常用目标检测算法对比

1、目标检测算法简述

目标检测（object detection）：指的是发现并且检测出图像中的物体，目标检测一般包含两个任务

- 物体识别：确定图像中物体的类别；
- 位置确定：确定图像中物体的候选框架位置（一般为矩形框）。

目标检测从某种程度上就是两个任务的集成，则最简单的实现思路也就是将上述两个单任务算法进行集成，也就提出了 R-CNN 算法，在 R-CNN 的基础上进行不同程度的优化，演化出 fast-RCNN。之后在不断的速度和精度的优化上，出现了 faster-RCNN、yolo 以及 SSD 算法。

2、two-stage model

2.1 R-CNN

RCNN，在这个算法中神经网络实际上就是一个特征提取器，作者用 selective search 的方法提取了一定数量（2000 个）region proposal，然后对 region proposal 做卷积操作，将 fc7 这一层的特征提取出来用于分类和坐标回归，这里分类用的还不是 softmax 而是 SVM。这个算法的贡献主要是提出了一种有效的特征利用方式，后续很多人在工程实践中都是用的 fc7 层的特征来做基于 faster RCNN 的应用。

算法步骤:

1. 获取输入的原始图片。
2. 使用选择性搜索算法（selective search）评估相邻图像之间的相似度，把相似度高的进行合并，并对合并后的区块打分，选出感兴趣区域的候选框，也就是子图。这一步大约需要选出 2000 个子图。
3. 分别对子图使用卷积神经网络（Alexnet、ZFnet、VGG），进行卷积-relu-池化以及全连接等步骤，提取特征。这一步基本就是物体识别的范畴了。
4. 对提取的特征进行物体分类（svm），保留分类准确率高的区块（非极大值抑制），以作为最终的物体定位区块。
5. 使用回归器精细修正候选框位置。

创新点：

- 借助 CNN 良好的特征提取和分类性能，通过 Region Proposal 方法实现目标检测问题的转化。

存在的问题：

- 每一个候选框依次进入 CNN 网络，存在冗余提取特征和冗余存储的问题。
- 候选框由传统的 selective search 算法完成，速度比较慢。
- 各个模块独立训练。无法实现 end-to-end。

2.2 SPP-Net

创新点：

- 在 R-CNN 的基础上做出改进，将 Region Proposal 的位置信息放在卷积层之后，这样使得图像可以在一次计算的基础上整体提取特征，减少 RCNN 带来的最大问题——冗余计算和冗余存储
- 加入了金字塔池化，不在使用裁剪和缩放归一化图片大小

存在的问题：

- 和 RCNN 一样，训练过程相互之间仍然给是分离的，候选框提取、卷积特征、SVM 分类、以及最后的 bounding box 回归都是独立训练。

◆ 依然使用 ss 算法生成 region proposal，耗时较长
◆ 由于加入了金字塔池化，卷积层不能 fine tune，不能反向传播

2.3 fast-RCNN

fast RCNN 将除 region proposal 提取以外的部分都用一个网络来实现，与 RCNN 不同的是：

1. 提出多任务 Loss，加入候选框映射功能，他的分类和坐标回归的 loss 一起通过反向传播来更新网络参数。
2. 它在提取 feature 时并不会把每个 region proposal 都放入提取，而是将整幅图提取特征后，用坐标映射的方式提取 feature，这样有两个好处：
   - 快，因为一张图片只走一次网络；
   - feature 的特征受感受野的影响，能融合相邻的背景的特征，这样“看”得更远一些。
3. 提出 ROI pooling 池化层结构，解决了候选框子图必须将图像裁剪缩放到相同尺寸大小的问题。由于 CNN 网络的输入图像尺寸必须是固定的某一个大小（否则全连接时没法计算），故 R-CNN 中对大小形状不同的候选框，进行了裁剪和缩放，使得他们达到相同的尺寸。这个操作既浪费时间，又容易导致图像信息丢失和形变。fast R-CNN 在全连接层之前插入了 ROI pooling 层，从而不需要对图像进行裁剪，很好的解决了这个问题。

算法步骤:

❶ 获取输入的原始图片。
❷ 使用选择性搜索算法（selective search）评估相邻图像之间的相似度，把相似度高的进行合并，并对合并后的区块打分，选出感兴趣区域的候选框，也就是子图。这一步大约需要选出 2000 个子图。
❸ 分别对子图使用卷积神经网络（Alexnet、ZFnet、VGG），进行卷积-relu-池化以及全连接等步骤，提取特征。
❹ 对最后一个卷积层的特征进行候选框映射功能，并用 ROI 池化候选框。
❺ 对提取的特征进行多任务损失函数回归和分类，保留分类准确率高的区块（非极大值抑制），以作为最终的物体定位区块。

创新点：

● 借鉴了 SPP 的思路，在不用图像金字塔池化的方式下，提出简化版的 ROI 池化层（单层 sppnet 的网络层）。
● 通过加入候选框映射功能，使网络能够反向传播。同时提出多任务 Loss，实现了 training and testing end-to-en。

存在问题：

◆ 提取候选框依然十分耗时，这个过程十分耗时，制约 fast-RCNN 的速度瓶颈

2.4 faster-RCNN

作者发现 selective search 的方法导致算法没有实时性的可能，因此，作者尝试用 region proposal network 来取代 selective search 的方法，并且与 fast RCNN 的分类和回归网络共用特征提取层，因此这样并不会带来太多额外的计算量，而实验结果也表明了，作者这样做确实提高的速度，并且还提高了准确率。因此，综上所述，region proposal network 是 faster RCNN 的精华所在，也是精度高于以及速度慢于后续 YOLO 和 SSD 算法的原因。

算法步骤:

❶ 卷积层。原始图片先经过 conv-relu-pooling 的多层卷积神经网络，提取出特征图。供后续的 RPN 网络和全连接层使用。faster R-CNN 不像 R-CNN 需要对每个子图进行卷积层特征提取，它只需要对全图进行一次提取就可以了，从而大大减小了计算时间。
❷ RPN 层，region proposal networks。RPN 层用于生成候选框，并利用 softmax 判断候选框是前景还是背景，从中选取前景候选框（因为物体一般在前景中），并利用 bounding

box regression 调整候选框的位置，从而得到特征子图，称为 proposals。

❸ ROI 层，fast R-CNN 中已经讲过了 ROI 层了，它将大小尺寸不同的 proposal 池化成相同的大小，然后送入后续的全连接层进行物体分类和位置调整回归

❹ 分类层。利用 ROI 层输出的特征图 proposal，判断 proposal 的类别，同时再次对 bounding box 进行 regression 从而得到精确的形状和位置。

创新点：

● 创新性的提出候选框提取不一定非要在原图上做，可以考虑在特征图上做。继而提出了 RPN（RegionProposal Network），使得其可以抛弃传统的 Region proposal 的方法，大幅加快训练速度。

存在问题：

◆ Faster 实现了端到端的检测，并且几乎达到了效果上的最优，速度方向的改进仍有余地（region proposal 耗时严重）。

2.5 R-FCNN

之前的网络（如 R-CNN，Fast R-CNN 等）利用 region proposal 结合深度网络用来进行目标检测是有效的。但是**在 R-CNN 中计算那些关于 crop 裁剪和 warp 变换的区域的计算是不共享的**。而 SPP-Net，Fast R-CNN 和 Faster R-CNN 是 “半卷积”的，在卷积网络中是计算共享的，但是在另一个子网络是各自计算独立的区域。

综上所示，目前来说，对于物体检测可以通过 ROI Pooling layer 分成两个方法：

◆ **独立于 ROI 的共享的、全卷积的子网络（a shared, “fully convolutional” subnetwork independent of RoIs）**

◆ **不共享计算的 ROI-wise 子网络(an RoI-wise subnetwork that does not share computation)**

造成这两种不同来源于之前提出的分类网络结构，比如说 AlexNet，VGG 等。它们由两个子网络组成，一个是以 spatial pooling layer（空间池化层）结束的卷积子网，另一个是若干个 fully-connected layers（全卷积层）。因此，在目标检测任务中，spatial pooling layer 很自然的也就转换成了 ROI pooling layer。

