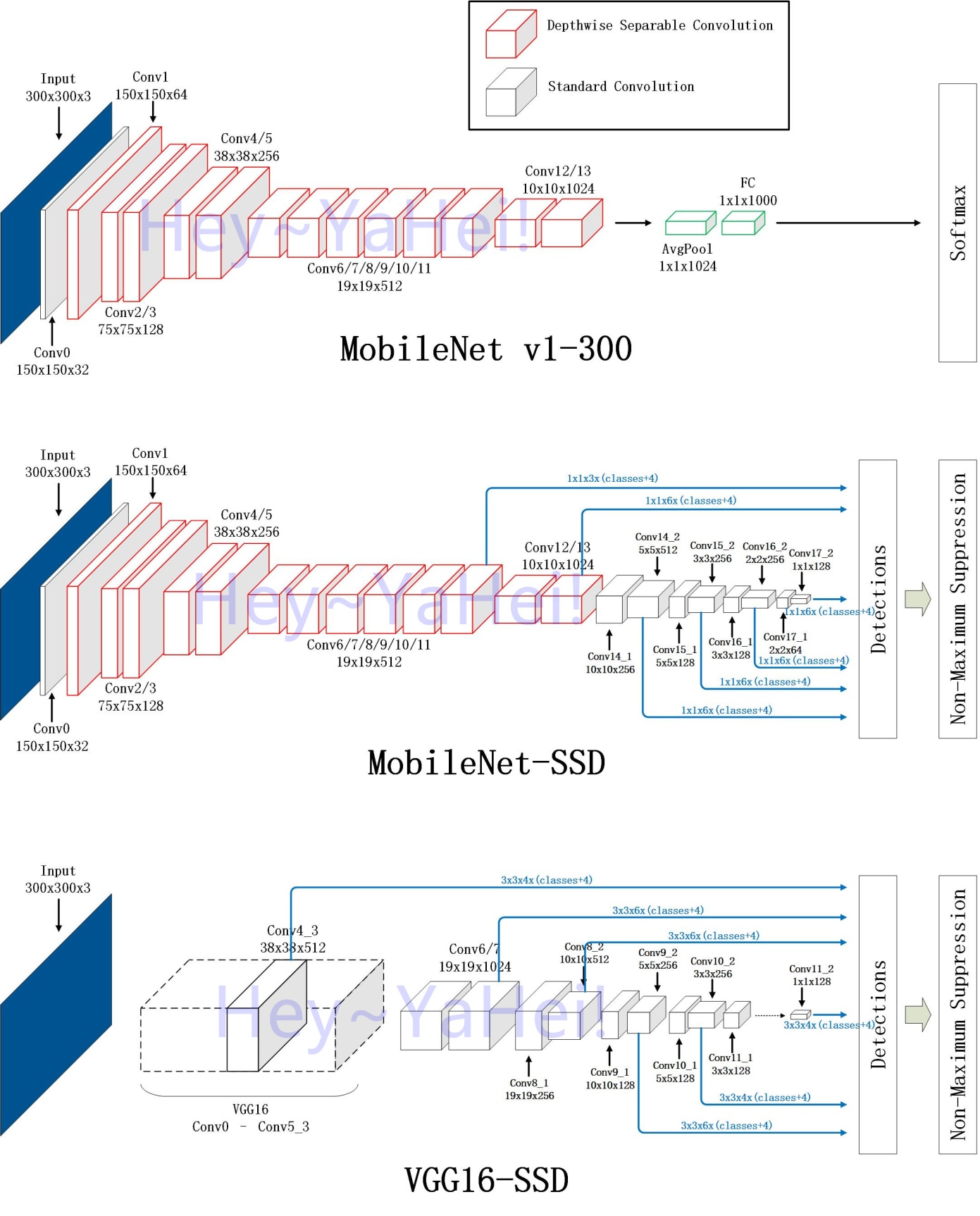
MobileNet-SSD网络解析

**网络结构**

参照 [MobileNet-SSD(chuanqi305)的caffe模型（prototxt文件） | github](https://github.com/chuanqi305/MobileNet-SSD/tree/master/template" \t "_blank)，绘制出MobileNet-SSD的整体结构如下（忽略一些参数细节）：  


图片中从上到下分别是MobileNet v1模型（统一输入大小为300x300）、chuanqi305的Mobilenet-SSD网络、VGG16-SSD网络。且默认都是用3x3大小的卷积核，除了MobileNet-SSD的Conv14\_1、Conv15\_1、Conv16\_1、Conv17\_1和VGG16-SSD的Conv8\_1、Conv9\_1、Conv10\_1、Conv11\_1用的是1x1大小的卷积核。  
图中每个立方体代表对应层的**输出**特征图；

