论文题目： DetNet: A Backbone network for Object Detection

----from ECCV2018

一、 背景

1、关于专门为物体检测设计的backbone特征提取器的讨论很少。更重要的是，图像分类和物体检测的任务之间存在一些差异。

2、最近的物体探测器如FPN和RetinaNet通常涉及extra stages,以防止图像分类任务处理各种尺度的物体。

3、物体检测不仅需要识别物体实例的类别，还需要在空间上定位。较大的下采样因子带来了较大的有效感受野，有利于图像分类，但会损害对象定位能力。

二、创新点

1、作者在提出了 DetNet，这是一种专门用于物体检测的新型 Backbone 网络。

2、DetNet 还包括针对传统骨干网络的 extra stages（基于 FPN 网络），用于图像分类，同时在更深层中保持高空间分辨率。

3、采用低复杂度的扩张瓶颈（bottleneck）结构。

三、Result

基于DetNet的Backbone网络，在MSCOCO基准测试的对象检测和实例分割都获得了最好的结果。

四、DetNet

1、motivation

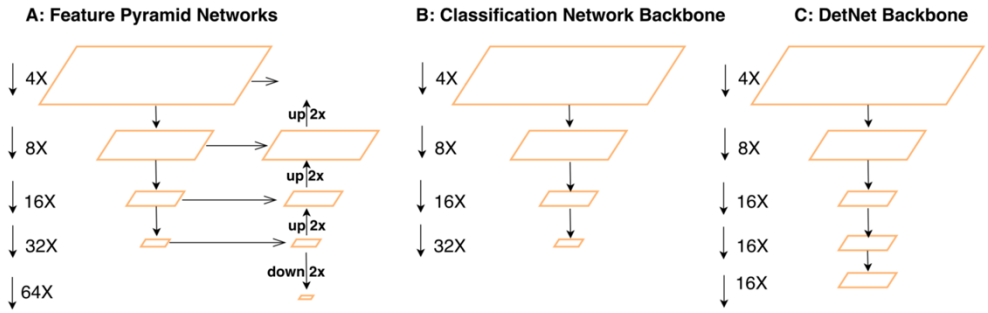


Fig. 1. Comparisons of different backbones used in FPN. Feature pyramid networks (FPN) with traditional backbone is illustrated in (A). Traditional backbone for image classification is illustrated in (B). Our proposed backbone is illustrated in (C), which has higher spatial resolution and exactly the same stages as FPN. We do not illustrate stage 1 (with stride 2) feature map due to the limitation of figure size.

作者列举了具有传统Backbone的特征金字塔网络FPN（如图1.A所示）、用于图像分类的传统Backbone网络（如图1.B所示）和自己提出的网络DetNet（如图1.C所示）。并介绍了前面两个网络的缺点：

1）The number of network stages is different.

上图1.B所示，典型的分类网络涉及5个阶段，每个阶段通过pooling 2x或stride 2个卷积来进行下采样特征映射,。在特征金字塔网络（FPN）中，添加了additional stage P6来处理更大的object，并且以类似的方式在RetinaNet中添加P6，P7。但是作者发现像P6这样的additional stage没有在ImageNet数据集中预先训练过。

2）Weak visibility of large objects.

在特征金字塔网络中，在较深层中生成并预测large object，这些对象的边界可能太模糊而无法获得准确的回归。当分类网络涉及更多stage时，这种情况甚至更糟，因为更多的下采样会给对象带来更多的stride。

3）Invisibility of small objects.

Large stride的另一个缺点是缺少small object。随着特征图的空间分辨率降低并且集成了大的上下文信息，来自small object的信息将容易变弱。如上图1.A所示特征金字塔网络通过采用自下而上的途径来缓解它。但是，如果较深层中缺少小对象，则这些上下文提示将同时丢失。

作者提出DetNet网络　1）阶段数直接设计用于物体检测 2）尽管涉及比传统分类网络更多的阶段（例如6个阶段或7个阶段），但保持feature map的高空间分辨率，同时保持较大的感受野。有以下优点：

1）DetNet与detector使用的阶段数完全相同，因此像P6这样的extra stage可以在ImageNet数据集中进行预训练。

2）受益于最后阶段的高分辨率特征图，DetNet在定位large object的边界和寻找丢失的small object方面更加强大。

2、DetNet Design

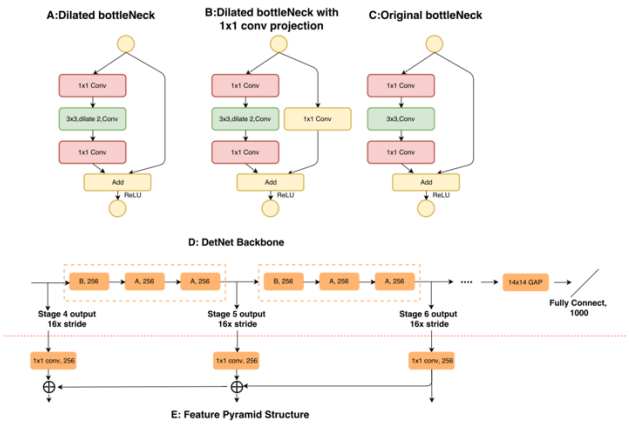


Fig. 2. Detail structure of DetNet (D) and DetNet based Feature Pyramid Net-Work (E). Different bottleneck block used in DetNet is illustrated in (A, B). Original bottleneck is illustrated in (C). DetNet follows the same design as ResNet before stage 4, while keeps spatial size after stage 4 (e.g. stage 5 and 6).

