PhD Forum: Impact of CNNs Pooling Layer Implementation on FPGAs Accelerator Design

A. Muñio-Gracia
Institute of Microelectronics
of Seville (IMSE-CNM),
CSIC-Univ. of Seville (Spain)
amunio@imse-cnm.csic.es

J. Fernández-Berni
Institute of Microelectronics
of Seville (IMSE-CNM),
CSIC-Univ. of Seville (Spain)
berni@imse-cnm.csic.es

R. Carmona-Galán
Institute of Microelectronics
of Seville (IMSE-CNM),
CSIC-Univ. of Seville (Spain)
rcarmona@imse-cnm.csic.es

Á. Rodríguez-Vázquez
Institute of Microelectronics
of Seville (IMSE-CNM),
CSIC-Univ. of Seville (Spain)
angel@imse-cnm.csic.es

ABSTRACT

Convolutional Neural Networks have demonstrated their competence in extracting information from data, especially in the field of computer vision. Their computational complexity prompts for hardware acceleration. The challenge in the design of hardware accelerators for CNNs is providing a sustained throughput with low power consumption, for what FPGAs have captured community attention. In CNNs pooling layers are introduced to reduce model spatial dimensions. This work explores the influence of pooling layers modification in some state-of-the-art CNNs, namely AlexNet and SqueezeNet. The objective is to optimize hardware resources utilization without negative impact on inference accuracy.

CCS CONCEPTS

• **Hardware** → **Reconfigurable logic and FPGAs; Hardware accelerators**; • **Computing methodologies** → *Neural networks*.

KEYWORDS

Convolutional Neural Networks, FPGA, hardware acceleration

ACM Reference Format:

A. Muñio-Gracia, J. Fernández-Berni, R. Carmona-Galán, and Á. Rodríguez-Vázquez. 2019. PhD Forum: Impact of CNNs Pooling Layer Implementation on FPGAs Accelerator Design. In *Association for Computing Machinery, September 09–11, 2019, Trento, IT*. ACM, New York, NY, USA, 2 pages. <https://doi.org/XX.XXXX/XXXXXXXX.XXXXXXX>

1 INTRODUCTION

The evolution of Convolutional Neural Networks (CNNs) have relegated traditional image processing algorithms in classification, detection, and segmentation tasks because of their superior performance. Their enhanced precision comes accompanied by higher requirements in terms of memory and computational power, inspiring the development of application-tailored accelerators. With shorter development time and lower cost than ASICs, FPGAs have become preeminent platforms to accelerate CNNs achieving better energy consumption rates when compared with CPUs or GPUs.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ICDSC '19, September 09–11, 2019, Trento, IT
© 2019 Association for Computing Machinery.
ACM ISBN XXX-X-XXXX-XXXX-X/XX/XX...\$15.00
<https://doi.org/XX.XXXX/XXXXXXXX.XXXXXXX>

Table 1: CNN models characteristics

	AlexNet[1]	SqueezeNet v1.1[2]
# Layers (POOL)	3	3
K_P	3	3
S_P	2	2
Top 1% Acc.	57.1	57.2
Top 5% Acc.	80.2	80.3

In practice, the power saving objective translates into the development of data movement minimization strategies including model and data size reduction, as well as data reuse optimization.

The insertion of pooling layers is normally used in CNN topologies to reduce their size and number of MAC operations. Pooling layer parameters vary on a per-model and per-layer basis, forcing accelerator architectures to be reconfigurable. In order to cope with the need of reconfigurability derived of CNN layers diverse structure, Network-on-Chip (NoC) based accelerators constructed with systolic-2D meshes of Processing Elements (PEs) linked by routing blocks are commonly used [3]. For external memory access reduction and data reuse improvement PEs enclose local-buffering elements (P_B) for both *fmaps* and filters [4]. This paper proposes an adaptable implementation of a pooling block for FPGAs.

2 MAPPING CNNs ON FPGAS

A CNN model can be decomposed in a pile of heterogeneous stacked layers. The exploration of CNNs design both in terms of layer description and organization have concluded in several unique models pursuing to increase inference accuracy while running on embedded devices, as those presented in Table 1. To perform an inference, neurons are arranged in 2D maps (named feature maps or *fmaps*) of dimensions $H \times W$ packed in blocks of C channels. These are convolved with a number N of filters of size $C \times K_x \times K_y$ applied with a stride S , with border padding size P . The calculation of an output *fmap* can be expressed as follows:

$$O[n][h][w] = f\left(B[n] + \sum_{c=0}^C \sum_{k_y=0}^{K_y} \sum_{k_x=0}^{K_x} I[c][h+k_y][w+k_x] \times W[n][c][k_y][k_x]\right) \quad (1)$$

Where I : input *fmaps*, W : filters or weights, O : output *fmaps*, and B : bias term. Activation layers are typically included after convolutions to normalize the *fmaps* using functions $f(\cdot)$ like ReLU.

