOLOv1的原理，并在此基础上实现了第一个由我们自己来手动搭建的单尺度目标检测网络：YOLOv1+。从检测的原理上来看，我们的YOLOv1+和YOLOv1是一样，仅仅是在实现上改用和加入了更贴近当下主流的做法，如使用ResNet、使用BN以及全卷积网络结构等。相较于YOLOv1，我们的YOLOv1+具有更加简洁的网络结构，更加出色的检测性能。

另外，我们还在1.10节中给出了增强版的YOLOv1，在COCO上的AP高达35。倘若不是对小目标有着很高的需求，那么增强版的YOLOv1可以胜任大多数检测任务，不失为一个好的工作。同时，我们实现的端到端的全卷积网络架构的YOLOv1也更加方便读者学习、理解和实践。

当然，这里并不是吹捧我们的YOLOv1如何的出色，毕竟这是笔者在2020年搭建的模型，自然会融合进来很多更现代的技术。YOLOv1的影响是很深远的，是开启了单阶段（one-stage）通用目标检测模型时代的先锋之作。既然叫做YOLOv1，那就表明YOLO这个工作并未就此止步。

在2016年的CVPR会议上，继YOLOv1工作后，原作者再次推出YOLOv2（或YOLO9000，但我们只关注YOLOv2这一部分）。相较于上一代的YOLOv1，YOLOv2在其基础之上做了大量的改进和优化，不仅仅是对模型本身做了优化，同时还引入了由Faster R-CNN工作提出的anchor box机制，并且使用了kmeans聚类方法来获得更好的anchor box，边界框的回归方法也因此做了调整。在VOC2007数据集上，YOLOv2超越了同年发表在ECCV会议上的SSD工作，是那个年代当之无愧的最强目标检测器之一。那么，接下来就让我们去看看YOLOv2究竟做了哪些改进吧。

二、解读YOLOv2

2.1 添加Batch Normalization层

在YOLOv1中，每一层卷积的结构都是线性卷积和激活函数，并没有使用诸如批归一化（batch normalization，简称BN）、层归一化（layer，normalization，简称LN）、实例归一化（instance normalization，简称IN）等任何归一化层。这一点是受限于那个年代的相关技术的发展，而以现在的眼光来看，这些归一化层几乎是搭建网络的标配，尤其是在计算机视觉领域中，BN层几乎随处可见。于是，在YOLOv2中，YOLO作者为YOLOv1添加了BN层，即卷积层的组成从原先的线性卷积与激活函数的组合改进为后来常用的“**卷积三件套**”：线性卷积、BN层以及激活函数，如图1所示。

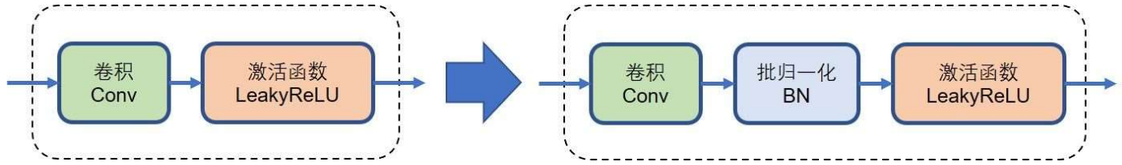


图1. YOLOv2使用的“卷积三件套”

于是，YOLOv1得到了第一次性能提升，在VOC2007测试集上，从原本的63.4% mAP提升到65.8% mAP。

2.2 高分辨率主干网络

在YOLOv1中，其backbone先在ImageNet上进行预训练，预训练时所输入的图像尺寸是224×224，而做检测任务时，YOLOv1所接收的输入图像尺寸是448×448，不难想到，训练过程中，网络必须要先克服由分辨率尺寸的剧变所带来的问题。毕竟，backbone网络在ImageNet上看得都是224×224的低分辨率图像，突然看到448×448的高分辨率图像，难免会“眼晕”。为了缓解这一问题，作者将已经在224×224的低分辨率图像上训练好的分类网络又在448×448的高分辨率图像上进行微调，共微调10个轮次。微调完毕后，再去掉最后的全局平均池化层和softmax层，作为最终的backbone网络。

于是，YOLOv1网络获得了第二次性能提升：从65.8% mAP提升到69.5% mAP。

由此可见，这一技巧确实有着明显的作用。不过，似乎这一技巧也只有YOLO工作在用，并未成为主流训练技巧，鲜在其他工作中用到。一个可能的原因就是这个问题可能确实不是很严重，稍微延长训练时间便可以了。

