基于深度学习的目标检测算法综述（一）

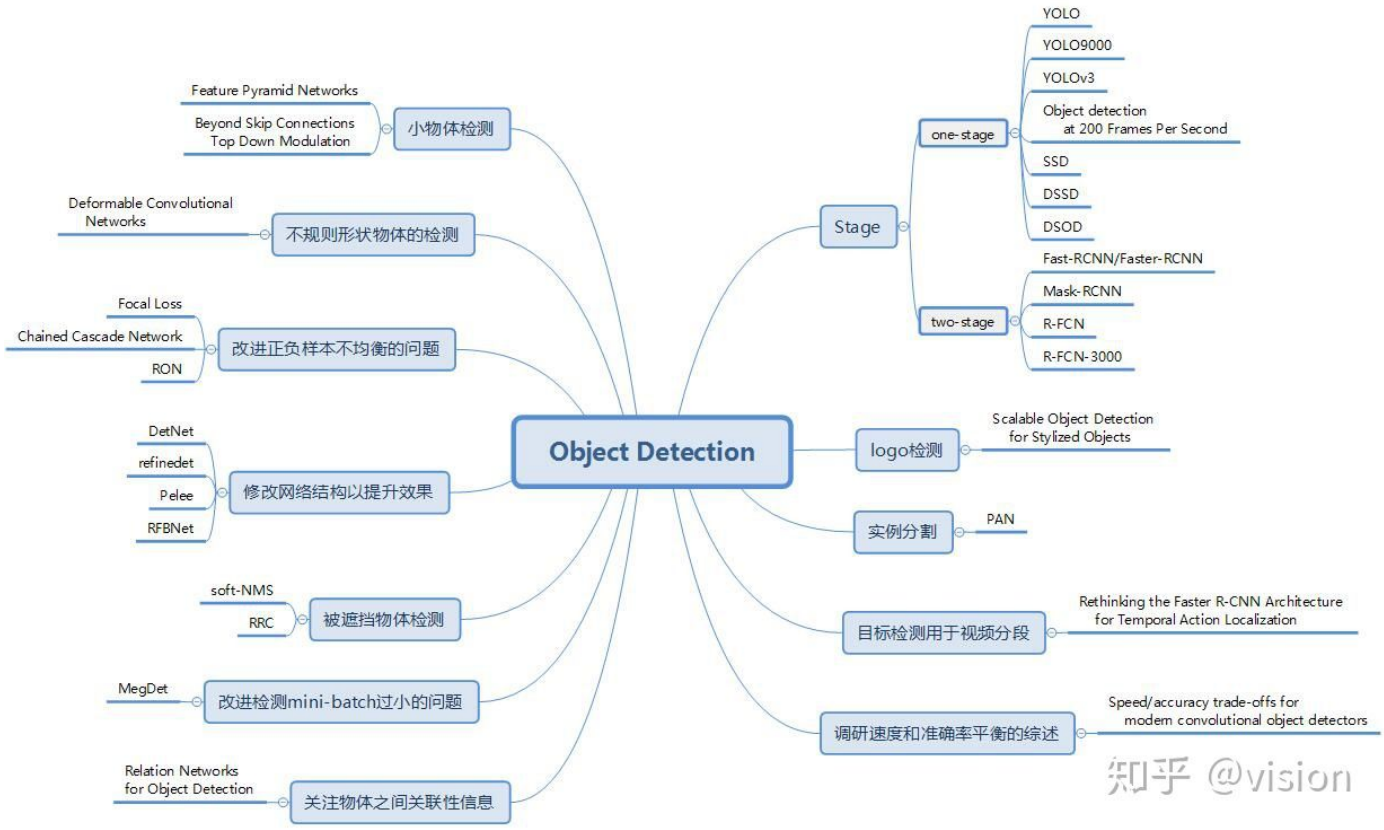
vision
算法攻城狮

关注他

本文内容原创，作者：美图云视觉技术部 检测团队，转载请注明出处

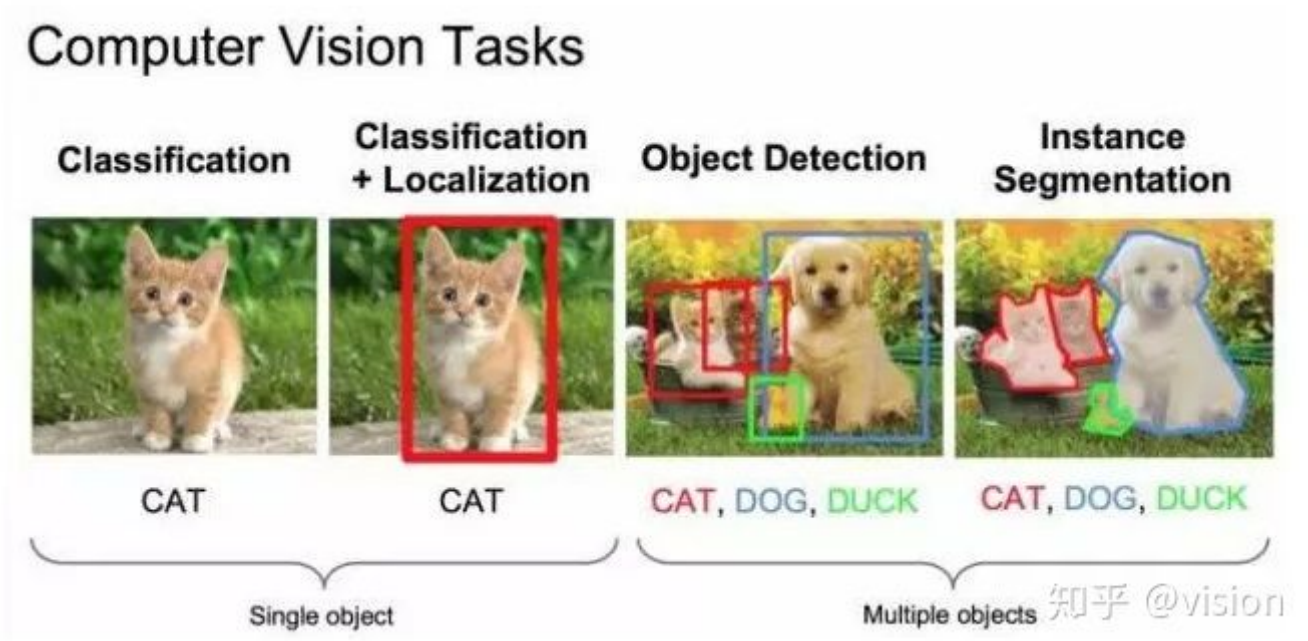
目标检测（Object Detection）是计算机视觉领域的基本任务之一，学术界已有将近二十年的研究历史。近些年随着深度学习技术的火热发展，目标检测算法也从基于手工特征的传统算法转向了基于深度神经网络的检测技术。从最初2013年提出的R-CNN、OverFeat，到后面的Fast/Faster R-CNN，SSD，YOLO系列，再到2018年最近的Pelee。短短不到五年时间，基于深度学习的目标检测技术，在网络结构上，从two stage到one stage，从bottom-up only到Top-Down，从single scale network到feature pyramid network，从面向PC端到面向手机端，都涌现出许多好的算法技术，这些算法在开放目标检测数据集上的检测效果和性能都很出色。

本篇综述的出发点一方面是希望给检测方向的入门研究人员提供一个技术概览，帮助大家快速了解目标检测技术上下文；另一方面是给工业界应用人员提供一些参考，通过本篇综述，读者可以根据实际业务场景，找到合适的目标检测方法，在此基础上改进、优化甚至是进一步创新，解决实际业务问题。本文对其中的27篇论文进行介绍，这27篇论文涵盖了2013以来，除SSD，YOLO和R-CNN系列之外的，所有引用率相对较高或是笔者认为具有实际应用价值的论文。R-CNN系列，SSD和YOLO相关的论文详解资源已经非常多，所以本文不再赘述。下图对这些方法进行了分类概括。



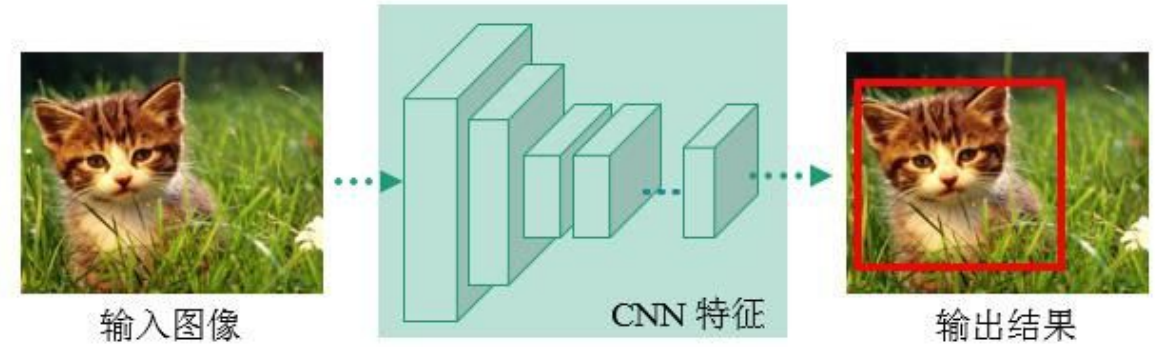
下文中，我们针对每篇文章，从论文目标，即要解决的问题，算法核心思想以及算法效果三个层面进行概括。同时，我们也给出了每篇论文的出处，录用信息以及相关的开源代码链接，其中代码链接以作者实现和mxnet实现为主。

一、 背景

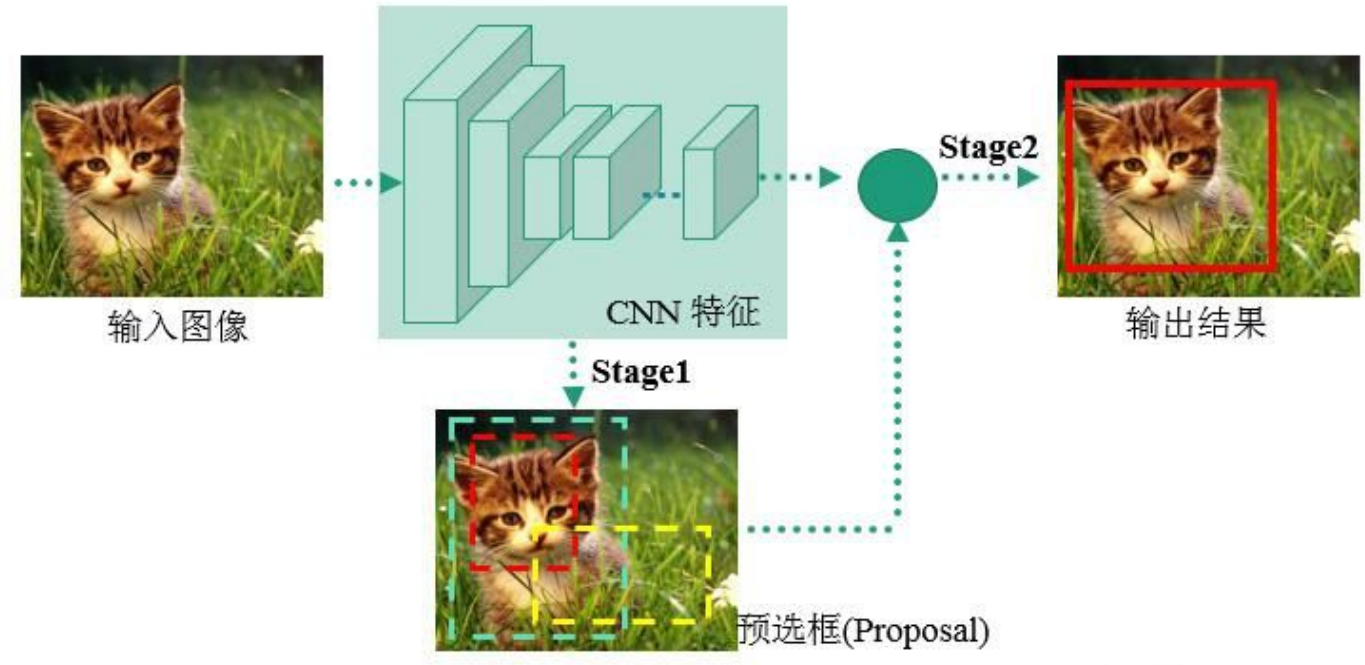


物体检测的任务是找出图像或视频中的感兴趣物体，同时检测出它们的位置和大小，是机器视觉领域的核心问题之一。

物体检测过程中有很多不确定因素，如图像中物体数量不确定，物体有不同的外观、形状、姿态，加之物体成像时会有光照、遮挡等因素的干扰，导致检测算法有一定的难度。进入深度学习时代以来，物体检测发展主要集中在两个方向：two stage算法如R-CNN系列和one stage算法如YOLO、SSD等。两者的主要区别在于two stage算法需要先生成proposal（一个有可能包含待检物体的预选框），然后进行细粒度的物体检测。而one stage算法会直接在网络中提取特征来预测物体分类和位置。



