POLITECNICO DI TORINO
Repository ISTITUZIONALE

WINNER: a high speed high energy efficient Neural Network implementation for image classification

Original
WINNER: a high speed high energy efficient Neural Network implementation for image classification / Antonietta, S.D., Coluccio, A., Turvani, G., Vacca, M., Graziano, M., Zamboni, M.. - (2019), pp. 29-32. (2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS)) [10.1109/ICECS46596.2019.8965001].

Availability:
This version is available at: 11583/2787904 since: 2020-03-17T09:32:10Z

Publisher:
IEEE

Published
DOI:10.1109/ICECS46596.2019.8965001

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
IEEE postprint/Author's Accepted Manuscript

©2019 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collecting works, for resale or lists, or reuse of any copyrighted component of this work in other works.

(Article begins on next page)

04 November 2026

WINNER: a high speed high energy efficient Neural Network implementation for image classification

Simone Domenico Antonietta, Andrea Coluccio
Giovanna Turvani, Marco Vacca, Mariagrazia Graziano, Maurizio Zamboni
Department of Electronics and Telecommunications, Politecnico di Torino, Italy

***Abstract*—Nowadays, Neural Networks are often used in many applications to solve very complex tasks like speech recognition or image classification. In applications like image recognition, time is a critical constraint and a Neural Network must be able to correctly classify an image in a very short time: in these cases, hardware accelerators are necessary. In this work, a hardware implementation of a Neural Network is proposed. As a reference model, AlexNet has been chosen and implemented as an ASIC, by adopting an In-Memory computing design. The concept of In-Memory is based on the idea of placing small computational units near the memory element. This choice brings to relevant benefits, reducing the bottlenecks coming from a Von Neumann's system (such as fetching latency and energy inefficiency due to the communication between logic and memory). The architecture has been synthesized on a 45nm CMOS technology and highlights remarkable results, with very high frame-per-second (FPS) (comparable to binary neural network implementations) combined with low values of Energy/FPS and a maximum operating frequency of 281MHz. The system is also partially reconfigurable, enabling the mapping of different kind of neural networks.**

I. Introduction

Recently, Deep Neural Networks have been used in almost any kind of applications, because of their versatility: they are able to perform complex tasks (such as speech recognition or image classification), reaching good levels of accuracy. The most used neural network model is a Convolutional Neural Network (CNN), which represents an efficient way to classify large amounts of data: one of them is AlexNet [1], that has been chosen as reference pre-trained model. It is a deep Neural Network made up by 8 main layers and used as image classifier, allowing to recognize up to 1000 categories of images, with a TOP-5 accuracy of 84.7%. Neural Networks are quite slow if implemented on a software solution due to their complexity, so an hardware accelerator can greatly improve the performance.

In this paper a reconfigurable hardware implementation of a Neural Network, focused on an In-Memory approach, is presented. In a standard Von Neumann's architecture, the CPU spends a big part of its computational time waiting for the data arriving from the memory, since the processor operates at an higher speed than the memory by itself, causing a non-negligible latency. By merging memory and computational units in a Logic-In-Memory (LIM) solution, it is possible to overcome these limitations: for this reason, the architecture is called WINNER, acronym for Weight In Memory Neural Network Embedded Ram. LIM/near-memory and classical architectures are analyzed from several works like [2], [3] based on a binarized approach and innovative technologies (RRAMs) or [4] that implements a XNOR-Net in a Wide IO2 DRAM architecture, using stacked DRAM memories and a logic layer to perform the computations. Other solutions, that are not based on an In-Memory approach are considered, such as Eyeriss [5], YodaNN [6] (ASICs) and [7] (FPGAs). Chain-NN [8] implements a fixed-point neural network architecture based on a chain of Processing Elements (PEs), reaching very good perfomance. WINNER has been synthesized on a 45 nm CMOS technology and state-of-the-art comparisons in terms of frame-per-seconds (FPS) and cost function Energy/FPS are proposed: our work reaches very high FPS (1326 FPS) and low Energy/FPS (40uJ/FPS) values combined with a TOP-5 rate of 84.7% (the same as [1]), opening the way for high speed applications that require high accuracy. Our architecture is reconfigurable, meaning that different kind of neural networks can be implemented.