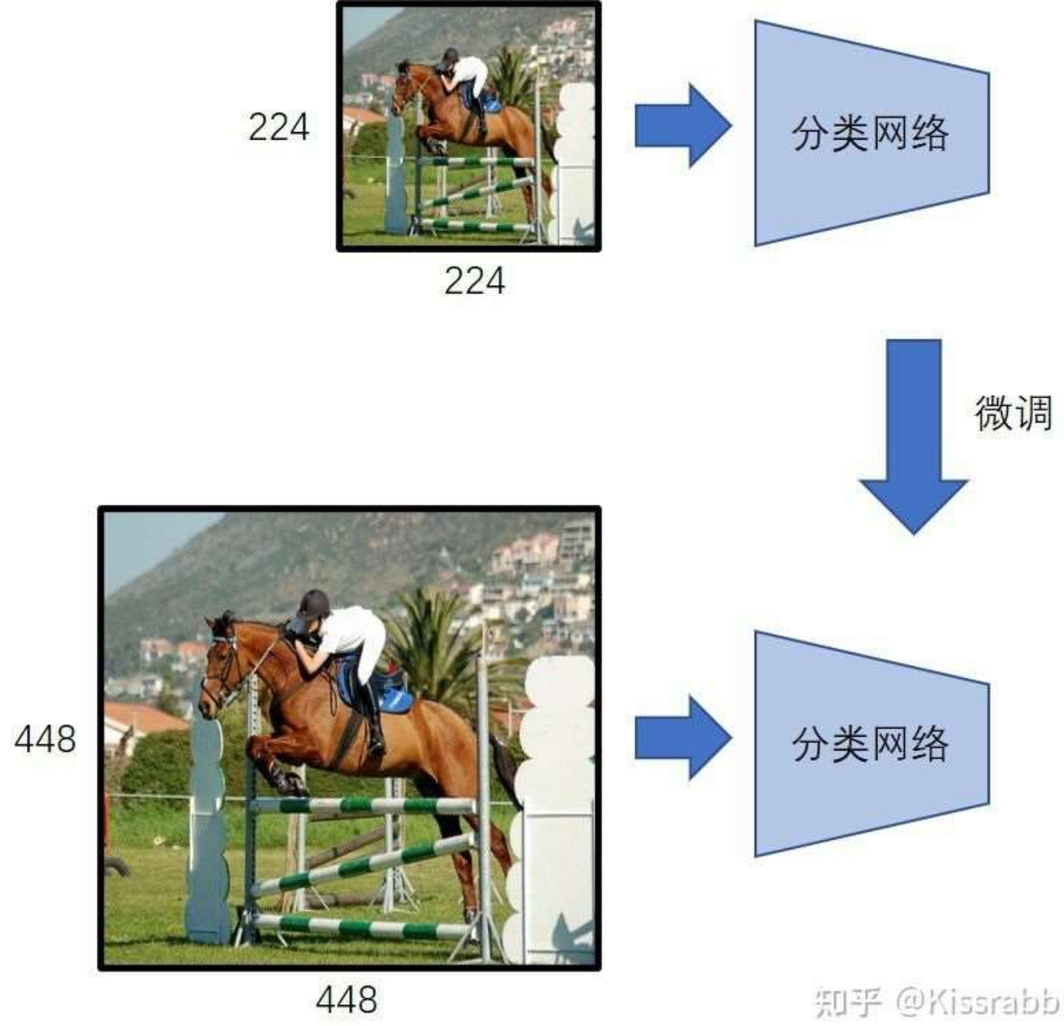


图2. 使用高分辨率图像训练主干网络

2.3 anchor box机制

在讲解YOLO的下一处改进之前，不妨先简单了解一下anchor box机制，这对我们掌握YOLO的下一次改进是有帮助的。

所谓的anchor box，字面翻译为“锚框”。如何理解“锚框”？我们很容易会想到一艘轮船到了岸边，要抛下锚从而将船身固定住，锚框的意思便是将一堆边界框放置在特征图网格的每一处位置，通常每个位置都放置相同数量的相同尺寸的锚框，如图3所示，注意，图中每两个网格绘制一次，仅是出于观赏性的考虑，实际上每一个网格都有，若是都画出来，委实令人眼晕，还请读者理解。