One Stage结构



Two Stage结构

基于深度学习的目标检测算法综述分为三部分：

1. Two/One stage算法改进。这部分将主要总结在two/one stage经典网络上改进的系列论文，包括Faster R-CNN、YOLO、SSD等经典论文的升级版本。

2. 解决方案。这部分我们归纳总结了目标检测的常见问题和近期论文提出的解决方案。

3. 扩展应用、综述。这部分我们会介绍检测算法的扩展和其他综述类论文。

本综述分三部分，本文介绍第一部分。

二、创新内容、改进方向

1.Two/One stage算法改进

1.1 Two stage

Faster R-CNN网络包括两个步骤：1. 使用RPN(region proposal network)提取proposal信息，2. 使用R-CNN对候选框位置进行预测和物体类别识别。本文主要介绍在Faster R-CNN基础上改进的几篇论文：R-FCN、R-FCN3000和Mask R-CNN。R-FCN系列提出了Position Sensitive(ps)的概念，提升了检测效果。另外需要注明的是，虽然Mask R-CNN主要应用在分割上，但该论文和Faster R-CNN一脉相承，而且论文提出了RoI Align的思想，对物体检测回归框的精度提升有一定效果，故本篇综述也介绍了这篇论文。

1.1.1 R-FCN: Object Detection via Region-based Fully Convolutional Networks

论文链接：arxiv.org/abs/1605.0640

开源代码：github.com/daijifeng001

录用信息：CVPR2017

论文目标：

对预测特征图引入位置敏感分数图提增强征位置信息，提高检测精度。

核心思想：

一. 背景

Faster R-CNN是首个利用CNN来完成proposals的预测的，之后的很多目标检测网络都是借助了Faster R-CNN的思想。而Faster R-CNN系列的网络都可以分成2个部分：

1.Fully Convolutional subnetwork before RoI Layer

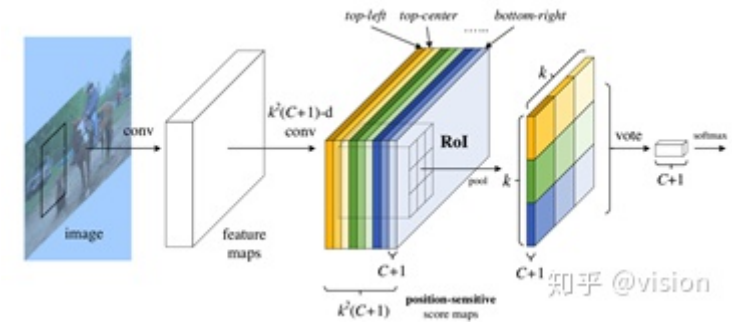
2.RoI-wise subnetwork

第1部分就是直接用普通分类网络的卷积层来提取共享特征，后接一个RoI Pooling Layer在第1部分的最后一张特征图上进行提取针对各个RoIs的特征图，最后将所有RoIs的特征图都交由第2部分来处理（分类和回归）。第二部分通常由全连接层组层，最后接2个并行的loss函数：Softmax和smoothL1，分别用来对每一个RoI进行分类和回归。由此得到每个RoI的类别和归回结果。其中第1部分的基础分类网络计算是所有RoIs共享的，只需要进行一次前向计算即可得到所有RoIs所对应的特征图。

第2部分的RoI-wise subnetwork不是所有RoIs共享的，这一部分的作用就是给每个RoI进行分类和回归。在模型进行预测时基础网络不能有效感知位置信息，因为常见的CNN结构是根据分类任务进行设计的，并没有针对性的保留图片中物体的位置信息。而第2部分的全连阶层更是一种对于位置信息非常不友好的网络结构。由于检测任务中物体的位置信息是一个很重要的特征，R-FCN通过提出的位置敏感分数图（position sensitive score maps）来增强网络对于位置信息的表达能力，提高检测效果。

二. 网络设计

2.1position-sensitive score map



上图展示的是R-FCN的网络结构图，展示了位置敏感得分图(position-sensitive score map)的主要设计思想。如果一个RoI含有一个类别c的物体，则将该RoI划分为k x k个区域，分别表示该物体的各个相应部位。其每个相应的部位都由特定的特征图对其进行特征提取。R-FCN在共享卷积层的最后再接上一层卷积层，而该卷积层就是位置敏感得分图position-sensitive score map。其通道数channels=k x k x (C+1)。C表示物体类别种数再加上1个背景类别，每个类别都有k x k 个score maps分别对应每个类别的不同位置。每个通道分别负责某一类的特定位置的特征提取工作。

2.2 Position-sensitive RoI pooling

位置敏感RoI池化操作了（Position-sensitive RoI pooling）如下图所示：

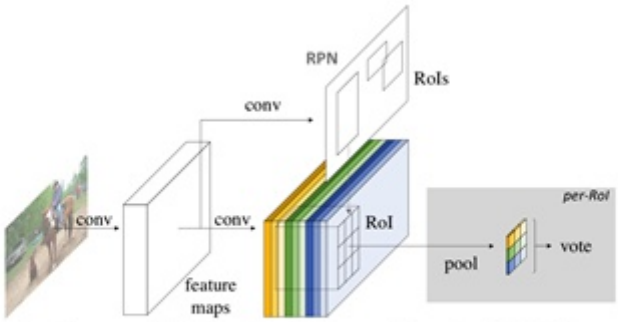
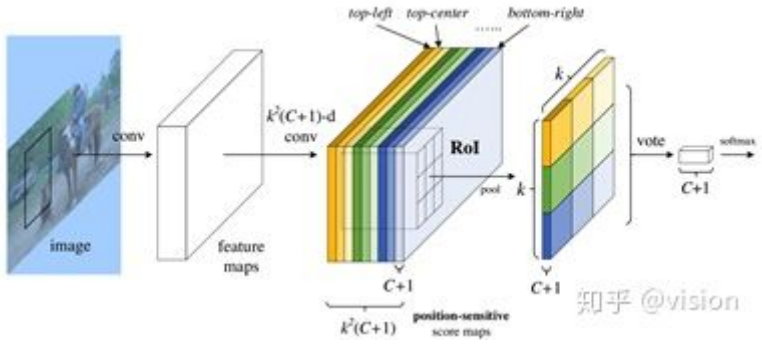


Figure 2: Overall architecture of R-FCN. A Region Proposal Network (RPN) [18] proposes candidate RoIs, which are then applied on the score maps. All learnable weight layers are convolutional and are computed on the entire image; the per-RoI computational cost is negligible.

该操作将每个RoIs分为k x k 个小块。之后提取其不同位置的小块相应特征图上的特征执行池化操作，下图展示了池化操作的计算方式。

$$r_c(i,j \mid \Theta) = \sum_{(x,y)\in \text{bin}(i,j)} z_{i,j,c}(x+x_0, y+y_0 \mid \Theta)/n.$$



得到池化后的特征后，每个RoIs的特征都包含每个类别各个位置上的特征信息。对于每个单独类别来讲，将不同位置的特征信息相加即可得到特征图对于该类别的响应，后面即可对该特征进行相应的分类。

$$r_c(\Theta) = \textstyle\sum_{i,j} r_c(i,j \mid \Theta).$$

$$s_c(\Theta) = e^{r_c(\Theta)} / \textstyle\sum_{c'=0}^{C} e^{r_{c'}(\Theta)}.$$

2.3 position-sensitive regression

在位置框回归阶段仿照分类的思路，将特征通道数组合为4 x k x k 的形式，其中每个小块的位置都对应了相应的通道对其进行位置回归的特征提取。最后将不同小块位置的四个回归值融合之后即可得到位置回归的响应，进行后续的位置回归工作。

三. 网络训练

3.1 position-sensitive score map高响应值区域

在训练的过程中，当RoIs包涵物体属于某类别时，损失函数即会使得该RoIs不同区域块所对应的响应通道相应位置的特征响应尽可能的大，下图展示了这一过程，可以明显的看出不同位置的特征图都只对目标相应位置的区域有明显的响应，其特征提取能力是对位置敏感的。

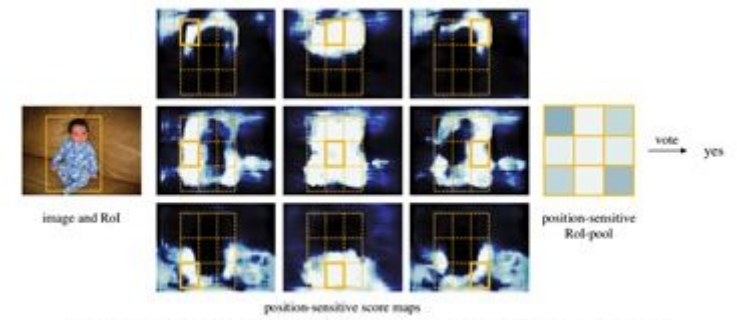


Figure 3: Visualization of R-FCN ($k \times k = 3 \times 3$) for the *person* category.

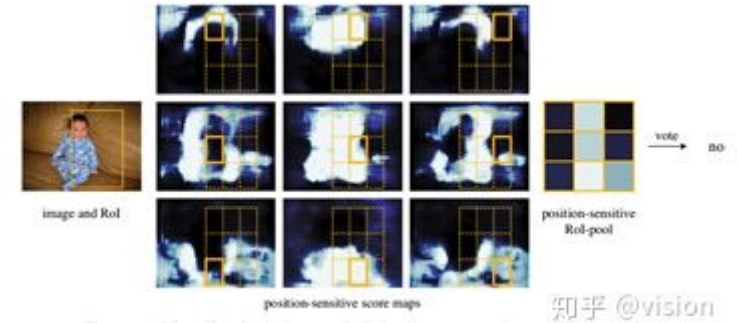


Figure 4: Visualization when an RoI does not correctly overlap the object.

3.2 训练和测试过程

使用如上的损失函数，对于任意一个RoI，计算它的Softmax损失，和当其不属于背景时的回归损失。因为每个RoI都被指定属于某一个GT box或者属于背景，即先让GT box选择与其IoU最大的那个RoI，再对剩余RoI选择与GT box的IoU>0.5的进行匹配，而剩下的RoI全部为背景类别。当RoI有了label后loss就可以计算出来。这里唯一不同的就是为了减少计算量，作者将所有RoIs的loss值都计算出来后，对其进行排序，并只对最大的128个损失值对应的RoIs进行反向传播操作，其它的则忽略。并且训练策略也是采用的Faster R-CNN中的4-step alternating training进行训练。在测试的时候，为了减少RoIs的数量，作者在RPN提取阶段就将RPN提取的大约2W个proposals进行过滤：

1.去除超过图像边界的proposals

2.使用基于类别概率且阈值IoU=0.3的NMS过滤

3.按照类别概率选择top-N个proposals

在测试的时候，一般只剩下300个RoIs。并且在R-FCN的输出300个预测框之后，仍然要对其使用NMS去除冗余的预测框。

算法效果：

	depth of per-RoI subnetwork	training w/ OHEM?	train time (sec/img)	test time (sec/img)	mAP (%) on VOC07
Faster R-CNN	10		1.2	0.42	76.4
R-FCN	0		0.45	0.17	76.6
Faster R-CNN	10	✓(300 RoIs)	1.5	0.42	79.3
R-FCN	0	✓(300 RoIs)	0.45	0.17	**79.5**
Faster R-CNN	10	✓(2000 RoIs)	2.9	0.42	N/A
R-FCN	0	✓(2000 RoIs)	0.46	0.17	79.3

上图比较了Faster-R-CNN 和R-FCN的mAP值和监测速度，采用的基础网络为ResNet-101，测评显卡为Tesla K40。

1.1.2 R-FCN-3000 at 30fps: Decoupling Detection and Classification

论文链接：arxiv.org/pdf/1712.0180

开源代码：无

录用信息：无

论文目标：

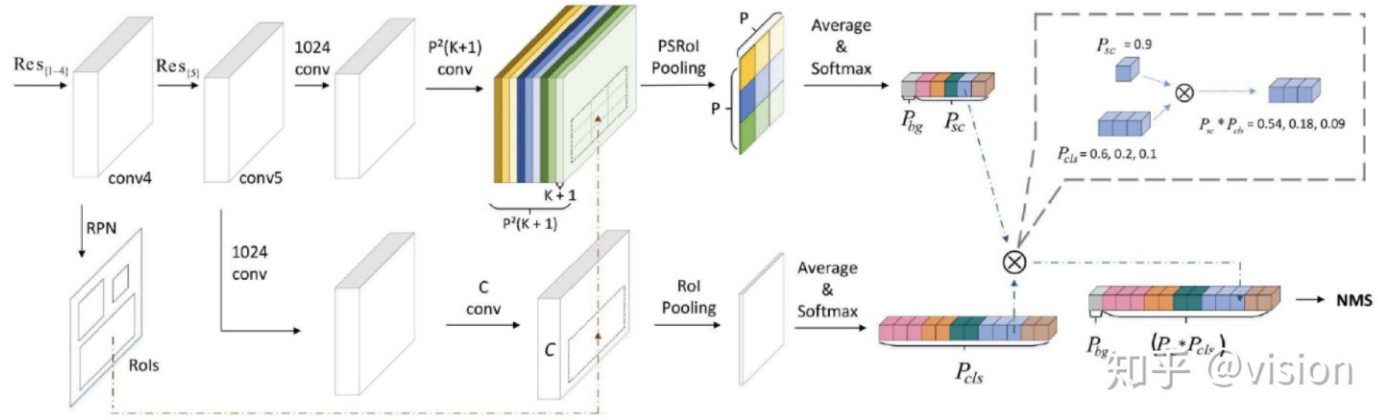
与YOLO9000（本论述后文会具体介绍YOLO9000）类似，本文的目标也是面向实际应用场景的大规模类别物体的实时检测。YOLO9000将检测数据集和分类数据集合并训练检测模型，但r-fcn-3000仅采用具有辅助候选框信息的ImageNet数据集训练检测分类器。

如果使用包含标注辅助信息（候选框）的大规模分类数据集，如ImageNet数据集，进行物体检测模型训练，然后将其应用于实际场景时，检测效果会是怎样呢？how would an object detector perform on "detection"datasets if it were trained on classification datasets with bounding-box supervision?