II. Convolutional neural networks: AlexNet

The neural network that we choose to implement is the AlexNet, which is a deep convolutional neural network able to classify complex RGB images of 227x227 pixels, winner of the ILSVRC 2012 competition [1]. We choose this network because, while it is not the actual best available, it has a good trade-off between accuracy and complexity, so it is particularly suited for an hardware implementation. AlexNet is composed of several layers as shown in Fig. 1: Convolutional (CONV), Max pooling (POOL), Cross-Channel Normalization (CROSS) and Fully Connected (FC). There are 5 CONVs, 3 FCs and 3 POOLs following the first, the second and the fifth CONVs. After the last FC layer, there is a 1000-way Softmax computation, that has been approximated with Max function, since WINNER is used only for inference purposes.

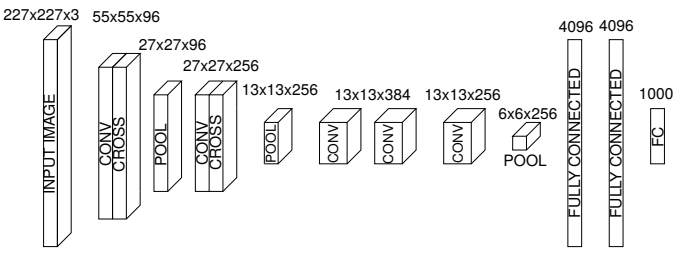


Fig. 1. AlexNet model

The complexity of each layer can be measured by the total number of multiplications and the number of neurons, that can be computed as $\#\text{neurons}_{i_{th}} = (D_x \times D_y \times D_z)_{(i_{th}+1)}$. D_x, D_y and D_z are the output feature maps (OFMAPs) sizes and i_{th} indicates the i-th layer. For the first CONV, the total number of neurons is equal to $\#\text{neurons}_1 = (55 \times 55 \times 96) = 290400$. More details about AlexNet model's parameters are reported in [1].

III. Circuit Architecture

Our architecture implements the whole neural network, in such a way that a good trade-off between performance and hardware requirements is reached.

A. Optimizing the number of neurons

In this paper, the term "neuron" is intended as a computational block that performs the sum-of-products of the inputs with the corresponding weights, following the Equation 1:

$$Y = \sum_{i=1}^{N} I_i \times W_i + Bias = \sigma + Bias \tag{1}$$

where N is the total number of neuron's inputs, I_i is the input, multiplied by the corresponding weight W_i. Since neurons are composed essentially of multiply-and-accumulate (MAC) PEs, their quantity has to be optimized, considering:

- Area-delay trade-off: area has to be as low as possible reducing the number of employed neurons, without degrading too much the performance;
- Neurons of the FC layer are the same PEs as CONV layer's ones, so the hardware can be reused;
- Good level of parallelization, in order to reach acceptable FPS values.

From [1], the maximum number of filters employed per layer are equal to 384: since each neuron represents an entire filter computation (as shown in Fig. 2), by choosing 384 neurons, all the kernels can be processed in parallel. However, in the tiny example reported in Fig. 2, there are only 4 inputs defined by the kernel size, while in AlexNet the maximum kernel size is 9216 ($6 \times 6 \times 256$, that is the dimension of the first FC input), meaning that a single neuron should have at least 9216 fixed point inputs: this is impracticable, and a trade-off between number of inputs and parallelization has to be considered.

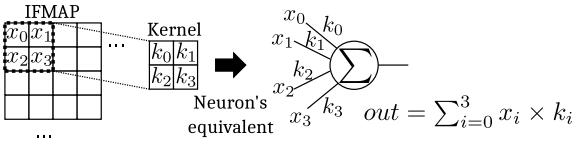


Fig. 2. Example of the correspondence between neuron and filter computation.

To reduce the complexity, in WINNER we have chosen a number of inputs equals to 64, and for those layers which require more than 64 contemporary inputs, the algorithm has been serialized requiring $\#\text{steps}_{\text{neuron}} = \#\text{inputs}_{\text{neuron}}/64$ steps to be completed.

B. Architecture overview

As depicted in Fig. 3(a), WINNER is composed of two blocks.

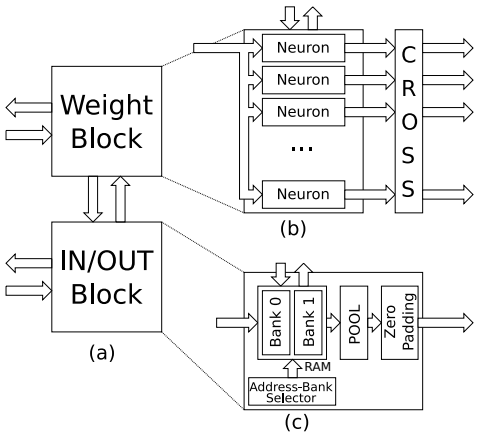


Fig. 3. WINNER architecture (a), Weight-Block (b) and IN/OUT-Block (c) schemes

1) Weight-Block: this is the part of the architecture where the neural computation is executed, which is shown in Fig. 3(b). It contains all the 384 neurons with 64 input bytes and 384 output bytes. Data are represented on 8 bits, in particular Q8.0 for inputs/outputs, Q2.6 for weights and Q6.2 for FC results, where the notation Qx.y indicates the total number of bits dedicated to the integer part (x) and fractional part (y). Lastly, it contains also the CROSS layer, connected to all the neurons' outputs. Neurons require lots of resources to be implemented: for this reason, in WINNER architecture we have developed an optimized version of a neuron block, which is reported in Fig. 4.