但是目前最好的图像分类网络，例如 ResNet 和 GoogleNets 都是使用 fully convolution 设计的，因此，类似的，很自然的可以想到通过使用全卷积在目标检测结构中构建一个共享计算的，卷积子网络，而不将 ROI-Wise 子网络作为 hidden layer(隐藏层)。然而，通过对这个想法的实验可以得出，使用这种方法构建的目标检测网络的**准确率与其分类的最优准确率有着很大的差距**。为了解决这个问题，**在 ResNet paper 中作者将 Faster R-CNN 检测方法中的 ROI pooling layer 很不自然的插入在两个卷积层中间，这样便通过一个更深的 ROI-wise 子网络来提升了检测的准确率，但是其代价是由于在每个 ROI 中未共享计算从而降低了检测速度。**

作者发现，前面这种“不自然”的设计是由于在**分类任务中的“平移不变性”和目标检测任务中要求的“平移转变性”的矛盾**。具体来说，一方面，在图像级别的分类任务中更加倾向于 translation invariance（平移不变性），也就是在一张图像中的**目标的移动应该不影响最终的分类结果。因此，深层（全）卷积网络结构更加倾向于保持整个结构的这种特性（平移不变性）**；但是另一方面，目标检测任务需要**定位一张图像中目标的具体位置，也就是需要存在 translation-variant (平移转变性)**。为了解决这个问题，在 ResNet paper 在目标检测流程中在两个卷积层之间插入了 ROI pooling layer。这种操作打破了平移不变形（？？？？？），以此 post-ROI 在评价不同 region 时不再保持平移不变性。然而，这种引入平移转变性的设计是以牺牲训练和测试效率为代价的，因为它引入了非常大量的 region-wise layers。

因此在本文中，提出了一个新的目标检测结构，也就是 R-FCN（ Region-based Fully

Convolutional Network），我们的网络由共享的全卷积结构组成，就像 FCN 网络一样。为了将上面提到的“平移转变性”融合到 FCN 中，我们通过在 FCN 的输出上使用一系列的特定卷积层构建了一系列的 position-sensitive score map（位置敏感得分图）。这些 位置敏感得分图 中的每一个都对相对空间位置对位置信息进行编码。在这个 FCN 的顶层，我们添加了一个位置敏感（position-sensitive） ROI pooling 层，以此用来管理从这些得分图得到的信息，并且这一层是没有任何附加权重的。可以看出 **整个结构是端到端(end-to-end)的**。所有需要学习的参数（权重）都是对于整张图像而言并且都是共享的，此外还对空间位置进行编码用来目标检测。

创新点：

- 提出 Position-sensitive score maps 来解决目标检测的位置敏感性问题。
- 区域为基础的，全卷积网络的二阶段目标检测框架。
- 比 Faster-RCNN 快 2.5-20 倍（在 K40 GPU 上面使用 ResNet-101 网络可以达到 0.17 sec/image）。

2.6 对比图

	使用方法	缺点	改进
R-CNN (Region-based Convolutional Neural Networks)	1、SS提取RP； 2、CNN提取特征； 3、SVM分类； 4、BB盒回归。	1、 训练步骤繁琐（微调网络+训练SVM+训练bbox）； 2、 训练、测试均速度慢 ； 3、 训练占空间	1、 从DPM HSC的34.3%直接提升到了66%（mAP）； 2、 引入RP+CNN
Fast R-CNN (Fast Region-based Convolutional Neural Networks)	1、SS提取RP； 2、CNN提取特征； 3、softmax分类； 4、多任务损失函数边框回归。	1、 依旧用SS提取RP(耗时2-3s，特征提取耗时0.32s)； 2、 无法满足实时应用，没有真正实现端到端训练测试； 3、 利用了GPU，但是区域建议方法是在CPU上实现的。	1、 由66.9%提升到70%； 2、 每张图像耗时约为3s。
Faster R-CNN (Fast Region-based Convolutional Neural Networks)	1、RPN提取RP； 2、CNN提取特征； 3、softmax分类； 4、多任务损失函数边框回归。	1、 还是无法达到实时检测目标； 2、 获取region proposal，再对每个proposal分类计算量还是比较大。	1、 提高了检测精度和速度； 2、 真正实现端到端的目标检测框架； 3、 生成建议框仅需约10ms。