DetNet网络结构如图2.E所示，首先作者采用ResNet-50作为baseline，作者前四个阶段与原始ResNet-50保持一致。在backbone中引入了extra stage，例如P6，后者将在FPN中用于对象检测。同时在第4阶段之后，将空间分辨率固定为16倍下采样.由于在第4阶段之后空间大小是固定的，为了引入一个新的stage，在每个阶段的开始采用扩张的瓶颈(bottleneck)和1x1卷积（图2.B所示），应用扩张颈(bottleneck)作为基本网络块，以有效扩大接收领域。第5阶段和第6阶段保持与阶段4相同的通道（256个输入通道用于瓶颈块）。这与传统的骨干设计不同，后者将在后期加倍通道。最后以自上而下的路径方式对这些stage的输出求和。

五、实验

Baseline Models : ResNet-50

数据集：VOC2007、VOC2012

实验环境

- Python 2.7 or 3.6
- Pytorch 0.2.0 or higher（not support pytorch version >=0.4.0）
- CUDA 8.0 or higher
- tensorboardX

作者代码用 pytorch 框架来编写，加载 ResNet-50 预训练模型，学习率设置为 0.001,总共设置 12 个 epoch，将代码跑完，总共耗时 12 小时。网络结构如上图二所示，在 VOC2007 上实测结果 Mean AP = 75.01，跟作者给出的 75.9 以及 ResNet-101 的 75.7 要低一点点，可能是因为训练批次的原因，我只训练了 12 个批次，随着训练次数的增加应该会提高准确率。下面给出自己测试的结果：

AP for aeroplane = 0.7839
AP for bicycle = 0.8044
AP for bird = 0.7877

AP for boat = 0.6581
AP for bottle = 0.5700
AP for bus = 0.8194
AP for car = 0.8572
AP for cat = 0.8756
AP for chair = 0.5374
AP for cow = 0.8397
AP for diningtable = 0.6766
AP for dog = 0.8847
AP for horse = 0.8449
AP for motorbike = 0.7843
AP for person = 0.7853
AP for pottedplant = 0.4812
AP for sheep = 0.7377
AP for sofa = 0.7302
AP for train = 0.8026
AP for tvmonitor = 0.7404
Mean AP = 0.7501
~~~~~~~~
Results:
0.784
0.804
0.788
0.658
0.570
0.819
0.857
0.876
0.537
0.840
0.677
0.885
0.845
0.784
0.785
0.481
0.738
0.730
0.803
0.740
0.750
~~~~~~~~

作者测试给出的结果如下：

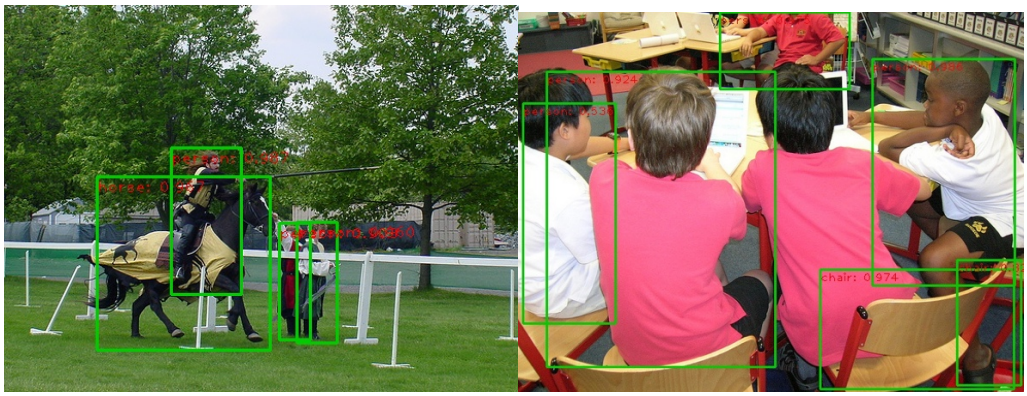
1). PASCAL VOC 2007 (Train/Test: 07trainval/07test, scale=600, ROI Align)

model（FPN）	GPUs	Batch Size	lr	lr_decay	max_epoch	Speed/epoch	Memory/GPU	mAP
ResNet-101	1 GTX 1080 (Ti)	2	1e-3	10	12	1.44hr	6137MB	75.7
DetNet59	1 GTX 1080 (Ti)	2	1e-3	10	12	1.07hr	5412MB	75.9

2). PASCAL VOC 07+12 (Train/Test: 07+12trainval/07test, scale=600, ROI Align)

model（FPN）	GPUs	Batch Size	lr	lr_decay	max_epoch	Speed/epoch	Memory/GPU	mAP
ResNet-101	1 GTX 1080 (Ti)	1	1e-3	10	12	3.96hr	9011MB	80.5
DetNet59	1 GTX 1080 (Ti)	1	1e-3	10	12	2.33hr	8015MB	80.7
ResNet-101(**using soft_nms when testing**)	1 GTX 1080 (Ti)	\	\	\	\	\	\	81.2
DetNet59(**using soft_nms when testing**)	1 GTX 1080 (Ti)	\	\	\	\	\	\	81.6

最后给出测试的部分图片：



六、总结

这篇代码是我最近跑的几个代码中效果是最好的，作者实际上是对 ResNet 网络的改进，通过第四个阶段之后加入残差网络，最终达到的效果还是不错的。论

文实验内容非常丰富，但是理论方面感觉好像没有什么，主要侧重点在实验上。