An Efficient Hardware Accelerator to Handle Compressed Filters and Avoid Useless Operations in CNNs

Adrián Alcolea[1], Javier Olivito[1], Javier Resano[1]

[1] Grupo de Arquitectura de Computadores de la Universidad de Zaragoza
Instituto de Investigación en Ingeniería de Aragón (I3A)
Universidad de Zaragoza, Mariano Esquillor s/n, 50018, Zaragoza, Spain.
Correos electrónicos: alcolea@unizar.es, jolivito@unizar.es, jresano@unizar.es

Abstract

Due to sparsity, a significant percentage of the operations carried out in Convolutional Neural Networks (CNNs) contains a zero in at least one of their operands. Different approaches try to take advantage of sparsity in two different ways. On the one hand, sparse matrices can be easily compressed, saving space and memory bandwidth. On the other hand, multiplications with zero in their operands can be avoided.

We propose the implementation in an FPGA of an architecture for CNNs capable of taking advantage of both, sparsity and filter compression.

Introduction

A significant percentage of the operations carried out in CNNs contains a zero in at least one of their operands. In activation matrices, sparsity is generated by the use of non-linear activation functions such as ReLU. In addition, pruning techniques also generate zero-elements in network filters, so zero-products increase significantly [1] [3].

On another note, the use of compressed filter matrices reduces the data that must be read from the off-chip memory, and maximizes the data that can be stored on-chip, which is very important since the access to external memories is usually the main energy consumption factor, and often a performance bottleneck [2].

Specific hardware accelerators could manage both, compression and zero-operations avoiding, to increase CNNS performance and energy efficiency.

Objectives

Our goal is to manage compression and avoid useless operations to increase CNNs performance and energy efficiency}, which requires:

- Design hardware support to decompress the filters matrices on the fly and carry out only non-zero operations.
- Integrate it into a proof of concept CNN architecture implemented on an FPGA.
- Evaluate both the benefits and the overheads generated.

Compression scheme

We propose a compression scheme that includes a bit for each filter value pointing out whether it is zero or not. We achieve a better compression ratio for most filters than the most common schemes.

As an example, we assume a 5 x 4 matrix with 60% sparsity and an 8-bit data size. Compression schemes with a list for the number of zeros need 2 x 8 x 8 = 128 bits, while our scheme needs 5 x 4 + 8 x 8 = 84 bits (See Figure 1).

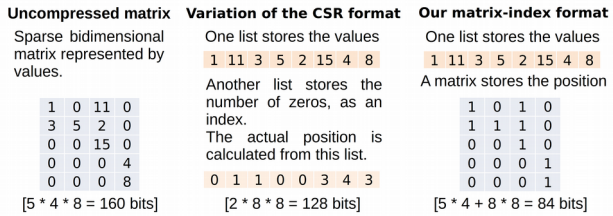

Figure 1. Comparison of compression schemes

Architecture

In the proposed architecture, N convolutions are processed in parallel in N processing units. Each of them targets a different filter and stores the

compressed filter information locally, whereas the activation memories are shared (See Figure 2).

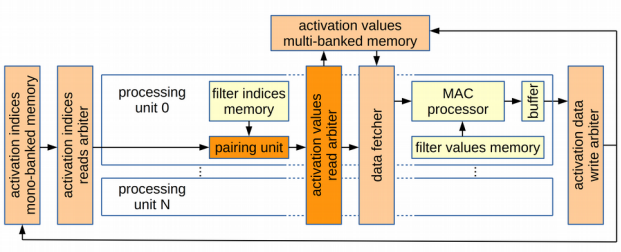

Figure 2. General architecture

The "pairing unit" takes advantage of the indices structures to efficiently find non-zero pairs. While the "activation values read arbiter" decides the fetching order to manage access conflicts.

Pairing unit

This module processes the matrices of indices of the activation map and the filters, identifying which computations must be carried out (those that do not have a zero in their operands). Its main function is to take advantage of the indices structures to efficiently find non-zero pairs.

Then it uses the filter values count and the convolution loop indices to generate the actual memory addresses of the filter and activation values in the current pair (See Figure 3).

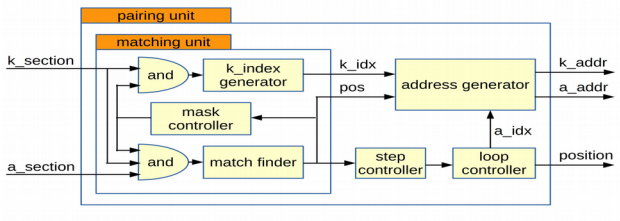

Figure 3. Pairing unit

Activation values read arbiter

Activation values are stored in a shared memory, so access conflicts between multiple processing units may arise (See Figure 4).

One approach to maintain the requested bandwidth consist in duplicating the number of pairs requested per processing unit, so the arbiter can take some decisions on the fetching order.

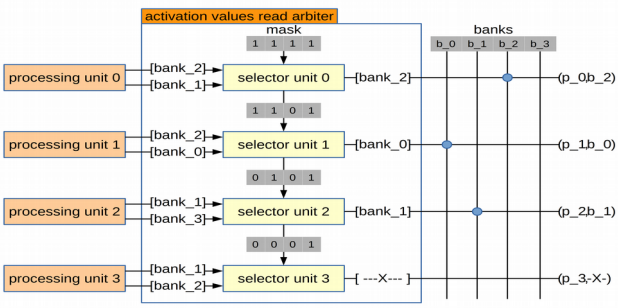

Figure 4. Activation values read arbiter

Contributions

- Our compression scheme can be used to efficiently pair the non-zero data.
- It also achieves better compression rate than state of the art.
- The proposed hardware pipeline handles compressed filters and discard all the operations where at least one operand is zero.
- We present a real implementation on an FPGA. It allows an accurate evaluation of the performance and energy efficiency of our proposal.

REFERENCES

[1]. ALBERICIO, P. JUDD, T. HETHERINGTON, T. AAMODT, N. E. JERGER, and A. MOSHOVOS. Cnvlutin: Ineffectual-neuron-free deep neural network computing. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)* , pages 1-13, June 2016.

[2]. SONG HAN, HUIZI MAO, and WILLIAM J. DALLY. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *Computing Research Repository*, eprint arXiv: 1510.00149, 2015.

[3]. SONG HAN, JEFF POOL, JOHN TRAN, and WILLIAM DALLY. Learning both weights and connections for efficient neural network. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1135-1143. Curran Associates, Inc., 2015.

ACKNOWLEDGEMENT

This work was supported in part by grants TIN2016-76635-C2-1-R (AEI/FEDER, UE) and Consolider NoE TIN2014-52608-REDC (Spanish Gov.), gaZ: T48 research group (Aragón Gov. and European ESF), and HiPEAC4 (European H2020/687698).