核心思想：

r-fcn-3000是对r-fcn的改进。上文提到，r-fcn的ps卷积核是per class的，假设有C个物体类别，有K*K个ps核，那么ps卷积层输出K*K*C个通道，导致检测的运算复杂度很高，尤其当要检测的目标物体类别数较大时，检测速度会很慢，难以满足实际应用需求。

为解决以上速度问题，r-fcn-3000提出，将ps卷积核作用在超类上，每个超类包含多个物体类别，假设超类个数为SC，那么ps卷积层输出K*K*SC个通道。由于SC远远小于C，因此可大大降低运算复杂度。特别地，论文提出，当只使用一个超类时，检测效果依然不错。算法网络结构如下：



上图可以看出，与r-fcn类似，r-fcn-3000也使用RPN网络生成候选框（上图中虚线回路）；相比r-fcn, r-fcn-3000的网络结构做了如下改进：

1. r-fcn-3000包含超类（上图中上半部分）和具体类（上图中下半部分）两个卷积分支。

2. 超类卷积分支用于检测超类物体，包含分类（超类检测）和回归（候选框位置改进）两个子分支；注意上图中没有画出用于候选框位置改进的bounding-box回归子分支；回归分支是类别无关的，即只确定是否是物体。

3. 具体类卷积分支用于分类物体的具体类别概率，包含两个普通CNN卷积层。

4. 最终的物体检测输出概率由超类卷积分支得到的超类类概率分别乘以具体类卷积分支输出的具体类别概率得到。引入超类和具体类两个卷积分支实现了‘物体检测’和‘物体分类’的解耦合。超类卷积分支使得网络可以检测出物体是否存在，由于使用了超类，而不是真实物体类别，大大降低了运算操作数。保证了检测速度；具体类分支不检测物体位置，只分类具体物体类别。

超类生成方式：对某个类别j的所有样本图像，提取ResNet-101最后一层2018维特征向量，对所有特征项向量求均值，作为该类别的特征表示。得到所有类别的特征表示进行K-means聚类，确定超类。

算法效果：

在imagenet数据集上，检测mAP值达到了34.9%。使用nvidia p6000 GPU，对于375x500图像，检测速度可以达到每秒30张。在这种速度下，r-fcn-3000号称它的检测准确率高于YOLO 18%。

此外，论文实验表明，r-fcn-3000进行物体检测时具有较强的通用性，当使用足够多的类别进行训练时，对未知类别的物体检测时，仍能检测出该物体位置。如下图：

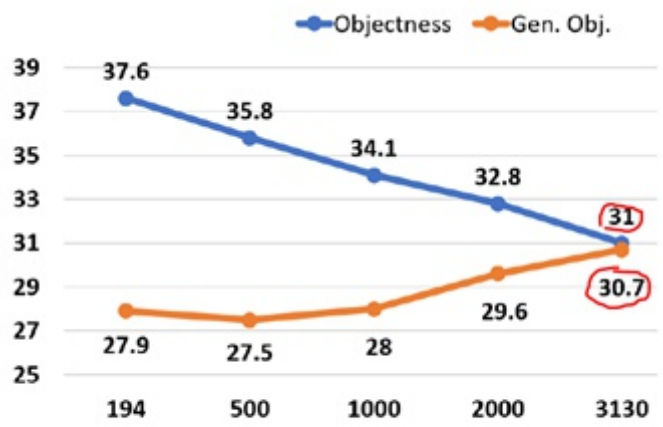


Figure 7. The mAP scores on a held out set of 20 classes for Generalized Objectness and Objectness baseline. 知乎 @vision

在训练类别将近3000时，不使用目标物体进行训练达到的通用预测mAP为30.7%，只比使用目标物体进行训练达到的mAP值低0.3%。

1.1.3 Mask R-CNN

论文链接：arxiv.org/abs/1703.0687

开源代码：github.com/TuSimple/mx-

录用信息：CVPR2017

论文目标

1. 解决RoIPooling在Pooling过程中对RoI区域产生形变，且位置信息提取不精确的问题。

2. 通过改进Faster R-CNN结构完成分割任务。

核心思想：

1. 使用RoIAlign代替RoIPooling，得到更好的定位效果。

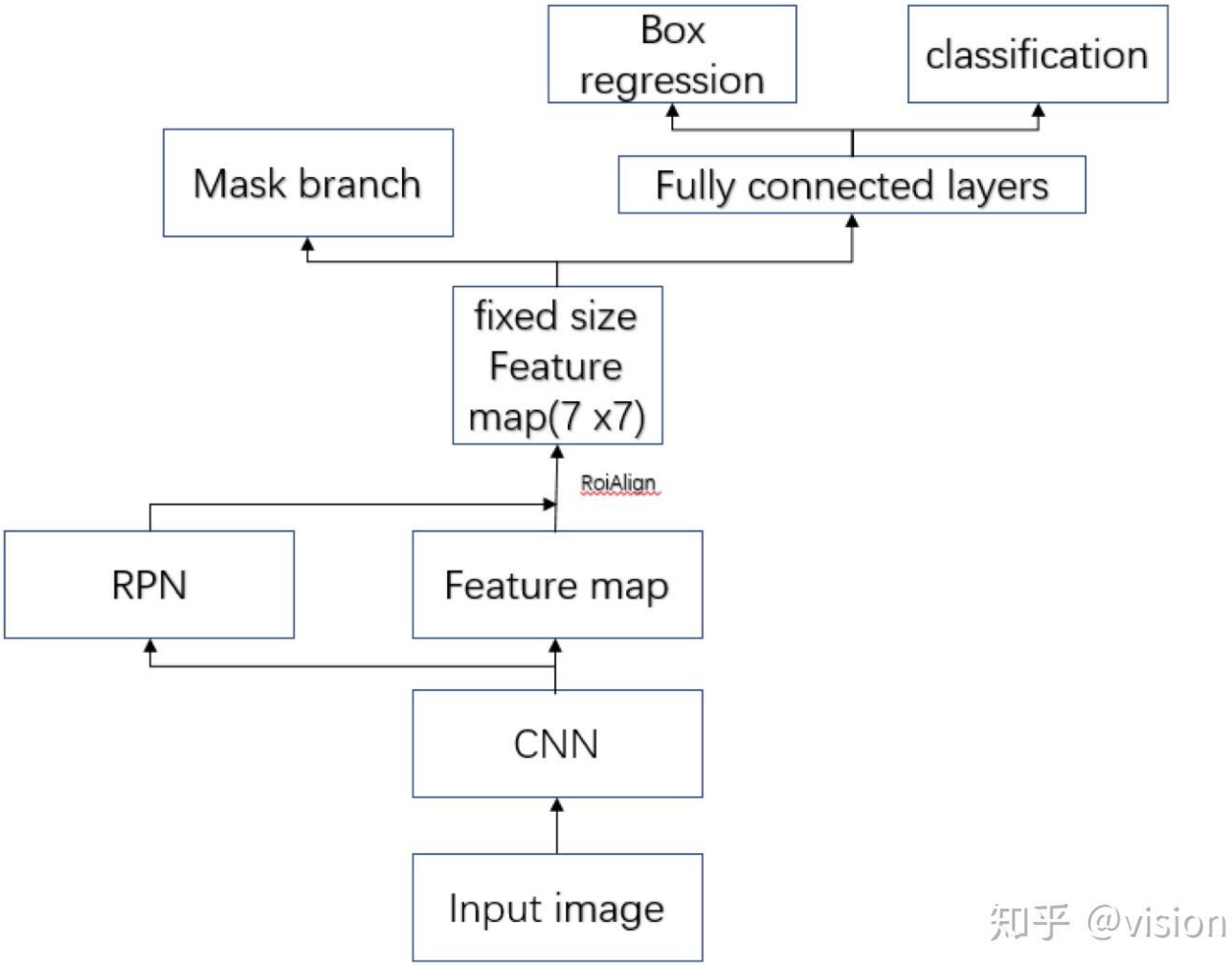
2. 在Faster R-CNN基础上加上mask分支，增加相应loss，完成像素级分割任务。

一、概述

Mask R-CNN是基于Faster R-CNN的基础上演进改良而来，不同于Faster R-CNN，Mask R-CNN可以精确到像素级输出，完成分割任务。此外他们的输出也有所不同。Faster R-CNN输出为种类标签和box坐标，而Mask R-CNN则会增加一个输出，即物体掩膜(object mask)。

二、网络结构介绍

Mask R-CNN结构如下图：

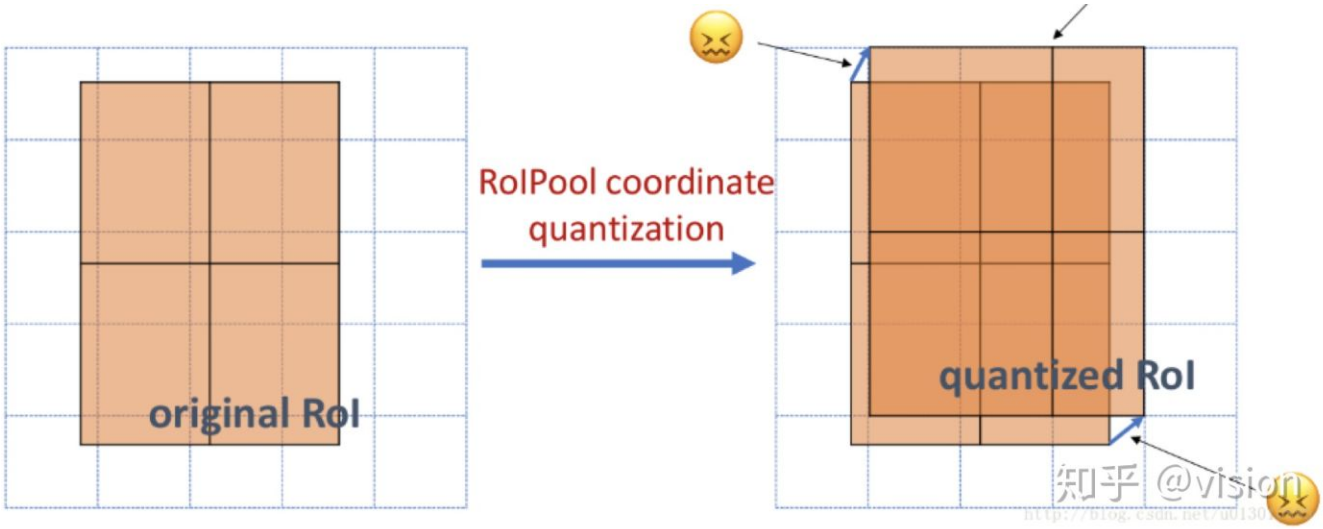


Mask R-CNN采用和Faster R-CNN相同的两个阶段，具有相同的第一层(即RPN)，第二阶段，除了预测种类和bbox回归，并且并行的对每个RoI预测了对应的二值掩膜(binary mask)。

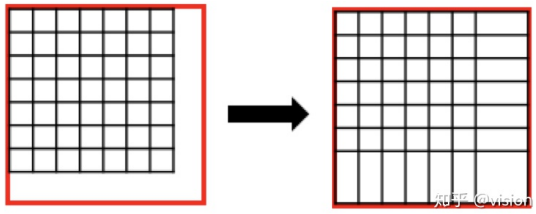
三、Mask R-CNN详细改进

1. RoIAlign

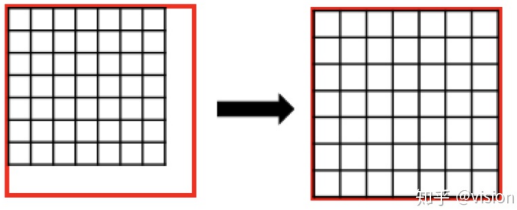
Faster R-CNN采用的RoIPooling，这样的操作可能导致feature map在原图的对应位置与真实位置有所偏差。如下图：