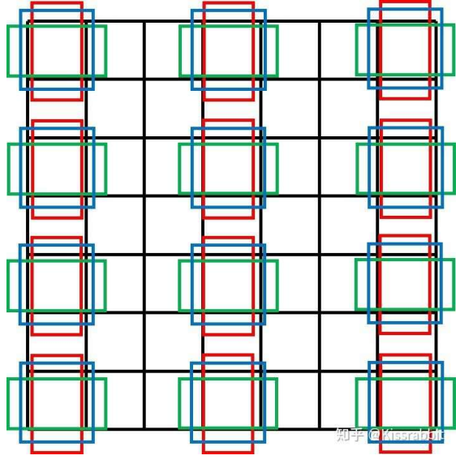


图3. Anchor box 机制

这一机制最早是在Faster R-CNN工作中提出的，用在RPN网络中，RPN网络在这些预先放置好的锚框上去为后续的预测提供**感兴趣区域**（Region of Interest，RoI）。每个网格处设定了k个不同尺寸、不同宽高比的anchor box，RPN网络会为每一个anchor box学习若干偏移量：**中心点的偏移量**和**宽高的偏移量**。用这些偏移量去调整每一个anchor box，得到最终的边界框。由此可见，anchor box的本质是提供边界框的尺寸先验，网络使用偏移量在这些先验值上进行调整，从而得到最终的尺寸，对于边界框的学习，不再是之前的“无中生有”了。因此，anchor box与其直接翻译成“锚框”，不如翻译成“先验框”更加贴切。加入先验框的目标检测网络，后来都被称为“Anchor-based”模型。

当然，有Anchor-based，自然也会有Anchor free概念，这是后话了，暂且不提。

设计先验框的一个难点在于设计多少个先验框，且每个先验框的尺寸（宽高比和面积）又是多少。对于宽高比，研究者们通常采取的配置是1 : 1、1 : 3以及3 : 1；对于面积，常用的配置是32、64、128、256以及512。

以上述两个配置为例，每一个面积都使用3个长宽比，因此，不难算出共有15个先验框，即k=15。对于一个13×13的网格，每一处的网格都要放置15个先验框，因此，这张网格上共有13×13×15=2535个先验框。如果我们用更多的网格，这个数量会更多。先验框越多，所需的参数量就越大，自然就会带来更多的计算量上的压力。

总之，先验框的作用就是提供边界框的尺寸先验信息，让网络只需学习偏移量来调整先验框去获得最终的边界框，相较于YOLOv1的直接回归边界框的宽高，基于先验框的方法表现得往往更好。

2.4 全卷积网络结构

在YOLOv1中，有一个很显着的问题就是网络在最后阶段使用了全连接层这不仅破坏了先前的特征图所包含的空间信息结构，同时也导致参数量爆炸。为了解决这一问题，作者便将其改成了全卷积结构，并且添加了Faster R-CNN工作所提出的anchor box机制。具体来说，首先，网络的输入图像尺寸从448改为416，去掉了YOLOv1网络中的最后一个池化层和所有的全连接层，修改后的网络的最大降采样倍数为32，最终得到的也就是13×13的网格，不再是7×7。每个网格处都预设了k

个的anchor box。网络只需要学习将先验框映射到真实框的尺寸的偏移量即可，无需再学习整个真实框的尺寸信息，这使得训练变得更加容易。

正如同我们在大学学习高等数学，有了高中的数学积累后，高等数学中的许多东西学起来也就有了一些基础，尽管高中数学的内容完全不够用，但总好过于一个从来没有学过数学的人直接学高等数学，后者的学习难度可想而知。

其实，在之前的YOLOv1中，我们已经知道每个网格处会有1个边界框输出，而现在变成了预测k个先验框的偏移量。原先的YOLOv1中，每个网格处的B个边界框都有一个置信度，但是类别是共享的，因此每个网格处最终只会有一个输出，而不是B个输出（置信度最高的那一个），倘若一个网格包含了两个以上的物体，那必然会出现漏检问题。加入先验框后，YOLOv1改为每一个先验框都预测一个类别和置信度，即每个网格处会有多个边界框的预测输出。因此，现在的YOLOv1的输出张量大小是S×S×k×(1+4+C)，每个边界框的预测都包含1个置信度、4个边界框的位置参数和个类别预测。