- Each neuron has 64 "WordLines" (indicated as WL_i in Fig. 4). Each "WordLine" is essentially a 2595 bytes register. This number, if multiplied by 64 WLs and 384 neurons, gives the total number of stored weights inside the architecture as $\#\text{Stored weights} = 384 \times 64 \times 2595 = 63774720$. $\#$Stored weights must be almost equal to the total number of weights required to perform the entire CNN algorithm, that in the AlexNet model are $\sim$ 60 millions [1]. The selectors (trapezoids indicated with ① in Fig. 4), implement a Wired-OR function and they select the corresponding weight-set to be processed in the current algorithm step.
- Parallel Prefix Units (PPUs) (boxes indicated by ② in Fig. 4) implement a radix-2 modified Booth encoding procedure to compute the multiplication of the results.
- Adders (③ in Fig. 4) are used to sum two WL contributions together to evaluate the intermediate convolution result. Each partial sum coming from ③ blocks, is added in a final adder tree to provide the partial convolution result. Lastly, an accumulation adder computes the final result, by performing a selection with the multiplexer (indicated as ④ in Fig. 4) between σ and Bias, depending on

the evaluation step, to compute the final result expressed in Equation 1.

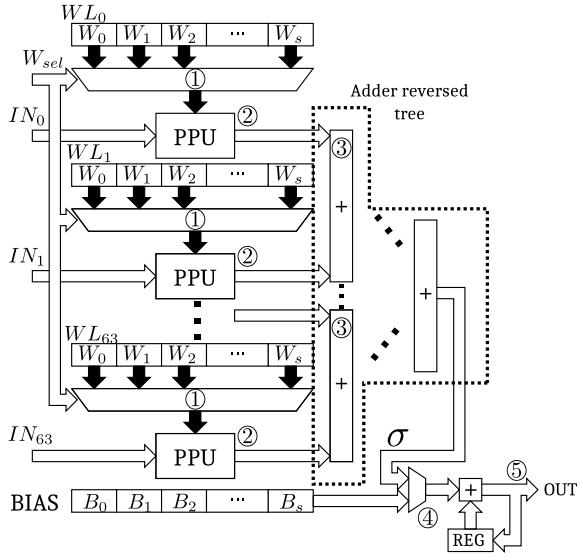


Fig. 4. WINNER neuron, an optimized version of a neuron

The CROSS layer, reported in Fig. 3(b), has a very complex formula for an hardware implementation, so an approximation has to be considered. From [1], it is equal to:

$$y_{cross} = \frac{x}{\left(K + \alpha \times \frac{\sum_{i=1}^{n} x_i^2}{WCS}\right)^{\beta}} \tag{2}$$

where x is the input element, $[K, \alpha, \beta]$ are constants (usually equal to $[2, 0.0001, 0.75]$ respectively), WCS is the window channel size (in AlexNet, it is equal to 5) and n is 5 [1]. By performing the Taylor series of the term $(1+x)^{\beta}$, a near-zero approximation is provided:

$$y_{cross} \sim 1 + \beta \cdot x + \frac{\beta \cdot (\beta - 1)}{2} \cdot x^2 \tag{3}$$

This approximation is good only in the range $[0, 1.5]$, otherwise the curve diverges from its original behavior. However, by performing extensive simulations with multiple batches, it has been demonstrated that only 5% of values above 1.5 threshold are present, so this approximation does not heavily affect the accuracy in our hardware implementation, which remains the same as the original paper [1].

2) IN/OUT-Block: this is the part of the architecture shown in Fig. 3(c), where the inputs/outputs of each layer are stored. It provides all the signals to the Weight-Block and can perform POOL and Zero Padding operations if required. It is made by a simple RAM divided into two identical banks. Bank 0 stores the inputs, while the Bank 1 stores the outputs. The address bank selector selects the Bank for read/write operations, depending on the execution step.