而通过引入RoIAlign很大程度上解决了仅通过 Pooling 直接采样带来的 Misalignment 对齐问题。

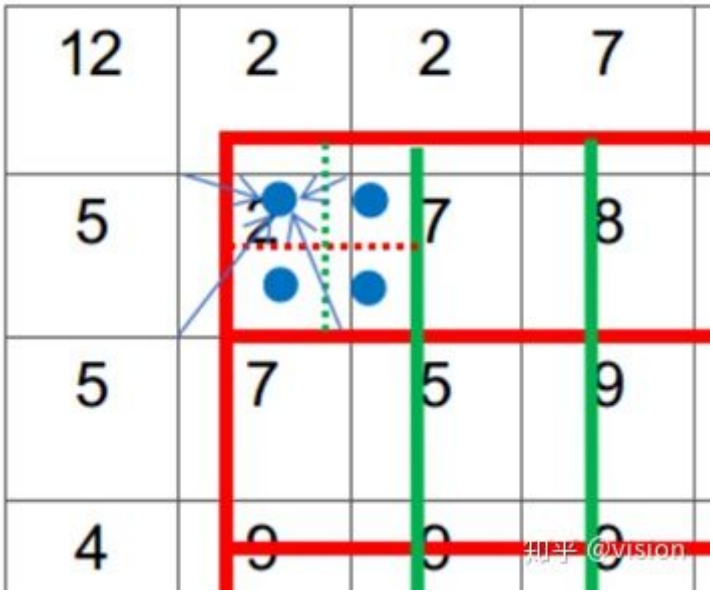


RoIPooling



RoIAlign

RoIPooling会对区域进行拉伸,导致区域形变。RoIAlign可以避免形变问题。具体方式是先通过双线性插值到14 x 14，其次进行双线性插值得到蓝点的值，最后再通过max Pooling或average pool到7 x 7。



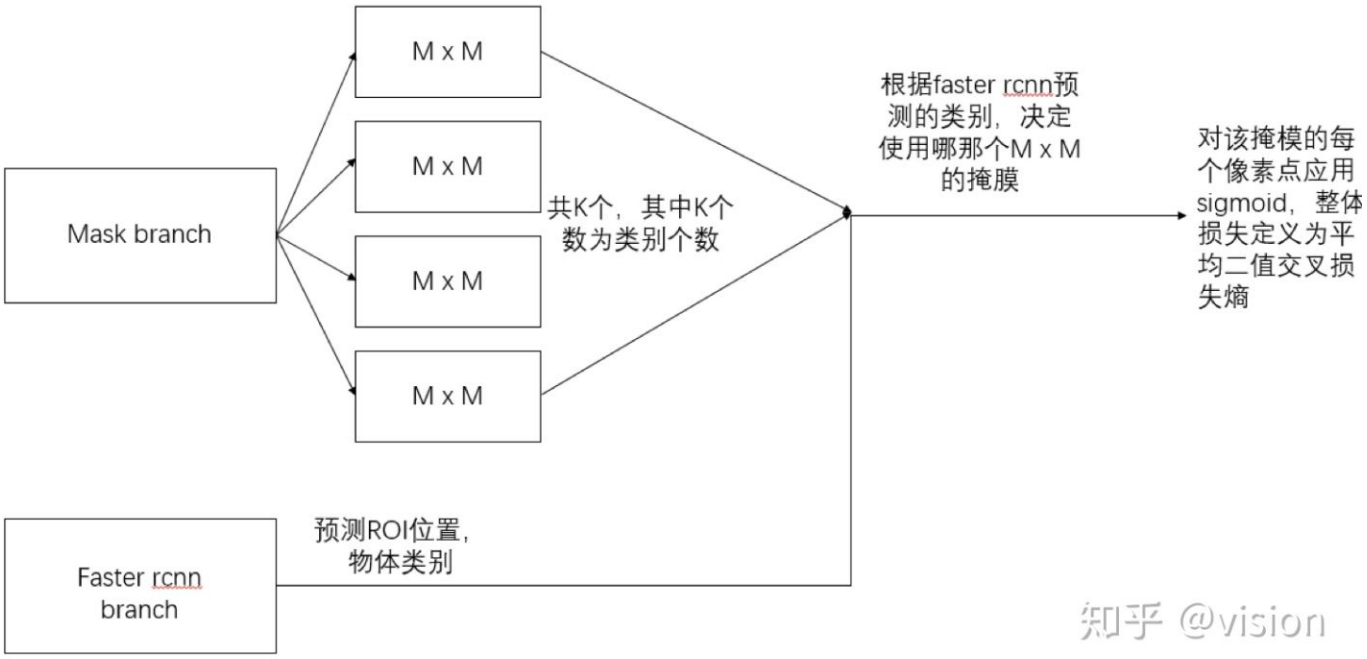
2.多任务损失函数

Mask R-CNN的损失函数可表示为：

$$L = L_{cls} + L_{box} + L_{mask}$$

其中 L_{cls} 和 L_{box} 与Faster R-CNN中的相似，所以我们具体看 L_{mask} 损失函数。

掩膜分支针对每个RoI产生一个K x M xM的输出,即K个M x M的二值的掩膜输出。其中K为分类物体的类别数目。依据预测类别输出，只输出该类对应的二值掩膜，掩膜分支的损失计算如下示意图：



1. mask branch 预测K个种类的M x M二值掩膜输出。

2. 依据种类预测分支(Faster R-CNN部分)预测结果：当前RoI的物体种类为i。

3.RoI的平均二值交叉损失熵（对每个像素点应用Sigmoid函数）即为损失 L_{mask} 。

此外作者发现使用Sigmoid优于Softmax，Sigmoid可以避免类间竞争。

算法效果：

	backbone	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{bb}_{S}	AP^{bb}_{M}	AP^{bb}_{L}
Faster R-CNN+++ [19]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [27]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [21]	Inception-ResNet-v2 [37]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [36]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
Faster R-CNN, RoIAlign	ResNet-101-FPN	37.3	59.6	40.3	19.8	40.2	48.8
Mask R-CNN	ResNet-101-FPN	38.2	60.3	41.7	20.1	40.1	50.2
Mask R-CNN	ResNeXt-101-FPN	**39.8**	**62.3**	**43.4**	**22.1**	**43.2**	51.2

体现了在COCO数据集上的表现效果。

1.2 One stage

提到one stage算法就必须提到OverFeat，OverFeat网络将分类、定位、检测功能融合在一个网络之中。随后的YOLO和SSD网络，都是很经典的one stage检测算法。

YOLO论文作者对原始YOLO网络进行了改进，提出了YOLO9000和YOLOv3。YOLO9000号称可以做到更好，更快，更强。其创新点还包括用小规模（指类别）检测标注数据集 + 大规模分类标注数据集训练通用物体检测模型。YOLOv3是作者的一个technical report，主要的工作展示作者在YOLO9000上的改进。另外本综述还将介绍新论文Object detection at 200 Frames Per Second，这篇论文在YOLO的基础上进行创新，能在不牺牲太多准确率的情况下达到200FPS（使用GTX1080）。

SSD算法是一种直接预测bounding box的坐标和类别的object detection算法，利用不同分辨率卷积层的feature map，可以针对不同scale的物体进行检测。本篇综述中主要介绍DSSD（原始作者的改进版本）和DSOD这两篇论文。

1.2.1 YOLO9000: better, faster, stronger

论文链接：arxiv.org/pdf/1612.0824

开源代码：github.com/pjreddie/dar

https://github.com/zhreshold/mxnet-YOLO **MXNet实现**

录用信息：CVPR2017

论文目标：

论文目标是要解决包含大规模物体类别的实际应用场景中的实时目标检测。实际应用场景中，目标检测应满足两个条件：1. 检测速度满足实际场景需求 2. 覆盖物体类别满足实际场景需求。实际场景包含很多类别的物体，而这些类别物体的标注数据很难拿到，本论文提出使用小规模（指类别）检测标注数据集 + 大规模分类标注数据集训练通用物体检测模型。

核心思想：

YOLO9000是在YOLO基础上的改进，相比YOLO，YOLO9000号称可以做到更好，更快，更强。下面从这三个方面介绍YOLO9000如何做到这三点。YOLO相关的论文解读可以参考：zhuanlan.zhihu.com/p/25

一、更好

准确率提升。相比R-CNN系列，YOLOv1的召回率和物体位置检测率较低，YOLO9000做了如下七点改进对其进行提升。

1.加入BN层。在所有的卷积层后加入BN操作，去掉所有dropout层。

2.使用高分辨率训练得到的分类模型pretrain检测网络。YOLOv1使用224x224训练得到的分类模型pretrain，而YOLO9000直接使用448x448训练得到的分类模型pretrain检测网络。

3.使用卷积层预测anchor box位置。YOLOv1基于输入图像的物理空间划分成7x7的网格空间，每个网格最多对应两个候选预测框，因此每张图像最多有98个bounding box，最后接入全连接层预测物体框位置。而YOLO9000移除全连接层，使用anchor box预测候选框位置，大大增加了每张图片的候选框个数。这个改进将召回率由81%提高到88%，mAP由69.5%稍微降低到69.2%。同时，由于去掉了全连接层，YOLO9000可以支持检测时不同分辨率的图像输入。

4.kmeans聚类确定候选框形状。使用k-means对训练数据集中的物体框的分辨率和比例进行聚类，确定anchor box的形状。为避免物体大小引起的统计误差，YOLO9000使用IoU而不是欧氏距离来作为距离度量方式。

5.预测‘候选框相对于图像的内部偏移’。以往RPN网络，通过回归候选框相对于当前anchor box的偏移来定位候选框的位置，由于偏移相对于anchor box外部，所以取值范围是不受限的，导致训练的时候难以收敛。因此YOLO9000采用与YOLO类似的方式，预测候选框相对于图像左上角的位置偏移，并将偏移量归一化到0-1区间，解决了训练难收敛问题。

6.使用更精细的特征。YOLOv1提取13x13的特征层进行后续物体检测，对于小物体的检测效果并不友好。为解决这个问题，YOLO9000将前一层26x26的特征与13x13层的特征进行通道concatenation。如26x26x512的feature map被拆分成13x13x2048，然后同后面的13x13特征层进行concatenation。mAP提升1%。

7.多尺度图像训练。YOLO9000采用不同分辨率的图像进行模型迭代训练，增强模型对多尺度图像的预测鲁棒性。

二、更快

YOLOv1的basenet基于GoogleNet改进得到，计算复杂度大概是VGG16的1/4，但在imagenet上224x224图像的top-5分类准确率比vgg16低2%。YOLO9000提出一个全新的basenet，号称darknet-19，包含19个卷积层和5个max pooling层，详细网络结构见论文，计算复杂度比YOLOv1进一步减少了34%，imagenet上top-5准确率提升了3.2%。

三、更强

更强是指，在满足实时性需求的前提下，能检测出的物体类别数更多，范围更大。YOLO9000提出使用词树‘wordtree’，将分类数据集和检测数据集合并，进行模型训练。反向传播时，检测样本的训练loss用于计算和更新整个网络的模型参数；而分类样本的训练loss仅用于更新与分类相关的网络层模型参数。这样以来，检测数据集训练网络学到如何检测出物体（是否是物体，位置），而分类数据集使得网络识别出物体类别。

算法效果：

下图给出了YOLOv2和对比算法的准确率和运行时间的综合性能结果。可以看出YOLOv2在保证准确率的同时，可以达到超过30fps的图像检测速度。相比SSD512和Faster R-CNN（使用ResNet），YOLOv2在准确率和运行性能上都更胜一筹（图中左边第一个蓝圈）。

