Title: Deep Tensor convolution on multicores

JX-Ma

2024/9/7

ABSTRACT

Deep Convolutional Neural Networks Improve the Ability of Video Image Analysis But on GPUs, it is limited by the size of the graphics memory, while on CPUs, although the memory size is sufficient, the speed is slow. This article mainly focuses on extending and optimizing the Winograd convolution algorithm to N dimensions for CPUs.

INTRODUCE

This section mainly introduces the successful application of convolutional neural networks in some image processing problems, followed by the advantages of network depth, and then proposes its performance limitations, mainly due to the limited memory capacity on GPUs. Finally, to address this issue, the method proposed in this article is to optimize the implementation of N-dimensional convolution on multi-core CPU systems to bypass GPU memory limitations.

PRIOR ART

This section mainly introduces two existing convolution algorithms, one is fast vector convolution and the other is Minimal winograd convolution

the fast vector convolution include three step, Step (1) is implemented by the kernel and data trans-forms, Step (2) by their transformed element-wise product and Step (3) by the final inverse transform. and for winograd, only a few handmade Winograd style algorithms have been implemented as fast CPU or GPU primitives .

Deep Tensor Convolution

This section define N -dimensional convolution and describe how existing fast algorithms can be extended to this general case.

Consider a network where for each layer i, kernel j and channel m, the kernel weights G(i,j,m) = (g p,q,r) and resulting feature map D(i,j) = (d x,y,z) are both 3D tensors. This calculation can be expressed element-wise as:

$$d_{x,y,z}^{(i+1,j)} = f(b^{(i,j)} + \sum_{m} \sum_{p,q,r} g_{p,q,r}^{(i,j,m)} d_{x+p,y+q,z+r}^{i,j})$$

it was introduced how to amortize transformation costs. On GPUs, conversion costs can be manually reduced, but not on CPUs because the memory limit is relatively small and CPUs can use the sparse structure of matrices to accelerate calculations. Therefore, for reducing conversion costs on CPUs, the sparsity of matrices can be used.

Optimizing for CPU Architecture

This section mainly introduces how to optimize convolution computation on CPU architecture. The article introduces many algorithm techniques that can be applied to reduce the number of computations required for convolution. Afterwards, it introduces how to utilize the utilization rate of a single core CPU, mainly by using vector registers and FMA instructions reasonably. Increasing utilization on multiple cores requires maximizing L3 cache utilization.

Summary

This article mainly explores the application of deep convolutional neural networks. The main work of the article is to optimize the Winograd convolution algorithm on the CPU architecture to adapt to multidimensional situations. Afterwards, some optimizations on the CPU architecture are introduced, among which the most important ones are to use simd reasonably to achieve optimization on a single core and maximize the use of L3 cache to achieve optimization on multiple cores.