Optimizing Direct Convolutions on ARM Multi-Cores

JX-Ma

2024/1/22

1 摘要

本文提出了提出了 nDirect，一种新的直接卷积方法，针对智能手机和高性能计算系统中常见的基于 arm 的多核 cpu。nDirect 旨在与主流深度学习框架使用的数据布局格式兼容，但为计算内核、数据打包和并行化提供了新的优化。

2 介绍

传统的 Im2col 卷积会增加内存占用，并且张量到矩阵的转换会导致性能欠佳的不规则矩阵。直接卷积它的工作原理是在输入张量上滑动一个 CONV 核，并在每个位置计算核与一小块输入之间的点积。直接卷积的两个优点直接卷积对内存的要求较低，因为它直接对输入张量进行操作，避免了将输入张量转换成更大的矩阵。这可以提高缓存局部性，减少内存使用其次，直接卷积可以利用卷积核的稀疏性，避免不必要的计算，导致更快的计算时间。

LIBXSMM 是最先进的基于库的直接卷积，它在 x86 和 ARMcpu 上提供了比 im2col+GEMM 更好的性能。缺点：它的数据布局设计与主流深度学习 (DL) 框架中使用的常见数据布局不兼容。LIBXSMM 仍然使用传统的基于 gem 的微内核，它无法利用卷积中潜在的数据重用机会来提高性能。

3 直接卷积

直接卷积上的优化有 4 种策略，包括本研究中针对的直接卷积、im2col+GEMM 方法、FFT(快速傅里叶变换) 和 Winograd，FFT 和 winograd 可以降低计算复杂度，但是这两种方法都会增加内存压力，降低预测精度。im2col 转换矩阵过程会有额外的内存消耗，并且可用的硬件内存带宽会限制其性能。nDirect 旨在通过保留主流 DL 格式并实现高性能来填补这一空白。虽然 LIBXSMM 已经取得了很好的卷积性能，但它引入了新的数据布局，旨在改进缓存局域性和利用向量化。然而，将这种新的数据布局纳入主流 DL 应用程序将需要对现有框架进行重大的重新开发，并需要大量的工程努力。一个更好的方案应该最大限度地减少对现有 DL 软件系统的中断，使它们更容易集成到现有的 DL 框架中，而无需大量的重新开发工作

虽然 LIBXSMM 使用优化的微内核和缓存友好的数据格式来实现快速的直接卷积，但我们发现它的循环块大小太小，无法充分利用现代 ARMv8 多核中可用的多级缓存和融合乘法累积 (FMA) 单元。我们观察到现有的并行化策略是粗粒度的，导致 ARM 多核上的卷积性能较差。计算是在多个批次上顺序执行的，导致线性成本积累。需要进一步的优化来克服这个问题。

4 优化方法

4.1 非直接设计

首先提到的就是改变循环顺序，让数据连续性表现的更好，循环顺序可以确定数据的访问顺序，我们可以尽可能的让循环顺序访问的数据连续性好，这样可以加快循环的速度。
后面提到的就是确定分块的大小也就是循环平铺，将循环按照某个数值平铺展开，循环平铺是提高缓存数据局部性的关键，在本次论文中提出了公式计算来确认平铺的大小计算方法大概是在 for 循环的最后一层也就是第七层，数据传输量要小于 L1 缓存，第六层的数据传输量要小于 L2 的缓存。

4.2 微内核设计

论文提出了两个微内核，其中一个用来加速卷积，第二个用于填充输入张量
nDirect 旨在改善直接卷积的数据重用，并利用 ARMNEON SIMD 扩展来提高指令级并行性。
在计算微内核中需要对输入张量进行打包处理，处理完在随后的每次迭代中，输入数据都从线性缓冲区中提取，可以提高 L 1 数据缓存的命中率。

4.3 并行化设计

理想情况下，所有内核将同时开始和完成工作，因此在任何时间点都没有任何内核空闲。然而，由于内存访问延迟和应用程序工作负载，这并不总是可能的。因此，我们需要仔细确定使用多少线程来并行化每个并行维度。
在并行化时不会去选择并行化卷积核的 width，height，和 channel, 因为这样会造成写入冲突，选择使用 PTk 去并行化卷积核的 batch,PTn 去并行化输入张量的宽，高和 batch, 且 PTk x PTn = PT (PT 为理想的线程大小)

5 实验部分

实验在三个高性能计算系统和一个 ARM 多核嵌入式系统上进行。对比的基准有 im2col+GEMM，LIBXSMM，XNNPACK，Ansor，还将 nDirect 与 MXNet 集成，并通过将我们的方法与 MXNet 使用的 im2col+ GEMM 和 CNN 模型 (由 Ansor 和 TVM 后端代码生成器调整) 进行比较，评估了 CNN 模型的端到端性能。实验结果展示 nDirect ，一种新的直接卷积解决方案，可在 A R M 多核 cpu 上提供高性能，高数据可重用性和深度学习 (DL) 框架兼容性，并在在大多数测试用例上都优于竞争基准，在所有硬件平台上都取得了更好的整体性能。

6 总结

这篇文章讲的是在 arm 多核上对直接卷积算法上的优化, 虽然我目前使用的架构是 AMD，但是从上述所给的优化方法也可以应用在 AMD 的主机上，这篇论文所讲的优化主要在 4 个地方，一个是循环排序，一个是分块大小的确定，还有微内核的设计，以及并行化设计，首先循环排序这部分我在很多篇论文中都能看到它，循环排序主要是让访问的数据在空间上尽可能的连续，这样的 cache 命中率就会很

高。而分块也是很重要的一部分，分块可以提高缓存数据的局部性，确定分块的大小是很重要的，分块小了的话会导致缓存利用不充分，过大会导致带宽限制。微内核相当于一个小的卷积过程，前面所实现的分块，循环排序，都是为了微内核进行卷积时能够具有良好的数据连续性从而加快卷积速度。并行化主要就是确定一个合适的线程数量，线程数量并不是越大越快，合理的分配即可避免资源浪费，也可以加快卷积速度。