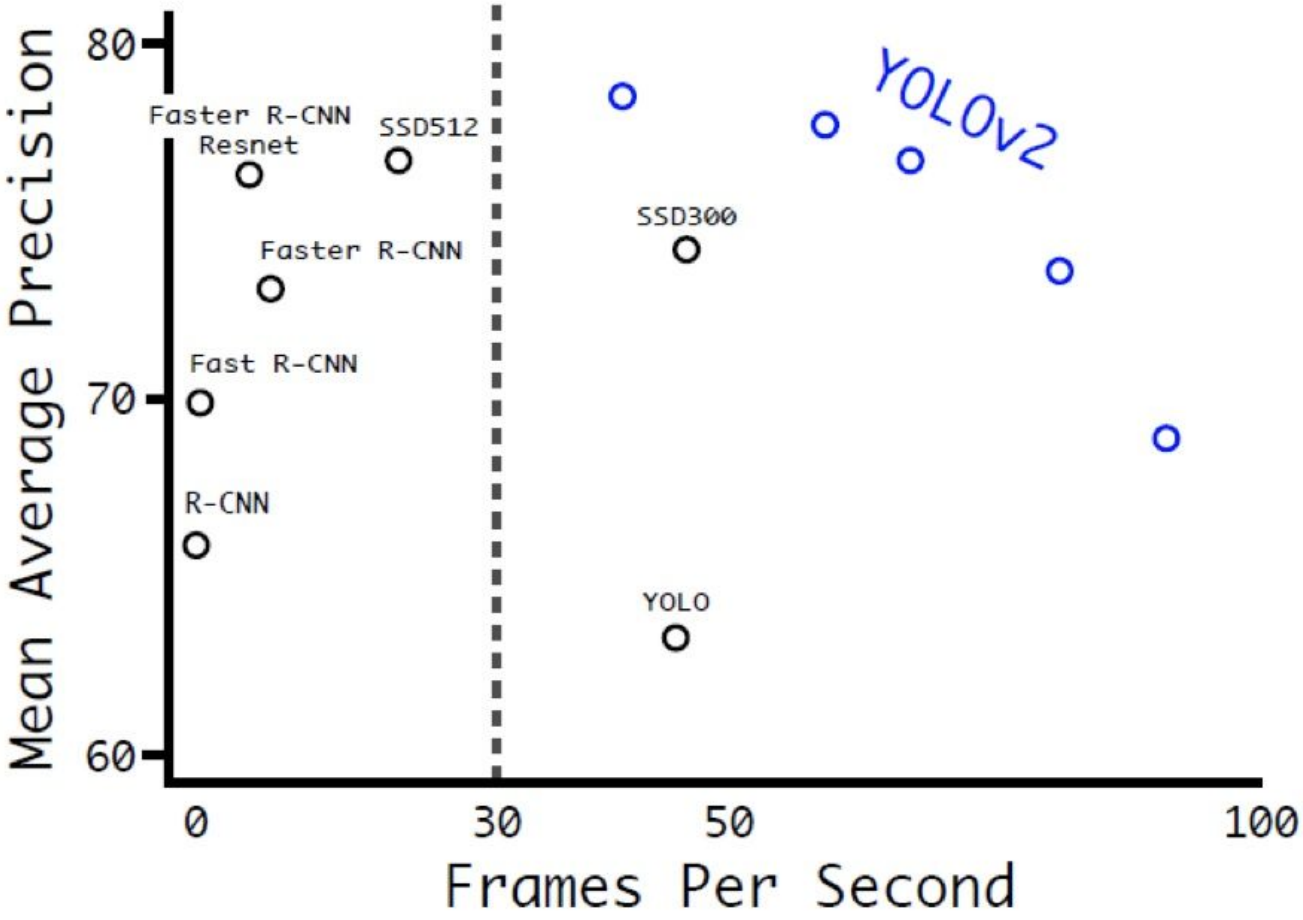


Figure 4: Accuracy and speed on VOC 2007.

1.2.2 YOLOv3: an incremental improvement

论文链接：arxiv.org/abs/1804.0276

开源代码：github.com/pjreddie/dar

录用信息：无。原文是4页technical report，2018年4月在arxiv放出。

论文目标：

保证准确率同时，更快。

核心思想：

YOLOv3对YOLO9000进行了改进，v3采用的模型比YOLO9000更大，进一步提高检测准确率，但速度比YOLO9000稍慢。相比其他检测算法，RetinaNet，SSD，DSSD等算法，YOLOv3的综合性能（准确率&速度）仍然很是最好的。但总的来说，文章的改进主要还是修修补补，换换网络，没有特别的突出创新点。具体改进如下：

1. 候选框预测时增加‘物体性’的预测，即增加对候选框 是否包含物体 的判断。这条改进借鉴Faster R-CNN的做法。区别在于，Faster R-CNN一个ground truth框可能对应多个检测候选框，而YOLO9000每个ground truth object最多对应到一个检测候选框。那么这会使得很多候选框对应不到ground truth box，这种候选框在训练时不会计算坐标或分类误差，而只会加入对‘物体性’的检测误差。

2. 多标签分类。每个候选框可以预测多个分类，使用逻辑归二分类器进行分类。

3. 多尺度预测。借鉴FPN思想，在3个尺度上进行预测，每个尺度对应3个候选框，每个候选框输出‘位置偏移‘，是否包含物体以及分类结果。YOLOv3对小物体的检测效果比YOLO9000有提升，但是对中大物体的检测准确率却有降低。文章没给出具体原因。

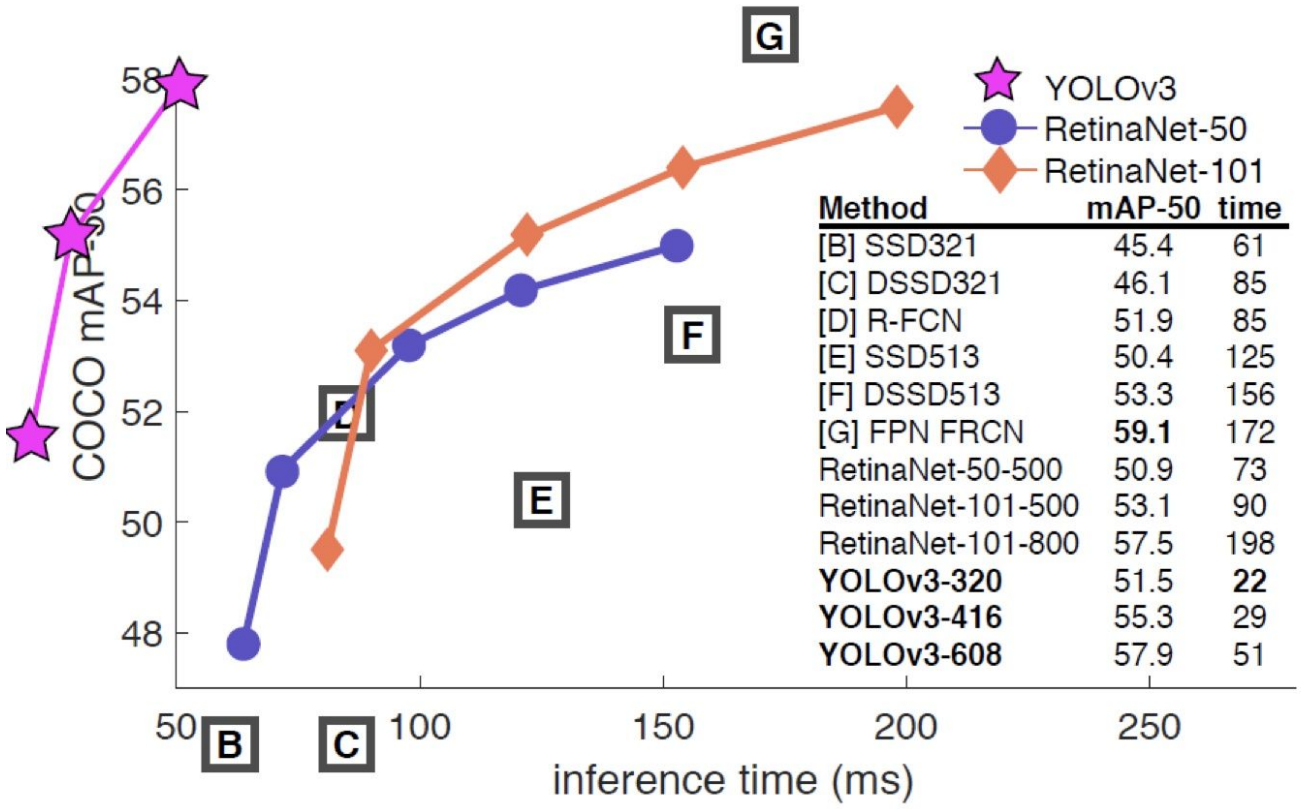
4. 提出新的basenet。YOLOv3采用一个53层卷积的网络结构，号称darknet-53，网络设计只采用3x3，1x1的卷积层，借鉴了ResNet的残差网络思想。该basenet在ImageNet上对256x256的Top-5分类准确率为93.5，与ResNet-152相同，Top-1准确率为77.2%，只比ResNet-152低0.4%。与此同时，darknet-53的计算复杂度仅为ResNet-152的75%, 实际检测速度（FPS）是ResNet-152的2倍。

5. 除以上改进外，YOLOv3还做了一些其他尝试，但效果都不理想。具体见论文，此处不列出。

算法效果：

对320x320的输入图像，YOLOv3在保证检测准确率与SSD一致（mAP=28.2）的前提下，处理每张图像的时间为22ms，比SSD快3倍。

值得注意的是，论文提出的darknet-53，是一个比ResNet152综合性能更好的分类网络。



Method	mAP-50	time
[B] SSD321	45.4	61
[C] DSSD321	46.1	85
[D] R-FCN	51.9	85
[E] SSD513	50.4	125
[F] DSSD513	53.3	156
[G] FPN FRCN	**59.1**	172
RetinaNet-50-500	50.9	73
RetinaNet-101-500	53.1	90
RetinaNet-101-800	57.5	198
YOLOv3-320	51.5	**22**
YOLOv3-416	55.3	29
YOLOv3-608	57.9	51

Figure 3. Again adapted from the [7], this time displaying speed/accuracy tradeoff on the mAP at .5 IOU metric. You can tell YOLOv3 is good because it's very high and far to the left. Can you cite your own paper? Guess who's going to try, this guy → [A].

1.2.3 Object detection at 200 Frames Per Second

论文链接：arxiv.org/abs/1805.0636

开源代码：无

录用信息：无

论文目标：

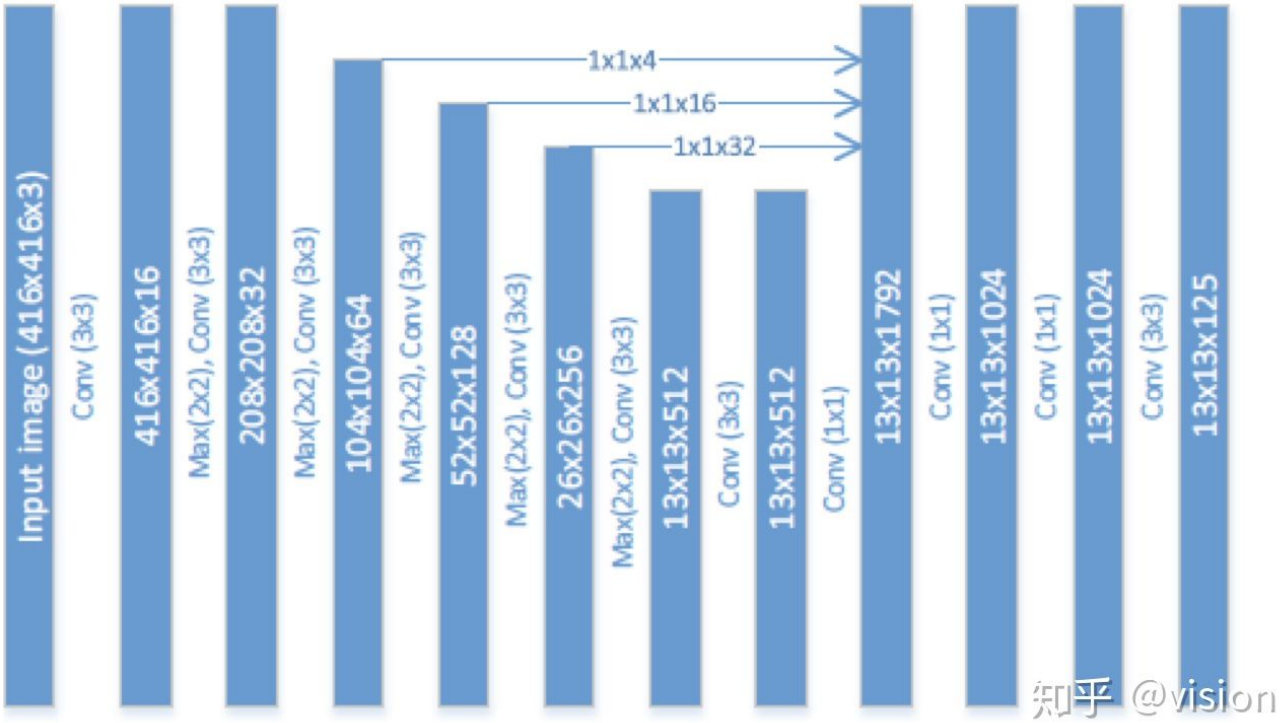
为了解决检测算法计算复杂度过高、内存占用过大的问题，本文提出了一种快而有效的方法，能够在保持高检测率的同时，达到每秒200帧的检测速度。