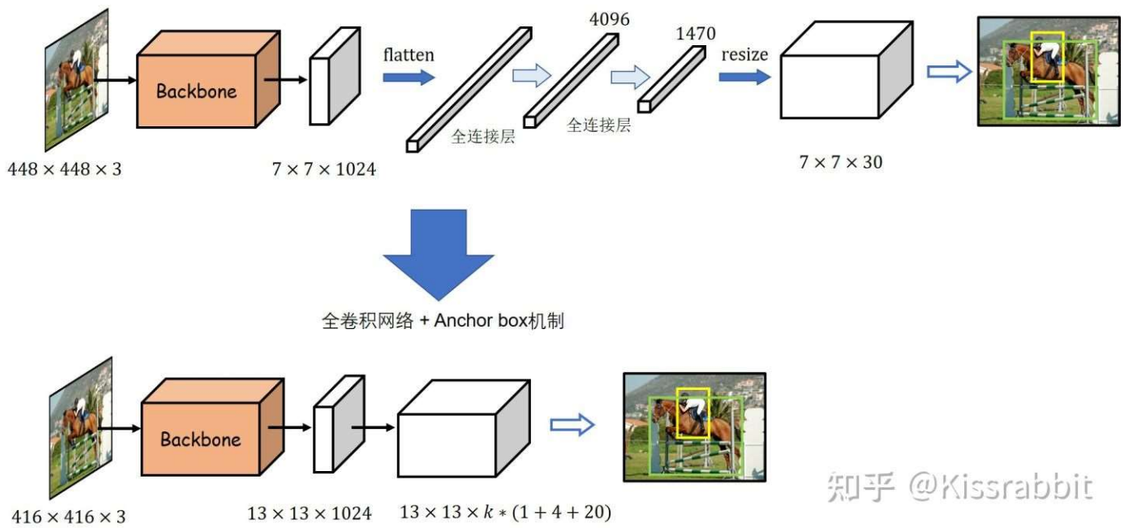


图4. 全卷积网络结构

尽管网络结构变成了全卷积网络，并使用了anchor box机制，但网络的精度并没有提升，反倒是略有所下降，69.5% mAP降为69.2% mAP，但召回率却从81%提升到88%。召回率的提升意味着YOLO可以找出更多的目标了，尽管精度下降了一点点。由此可见，每个网格输出多个检测结果确实有助于网络检测更多的物体。因此，作者并没有因为这微小的精度损失而放弃掉这一改进。

2.5 新的主干网络

随后，作者又设计了新的backbone网络取代原先的GoogLeNet风格的backbone网络，新网络被命名为DarkNet19，其网络结构如表1所示。

	层类型	卷积核数量	卷积核大小/步长	输出尺寸
1	卷积层	32	3×3	224×224
2	最大池化层		2×2 / 2	112×112
3	卷积层	64	3×3	112×112
4	最大池化层		2×2 / 2	56×56
5	卷积层	128	3×3	56×56
6	卷积层	64	1×1	56×56
7	卷积层	128	3×3	56×56
8	最大池化层		2×2 / 2	28×28
9	卷积层	256	3×3	28×28
10	卷积层	128	1×1	28×28
11	卷积层	256	3×3	28×28
12	最大池化层		2×2 / 2	14×14
13	卷积层	512	3×3	14×14
14	卷积层	256	1×1	14×14
15	卷积层	512	3×3	14×14
16	卷积层	256	1×1	14×14
17	卷积层	512	3×3	14×14
18	最大池化层		2×2 / 2	7×7
19	卷积层	1024	3×3	7×7
20	卷积层	512	1×1	7×7
21	卷积层	1024	3×3	7×7
22	卷积层	512	1×1	7×7
23	卷积层	1024	3×3	7×7
24	卷积层	1000	1×1	7×7
25	平均池化层		全局	1000
26	Softmax 层			

图5. DarkNet-19的网络结构

其中的卷积层即为前面所提到的“卷积三件套”：线性卷积、BN层以及LeakyReLU激活函数的组合。DarkNet19名字中的19是因为该网络共包含19个卷积层。

作者首先将DarkNet19在ImageNet上进行预训练，获得了72.9%的top1准确率和91.2%的top5准确率。在精度上，DarkNet19网络达到了VGG网络的水平，但前者模型更小。

预训练完毕后，去掉表6-1中的最后第24层的卷积层、第25层的平池化层以及第26层的softmax层，然后换掉原先的backbone网络。于是，YOLOv1网络从上一次的69.2% mAP提升到69.6% mAP。

2.6 kmeans聚类先验框

前面简介anchor box机制时，我们提到了这些先验框的一些参数需要人工设计，包括先验框的数量和大小。在Faster R-CNN中，这些参数都是由人工设定的，然而YOLO作者认为人工设定的不一定好。为了去人工化，作者采用kmeans方法在VOC数据集上进行聚类，一共聚类出k个先验框，通过实验，作者最终设定 。聚类的目标是数据集中所有检测框的宽和高，与类别无关。为了能够实现这样的聚类，作者使用IoU作为聚类的衡量指标，如公式(1)所示。

