Convolutional Neural Networks on embedded automotive platforms: a qualitative comparison

Gianluca Brilli, Paolo Burgio and Marko Bertogna
University of Modena and Reggio Emilia, Modena, Italy
88740@studenti.unimore.it, {paolo.burgio, marko.bertogna}@unimore.it

***Abstract*—In the last decade, the rise of power-efficient, heterogeneous embedded platforms paved the way to the effective adoption of neural networks in several application domains. Especially, many-core accelerators (e.g., GPUs and FPGAs) are used to run Convolutional Neural Networks, e.g., in autonomous vehicles, and industry 4.0. At the same time, advanced research on neural networks is producing interesting results in computer vision applications, and NN packages for computer vision object detection and categorization such as YOLO, GoogleNet and AlexNet reached an unprecedented level of accuracy and performance. With this work, we aim at validating the effectiveness and efficiency of most recent networks on state-of-the-art embedded platforms, with commercial-off-the-shelf System-on-Chips such as the NVIDIA Tegra X2 and Xilinx Ultrascale+. In our vision, this work will support the choice of the most appropriate CNN package and computing system, and at the same time tries to "make some order" in the field.**

***Index Terms*—Automotive systems, Convolutional Neural Networks, GPU,FPGA, YOLO, Tegra, Ultrascale**

I. Introduction

Convolutional neural network (CNN) are today adopted in tens of applications, ranging from industry 4.0, autonomous driving, medical, and more in general wherever computational-intensive tasks (such as computer vision tasks) must run in real-time [1]. In CNNs, thousands of logically independent convolutional kernels execute in parallel, making modern many-core accelerators, such as GP-GPUs and FPGAs, the preferable choice of system engineers to achieve the optimal performance/power tradeoff in their designs. These platforms typically couple a multi-core host, and accelerators that can be either based on i) GP-GPUs [13], [12], ii) on "pure" many-cores [6], or iii) on more flexible, reconfigurable logics [18], where hundreds of custom Processing Units (PUs) are designed using high-level design tools [3]. A typical example of these platforms systems is the NVIDIA Tegra family [13], [12], a GPU-based System-on-Chip (SoC) originally designed for smartphones and tablets, and more recently for autonomous-driving systems [13], [16], delivering a tremendous performance-per-watt. Also Field Programmable Gate Arrays have a great potential for powering up CNNs, not only for the extreme power efficiency at sub-byte operations, but also because they can be "hot" reconfigured on-the-fly within few milliseconds. The Xilinx Ultrascale+ [18] is at the state-of-the-art in the field. In this work, we plan to extensively analyze and validate state-of-the-art CNN packages running on cutting-edge heterogeneous many-core accelerators, in terms both of performance and power consumption. We especially focus on YOLO and its variants [14], [1] as a network for object detection, and on AlexNet [8] for image categorization.

This paper is structured as follows: Section II depicts the two architectures considered in this work, while Section III shows in details properties and structure of the considered neural networks. Section IV is the main contribution, and it shows our analysis, in terms of performance (frames-per-second, FPS), and power consumption. Section V some previous non-exhaustive works that are somehow related to ours, and, finally, Section VI highlights some conclusions and future works.

[1]Here we mean soft real-time, where average performance is still the major requirement, as opposite to hard RT, where worst case performance and timing predictability are.

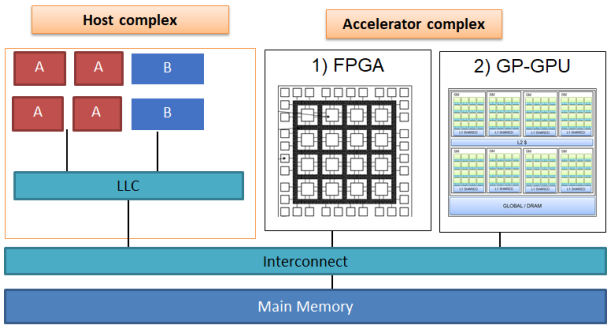

Fig. 1. Host-Accelerator Architecture

II. Automotive Architectures

We target a typical architectural template of modern embedded systems, where a host multi-core is coupled with a power-efficient many-core accelerator (see Figure 1). We target two different architectural "flavors", namely one based on GP-GPUs accelerator, and one based on reconfigurable FPGA logic. The considered SoCs, respectively a NVIDIA Tegra X2 [12], and a Xilinx Ultrascale+ [18], represent the most advanced technologies available on each market.

