Direct Convolution: Performance Effects of Loops Ordering and Parallelization

Mirco Mannino[a,*], Andrea Mondelli[b], Sandro Bartolini[a]

[a]*Department of Information Engineering and Mathematics, University of Siena, via Roma 56, Siena, 53100, Italy*

[b]*Huawei Technologies Research Development (UK) Ltd, Unit 101, Science Park, Milton Road, Cambridge, CB4 0FY, England, United Kingdom*

[*]*mannino@diism.unisi.it*

Nowadays Convolutional Neural Networks (CNNs) are widely used in several fields (e.g., computer vision, bioinformatics, natural language processing). Popular deep learning frameworks implement convolution operation through a transformation of tensors into matrices (i.e., *im2col* operation) and then use highly optimized matrix-multiplication routines. The main drawbacks of this method are the memory overhead used to transform tensors into matrices and lack of scalability [1]. In order to avoid latter problems, direct convolution techniques (i.e., methods that do not use additional memory to perform convolution) are drawing increasing attention.

On the other hand, direct convolution, if not properly implemented, may incur performance degradation due to non-continuous memory accesses or not full utilization of CPU functional units.

In this work, we investigate the effects of direct convolution in terms of memory access and scalability, with the goal of thoroughly understanding the aforementioned behaviors as to steer the definition of dedicated functional units and/or ad-hoc accelerating structures in a RISC-V CPU [2].

Direct convolution can be implemented as a set of nested loops around an accumulation operation on output elements. In particular, the order in which the cycles are arranged does not alter the result but can change the performance. Additionally, we can say that convolution operation is potentially highly parallelizable, since the computation of each output element is independent from other elements.

Aiming to performance, we also use vectorized instruction, in line with already proposed ones for RISC-V CPUs [2], and we are evaluating to extend such ISA for an even better support of direct convolution.

We carefully select 3 orders of loops, based on tensors' memory layout, and we analyze performance and scalability in multi-threaded scenario. HWC memory layout is selected and parallelization is done along height dimension of the output, therefore a portion of output rows is assigned to each thread.

Considered loop orders exhibit similar performance but different loads on the cache hierarchy since each order generates different memory access pattern. Both in mono-threaded and multi-threaded scenarios, the full potential of cores is not exploited due to memory hierarchy pressure and in the latter case also interference between threads plays a crucial role.

This work focuses on the implications of the proposed mechanism in an open architecture such as RISC-V, highlighting the challenges of the current implementation of vectorized instructions in RISC-V, along with the implications of the RISC-V architecture on the design flexibility of the cache memory hierarchy. In particular, the ongoing work focuses on the definition of ad-hoc computational and memorization units in RISC-V, supporting direct convolution workload.

[1] J. Zhang, F. Franchetti, and T. M. Low, "High performance zero-memory overhead direct convolutions," in *Int. Conf. on Machine Learning*. PMLR, 2018, pp. 5776–5785.

[2] "The risc-v instruction set manual volume i: User-level isa document version 2.2." [Online]. Available: https://riscv.org/wp-content/uploads/2017/05/riscv-spec-v2.2.pdf