(1). $d\left(box, centroid\right) = 1 - IoU\left(box, centroid\right)$

通过kmeans聚类的方法所获得的先验框显然会更适合于所使用的数据集，但这也会带来一个问题：从A数据集聚类出的先验框显然难以适应新的B数据集。尤其A和B两个数据集中所包含的数据相差甚远时，这一问题会更加的严重。因此，当我们换一个数据集，如COCO数据集，则需要重新进行一次聚类，如果样本不够充分，这种聚类出来的先验框也就不够好，这也是YOLOv2以及后续的YOLO版本的潜在问题之一。另外，由聚类所获得的先验框严重依赖于数据集本身，倘若数据集规模过小、样本不够丰富，那么由聚类得到的先验框也未必会提供足够好的尺寸先验信息。当然，即便是人工设计的先验框也有着类似的问题，甚至整个anchor-based模型都有这一类问题，所以才有了后来的anchor free工作，这是后话，暂且不提。

回到YOLOv2工作来，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 t_x 和 t_y 。我们知道，这个中心点偏移量是介于01范围之间的数，在YOLOv1时，作者没有在意这一点，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。于是，作者使用sigmoid函数使得网络对偏移量的预测是处在01范围中。我们的YOLOv1+正是借鉴了这一点。

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。假设某个先验框的宽和高分别为 p_w 和 p_h ，网络输出宽高的偏移量为 t_w 和 t_h ，则使用公式（2）和公式（3）即可解算出边界框的宽和高。

(2). $w = p_w e^{t_w}$

(3). $h = p_h e^{t_h}$

我们的YOLOv1+的exp-log方法正是借鉴了YOLOv2的这两个公式。这种边界框预测方法，在论文中被命名为“location prediction”。

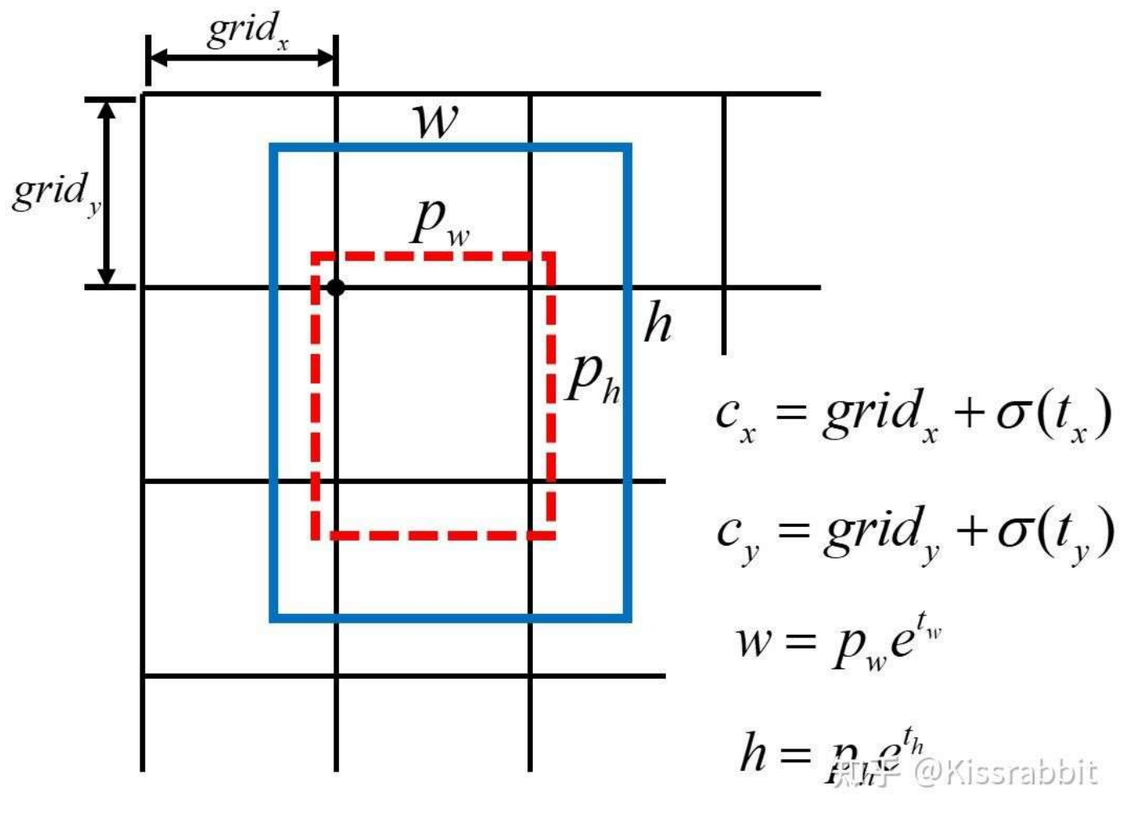


图5. 基于anchor box的边界框预测

使用kmeans聚类方法获得先验框，再配合“location prediction”的边界框预测方法，YOLOv1的性能得到了显着的提升：从69.6% mAP提升到74.4% mAP。不难想到，性能提升的主要来源在于kmeans聚类，更好的先验信息自然会有效提升网络的检测性能。只不过，这种先验信息是依赖于数据集的，这是一个潜在问题，感兴趣的读者不妨在这一点上稍作思考。