3、one-stage model

3.1 YOLO

针对于 two-stage 目标检测算法普遍存在的运算速度慢的缺点，yolo 创造性的提出了 one-stage。也就是将物体分类和物体定位在一个步骤中完成。不再采用 two-stage 的这种思路，将图像分类和位置检测完全集成在一个网络里，将边框位置设计称可以回归的参数，直接由网络回归得出。速度极大的得到提升，但是检测精度跟 faster-RCNN 相比，降低了不少。不再显式提取候选框

3.1.1 YOLO v1

YOLO 的一个贡献是将检测问题转化为了回归问题，相信这句话很多人见过很多次了。那到底是什么意思呢？指的就是之前 faster RCNN 是先分两步，先提取 region proposal，也就是判断是前景还是背景的问题，之后再分类，具体看前景是什么东西。而 YOLO 直接通过 regression 一次既产生坐标，又产生每种类别的概率。

YOLO 的特点在于快，其中一方面来源于 regression 机制，还有一个原因就在于 region proposal 的提取过程了。再 YOLO 中很少提 region proposal 的概念，但是为了类比 faster RCNN 我们可以这样理解，YOLO 中粗暴地分成了 7X7 的网格，每个位置默认可能属于 2 个 object，那么事实上就是提取了 98 个 region proposal，而 faster RCNN 是一种滑动窗口机制，每个

feature map 上都回归出 9 个 anchor，大约一共 20k 个 anchor，在通过非极大值抑制等方法最终会得到 300 个 region proposal。两者之间候选框差别巨大，因此，faster RCNN 会准一点也是情理之中，而既然每个位置都要精修，当然效率就会低很多，也就不能满足实时性要求了。另外，YOLO 精简了网络，比 VGG 要稍微计算量小一些，可能也会加快一些速度，但这些计算量比起前面提到的两点已经不足为道。

创新点：

- yolo 算法开创了 one-stage 检测的先河，它将物体分类和物体检测网络合二为一，都在全连接层完成。故它大大降低了目标检测的耗时，提高了实时性。
- 改革了区域建议框式检测框架：RCNN 系列均需要生成建议框，在建议框上进行分类与回归，但建议框之间有重叠，这会带来很多重复工作。

存在问题：

- 每个网格只对应两个 bounding box，当物体的长宽比不常见（也就是训练数据集覆盖不到时），效果很差。
- 原始图片只划分为 7x7 的网格，当两个物体靠的很近时，效果很差
- 最终每个网格只对应一个类别，容易出现漏检（物体没有被识别到）。
- 对于图片中比较小的物体，效果很差，由于原理上的限制，YOLO 仅检测最后一层卷积输出层，小物体像素少，经过层层卷积，在这一层上的信息几乎体现不出来，导致难以识别。
- 因为存在全连接层，输入图片大小固定。

3.1.2 YOLO v2

针对 **yolo 准确率不高，容易漏检(低召回率)，对长宽比不常见物体效果差等问题**，结合 SSD 的特点，提出了 yoloV2。它主要还是采用了 yolo 的网络结构，在其基础上做了一些优化和改进，如下:

- 网络采用 DarkNet-19：19 层，里面包含了大量 3x3 卷积，同时借鉴 inceptionV1，加入 1x1 卷积核全局平均池化层。
- 去掉全连接层：和 SSD 一样，模型中只包含卷积和平均池化层（平均池化是为了变为一维向量，做 softmax 分类）。这样做一方面**是由于物体检测中的目标，只是图片中的一个区块，它是局部感受野，没必要做全连接。二是为了输入不同尺寸的图片，如果采用全连接，则只能输入固定大小图片了。**
- batch normalization：卷积层后加入 BN，对下一次卷积输入的数据做归一化。可以在增大学习率的前提下，同样可以稳定落入局部最优解。从而加速训练收敛，在相同耗时下，增大了有效迭代次数。
- 使用 anchors：**yolo v1 使用全连接层数据进行 bounding box 预测（要把 1470*1 的全链接层 reshape 为 7*7*30 的最终特征），这会丢失较多的空间信息定位不准。**YOLOv2 借鉴了 Faster R-CNN 中的 anchor 思想： 简单理解为卷积特征图上进行滑窗采样，每个中心预测 9 种不同大小和比例的框。**由于都是卷积不需要 reshape，很好的保留的空间信息，最终特征图的每个特征点和原图的每个 cell 一一对应。**而且用预测相对偏移（offset）取代直接预测坐标简化了问题，方便网络学习。
- pass through layer：yolo 原本最终特征图为 13x13x256。yoloV2 还利用了之前的 26x26 的特征图进行目标检测。26x26x256 的 feature map 分别按行和列隔点采样，得到 4 幅 13x13x256 的 feature map，将他们组织成一幅 13x13x2048 的 feature map。这样做的目的是提高小物体的识别率。因为越靠前的卷积，其感受野越小，越有利于小物体的识别。
- 高分辨率输入 Training：yolo 采用 224x224 图片进行预训练，而 yoloV2 则采用 448x448
- Multi-Scale Training：输入不同尺寸的图片，迭代 10 次，就改变输入图片尺寸。由于模

