1. AlexNet网络架构

这小节是本文的精彩点之一。我们开始讨论AlexNet的网络结构，先看看前面GPU章节所展示的AlexNet架构图。从图中，我们可以看到这是是一个8层网络架构，我在笔记里用带圈数字给每一层都标了号，前5层是卷积层，后3层是全连接层。其中最后的全连接层输出是一个1000通道softmax映射归一化结果，表示输入在1000类别的响应情况，或者说在归属类上的概率分布，再细致的说就是每个通道的softmax输出表示输入属于该类的可能性（而不是得分，见后文说明）。我们网络训练的目标是最大化多项的logistic回归，其实就是最大化这个网络所有输出正确类别的响应的平均值，即max1N∑Ni=0log(p(yi|xi))，其中N是样本数，yi代表第i个样本的标记，xi代表第i个样本的输入，p(yi|xi)表示网络输入第i个样本，输出其样本标记的可能性，也就是标记通道对应的那个值。

关于softmax，虽然我不想做详细介绍，但我想用知乎上的一段话来总结一下它的作用。“看名字就知道了，就是如果某一个zj大过其他z,那这个映射的分量就逼近于1,其他就逼近于0，主要应用就是多分类，sigmoid函数只能分两类，而softmax能分多类，softmax是sigmoid的扩展”，这里z指的是所有全连接的结果。还有一句“softmax模型是logistic模型在多分类问题上的推广， logistic 回归是针对二分类问题的”。也就是说，由于前面提到过的sigmoid函数对于每个神经元是一个二值饱和函数，在多分类问题上其实挺容易遇到两类的得分很高也接近的情况，这时输出容易模糊(Fuzzy)。而softmax是一个类别的得分的分布情况，要么一起低，要不就有高有低，所以在多分类的时候，容易落实。详细可参考http://www.zhihu.com/question/23765351.

我们再从架构图中观察一些这8层网络的特点。不难发现，第2、4、5个卷积层的输入竟然只跟同一个GPU下前级输出有关系，与之前提到过的一样，而仅有第3层跟前级的所有输出，包括另一个GPU的输出，有关系。这样几乎整个卷积层都实现了并行了，但并非毫无代价的，稍后再解释。然后全连接层肯定是所有输入都得跟所有的前级输出有关，这是毋庸置疑的。之前提到的带交叠的Max-pooling层，我们将它放到1、2、5层。局部归一化（侧抑制）我们让它作用在1、2层跟在Max-pooling的前面。而ReLU是对每层的卷积和全连接作用结果进行激活。

激动人心的时刻到了！经过前三段枯燥的铺垫，我们已经掌握了AlexNet的基本流程与相关的组成部分。现在我们要分析它的具体规模，并且还要分析接近它规模的单GPU网络做对比。

看下用铅笔画的单GPU网络图的规模，输入层大小是224×224×3，第一层卷积层的大小是55×55×96，第二层为27×27×128，第三层为13×13×384，第四层为13×13×192，第五层是13×13×256，最后三个全连接层分别是4096×1，4096×1，1000×1。利用这些数据，我们又可以分析一些卷积层之间的卷积核信息。比如，输入层的深度为3，而第一卷积层的深度是96，也就是说我们这两层之间需要96个？×？×3大小的卷积核。可是我们还没有办法知道这个核的大小是多少，因为没有卷积核遍历输入层的步长信息。幸好，作者说这个步长是4，所以我们通过方程224+?2−?4=55得到？＝11。然而，其他卷积核的大小没这好办了，卷积层大小的压缩主要由Max-pooling实现，比如3、4、5层的大小都没有变化（这三层之间没有Max-pooling），而卷积核的大小一般是不直接影响卷积映射结果的大小，因此卷积核大小其实是一个待设定参数。不过，卷积核的深度和个数都是可以计算的，我这里直接按顺序给出剩下3个卷积核，128个5×5×96，384个3×3×128， 192个3×3×384，256个3×3×192。你可能已经发现规律！是的，每一层卷积核的个数会变成下一层卷积核的深度，这是由卷积形式所决定的。因为每一个卷积核（注意概念，不是一层的卷积核，而是一层卷积核里面的其中一个）对上一层的结果进行操作，都会出来一个属于这个卷积核的结果，那么如果我有128个核就有128个结果，每个结果是一个特征图，然后它们放在一起形成一个多维特征结构，所以我们能看出，CNN卷积层的特征提取操作可以理解为一个多维度分析过程，其实也就是小波变换（如果将卷积核理解成滤波器的话）。那么特征学习其实就是在学习到底哪些维度更能描述这个分类问题。