2.7 使用更高分辨率的特征

随后，YOLO作者又借鉴了同年的SSD工作：使用更高分辨的特征。在SSD工作中，检测是在多张特征图上进行的，不同的特征图的分辨率也不同。可以理解为，特征图的分辨率越高，所划分的网格也就越精细，能够更好地捕捉目标的细节信息。相较于YOLOv1只在一张7×7的过于粗糙的网格上做检测，使用多种不同分辨率的特征图自然会更好。

于是，YOLO作者借鉴了这一思想。具体来说，之前的改进中，YOLOv1都是在最后一张大小为13×13×1024的特征图上进行检测，为了引入更多的细节信息，作者将backbone的第17层卷积输出的26×26×512特征图拿出来，做一次特殊的降采样操作，得到一个13×13×2048特征图，然后将二者在通道的维度上进行拼接，最后在这张融合了更多信息的特征图上去做检测。

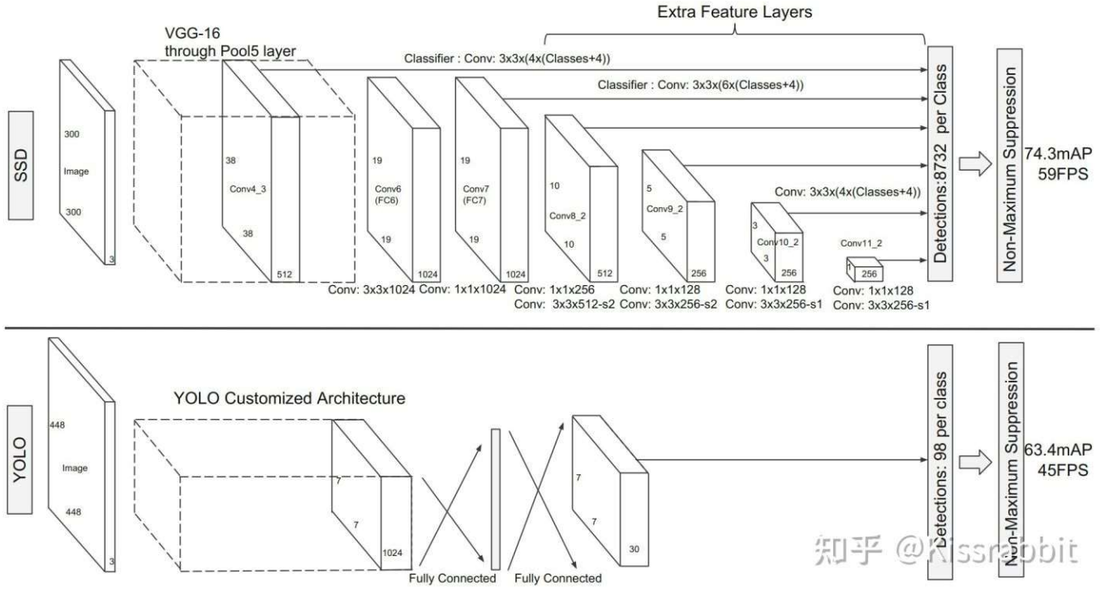


图6. SSD目标检测网络与YOLOv1的对比

需要说明的是，这里的特殊降采样操作并不是常用的步长为2的池化层或步长为2的卷积操作，而是一种类似于在图像分割任务中常用到的pixelshuffle操作的逆操作，如图7所示。依据YOLO官方配置文件中的命名方式，我们暂且称之为reorg操作。