* 首先观察基础网络部分  
  MobileNet-SSD从Conv0到Conv13的配置与MobileNet v1模型是完全一致的，相当于只是去掉MobileNet v1最后的全局平均池化、全连接层和Softmax层；
* 再看SSD部分
  + 在VGG16-SSD的方案中，用Conv6和Conv7分别替代了原VGG16的FC6和FC7；
  + MobileNet-SSD和VGG16-SSD都是从六个不同尺度的特征图上提取特征来做Detections，它们的大小为：
  + MobileNet-SSD | VGG16-SSD
  + ----------------+-----------------
  + 19 x 19 x 512 | 38 x 38 x 512
  + 10 x 10 x 1024 | 19 x 19 x 1024
  + 5 x 5 x 512 | 10 x 10 x 512
  + 3 x 3 x 256 | 5 x 5 x 256
  + 2 x 2 x 256 | 3 x 3 x 256
  + 1 x 1 x 128 | 1 x 1 x 128
    - 从通道数量上看，两者是完全一致的
    - 从特征图分辨率上看，MobileNet-SSD都只有VGG16-SSD的一半
      * **这意味着什么？**  
        打个比方，假设对于那个分辨率最大的特征图，都能用4x4的感受野检测出一只猫，如下图所示，黑色是头，红色是身体，棕色是腿，黄色是尾巴。

http://hey-yahei.cn/imgs/MobileNet-SSD/cat.png

那用MobileNet-SSD可以检测出占原图419≈0.211419≈0.211大小的猫，而VGG16-SSD却可以检测出占原图438≈0.105438≈0.105大小的猫；

* + - * **那为什么MobileNet-SSD为什么不和VGG16-SSD一样，从38x38分辨率的特征图开始做Detections呢？**  
        回到上一篇博文《[SSD框架解析 - 网络结构| Hey~YaHei!](http://hey-yahei.cn/2018/08/06/SSD/#%E7%BD%91%E7%BB%9C%E7%BB%93%E6%9E%84)》，VGG16是从Conv4\_3也就是第10层卷积层取出38x38分辨率的特征图；  
        再观察一下MobileNet v1-300的模型，想要取出38x38分辨率的特征图，最深也只能从Conv5也就是第6层卷积层取出，这个位置比较浅，实在很难保证网络提取出了足够有用的特征可以使用；
      * **那可以通过增加最初输入图片的分辨率来解决这个问题吗？**  
        倒也可以，比如把输入图片大小扩大到512x512，那么Conv11的输出就变为32x32，按上上一点的描述，可以检测出占原图432=0.125432=0.125大小的猫；  
        但要付出相应的代价，仅考虑基础网络部分（Conv0到Conv13），参数数量和乘加运算量均提高为原来的 (512300)2≈2.913(512300)2≈2.913 倍（不考虑padding的影响，计算方式可以参考《[MobileNets v1模型解析 - 效率比较 | Hey~YaHei!](http://hey-yahei.cn/2018/08/05/MobileNets_v1/#%E6%95%88%E7%8E%87%E6%AF%94%E8%BE%83)》），MobileNet本身小模型的低参数量、低运算量优势变得不再明显。
  + 还有一个小细节，观察特征图到Detections的路径  
    VGG16-SSD中用的都是3x3大小的卷积核，缺省框数量依次是4、6、6、6、4、4；  
    MobileNet-SSD中用的都是1x1大小的卷积核，缺省框数量依次是3、6、6、6、6、6；  
    *这一部分的改动不是很能理解，3x3卷积改1x1卷积可能是实践中发现改动后效果差不多但可以减少运算量；缺省框数量改动的原因就不得而知了~*

**BN层合并**

对比chuanqi305的 [train模型](https://github.com/chuanqi305/MobileNet-SSD/blob/master/template/MobileNetSSD_train_template.prototxt) 和 [deploy模型](https://github.com/chuanqi305/MobileNet-SSD/blob/master/template/MobileNetSSD_deploy_template.prototxt) 还能发现一件有趣的事情——  
**deploy模型中的BN层和scale层都不见啦！！！**  
BN层是这样随随便便就能丢弃的么？没道理啊！

几经辗转，查阅资料之后发现，原来BN层是可以合并进前一层的卷积层或全连接层的，而且这还有利于减少预测用时。  
参考《[Real-time object detection with YOLO - Converting to Metal](http://machinethink.net/blog/object-detection-with-yolo/" \l "converting-to-metal" \t "_blank)》