IV. Results

The procedure used to estimate the performance starts with a circuit synthesis with Synopsys Design Compiler and a functional simulation with Modelsim, that writes on a SAIF file all the informations regarding the switching activity of each node of the network. This procedure enables a more precise estimation, instead of considering the worst case maximum activity equal to 1. The technology used is the Nangate CMOS 45nm. Synthesis procedure indicates also the critical path delay, which results equal to 3.55ns, meaning that WINNER is able to reach up to 281MHz as maximum operating frequency. In the design, RAMs are synthesized as registers, so the power and area obtained represent an overestimation of a real case, with a more correct model of the memories.

1) In/Out-Block characterization: in Fig. 5 are reported the main contributions to the area and power into the In/Out-Block. Considering the area graph in Fig. 5(a), the most important parts are the logic, composed of the circuit that establishes the output selection, and the comparators, forming together the POOL. As expected, the RAM part has an irrelevant contribution in terms of area, since in the In/Out block only inputs/outputs are stored. Regarding the power distribution in Fig. 5(b), the main contribution is given by the comparators because of their complexity and, secondly, by the logic layer. RAM's contribution is only 4% of the total power, that can be further reduced if a more real memory model is employed.

2) Weight-Block characterization: A single neuron's power and area are evaluated. In Fig. 6(a), the main area contribution is given by the selectors, since 64 of them are required with 2595 inputs of 1 byte each, as discussed in subsection III-B. They have also an important power contribution, secondly only to RAM, according to Fig. 6(b). The computational part (composed of PPUs and adders) and the CROSS layer introduce small overheads in terms of power (27% and <1% respectively).

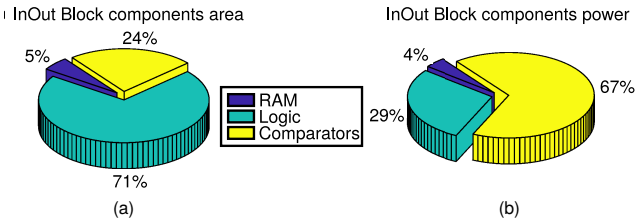


Fig. 5. Power and Area of the In/Out Block internal components

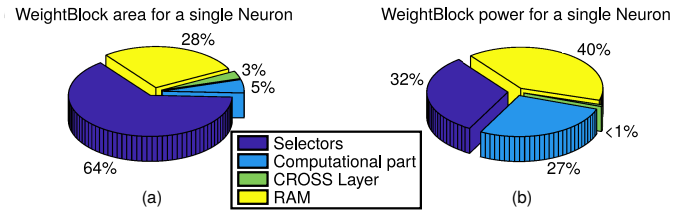


Fig. 6. Power and area of the Weight-Block internal components. Only one neuron is considered in this estimation.

A. Comparison with the state-of-the-art

A comparison with the state-of-the-art implementations of the AlexNet is presented. The WINNER architecture is able

to process a frame in a very short time (0.75ms). To evaluate performance, a cost function Energy/FPS is considered: it indicates how efficient the architecture is to minimize the energy, so it should be as low as possible. Table I summarizes the performance of our architecture, compared with state-of-the-art implementations of the AlexNet neural network. The comparison is made with very different implementations, some of them use a fixed point representation [5][7][3][8], some others a floating point representation [7]. Also, binary implementations are analyzed [3][4][6] and other ones are based on emerging technologies [3]. As can be seen from the results in Table I, our architecture is the third fastest implementation, overcome only by binary implementations that trade speed for lower accuracy. Fig. 7 depicts the relative energy efficiency of each architecture, where the function f on the y-axis is used to obtain a more clear graphical comparison, in fact it transforms the values of Energy/FPS in percentages relative to the XNOR-POP [4], which has the highest efficiency according to Table I. In Fig. 7, WINNER has the fourth highest value: two binary implementations are better, but at the cost of accuracy and the only non-binary architecture that has a better value is the Chain-NN [8], that is however considerably slower. As can be seen, WINNER has around 70% the relative energy efficiency of the XNOR-POP [9], while having better accuracy of 84.7%. Overall our architecture is ideal for all the applications that require an high acquisition speed while maintaining a very good energy efficiency. Furthermore, results can be greatly improved by scaling the technology.