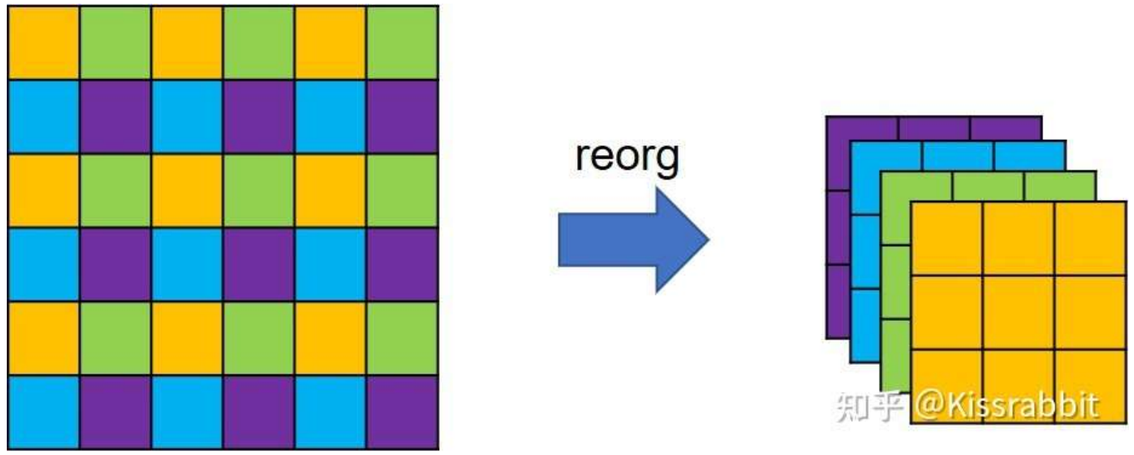


图7. reorg操作

不难发现，特征图在经过reorg操作的处理后，特征图的宽高会减半，而通道则扩充至4倍，因此，从backbone拿出来的26×26×512特征图就变成了13×13×2048特征图。这种特殊降采样操作的好处就在于降低分辨率的同时，没丢掉任何细节信息，信息总量保持不变。

这一改进在论文中被命名为“passthrough”。加上该操作后，在VOC 2007测试集上的mAP从74.4%再次涨到了75.4%。由此可见，引入更多的细节信息，确实有助于提升模型的检测性能。

2.8 多尺度训练

在计算机视觉中，一个十分常见的图像处理操作是图像金字塔，即将一张图像缩放到不同的尺寸，同一目的，在不同分辨率的图像中，所包含的信息量也不一样，直观的体现便是分辨率越高，构成目标所需要的像素量就越多，目标本身的大小（或像素面积）也就越大。通过使用图像金字塔的操作，网络能够在不同尺寸下去感知同一目标，从而增强了其本身对目标尺寸变化的鲁棒性，如图8所示。YOLO作者便将这一思想用到了模型训练中，以提升YOLO对物体的尺度变化的适应能力。

图8. 图像金字塔

具体来说，在训练网络时，每训练迭代10次（常用iteration表示训练一次，即一次前向传播和一次反向传播，而训练一轮次则用epoch，即数据集的所有数据都经过了一次迭代），就从{320，352，384，416，448，480，512，576，608}选择一个新的图像尺寸用作后续10次训练的图像尺寸。注意，这些尺寸都是32的整数倍，因为网络的最大降采样倍数就是32，倘若输入一个无法被32整除的图像尺寸，则会遇到些不必要的麻烦。这里不建议大家尝试这一点。

这种多尺度训练的好处就在于可以改变数据集中各类物体的大小占比，比如说，一个物体在608的图像中占据较多的像素，面积较大，而在320图像中就会变少了，就所占的像素数量而言，相当于从一个较大的物体变成了较小物体。通常，多尺度训练是常用的提升模型性能的技巧之一。不过，技巧终归是技巧，并不总是有效的，若是我们的目标几乎不会有明显的尺寸变化，那么也就没必要进行多尺度训练了。

配合多尺寸训练，YOLOv1再一次获得了提升：从75.4% mAP提升到76.8% mAP。

既然已经使用了多尺度训练，YOLOv1模型不仅可以使用416这个尺寸去做测试，也可以使用更大的图像尺寸去做测试。于是，作者又使用544尺寸去测试mAP，意料之内地得到了更高的测试结果：78.6% mAP。