型中去掉了全连接层，故可以输入不同尺寸的图片了。从 320x320，到 608x608

存在问题：

- yolo v1 和 yolo v2 只能识别 20 类物体
- 对小物体检测不理想

3.1.2 YOLO 9000

yolo 和 yoloV2 只能识别 20 类物体，为了优化这个问题，提出了 yolo9000，可以识别 9000 类物体。它在 yoloV2 基础上，进行了 imageNet 和 coco 的联合训练。这种方式充分利用 imageNet 可以识别 1000 类物体和 coco 可以进行目标位置检测的优点。当使用 imageNet 训练时，只更新物体分类相关的参数。而使用 coco 时，则更新全部所有参数。

3.1.3 YOLO v3

YOLOv3 的先验检测（Prior detection）系统将分类器或定位器重新用于执行检测任务。他们将模型应用于图像的多个位置和尺度。而那些评分较高的区域就可以视为检测结果。此外，相对于其它目标检测方法，我们使用了完全不同的方法。我们将一个单神经网络应用于整张图像，该网络将图像划分为不同的区域，因而预测每一块区域的边界框和概率，这些边界框会通过预测的概率加权。我们的模型相比于基于分类器的系统有一些优势。它在测试时会查看整个图像，所以它的预测利用了图像中的全局信息。与需要数千张单一目标图像的 R-CNN 不同，它通过单一网络评估进行预测。这令 YOLOv3 非常快，一般它比 R-CNN 快 1000 倍、比 Fast R-CNN 快 100 倍。

改进之处：

- **anchor bbox prior 不同：**v2 作者用了 5 个 anchor，一个折衷的选择，所以 v3 用了 9 个 anchor，提高了 IOU。选择 9 个簇以及 3 个尺度，然后将这 9 个簇均匀的分布在这几个尺度上。
- **detection 的策略不同：多尺度预测，**v2 只有一个 detection，原来的 YOLO v2 有一个层叫：passthrough layer，假设最后提取的 feature map 的 size 是 13*13，那么这个层的作用就是将前面一层的 26*26 的 feature map 和本层的 13*13 的 feature map 进行连接，有点像 ResNet。当时这么操作也是为了加强 YOLO 算法对小目标检测的精确度。v3 一下变成了 3 个，分别是一个下采样的，feature map 为 13*13，还有 2 个上采样的 eltwise sum，feature map 为 26*26，52*52，也就是说 v3 的 416 版本已经用到了 52 的 feature map，而 v2 把多尺度考虑到训练的 data 采样上，最后也只是用到了 13 的 feature map，这应该是对小目标影响最大的地方。
- **backbone 不同：**这和上一点是有关系的，v2 的 darknet-19 变成了 v3 的 darknet-53，为啥呢？就是需要上采样啊，卷积层的数量自然就多了，另外作者还是用了一连串的 3*3、1*1 卷积，**3*3 的卷积增加 channel，而 1*1 的卷积在于压缩 3*3 卷积后的特征表示，**这波操作很具有实用性，一增一减，效果棒棒。
- **loss 不同，**作者 v3 替换了 v2 的 softmax loss 变成 logistic loss，而且每个 ground truth 只匹配一个先验框。