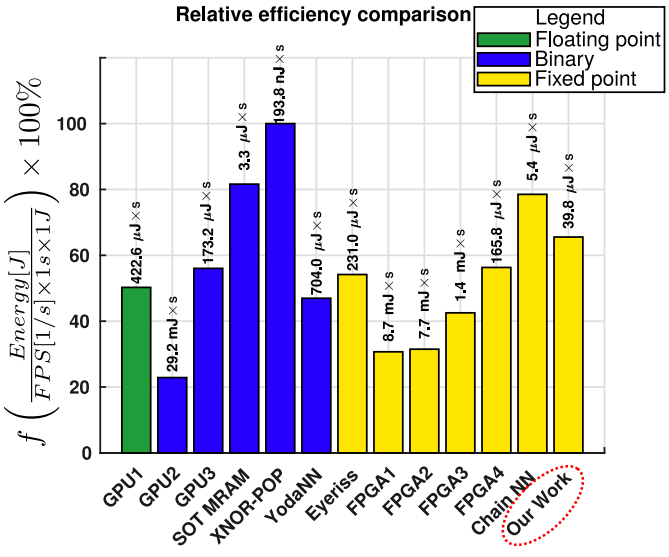


Fig. 7. Relative efficiency comparison. In the plot, the values have been rescaled in percentage; 100% value indicates that the architecture has the highest efficiency. The absolute values are indicated over each bar. For example, WINNER has 70% efficiency compared to the XNOR-POP [4].

TABLE I
STATE-OF-THE-ART PERFORMANCE COMPARISON WITH ALEXNET MODEL. PROCESS TIME IS RESCALED TO BATCH SIZE EQUAL TO 1.

Architecture[a]	Process time[ms]	FPS[1/s]	Energy[mJ/frame]	Impl.[b]
Intel XEON E5-2637 [7]	195.00	5.0	25400.00	*
GPU1 [7]	1.30	769.0	325.00	*
GPU2 [3]	90.00	11.1	324.00	**
GPU3 [3]	0.73	1369.9	237.25	**
SOT MRAM [3]	10.70	93.5	0.31	**
XNOR-POP [4]	0.29	3390.0	0.66	**
YodaNN [6]	2000.00	0.5	0.35	**
Eyeriss [5]	28.825	34.7	8.01	***
FPGA1 [7]	21.61	46.3	402.00	***
FPGA2 [7]	20.10	50.0	384.00	***
FPGA3 [7]	2.56	391.0	77.00	***
FPGA4 [3]	5.94	168.4	27.92	***
Chain NN [8]	3.07	325.4	1.74	***
Our Work	0.75	1326.5	52.77	***

[a]GPU1 = GTX Titan X, GPU2 = NVIDIA Jetson TK1, GPU3 = NVIDIA Tesla K40, FPGA1 = Virtex-7 VX485T, FPGA2 = Stratix-V GSD8, FPGA3 = Virtex-7 VC709, FPGA4 = Xilinx Zynq-7000

[b]Impl. implementation type. *, ** and *** indicate floating point, binary approximation and fixed point architectures respectively.

V. CONCLUSIONS

We have developed an innovative hardware implementation of an AlexNet based on an In-Memory approach. The architecture reaches very good values of Energy/FPS, FPS (this last one is comparable to a binary neural network approximation) and accuracy. By scaling the technology, performance can be greatly improved. As a future work we are trying to modify this architecture, enabling the implementation of other neural networks, keeping at the same time high FPS and high efficiency in terms of energy and accuracy.

REFERENCES

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[2] T. Tang, L. Xia, B. Li, Y. Wang, and H. Yang. Binary convolutional neural network on rram. In *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 782–787, Jan 2017.

[3] D. Fan and S. Angizi. Energy efficient in-memory binary deep neural network accelerator with dual-mode sot-mram. In *2017 IEEE International Conference on Computer Design (ICCD)*, pages 609–612, Nov 2017.

[4] L. Jiang, M. Kim, W. Wen, and D. Wang. Xnor-pop: A processing-in-memory architecture for binary convolutional neural networks in wide-io2 drams. In *2017 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pages 1–6, July 2017.

[5] Y. Chen, T. Krishna, J. Emer, and V. Sze. 14.5 eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. In *2016 IEEE International Solid-State Circuits Conference (ISSCC)*, pages 262–263, Jan 2016.

[6] R. Andri, L. Cavigelli, D. Rossi, and L. Benini. Yodann: An ultra-low power convolutional neural network accelerator based on binary weights. In *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 236–241, July 2016.

[7] Huimin Li, Xitian Fan, Li Jiao, Wei Cao, Xuegong Zhou, and Lingli Wang. A high performance fpga-based accelerator for large-scale convolutional neural networks. In *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–9, Aug 2016.

[8] S. Wang, D. Zhou, X. Han, and T. Yoshimura. Chain-nn: An energy-efficient 1d chain architecture for accelerating deep convolutional neural networks. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, pages 1032–1037, March 2017.

[9] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542. Springer, 2016.