Systematic Dataflow Exploration and Code Generation for Efficient Neural Network Inference using SIMD Architectures on CPUs

JX-Ma

2024/3/16

1 笔记

本文把输出保持在 SIMD 寄存器中，最大化输入和权重重用的数据流始终为各种推理工作负载提供最佳性能。在本文中引入了数据流的概念，数据流是指神经网络计算操作的执行顺序。simd 确定了不同变量的重用机会，我们可以指导如何最好地分配有价值的 SIMD 寄存器资源以最大化重用。本文的工作为 1. 扩展了现有的数据流，这些数据流通常只指定要重用的一种类型的变量，以允许所有类型的变量都可以重用。2. 形式化了一组启发式方法，基于数据移动成本，通过最大化每个数据流内的数据重用来优化三个基本的、通用的神经网络数据流。3. 实现了一个代码生成器，它自动使用 SIMD 指令来实现各种扩展和基本数据流，适用于任何给定的神经网络配置。4. 使用代表性的工作负载，定量地比较了最好的实现与最先进的实现。这篇文章采用的数据布局是 NCHW 布局，这篇文章的数据布局和我之前在直接卷积上实现的类似，也是把输入张量通道按照寄存器大小的倍数展开，也叫作通道维度矢量化，我之前实现的是按照通道 8 展开，因为 256 位寄存器可以存储 8 个 Float 元素。然后它的输出张量也拿了 simd 去存储，后面由于考虑到寄存器的数量，于是本文实现了一个代码生成器，会根据神经网络层配置 (类型、张量维度、精度)、硬件信息 (ISA、SIMD 向量大小、向量寄存器数量) 以及所需的锚定和辅助平稳性，从而生成合适的对输入张量展开大小。

2 心得

这篇文章的做法我在直接卷积上试过，首先我在优化直接卷积时使用了两个方法加快卷积，一个就是之前提到了关于 simd 存储元素的方式，之前提到的方式一是存储输入张量单个窗口内连续的元素，另一个就是存储输入张量连续窗口的第 i 个元素，方式二要求步长为 1，方式二的话输出张量就能使用 simd 存储连续的几个元素，在步长为 1 的情况下，方式二的速度要远大于方式一的速度。方式二的好处在于数据 cache 命中率较高，假设我宽为 40，存储输入张量寄存器第一次存储的数据是 1-8，第二次存储的数据是 2-9... 其二就是通过 fma 算出来的结果直接就是输出张量的结果，当然主要影响性能的还是数据搬运速度不够。本篇论文也提到过使用 simd 去存储输出张量中连续的元素可以加快卷积的速度。另一个方法就是按照输入张量通道展开，之前看到过高性能零内存卷积时提到过他们的数据布局按照通道展开，我当时就想着通道能被 8 整除的可以按照通道 8 展开，这样可以避免填充，减少寄存

器做多于不必要的计算。从数据局部性来说，按照通道展开后，输入张量相邻窗口的之间所重用的元素数量提高了 8 倍。对于卷积核较小，并且输入张量宽度也小的 benchmark 来说，可以通过提高通道展开的倍数从而提高性能。本文的做法是把这两种都结合，即把输入张量按照通道展开，也使用了 simd 去存储了输出张量的元素。至于代码生成器我觉得后面肯定需要了解一下，因为现在我的做法是根据输出张量宽度维度大小的不同，我把每个 benchmark 分了不同的块大小，因此对于每个 benchmark 都有不同的方法，后面需要统一成一个卷积方法。