核心思想：

为了实现又快又强的检测目标，本文从三个方面提出了创新：网络结构、损失函数以及训练数据。在网络结构中，作者选择了一种深而窄的网络结构，并探讨了不同特征融合方式带来的影响。在损失函数设计中，作者提出了蒸馏损失函数以及FM-NMS方法以适应one-stage算法的改进。最后，作者在训练时同时使用了已标注数据和未标注数据。下面具体介绍下本文在这三方面的创新工作。

一、网络结构

一般来说，网络越深越宽，效果也会越好，但同时计算量和参数量也会随之增加。为了平衡算法的效果与速度，作者采用了一个深而窄的网络结构。示意图如下:



说明一下，本文的baseline算法是Tiny-Yolo（Yolo 9000的加速版）。

为了实现更窄，作者将卷积的通道数做了缩减，从Yolo算法的1024缩减为了512；为了实现更宽，作者在最后添加了3个1*1的卷积层。为了加深理解，建议读者结合Yolo的网络结构图，对比查看。

从上图中，我们还可以看出，作者采用了特征融合的方式，将前几层提取的特征融合到了后面层的特征图中。在融合的过程中，作者并没有采取对大尺寸特征图做max pooling然后与小尺寸特征图做融合的方式，而是采用了stacking方法，即先将大尺寸特征图进行resize然后再和小尺寸特征图做融合。具体到上图中，对104*104*64的特征图用卷积核数量为4，大小为1*1的卷积层进行压缩，得到104*104*4的特征图，然后做resize得到13*13*256的输出。

二、损失函数

蒸馏算法是模型压缩领域的一个分支。简单来说，蒸馏算法是用一个复杂网络（teacher network）学到的东西去辅助训练一个简单网络（student network）。但直接将蒸馏算法应用于one stage的Yolo算法还存在着一些困难。

困难1：对于two stage算法，在第一阶段就会去除很多背景RoI，送入检测网络的RoI相对较少，并且大部分包含object；而one stage 算法，输出中包含大量背景RoI。如果直接对输出进行学习，会导致网络过于关注背景，而忽视了前景。

鉴于此，本文作者提出objectness scaled distillation，主要考虑了teacher network中输出的objectness对损失函数的影响。作者认为只有objectness比较大的才应该对损失函数有贡献。

为了更好地理解作者的思路，我们先回顾一下Yolo算法的损失函数，如下所示：

$$L_{Yolo} = f_{obj}(o_i^{gt}, \hat{o}_i) + f_{cl}(p_i^{gt}, \hat{p}_i) + f_{bb}(b_i^{gt}, \hat{b}_i)$$

其中 $o_i^{\wedge}$ ，$p_i^{\wedge}$ ，$b_i^{\wedge}$ 分别代表student network预测的objectness、class probability以及bbox坐标， o_i^{gt} ，p_i^{gt} ，b_i^{gt} 则代表了真实值。

作者提出的蒸馏损失函数如下：

$$f_{obj}^{Comb}(o_i^{gt}, \hat{o}_i, o_i^T) = \underbrace{f_{obj}(o_i^{gt}, \hat{o}_i)}_{\text{Detection loss}} + \underbrace{\lambda_D \cdot f_{obj}(o_i^T, \hat{o}_i)}_{\text{Distillation loss}}$$

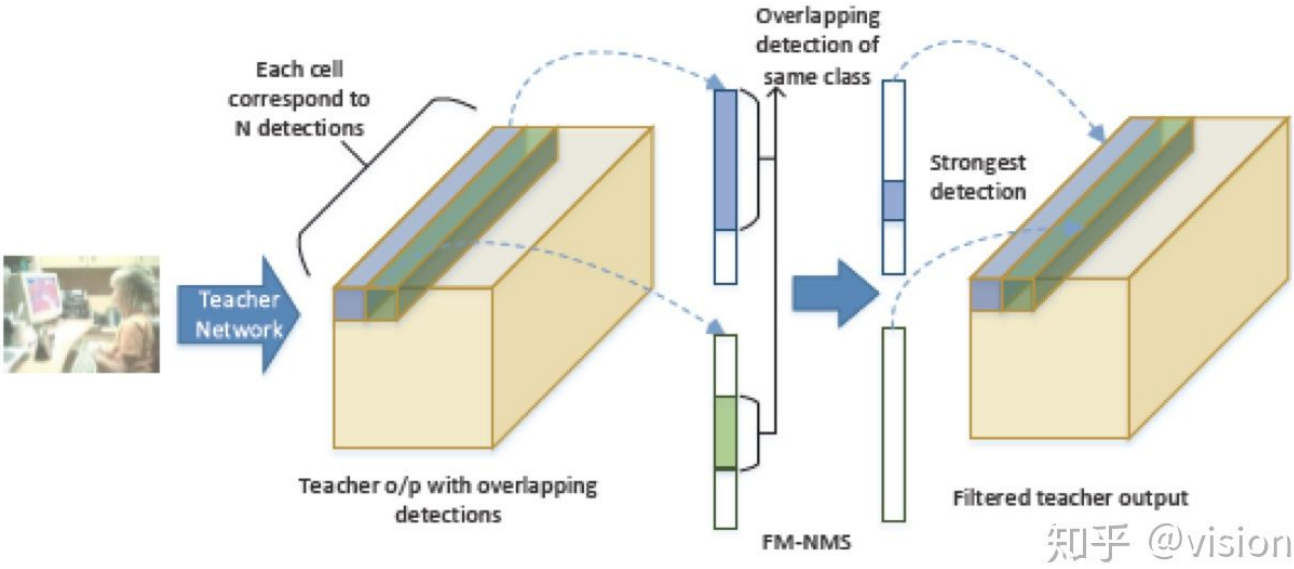
$$f_{cl}^{Comb}(p_i^{gt}, \hat{p}_i, p_i^T, \hat{o_i^T}) = f_{cl}(p_i^{gt}, \hat{p}_i) + \hat{o_i^T} \cdot \lambda_D \cdot f_{cl}(p_i^T, \hat{p}_i)$$

$$f_{bb}^{Comb}(b_i^{gt}, \hat{b}_i, b_i^T, \hat{o_i^T}) = f_{bb}(b_i^{gt}, \hat{b}_i) + \hat{o_i}^T \cdot \lambda_D \cdot f_{bb}(b_i^T, \hat{b}_i)$$

$$L_{final} = f_{bb}^{Comb}(b_i^{gt}, \hat{b}_i, b_i^T, \hat{o_i^T}) + f_{cl}^{Comb}(p_i^{gt}, \hat{p}_i, p_i^T, \hat{o_i^T}) + f_{obj}^{Comb}(o_i^{gt}, \hat{o}_i, o_i^T)$$

困难2：对于检测算法来说，如果不做NMS，直接将teacher network的预测RoI输出给student network，会因为某些box有很多的相关预测RoI而导致这些box容易过拟合。

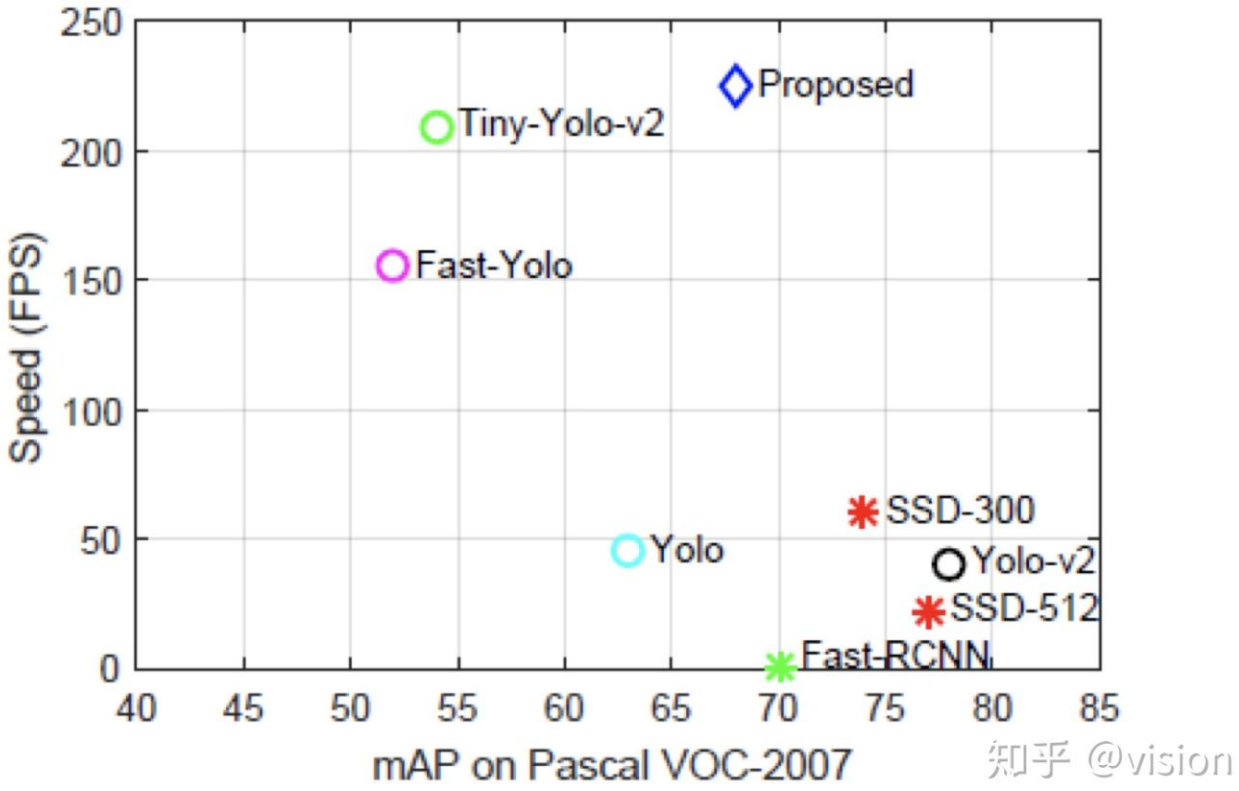
鉴于此，本文作者提出FM-NMS。取3*3区域内的相邻grid cell，对这9个grid cell中预测相同类别的bbox按照objectness 进行排序，只选择得分最高的那个bbox传给student network。2个grid cell做FM-NMS的示意图如下：



三、训练数据

鉴于作者使用了蒸馏算法，在训练时，可以非常方便地使用已标注数据和未标注数据。如果有标注数据，就使用完整的蒸馏损失函数。如果没有标注数据，就只使用蒸馏损失函数的distillation loss部分。

算法效果：



1.2.4 DSSD: Deconvolutional Single Shot Detector

论文链接：arxiv.org/pdf/1701.0665

开源代码：https://github.com/MTCloudVision/mxnet-DSSD 综述笔者实现版本

录用信息：未被会议收录

论文目标：

大小物体通吃。使用Top-Down网络结构，解决小物体检测的问题。

DSSD论文的详细解读可以参见zhuanlan.zhihu.com/p/33。

DSSD与FPN类似，都是基于Top-Down结构解决小物体检测，不同的是，如FPN的网络结构只是针对ResNet做了优化，文章中也没有提及过更换其他的基础网络的实验结果，普适度不够。DSSD作者提出一种通用的Top-Down的融合方法，使用vgg和ResNet网络将高层的语义信息融入到低层网络的特征信息中，丰富预测回归位置框和分类任务输入的多尺度特征图，以此来提高检测精度。

