A Data-Reuse Aware Accelerator for Large-Scale Convolutional Networks

Maurice Peemen, Bart Mesman, and Henk Corporaal
Department of Electrical Engineering, Electronic Systems Group
Eindhoven University of Technology, Eindhoven the Netherlands
m.c.j.peemen@tue.nl, b.mesman@tue.nl, h.corporaal@tue.nl

Abstract

This paper presents a clustered SIMD accelerator template for Convolutional Networks. These networks significantly outperform other methods in detection and classification tasks in the vision domain. Due to the excessive compute and data transfer requirements these applications benefit a lot from a dedicated accelerator. The proposed accelerator reduces memory traffic by loop transformations such as tiling and fusion to merge successive layers. Although fusion can introduce redundant computations it often reduces the data transfer, and therefore can remove performance bottlenecks. The SIMD cluster is mapped to a Xilinx Zynq FPGA, which can achieve 6.4 Gops performance with a small amount of resources. The performance can be scaled by using multiple clusters.

1. Introduction

Nowadays embedded computer architectures embrace customization to realize the best possible performance within a low power envelope. This trend has successfully brought complex tasks such as HD video playback to Smartphones, and in the extreme to wearable devices such as Smartwatches or Glasses. A current trend is to improve intelligence of these so called smart devices by releasing applications for speech, image, and video recognition e.g., Siri (speech), and Impala (image categorization).

Recently the state-of-the-art for such complex classification tasks is improved significantly by the availability of huge amounts of training data combined with the use of Deep Convolutional Networks [6]. These networks are all-in models that perform Feature Extraction and Classification in a single trainable model, consisting of 5 to 8 successive layers of processing [7]. Mapping these large-scale models to embedded processors is very challenging. Firstly, the compute workload is huge e.g., the winning network in the ILSVRC-2012 [6] image classification task has $6.4 * 10^8$ connections, resulting in $6.4 * 10^8$ Multiply Accumulate (MACC) operations to process a RGB image patch of 224x224 pixels. Secondly, the data transfer requirements are huge, this network has $6 * 10^7$ parameters while the intermediate neuron data in successive layers holds $7.7 * 10^5$ values. To make things even worse, if this network is used for a detection task on larger images the compute and transfer requirements increase substantially.

1.1. Prior work

Due to the success of Convolutional Networks other works have mapped this model to different platforms. Often GPUs are used to accelerate workload that involves the training or detection tasks [4, 8, 3]. With massive amounts of parallelism and huge data transfer rates a GPU platform seems a natural choice for Convolutional Networks. However, from an energy perspective a GPU is far from ideal for embedded devices. As a result, a number of researchers have successfully proposed dedicated accelerators for FPGA or ASIC platforms [1, 5, 9, 2]. Most of these works focus on efficient implementation of the computational primitives by using systolic implementations. Consequently, data transfer is considered secondary and solved by advanced DMA or neglected for simplicity.

The most recent works admit that data transfer should be a first order concern to achieve efficient processing [9, 2]. Both works consider a layer of a Convolutional Network as a deep nested loop on which transformations such as interchange and tiling can be applied to improve data locality. However architectural design choices prevent merging of successive layers, which forces them to transfer the hidden layer outputs to- and from-external memory.

1.2. Contributions

To reduce data transfer requirements further we propose a clustered SIMD architecture with local buffers that can merge successive layers. Layer merging is performed by adding loop fusion to the list of optimization, which is possible due to our flexible SIMD architecture. More specifically we make te following contributions:

- We present loop fusion as an additional transformation to further reduce data transfer for Convolutional Networks.
- We propose a clustered SIMD template to exploit the benefits of merging network layers by fusion.
- We mapped the cluster to a Xilinx Zynq FPGA to evaluate performance, and resource usage.