3.2 SSD

Faster R-CNN 准确率 mAP 较高，漏检率 recall 较低，但速度较慢。而 yolo 则相反，速度快，但准确率和漏检率不尽人意。SSD 综合了他们的优缺点,在 YOLO 和 faster-RCNN 的基础上，增加了多尺度的卷积层（不同尺度下的多个卷积层送入判定网络），这使得精度几乎与 faster-RCNN 保持不变，同时速度比 YOLO 更快。

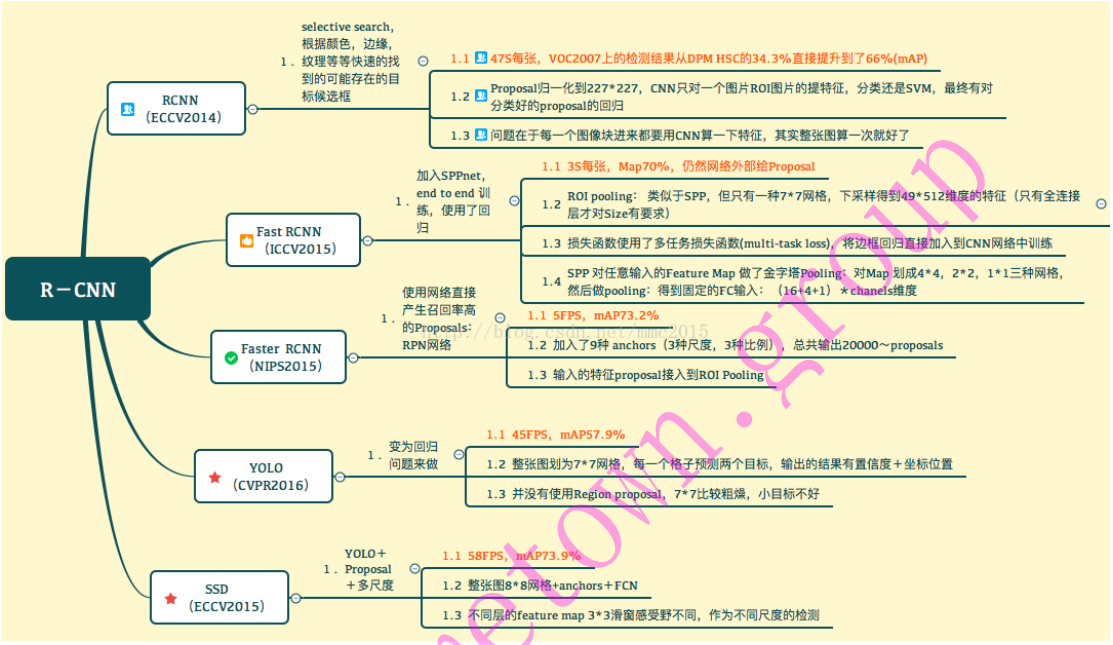
SSD 有人说是 faster RCNN 和 YOLO 的结合体，是有道理的。首先说 SSD 的贡献，它的贡献在于它利用了多层网络特征，而不仅仅是 FC7。那么为什么说它像 YOLO 呢，这主要是因为，SSD 还是借鉴了 detection 转化为 regression 的机制，而说它像 faster RCNN 是因为借鉴