笔者认为，虽然Top-Down结构也许有效，但毕竟DSSD比FPN放出时间更晚一些，且在网络结构上这并没有太大创新，也许这就是本文未被会议收录的原因之一。

核心思想：

DSSD是基于SSD的改进，引入了Top-Down结构。下文分别从这两方面出发，介绍DSSD思想。

一、DSSD之于SSD

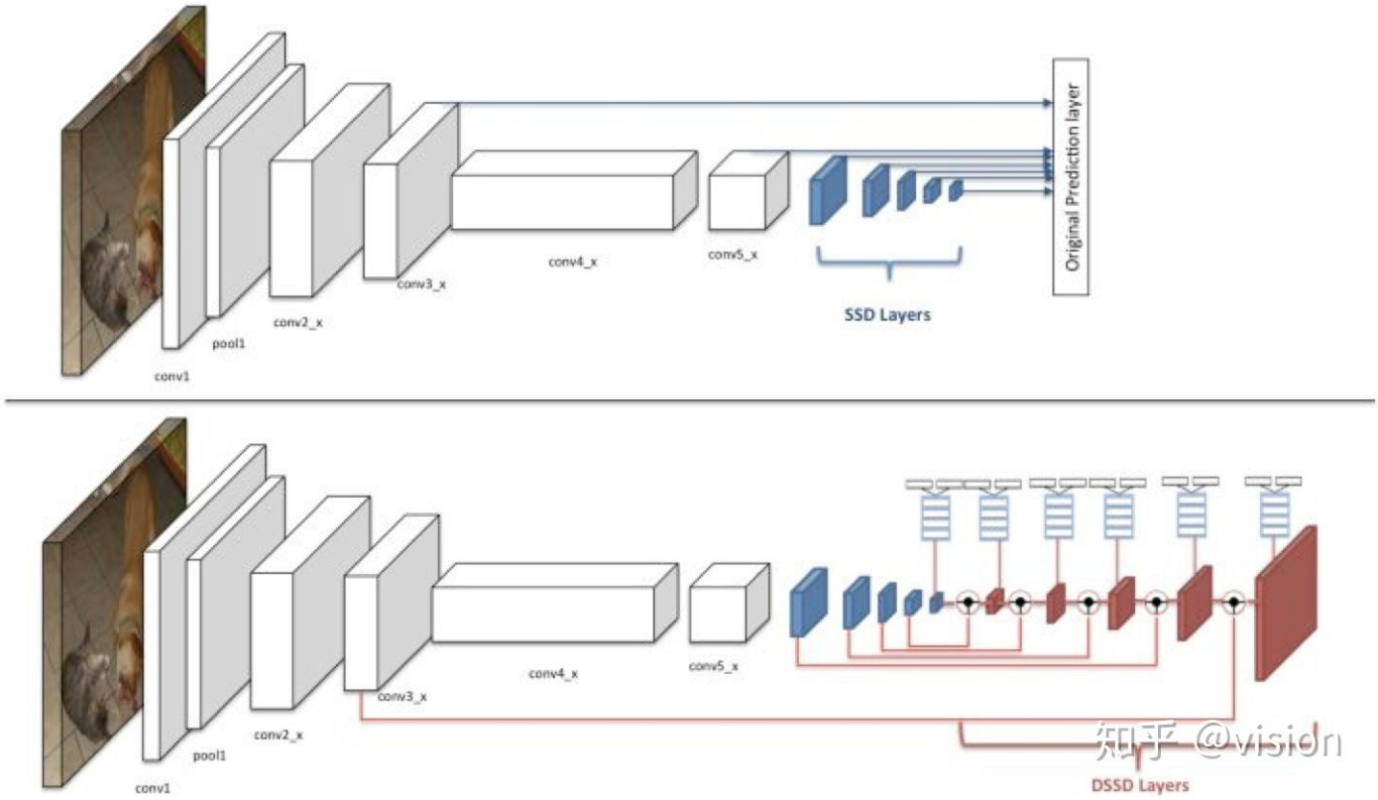
DSSD相对于 SSD算法的改进点，总结如下：

1.提出基于Top-Down的网络结构，用反卷积代替传统的双线性插值上采样。

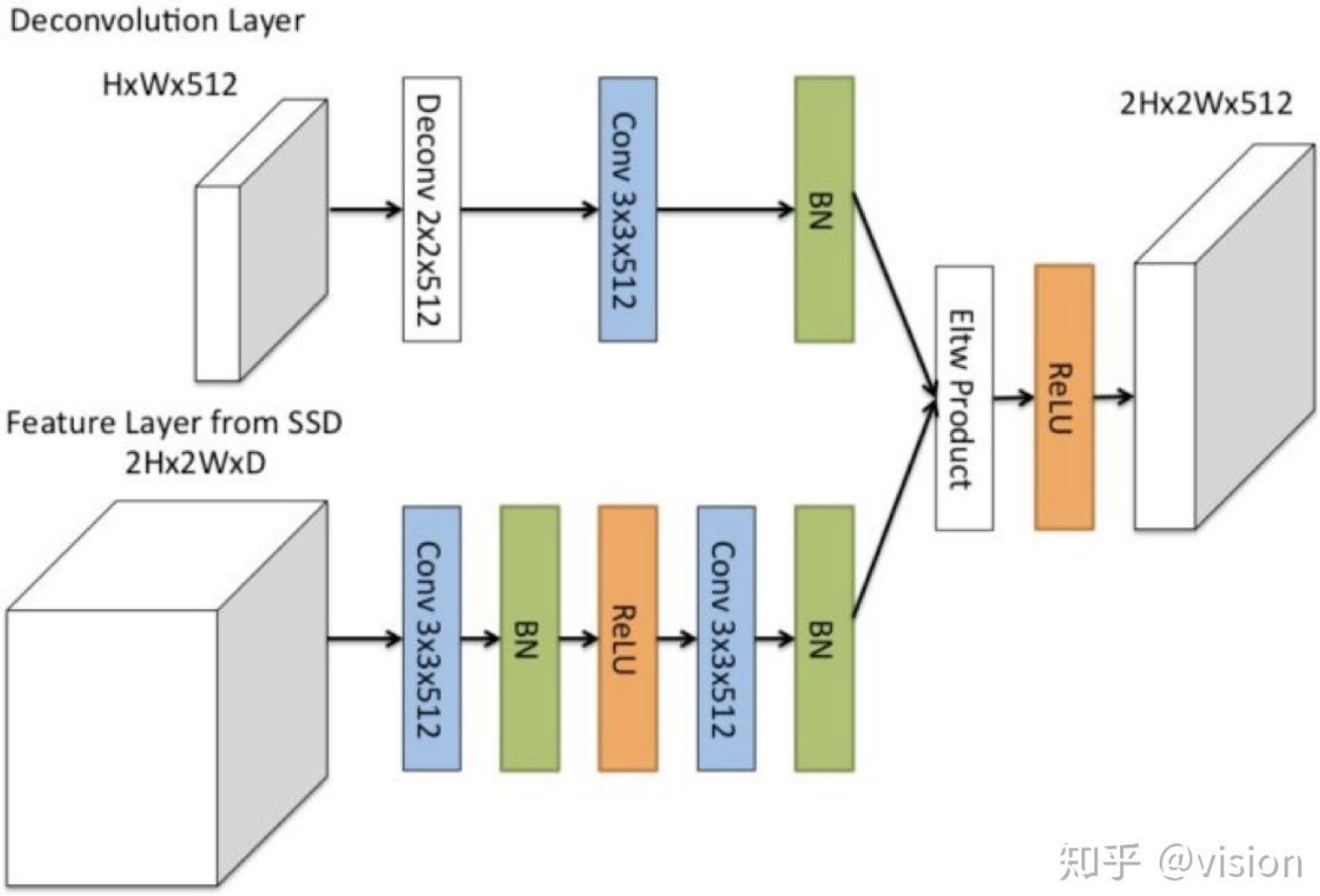
2.在预测阶段引入残差单元，优化候选框回归和分类任务输入的特征图。

3. 采用两阶段训练方法。

DSSD的网络结构与SSD对比如下图所示，以输入图像尺寸为为例，图中的上半部分为SSD-ResNet101的网络结构，conv3_x层和conv5_x层为原来的ResNet101中的卷积层，后面的五层是SSD扩展卷积层，原来的SSD算法是将这七层的特征图直接输入到预测阶段做框的回归任务和分类任务。DSSD是将这七层特征图拿出六层（去掉尺寸为的特征图）输入到反卷积模型里，输出修正的特征图金字塔，形成一个由特征图组成的沙漏结构。最后经预测模块输入给框回归任务和分类任务做预测。

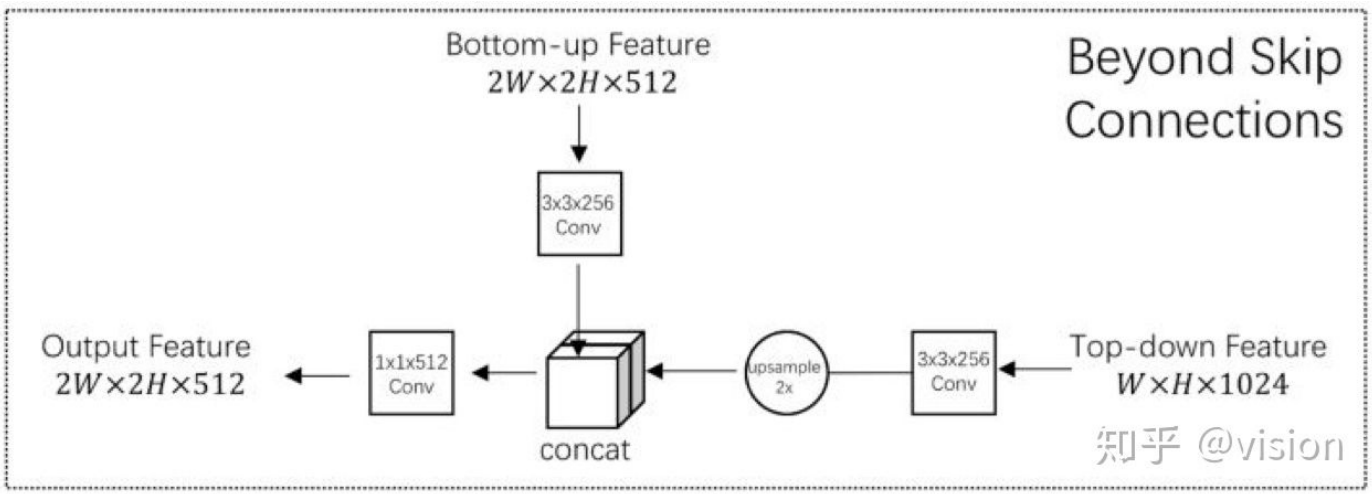


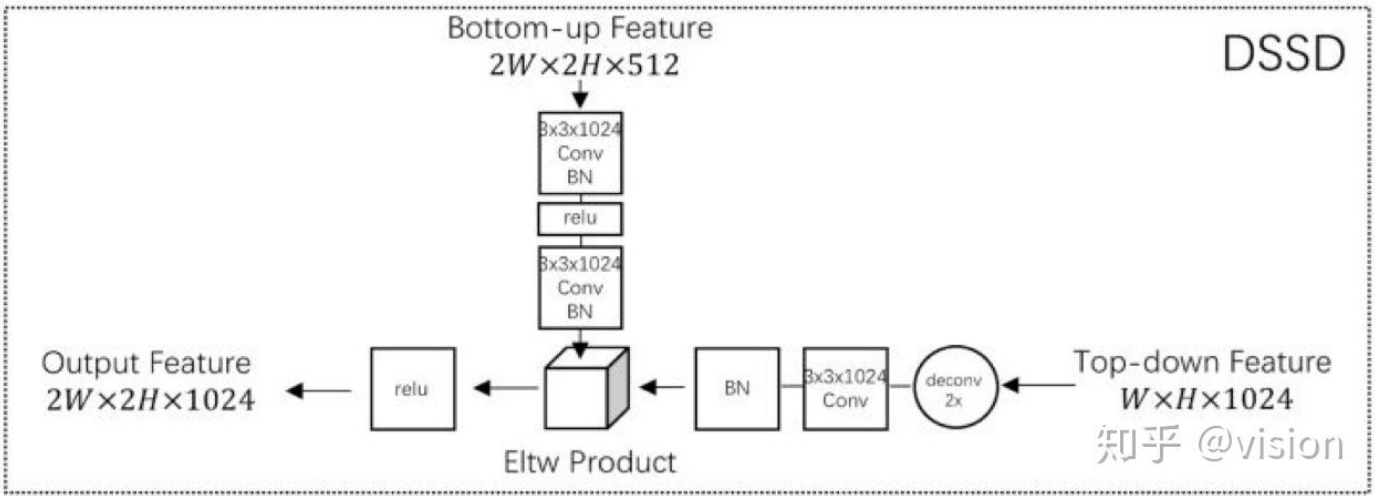
DSSD中的D，即反卷积模型，指的是DSSD中高层特征和低层特征的融合模块，其基本结构如下图所示：

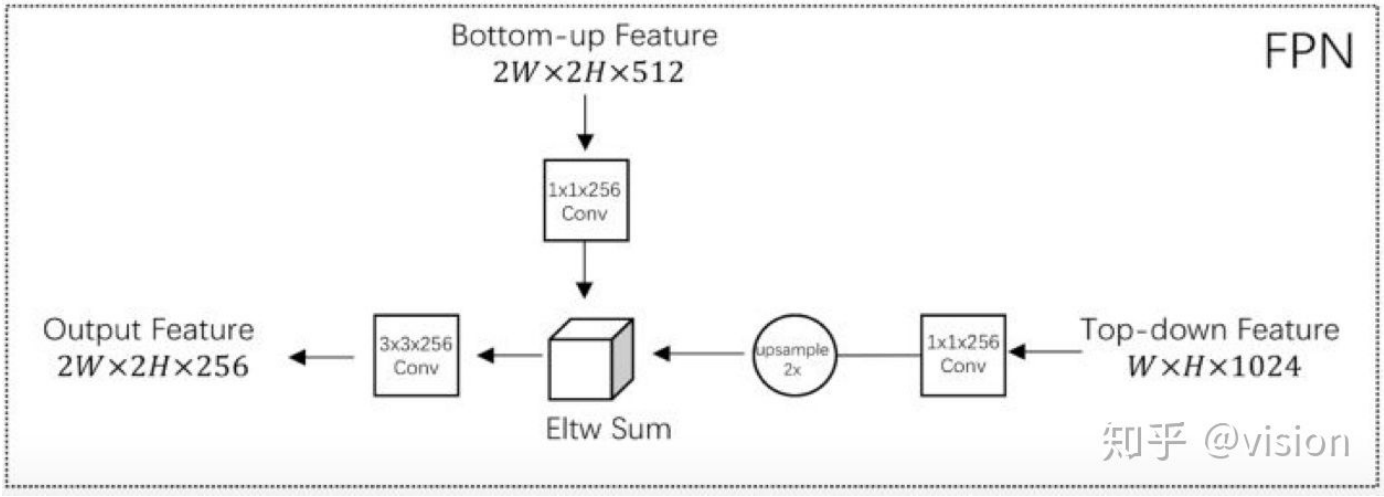


二、DSSD之于FPN和TDM

同样是采用Top-Down方式，DSSD与FPN 和TDM（这两篇论文将在本论述后文中详细介绍）的网络结构区别如下图。可以看出，TDM使用的是concat操作，让浅层和深层的特征图叠在一起。DSSD使用的是Eltw Product（也叫broadcast mul）操作，将浅层和深层的特征图在对应的信道上做乘法运算。FPN使用的是Eltw Sum（也叫broadcast add）操作，将浅层和深层的特征图在对应的信道上做加法运算。







算法效果：

DSSD 当输入为513x513的时候在VOC2007数据集赏达到了80.0%mAP。

更详细的实验复现和结果对比见zhuanlan.zhihu.com/p/33

1.2.5 DSOD : learning deeply supervised object detectors from scratch

论文链接：arxiv.org/pdf/1708.0124

开源代码：github.com/szq0214/DSOD

录用信息：ICCV2017

论文目标：

从零开始训练检测网络。DSOD旨在解决以下两个问题：

1. 是否可以从零开始训练检测模型?