2. Fusing Network Layers

Deep Convolutional Networks, as depicted in figure 1, can be written as a series of deep nested loops representing successive layers in the algorithm. The order of computations in layers is not fixed, so it can be changed by loop transformations. The upper part of figure 1 outlines tiling of neurons in a single layer to improve data locality. In addition, we propose fusion

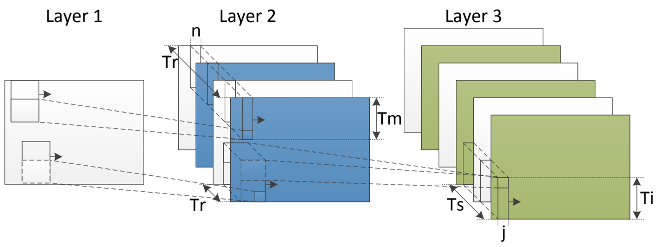


Figure 1: Graphical overview of Convolutional Network layers. Layer 1 hold an input image which is convolved by different filters to extract features. Tiling transformations can improve data locality in a layer, which can also span multiple layers removing the need to store intermediate layers externally.

to prevent external storage of a hidden layer. However, for fusion to be valid dependencies over layers must be taken into account. In figure 2, we show the data access patterns of our pipelined fusion approach, which involves redundant computations to solve dependencies. Figure 2, shows that for small tiles the redundant work is significant; therefore tile size must be optimized to minimize redundancy. If applied correctly, fusion reduces data transfer between 1.5 - 2x for our example networks.

3. SIMD Accelerator Template

To balance flexibility and efficiency we use a clustered SIMD based accelerator. This gives us programmable control over data streams, and in addition we can split convolutions in separate MACC operations. As a result we have more scheduling freedom compared to the bigger convolution primitives. Figure 3 outlines the pipelined cluster template, which can process a stream of overlapping tiles as shown in figure 1-2. Key in this design is the dual-port vector reuse buffer, which enables flexible local storage of coefficients, inputs, and outputs.

For control a distributed micro-program is used, which controls the individual pipeline stages. The instruction sequences

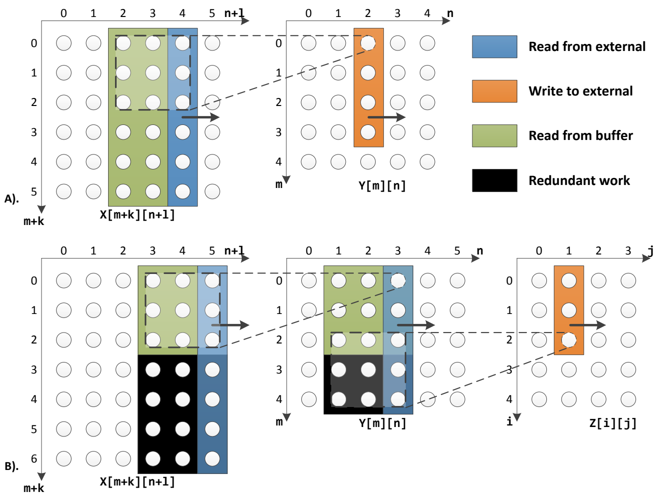


Figure 2: A). Data access, and buffer storage pattern when only inter-tile reuse optimization is applied. B). Access pattern when layer fusion used, introducing redundant compute workload, but without external storage of layer Y results

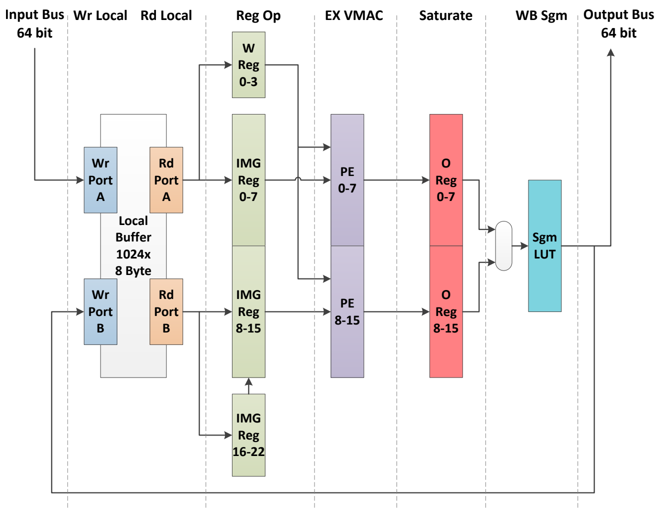


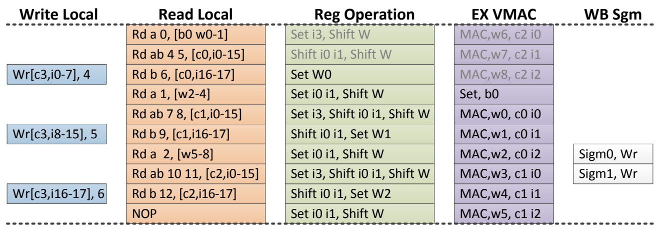
Figure 3: Pipelined data path for a SIMD compute cluster. The cluster contains a 1024 entry local buffer, a vector register file with shift functionality, and 16 MACC units. The activation LUT has a feedback to the local buffer to enable merging of successive layers.

are very similar to software-pipelining for VLIW processors. An example instruction sequence is given in figure 4.