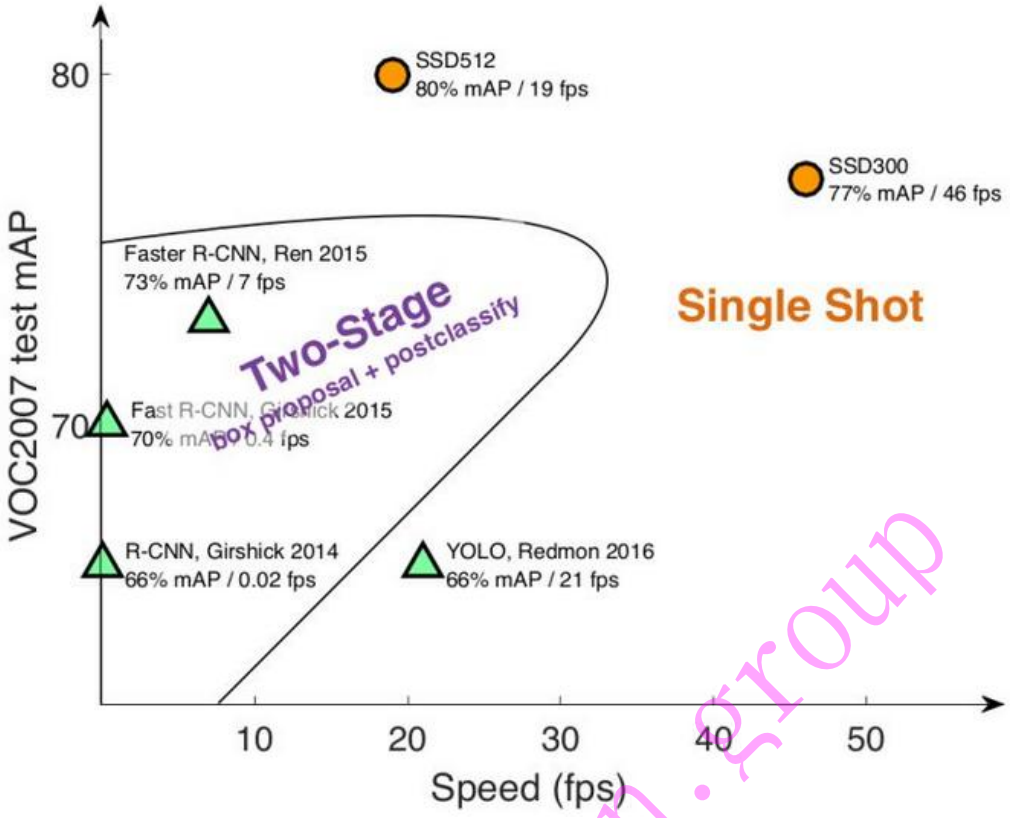
了 anchor 的机制，只不过它的 anchor 不是每个位置的精调，是跟 YOLO 一样画网格，然后在网格上产生 anchor，由于利用了多层特征，anchor 的 scale 每层都不同，因此产生了较多的超参数，增加了训练难度。

创新点：

- 从 YOLO 中继承了将 detection 转化为 regression 的思路，同时一次即可完成网络训练
- 基于 Faster RCNN 中的 anchor，提出了相似的 prior box；
- 加入基于特征金字塔（Pyramidal Feature Hierarchy）的检测方式，相当于半个 FPN 思路

存在问题：





4、总结

目标检测领域的深度学习算法，需要进行目标定位和物体识别，算法相对来说还是很复杂的。当前各种新算法也是层不出穷，但模型之间有很强的延续性，大部分模型算法都是借鉴了前人的思想，站在巨人的肩膀上。我们需要知道经典模型的特点，这些 tricks 是为了解决什么问题，以及为什么解决了这些问题。这样才能举一反三，万变不离其宗。综合下来，目标检测领域主要的难点如下：

- 检测速度：实时性要求高，故网络结构不能太复杂，参数不能太多，卷积层次也不能太多。
- 位置准确率：（x y w h）参数必须准确，也就是检测框大小尺寸要匹配，且重合度 IOU 要高。SSD 和 faster RCNN 通过多个 bounding box 来优化这个问题
- 漏检率：必须尽量检测出所有目标物体，特别是靠的近的物体和尺寸小的物体。SSD 和 faster RCNN 通过多个 bounding box 来优化这个问题
- 物体宽高比例不常见：SSD 通过不同尺寸 feature map，yoloV2 通过不同尺寸输入图片，来优化这个问题。
- 靠的近的物体准确率低
- 小尺寸物体准确率低：SSD 取消全连接层，yoloV2 增加 pass through layer，采用高分辨率输入图片，来优化这个问题

参考链接：

1. https://blog.csdn.net/sum_nap/article/details/80388110 (R-CNN 系列到 yolo、SSD 简要)
2. https://blog.csdn.net/lanmengyiyu/article/details/79680022
3. https://blog.csdn.net/mmc2015/article/details/72957372
4. https://www.cnblogs.com/guoyaohua/p/8994246.htm (目标检测算法总结)
5. https://yq.aliyun.com/articles/598428 (一文读懂目标检测 AI 算法)
6. https://blog.csdn.net/xiaoye5606/article/details/71191429