如何设计网络结构更有利于ARM架构底层加速

authored by kailiang

对于ARM而言，由于其本身计算能力的限制，因此网络的耗时基本集中在卷积操作上。

PPL目前主要针对如下的几种常用的convolution的case进行了针对性的优化:

filter_h = filter_w = 1, 3, 5 && stride = 1, 2

通常而言，conv的计算量可以表示为2 * h * w * c * k * flt_h * flt_w。其中，2表示一个乘法和一个加法，h是输出的高，w是输出的宽，c是输入的通道数，k是输出的通道数，flt_h和flt_w是卷积核的宽和高。

在总计算量不变的前提下，c和k越大，通常性能会越好。当然这个“好”的程度是有明显的边际效应递减的。

尤其是对于flt_h = flt_w = 3, 5 && stride = 1, 2的case，我们使用了winograd算法。这个计算方法改变了进行计算的方式。其由新的三个部分组成，一部分仅与h, w, c有关，一部分仅与h, w, k有关。因此借鉴均摊分析的思路可知，当c和k越大，这两个部分占时比例就会越低，性能的提升会更加明显。

另外一方面，深度可分离卷积(depthwise)由于计算量相比普通的卷积计算量要低得多，所以如果直接进行替换也能有很明显的耗时的减少。这里的比较前提是depthwise的group == c == k的情形。此时的depthwise的计算量是2 * h * w * group * flt_h * flt_w。可以看出相比于普通的卷积少了k倍。但是因为depthwise本身是一个对于底层优化不那么友好的类型，且不能使用winograd算法，所以其绝对效率没有普通的卷积那么高。所以如果两者不是group == c == k，而是计算量相等，那么depthwise的耗时会明显长于普通的卷积。

在总计算量不变的前提下，stride==1的case性能会比stride==2的好。这是因为stride==1的数据在读入后能够被复用得更多，对于底层的优化而言会更加容易发挥硬件的计算的能力。

在PPL3中，目前除了开头的第一层外，内部的绝大部分操作中， channel这一维度（包括input和output）会补齐到4的倍数进行计算。因此去对模型裁剪时，如果只是将channel从比如80降到77，对于耗时其实不会有帮助。