4. Conclusions

We presented layer merging by nested loop fusion as an additional transformation to improve data locality for Convolutional Networks. In our example networks fusion can reduce external memory traffic by 1.5 to 2 times. In addition we presented a clustered SIMD accelerator that can take advantage of the improved data locality. We mapped the cluster to a Xilinx Zynq FPGA board on which it can achieve 6.4 Gops, at 200 Mhz. The cluster design requires 16 DSP blocks, 4 BRAMs, 660 LUTs and 527 Flip-Flops, which is only a small amount (7.3%). As a result we can scale performance by the use of multiple clusters that share a dedicated DMA for efficient memory access.

Although this design is mapped to FPGA it can be implemented as accelerator core in a SOC. This would enable users of Smartphones to run a range of complex vision applications without depleting their battery within a few minutes.



Write Local	Read Local	Reg Operation	EX VMAC	WB Sgm
	Rd a 0, [b0 w0-1]	Set i3, Shift W	MAC,w6, c2 i0	
	Rd ab 4 5, [c0,i0-15]	Shift i0 i1, Shift W	MAC,w7, c2 i1	
Wr[c3,i0-7], 4	Rd b 6, [c0,i16-17]	Set W0	MAC,w8, c2 i2	
	Rd a 1, [w2-4]	Set i0 i1, Shift W	Set, b0	
	Rd ab 7 8, [c1,i0-15]	Set i3, Shift i0 i1, Shift W	MAC,w0, c0 i0	
Wr[c3,i8-15], 5	Rd b 9, [c1,i16-17]	Shift i0 i1, Set W1	MAC,w1, c0 i1	
	Rd a 2, [w5-8]	Set i0 i1, Shift W	MAC,w2, c0 i2	Sigm0, Wr
	Rd ab 10 11, [c2,i0-15]	Set i3, Shift i0 i1, Shift W	MAC,w3, c1 i0	Sigm1, Wr
Wr[c3,i16-17], 6	Rd b 12, [c2,i16-17]	Shift i0 i1, Set W2	MAC,w4, c1 i1	
	NOP	Set i0 i1, Shift W	MAC,w5, c1 i2	

Figure 4: Assembly description of the steady-state program that computes a 3x3 convolution in a feature map. Note that control is distributed over the successive stages in the architecture pipeline. Image read and write actions with the local buffer use modulo addressing over steady-state iterations to maximize the efficiency of the memory.

References

[1] S. Chakradhar, M. Sankaradas, V. Jakkula, and S. Cadambi, “A dynamically configurable coprocessor for convolutional neural networks,” in *ISCA*, 2010, pp. 247–257.

[2] T. Chen, Z. Du, N. Sun, J. Wang, C. Wu, Y. Chen, and O. Temam, “Diannao: a small-footprint high-throughput accelerator for ubiquitous machine-learning,” in *Proceedings of ASPLOS ’14*. ACM, pp. 269–284.

[3] D. C. Cireşan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber, “Flexible, high performance convolutional neural networks for image classification,” in *Proceedings of AAAI ’11*, 2011, pp. 1237–1242.

[4] A. Coates, B. Huval, T. Wang, D. Wu, B. Catanzaro, and N. Andrew, “Deep learning with cots hpc systems,” in *Proceedings of The 30th International Conference on Machine Learning*, 2013, pp. 1337–1345.

[5] C. Farabet, B. Martini, B. Corda, P. Akselrod, E. Culurciello, and Y. LeCun, “Neuflow: A runtime reconfigurable dataflow processor for vision,” in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*, 2011, pp. 109–116.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks.” in *NIPS*, vol. 1, no. 2, 2012.

[7] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, 1998.

[8] M. Peemen, B. Mesman, and H. Corporaal, “Efficiency optimization of trainable feature extractors for a consumer platform,” in *Advances Concepts for Intelligent Vision Systems*. Springer, 2011, pp. 293–304.

[9] M. Peemen, A. A. Setio, B. Mesman, and H. Corporaal, “Memory-centric accelerator design for convolutional neural networks,” in *Computer Design (ICCD)*. IEEE, 2013, pp. 13–19.