Anatomy Of High-Performance Deep Learning Convolutions On SIMD Architectures

JX-Ma

2024/1/2

1 笔记

反向传播实现：反向传播实现的循环结构和前向传播一致，他们之间的区别是前向传播累加的是输出张量，而反向传播实现累加的是输入张量，在反向传播中也能重用高性能前向传播。
权重梯度更新实现: 通过将梯度输出张量和输入张量进行卷积得到新的卷积核，在此过程中我们使用到了前向传播所应用的优化，例如将权重梯度分为 VLENxVLEN 的子张量，在梯度更新中采用并行化策略，在梯度更新中可以选择不同的并行化策略，一个是我们假设有 RxSxKbxCb 独立任务。如果这个并行量对于 T 个线程是足够的，并假设完美的工作分配，那么每个线程计算 (R*S*C*K)/T 项的权重梯度张量。另一种是通过分配小批量维度 N 来计算自己的梯度权重的部分、局部副本，那么我们可以提取更多的并行性。
降低精度: 深度神经网络训练的另一个大趋势是降低精度以加快训练时间。
性能评估：性能评估分为只使用内核的性能，后面是完全基于图的执行性能。接下来介绍了几种卷积方法
im2col: 输入数据被扁平化，随后执行标准的矩阵乘法调用。
libxsmm: 这种方法使用被适当阻塞的直接卷积循环的实现，以容纳小的矩阵乘法作为最内层的微内核。
blas: 与上面相同的实现，但我们使用 MKL GEMM 调用而不是利用 LIBXSMM
autovec: 与上面相同的实现，但不是使用 MKL GEMM 调用，我们显式地将小 GEMM 拼成三个嵌套循环，并依赖编译器自动向量化循环 (编译器版本 icc v2017.0.4)。
MKL: 为了完整性，我们对专门用于直接卷积的 MKL-DNN li- library v0.12[22] 进行了基准测试。
本文在 Skylake-SP (SKX) 和 Knights Mill (KNM) 之间把不同的卷积方式进行了性能评估，可以看到本文中使用的卷积在很多层的表现情况下都要优于其他卷积方法，有些时候会低于 MKL 卷积方法，而后在 MKL 和这项工作这项工作之间，通过降低精度的方式来对比这两种卷积之间的性能，最后得出结论本文提出的方法和 MKL-DNN 实现了相当的内核性能，但是由于缺乏融合、低效的内存分配或线程调度等各种原因，这种良好的 MKL-DNN 的大部分性能在框架集成期间会丢失。

2 心得

上次看这篇论文只看到卷积神经网络前向传播部分，并了解了前向传播实现中的几个优化的方法，首先就是向量化和寄存器阻塞，向量化也就是使用向量寄存器去存储数据，我们使用 VLEN 代表向量寄

存器存储数据的个数，VLEN 取决去存储的数据类型和数据大小，我上周使用了 AVX2 指令集中的 _m256 代表可以储存 256 位大小的寄存器，然后我使用的是 float 也就是 FP32 数据类型，这个时候设置的 VLEN 是 8，寄存器阻塞可以用来提高寄存器的数据重用，最重要的是隐藏 FMA 指令的延迟，256 位 FMA 指令是 _mm256_fmadd_ps(), 主要进行一次乘法和一次加法，因为运行 FMA 指令需要取读取数据，读取数据存在延迟，之前学过 hoist 优化就是可以把通过寄存器把读取数据的时间藏起来，而使用向量寄存器就可以把更多读取数据的时间藏起来，还有优化就是调整循环顺序，通过把矢量化块作为张量的最内层、快速运行的维度。还有就是微内核，把寄存器分块的部分作为微内核，但是一个微内核不可能实现任意卷积层的性能，也就是说寄存器分块的数量不可能刚好分完，会有剩下的部分，对于剩下部分处理，论文中给出两种处理方案，一种是剩下的部分可以 hoist 输出张量的元素，按照单个元素直接卷积的方法进行卷积，另一种是当剩下的部分宽可以分块时，我们把剩下的每一行的元素存入向量寄存器进行计算。后面的优化方法包括软件预取，目的为了减少 cache 命中率，并行化策略，通过合理利用并行性。层融合，利用时间局部性节省内存带宽，内核流，处理输出张量的宽和高不能完全分块的情况。反向传播实现的目的是更新卷积核参数，通过不断调整卷积核的参数来使卷积达到正确的结果，在反向传播的过程中，我们也可以使用前向传播所介绍的各种优化方法，在从最后的性能评估方面得到结论通过优化直接卷积可以和 MKL-DNN 实现了相当的内核性能，这样的话我们可以避免很多多余的内存消耗，这样优化直接卷积算法就可以适用在更多资源有限的边缘性设备上。