Figure 1: Network-on-Chip mapping and pooling halos

Pooling layers downsample input $fmaps$ dimensions by a factor of S_P by the application of filters on windows of $K_P \times K_P$ size. Convolutional layers dominate model calculations, constituting 90% of them. Because of that, recent investigations about CNNs mapping into RTL focus on their execution optimization exploiting either inter-layer [5] or intra-layer [6] parallelization.

3 HARDWARE IMPLEMENTATION OF THE POOLING BLOCK

The total number of elements in a CNN layer rarely fits entirely on the limited FPGA hardware resources for fully simultaneous computation, this is solved by CNN model partitioning. In order to reduce required buffer allocation, CNN accelerator design benefits of merging both convolution and pooling layers hence avoiding intermediate $fmaps$ storage. The pooling block must be able to dynamically adjust its behavior conditioned by NoC mapping, pooling kernel size K_P , stride S_P , and type T_P . To the best of our knowledge, actual CNNs pooling parameter values are narrowed down to few combinations: $K_P=[2,3]$, $S_P=2$, and $T_P=[MAX, AVG]$.

Subject to accelerator configuration, on-the-fly pooling calculation can demand halo data buffering, this is the case depicted in Fig. 1. Fig. 2 details the RTL model of a pooling block compatible with Table 1 parameters, which can be further simplified in case of reducing K_P range of allowed values from $[2,3] \rightarrow [2]$. Required halo storage blocks depth ($halo_{DEPTH}$) is proportional to inter-layer $\max(N \cdot H)$, in case of *Halo-C*, and $\max(N)$ for *Halo-R*. Conditional to $fmaps$ bit resolution and platform memory block features, requested number of per-halo buffers is:

$$\#buffer = \left\lceil \frac{fmaps_{BITS}}{buffer_{BITS}} \right\rceil \cdot \left\lceil \frac{halo_{DEPTH}}{buffer_{DEPTH}} \right\rceil \cdot halo_P \quad (2)$$

Where $halo_P$ is the number of halo buffers that need to be read in parallel, determined by the NoC mapping strategy.

Table 2: Pooling Block #buffers in Zynq104 ($fmaps$ 16b)

	$\max(N)$	$\max(N \cdot H)$	#BRAM ($halo_P=1$)	#LUTRAM ($halo_P=P_C$)
AlexNet	256	21,792	12	$64 \cdot P_C$
SqueezeNet v1.1	128	14,528	8	$32 \cdot P_C$

Figure 2: Pooling Block, blue-logic unused when $K_P=2$

For those CNN models described in Table 1, in Zynq104 FPGA the amount of data to store determines to better use LUTRAM for *Halo-R* buffers and BRAM in *Halo-C* case, resulting in the hardware utilization shown in Table 2.

4 CONCLUSION

A dynamically configurable pooling hardware module is presented enabling on-the-fly $fmaps$ down-sampling thus reducing data movement which entails a power consumption diminution.

State-of-the-art CNN models are retrained with pooling kernel $K_P=2$, with a positive impact on accuracy (Table 3) leading to further pooling block simplification.

Table 3: CNN models accuracy deviation with $K_P=3 \rightarrow K_P=2$

	New Top-1%	Top-1% Deviation
AlexNet	57.38	+0.28
SqueezeNet v1.1	58.71	+1.51

ACKNOWLEDGMENTS

This work was supported by EU H2020 MSCA through Project ACHIEVE-ITN (Grant No 765866), by the Spanish MINECO and European Region Development Fund (ERDF/FEDER) through Project RTI2018-097088-B-C31 and by the US Office of Naval Research through Grant No. N00014-19-1-2156.

REFERENCES

- [1] A. Krizhevsky et al. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1, NIPS'12*, pages 1097–1105, 2012.
- [2] F. N. Iandola et al. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size. *CoRR*, 2016.
- [3] F. Tu et al. Deep convolutional neural network architecture with reconfigurable computation patterns. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 25(8):2220–2233, Aug 2017.
- [4] Y. Chen et al. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE Journal of Solid-State Circuits*, 52(1):127–138, Jan 2017.
- [5] M. Alwani et al. Fused-layer cnn accelerators. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1–12, Oct 2016.
- [6] Y. Ma et al. Optimizing loop operation and dataflow in fpga acceleration of deep convolutional neural networks. In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, FPGA '17*, pages 45–54, 2017.