合并的原理：卷积层、全连接层和BN层都是纯粹的线性转换。

数学推导也很简单：  
*假设图片为*xx*，卷积层权重为*ww*。*  
那么对于卷积运算有，

conv[j]=x[i]w[0]+x[i+1]w[1]+x[i+2]w[2]+…+x[i+k]w[k]+bconv[j]=x[i]w[0]+x[i+1]w[1]+x[i+2]w[2]+…+x[i+k]w[k]+b

BN层运算为，

bn[j]=γ(conv[j]−mean)variance−−−−−−−√+β=γ⋅conv[j]variance−−−−−−−√−γ⋅meanvariance−−−−−−−√+βbn[j]=γ(conv[j]−mean)variance+β=γ⋅conv[j]variance−γ⋅meanvariance+β

代入conv[j]conv[j]变为，

bn[j]=x[i]γ⋅w[0]variance−−−−−−−√+x[i+1]γ⋅w[1]variance−−−−−−−√+…+x[i+k]γ⋅w[k]variance−−−−−−−√+γ⋅bvariance−−−−−−−√−γ⋅meanvariance−−−−−−−√+βbn[j]=x[i]γ⋅w[0]variance+x[i+1]γ⋅w[1]variance+…+x[i+k]γ⋅w[k]variance+γ⋅bvariance−γ⋅meanvariance+β

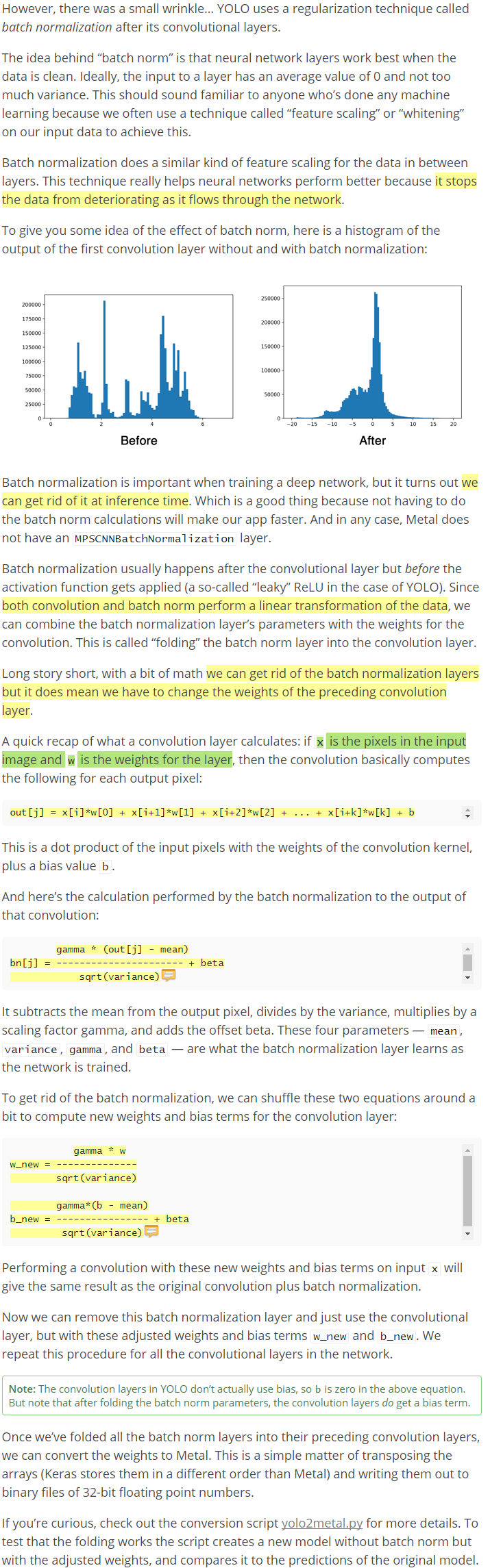
两式对比可以得到，

wnew=γ⋅wvariance−−−−−−−√wnew=γ⋅wvariance

βnew=β+γ⋅bvariance−−−−−−−√−γ⋅meanvariance−−−−−−−√=β+γ(b−mean)variance−−−−−−−√βnew=β+γ⋅bvariance−γ⋅meanvariance=β+γ(b−mean)variance

注意，其中 γγ、meanmean、variancevariance、ββ 都是训练出来的量，在预测阶段相当于一个常量。

原文摘录如下：



本文介绍了chuanqi305的MobileNet-SSD网络是如何组成的以及实用的MergeBN技术，在下一篇博文中我们将尝试用该网络进行训练并部署在RK3399的Tengine平台上，并且进一步对该网络进行改进以满足我们实际场景的需要。