至此，YOLOv1彻底进化为YOLOv2，一个十分强大的基于anchor box的单尺度目标检测网络。

在本节的最后，笔者参照YOLOv2官方的配置文件，提供了YOLOv2网络的结构图，如图9所示。

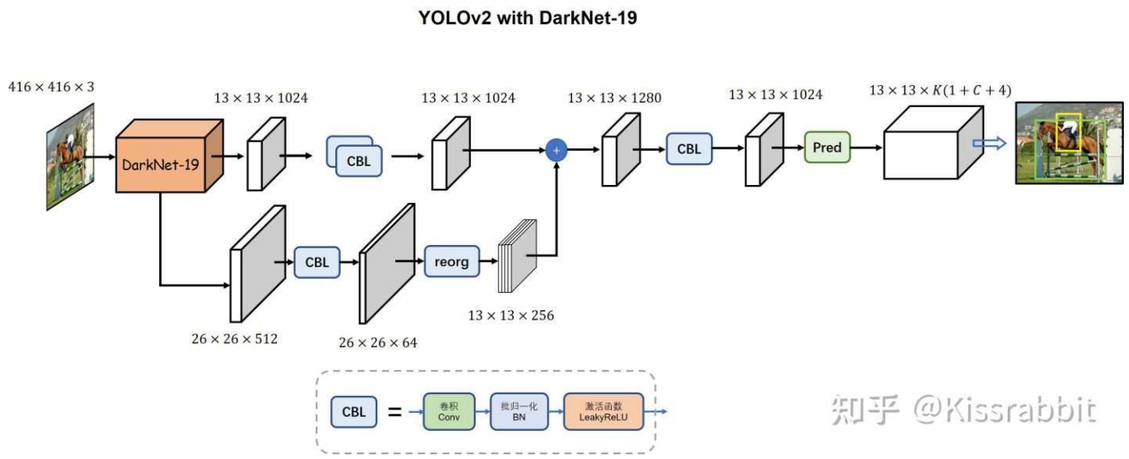


图9. YOLOv2网络结构图

下一节，我们将在YOLOv1+的工作基础之上，来构建我们的YOLOv2模型，并进行训练和测试，我们仍会在YOLOv2的工作基础上，做些小的改动，提升网络的性能，因此，由我们自己手动实现的第二个网络就命名为YOLOv2+。我们仍以VOC数据集为主，来完成实践环节，在最后的一小节中使用更大的COCO数据集。数据预处理和数据增强代码无需改动，仍沿用YOLOv1+的代码，我们只需要单独写YOLOv2+的模型代码、制作正样本的代码以及损失函数的代码。

当然，在实现完我们的YOLOv2+之后，笔者还会再最后一节提供一个增强版的YOLOv2，在COCO val上的AP高达36.5，增强版的YOLOv2主要就是在之前的增强版的YOLOv1基础之上加入了anchor box。

敬请期待！

科普不易，还请各位读者多多支持，十分感谢！

编辑于 2022-01-28 12:16

「真诚赞赏，手留余香」

赞赏

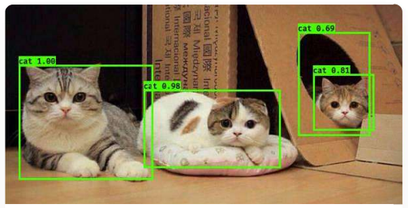
还没有人赞赏，快来当第一个赞赏的人吧！

YOLO算法　目标检测　计算机视觉

文章被以下专栏收录

第二卷-基于YOLO的目标检测入门教程
本专栏将从零开始完整实现YOLOv1至v3模型

推荐阅读



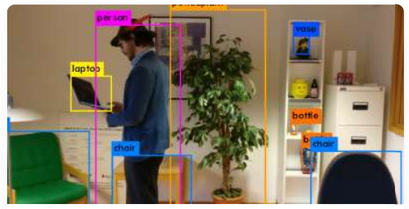
目标检测算法之YOLOv2

BBuf　　发表于Giant...

《目标检测》-第0章-序

本专栏的 核心内容是围绕着YOLOv1，讲得最为细致，因为YOLOv1是最适合作为目标检测的入门模型。其次会上升到YOLOv2和YOLOv3，但讲得就没有v1那么细致了，相较于v1，v2和v3也就...

Kissr...　　发表于第一卷-目...



目标检测算法之YOLOv2损失函数详解

BBuf　　发表于Giant...

目标检测之YOLOv2/3

Dwzb　　发表于Data ...

17 条评论

切换为时间排序

写下你的评论...

书昨日　　2021-11-12

哇！终于更了！期待

1

Kissrabbit (作者) 回复 书昨日　　2021-11-12

谢谢支持！后面会稳定更新，请放心食用🧐

2

书昨日 回复 Kissrabbit (作者)　　2021-11-12

好哒~感觉您写的挺不错哈哈，对我一个小白很友好🥹

1

一条乐　　03-27

博主码字不易！给赞支持一下~

赞

余忆　　02-21

感谢博主，可以引用上面一些知识吗？感谢博主了

赞

Kissrabbit (作者) 回复 余忆　　02-21

当然可以~如果对内容有问题，请在下方留言

赞

来学习不看装13 01-25

你敲到里面的公式好像被吞了，类似图像大小224*224这种的

赞

Kissrabbit (作者) 回复 来学习不看装13 01-25

这个可能是网卡了一下或者知乎有了点日常小毛病～重新打开过着过一会儿就好了

赞

Kemlz 回复 Kissrabbit (作者) 01-25

在YOLOv1中，其backbone先在ImageNet上进行预训练，预训练时所输入的图像尺寸是 ，而做检测任务时，YOLOv1所接收的输入图像尺寸是 。(2.2节第一句)我也看不到这些~

赞

查看全部 9 条回复

duxingwei 2021-12-09

👍

赞