然后我们回到AlexNet的架构图（印刷图），如果你注意到我的单GPU架构图部分的一些长方体的虚线以及两个小正方体相加的图案，并开始琢磨这俩的含义时，我会说“恭喜你，你看到了点子上！”。其实AlexNet的双GPU架构是，从我所画的这个单GPU架构，通过窍门将它分成两部分。如果还能注意到我跟Alex这个架构图的在规模上的差别的话，你还会发现其实这个窍门的本质几乎所有程序猿都认识——空间换时间。我拿前三层打个比方，第一个卷积层实际上是“无损的”，他只将我原来的55×55×96的卷积层模型从深度上切分成两块55×55×48的模型，放到两个不同的GPU位置而已（这里要注意一下哈，我说的卷积层是指感受野或hidden layer，请区别与前面的每层的卷积核，kernel）。逻辑严谨的朋友会发现，这里有一个问题：卷积的时候，一个卷积核需要用到两个GPU的结果，那么多卷积核，GPU之间的交互不应该很频繁么？哪能像Alex 图那样的还光GPU自己里面能倒腾出来，轻松并行呀？（提醒一下，有的人可能没注意到一个卷积核是有深度的）我觉着这个问题非常到位，那要不咱们也把卷积核按照对应深度切开呢？诶，好像有戏！如果确实我们现在看第2层，将原来128个5×5×96切分成两组128个5×5×48，得到256个5×5×48卷积核，分别作用在两个不同GPU上的结果的到两个27×27×128的卷积层结果。（大家应该都注意到了，这里所需的存储空间已经比之前翻倍了，后文详述。）但是，主要矛盾还没解决，本来两结果要做运算和在一起的，现在还分居两地呢。“要做啥运算啊？”，“好像是直接相加！”，“那很淡定！不还有加法分配律么？”。如果我们将384个3×3×128复制一份，变成两组相同384个3×3×128，即一组深度为256的核，分别作用再相加应该就跟原来单GPU一样了。然而，拷贝在训练阶段可能是不靠谱的，因为我们只用了一半的特征做学习而已，到底哪一半数据可行度最高还是未知（不过这个问题可以考虑通过不同时刻利用不同GPU的特征，切换着来训练从而减少核数据的存储压力），而且Alex和我的模型有着本质上的区别，就是ReLU和Max-pooling对每个分离层的作用引入了非线性因素，导致叠加与原来单GPU的情况有很大差别，但我们也能看出选择ReLU的非饱和线性因素在这个地方有点意义。因此我觉着Alex在这依然选择依然采取双核是有他的道理的。经过这点改造后，我们就能得到单GPU的13×13×384（即深度为384）的卷积层。再将其从深度切成两块放到两个不同的GPU，就是AlexNet的第三层。将上述功夫重复一遍，就是AlexNet的卷积网络部分，这里不做详细介绍Alex的剩下网络，有兴趣看下原文第3章。

最后，我们再点评一下AlexNet的并行架构设计。它实际上只是一个单GPU网络的切割分离，核心技巧是“空间换时间”，它的美妙之处是Alex是刚刚好让卷积网络的分离并行流程做了两次而已，前面介绍的第1、2、3层是一次，第4、5、6层又是一次。而且分离合并的时机也很恰当，假设第2和3层之间不做数据交互合并的话，估计与单GPU模型有着很大的差别，因为非线性因素引入太多了，核也很难再切了。另外，通过对比分析这俩个模型，我怀疑作者前面的单GPU实验设计是有点问题的。他只用了一半AlexNet的kernels来做实验，就暗示单GPU的精度没有双GPU高的结论，我觉得有些欠妥。实际上，单GPU的kernels在其中两层的数目是应比AlexNet的对应的一半要多的，所以这里应该补充一组实验数据。