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

JX-Ma

2024/8/10

ABSTRACT

This paper present two high-performance implementations of the convolution operator via the direct algorithm. the one have zero-memory overhead, and another type of workspace used is much smaller than im2col+gemm.

INTRODUCE

This section introduce the concept of convolution (operator), then propose the im2col disadvantage that it need a lot of memory.

this paper's work as follow:

1. it reserve the standard NHWC layout of input tensor and only specially packing for the filter tensor.
2. They proposed an alternative blocking variant that packages a small portion of the input tensor into a buffer during convolution operations to ensure that both the filter and input tensor use unit step memory during microkernel execution.
3. finally, they use GoogleNet and ResNet-50 to represent the experiment result, the speed of new algorithm is 13% to 85% faster than the method of basing on im2col.

Related work

This section introduce four convolution method,1. the direct algorithm, 2. the lowering approach 3,base on FFT algorithm, 4. base on Winograd convolution.

for direct convolution, the Zhang et al work include selecting appropriate data layout by input tensor and filter tensor. this transform occur before the DNN

first level or after last level, so this method not widely apply. this paper improved it.

Li et al. introduced a block hardware friendly direct convolution implementation that completely avoids off chip memory transfer of intermediate feature maps on devices with limited memory, such as FPGA (Field Programmable Gate Array) or MCU (Microcontroller Unit).

Zlateski et al propose the meta-programming techniques, like auto-tuning compilation stacks.

Algorithms for the direct convolution

This section introduce simple implement of direct convolution. next, Introduce some optimization of direct convolution.

Attention, this section adopt data layout of input and output are NHWC, and the data layout of filter tensor is HWCN.

the object of blocking is batch of filter tensor, it define the batch of filter tensor size inputed by user, they vectorization in batch of filter tensor, and use it as the last layer of the loop.

the loop order is N(output->batch) --> H_o (output->height) --> H_f (filter->height)--> W_f -->C_i --> W_o --> C_o

next, it use $C_{O,b}$ and $W_{o,b}$ to blocking channel and width of output tensor. it define a micro-kernel and size is $C_{o,b} \times W_{o,b}$, now, the loop order become N,C_o , C_i H_o, W_o, H_f

if we intend to achieve it, we assume the size of micro-kernel is 4 x 8, we need 4 register to store the output in micro, (we use the 256 bit register to store 8 float data), they store separately per row in micro-kernel, for input tensor, if the stride is 1, we need four element in width, we need 4 register to store them, The 8 elements stored in each register are the same. for filter tensor, we only need one register to store 8 element that are continuous in batch.

the 4 element of input tensor in micro-kernel interval is in units of row stride, we can reduce the $C_{i,b}$ to ensure they in the same cache line.

New blocked NHWC-preserving algorithms for the direct convolution

This section introduce two new algorithm.

in new algorithm A, introduce how to allocate vector registers for micro-kernel to maximize the use of vector registers.

The new algorithm B change the row stride to 1 by pack, Make the elements of input tensor in the micro-kernel continuous.

Experimental results

this section describe experimental environment, includes, Hardware setup, Data setup, and using algorithm ,the performance evaluation mainly include the Gflops and memory usage.

Summary

The direct convolution proposed in this paper differs from our previous work in the following ways:

1. This paper in usage of the data layout of filter tensor is HWCN, the data layout of output tensor is NHWC. Our previous data layout using filter tensor was NHWC, and the data layout of output is NCHW.

1. The dimension of vectorization proposed in this paper is the batch of filter tensor, and we vectorize the width of filter tensor.

2. we make the micro-kernel is one-dimensional (the width of output tensor), this paper use the micro-dimensional is two-dimensional (width and channel of output tensor).

Because the output tensor of the first convolutional layer needs to be used as the input tensor for the next layer, our algorithm input tensor need to transform in every layer, and the algorithm only need a transform, because the data layout of the input tensor remains unchanged after convolution calculation. in the convolutional computation, we have better locality data for input tensor ,they keep the locality of input tensor by packing.

There are two main reasons why we select one-dimensional micro-kernel.

One is the number of registers is limited, it need much register to use two-dimensional micro-kernel.

and another is the data continuity, for instance, We store two rows of elements in the output tensor. If the width of the output tensor is small, then a cache line may contain the first addresses of two rows. At this point, only one cache read is needed, but when the output tensor width is large, cache miss may be detected, and the cost of data reading is high. This issue was also mentioned in this article.