Reformulating the direct convolution for high-performance deep learning inference on ARM processors

JX-Ma

2024/1/26

1 摘要

这篇文章提出了卷积算子的两种高性能实现，其性能优于基于 im2col 变换和基于 ARMv8 处理器的 gem 内核的所谓降低方法。第一种提供了零内存开销的额外优势，另一种方法采用了额外的但相当适中的工作空间，比 im2col+gemm 解决方案所需的工作空间小得多。

2 介绍

卷积 (算子) 是一个关键的计算内核，它集中了在信号处理和计算机视觉任务中经常使用的深度神经网络 (dnn) 类型的很大一部分算法成本。直接卷积算法，这篇文章提出的直接卷积算法的循环顺序和我们之前提到的有点差别, 并且论文指出算法循环可以以任何顺序重组，同时仍然产生正确的结果。文章还提出需要卷积友好型数据布局。确保以单位步长访问卷积运算符的输出，这篇论文的操作是更改了数据布局，原本存储的连续的空间是宽度这个维度上的空间，经过更改后，存储的数据为 a(0,0,0,0),a(0,1,0,0),a(0,0,0,1),a(0,1,0,1)... 这些数据在空间上连续，这一步的操作也可以叫做平铺，上周所看的论文中也提到了这个相关的概念，这样做的好处是为了避免可能导致性能下降的寄存器溢出，或者寄存器存储了无效的数据，进行不必要的无效的运算，

3 算法 A

算法 A 主要是在不使用额外的内存的情况下，使用向量寄存器去加速微内核的卷积，对于微内核的大小需要根据寄存器数量来决定。循环顺序也对卷积的性能有着很大的影响，对于本文中提到的卷积来说，需要使用行为主的循环顺序去进行卷积。下面是文中举的示例：首先我们使用 128 位寄存器去存储 4 个 fp32 的数据，我们假设微内核的维度为 7x12, 这个时候需要使用 21 个 128 位寄存器去存储这些数据，这篇文章的做法是使用 7 个寄存器去存储每一行连续的四个元素，然后再利用 3 个寄存器去存储卷积核的窗口元素。在这里也指出微内核的尺寸受到处理器体系结构中向量寄存器数量的限制。当向量寄存器数量达到最高时，再继续增加使用向量寄存器的数量会影响性能。

4 算法 B

算法中采用的替代策略是采用 BLIS 策略，也就是将输入张量打包，且打包是在遍历循环的时候完成的，打包后的结果能够确保输入张量的窗口是以单位步长进行移动，这种算法会带来一定的负面影响，例如由于 buffer 的额外工作空间，这种变体引入了内存 开销，由于将数据复制到变量中，存在一定的成本 。好的效果就是这个新算法 B 可以利用自然的 BLIS 微内核来执行小 gem，在内部循环中，数据复制的成本很可能在足够的计算中得到很好的平摊。重排保证了单位跨步访问 from 微内核的微面板。微内核的开发是一个系统过程，可以在一定程度上实现自动化，并获得良好的性能结果.

5 实验部分

硬件配置：实验都是在 NVIDIA Carmel (ARMv8.2) 处理器的单核上使用 IEEE 单精度 (FP32) 进行的。该架构配备了每核 64 KiB 的私有 4 路关联 (数据)L1 缓存，每对核之间共享 2 MiB 的 16 路关联 L2 缓存，所有 8 核共享 4 MiB 的 16 路关联 L2 缓存以及 4gib 的 RAM。避免了因利用率不同而造成的性能扭曲。
卷积层选用了 GoogleLeNet[29] 和 ResNet-50 (v1.5)
算法选择了 lowering，BLOKED,NEW-A,NEW-B,WINOGRAD, 这 5 中算法的性能进行对比。
实验结果表明，新的算法比传统的基于 im2col 的方法要快，算法 A 没有消耗额外的内存但是速度比较算法 B 还是要慢，虽然算法 B 有额外内存的消耗但是相比较传统的 im2col 内存消耗要少很多。

6 总结

这篇文章主要是 arm 架构上对直接卷积的优化，其中提出的算法 A 是零内存直接卷积，没有其他多余内存的消耗，这种卷积适用于一些内存有限的边缘性设备，而且论文提出的算法 A 性能的展示也比较好。算法 B 主要是使用了一定的内存去存储数据，虽然有多余的内存消耗，并且复制数据存在一定的成本，但是速度方面有着很大的提升，而且内存消耗远小于传统的 im2col 卷积，对于一些内存空间足够的设备上进行卷积操作，完全可以牺牲一定的空间去提升自己卷积的速度。