2. 如果可以从零训练，什么样的设计会让网络结果更好?

DSOD是第一个不使用图像分类预训练模型进行物体检测训练初始化的检测算法。此外，DSOD网络参数只有SSD的1/2，Faster R-CNN的1/10。

核心思想：

一、从零开始训练检测任务

现有的物体检测算法如Faster R-CNN、YOLO、SSD需要使用在大规模分类数据集上训练得到的分类模型进行backbone网络初始化。比如使用ImageNet分类模型。这样做的优势在于：1.可以使用现有的模型，训练较快；2. 由于分类任务已经在百万级的图像上进行过训练，所以再用做检测需要的图片数量会相对较少。但其缺点也很明显：1.很多检测网络都是分类网络改的。图像分类网络一般都较大，检测任务可能不需要这样的网络。2.分类和检测的策略不同，可能其最佳收敛区域也不一样。3. 分类任务一般都是RGB图像训练的，但检测有可能会使用深度图像、医疗图像等其他类型的图像。导致图像空间不匹配。

为解决以上问题，DSOD提出从零开始训练检测模型。

二、网络结构

DSOD网络由backbone sub-network和front-end sub-network构成，Backbone的作用在于提取特征信息，Front-end网络是检测模块，通过对多层信息的融合用于物体检测。

基础网络Backbone sub-network部分，是一个DenseNets的变种,由一个stem block, 四个dense blocks, 两个 transition layers ,两个transition w/o pooling layers构成,用来提取特征。如下图所示:

Layers		Output Size (Input 3×300 × 300)	DSOD
Stem	Convolution	64×150×150	3×3 conv, stride 2
	Convolution	64×150×150	3×3 conv, stride 1
	Convolution	128×150×150	3×3 conv, stride 1
	Pooling	128×75×75	2×2 max pool, stride 2
Dense Block (1)		416×75×75	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)		416×75×75	1×1 conv
		416×38×38	2×2 max pool, stride 2
Dense Block (2)		800×38×38	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 8$
Transition Layer (2)		800×38×38	1×1 conv
		800×19×19	2×2 max pool, stride 2
Dense Block (3)		1184×19×19	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 8$
Transition w/o Pooling Layer (1)		1120×19×19	1×1 conv
Dense Block (4)		1568×19×19	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 8$
Transition w/o Pooling Layer (2)		1568×19×19	1×1 conv
DSOD Prediction Layers		–	Plain/Dense

Table 1: DSOD architecture (growth rate $k = 48$ in each dense block).

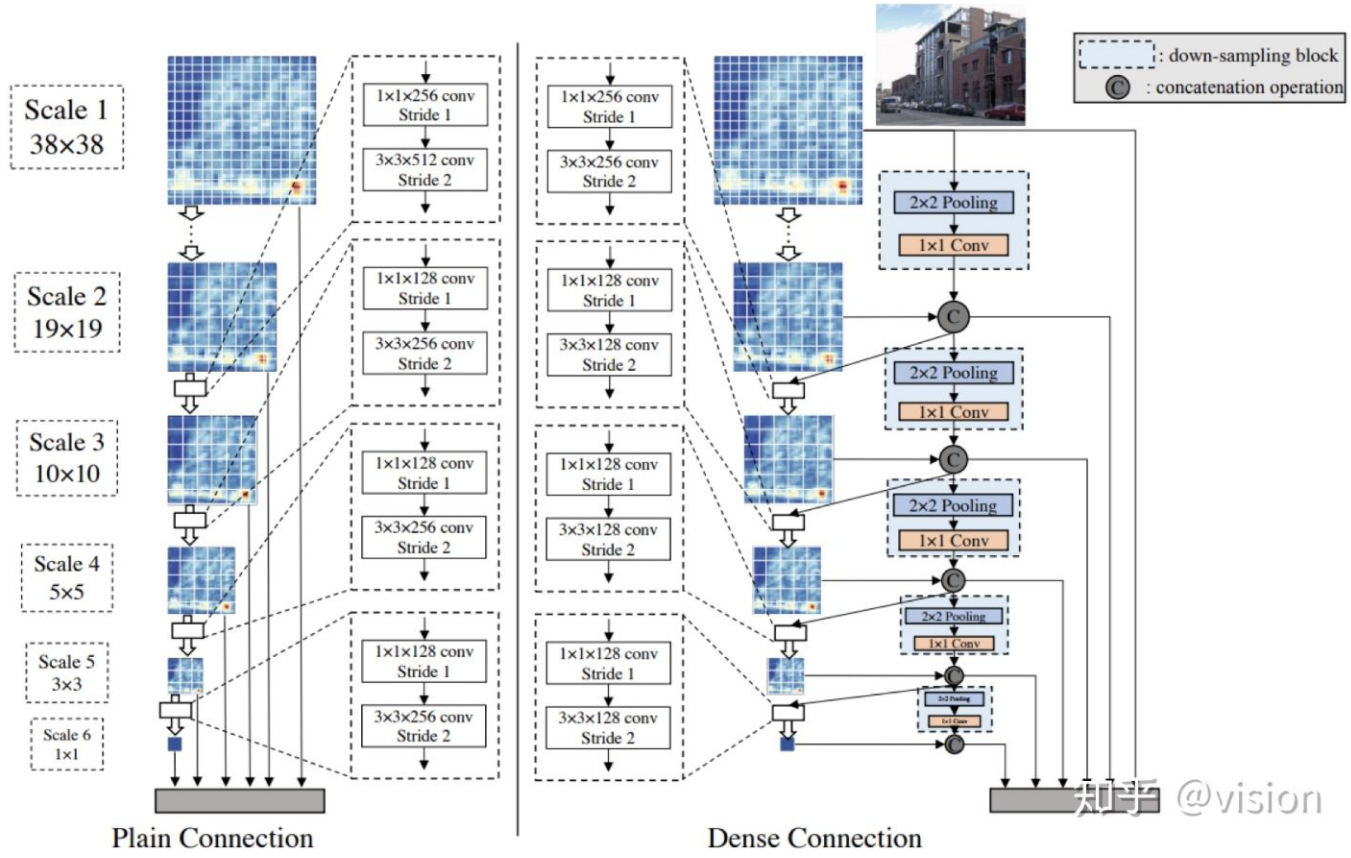
Stem block中作者没有使用DenseNet的7*7卷积，而是使用了两个3*3的卷积（这点和Inception-V3的改进很像）。作者指出这种设计可以减少从原始图像的信息损失，对检测任务更有利。其他的模块和Densenet很类似。作者使用了详细的实验证明了基础网络的设计部分的规则，如Densenet的过渡层transition layer通道数不减少、Bottleneck结构的通道更多、使用stem block而非7*7卷积对最终的识别率都是有提升的。

	DSOD300							
transition w/o pooling?		✓	✓	✓	✓	✓	✓	✓
hi-comp factor θ?			✓	✓	✓	✓	✓	✓
wide bottleneck?				✓	✓	✓	✓	✓
wide 1st conv-layer?					✓	✓	✓	✓
big growth rate?						✓	✓	✓
stem block?							✓	✓
dense pred-layers?								✓
VOC 2007 mAP	59.9	61.6	64.5	68.6	69.7	74.5	77.3	77.7

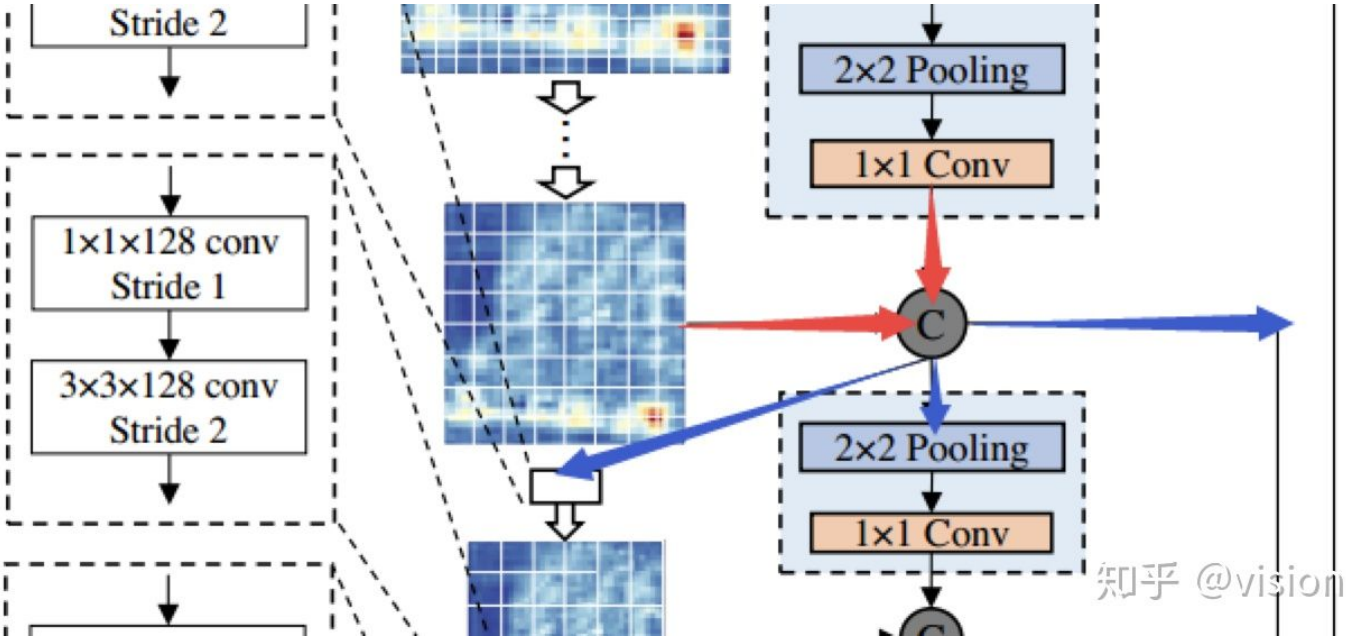
Table 2: **Effectiveness of various designs on VOC 2007 `test` set.**

经过基础特征提取后，检测的网络Front-end sub-network有两种实现方式：Plain Connection和Dense Connection。其中Plain Connection 就是SSD的特征融合方法。注意虚框中的是

Bottlenet结构，即使用1*1的卷积先降维然后再接3*3的卷积。



笔者认为Dense Connection结构就是DSOD的主要创新点，这部分也很巧妙的采用了densenet的思想，一半的Feature map由前一个scale学到，剩下的一半是直接down-sampling的高层特征。



以第一个链接结构为例，该结构的输入一半为上一层的降采样的Feature Map，其中通道的改变由1*1的卷积完成。另一半为这个尺度学习到的feature。经过Concat后的输出是三个部分，1. 经过1*1卷积和3*3卷积作为下一层的输入。2. 直接降采样并修改通道作为下一层的输入。3.输入这一层的feature到最后的检测任务。

算法效果：

DSOD的检测速度(17.4fps)比SSD、YOLO2略差，但在模型准确率和模型大小方面却更胜一筹，最小的网络只有5.9M，同时mAP也能达73.6%。作者在实验部分还使用了pre-trained model 初始化DSOD，结果反而没有从零开始训练效果好,未来可能去探究一下。