GP-GPU: the NVIDIA Tegra family The Tegra X2 platform embeds an esa-core host with Big.SUPER configuration and a GPU with 2 NVIDIA Streaming Multiprocessors (SM) of the Pascal family, summing up to 256 CUDA cores. *Big* cores are four ARM A57, while *SUPER* cores are two ARM-based, NVIDIA's proprietary technology named *Denver*. The platform is claimed to achieve 1 TFLOP of computing power, within approximately 20 Watts. We indeed experience these numbers in our experiments.

Reconfigurable logics: the Xilinx Ultrascale+ The second platform we consider is Xilinx Ultrascale+ [18] (XU+), a next-generation reconfigurable heterogeneous platform for embedded systems. Also the XU+ board features a multi-core subsystem, which couples small, power-efficient ARM

A53 cores and a real-time grade core, ARM Cortex R5. The SoC also features programmable logic, and a Mali GPU, which we leave out of the picture, for the moment. The XU+ board has richer I/O connectivity than the TX2; hence, in our experiments, we expect its power consumption to be higher.

III. Target neural networks

Convolutional Neural Networks or *CNNs* are composed by multiple neuron layers, and, at each layer, features (information) are collected from the input images (called *input feature maps – fmaps*). Typical CNNs are composed by tens of layers, summing up to thousands of neurons concurrently executing, hence they take great benefits from acceleration on many-core platforms. The most compute-intensive operation performed by CNN layers is the discrete 2D-convolution, defined as:

$$o^k_{i,j} = \sum_{c=0}^{D_{in}} \sum_{h=0}^{K_H} \sum_{w=0}^{K_W} (w^k_{h,w,c} x_{i+h,j+w,c}) + b_k$$

In today's embedded systems, CNNs are mainly adopted to perform two tasks, that is, i) image classification into one or more categories, which is probably the most well-known problem in computer vision, and ii) object detection inside image/video frames, which is more computationally complex. In our work, we consider representative CNNs from both domains.

Detection: You-Only-Look-Once (YOLO) The networks for objects detection studied in this work are YOLO (You-Only-Look-Once) and some variants. They were proposed by Redmon et al. in [14]. In particular, the YOLO-based models tested are i) the original network, which has 23 convolutional layers, and processes images with resolution 408×408; *Small-YOLO*, a variant with fewer fully connected layers (and smaller memory footprint), and iii) *Tiny-YOLO*, which is composed by only nine convolutional layers, hence has the lowest end-to-end latency, but also the lowest accuracy.

Classification: AlexNet AlexNet was proposed by Krizhevsky et al. [8], and has only 5 convolutional layers. It is a Deep Convolutional Neural Network for image classification, and it won the ILSVRC-2012 competition [15], achieving a winning top-5 error rate of 15.3%, compared to 26.2% by the runner-up.

IV. Experimental Evaluation

Experimental methodology. We are interested in assessing both performance metrics, i.e., processed *frames-per-seconds* and power consumption. In our experiments, we used the same (synthetic) data-set of images for both the types of networks, in case resized to match the size of each CNN fmap. We collect power consumption by connecting an amperometer to the board power supply, and measure the dynamic power consumption as $P_{dyn} = \int_0^\infty i(t) V_{DD} dt$. This formula has been approximated by multiplying the mean current absorbed and the voltage supply, as follows: $P_{board} = V_{DD} \times I_{mean}$. Then, we derived P_{inf} by subtracting the power of the inference phase, and the one consumed when the SoC is idle: $P_{inf} = P_{board} - P_{idle}$. The metric roughly captures the power consumed during network inference, and, in the future, we want to measure the power consumption of the sole SoC using more accurate, industrial (yet expensive) profiling tools, such as Lauterback.

As said, we report the power consumption and throughput, measured in frames-per-seconds, of the targeted networks. For TX2, we show 3d-plots, where GPU frequency is scaled (x-axis) from about 100MHz up to 1.3GHz, and CPU frequency (y-axis) ranges from about 300MHz up to about 2GHz. The metric of interest is shown in the z-axis. Colors of surface charts ease the reading of y-axes values in the 3d plots. TX2 runs an Ubuntu Linux, and we set the highest (real-time) priority for the task that offloads data and computation to the network running on the accelerator, to avoid context switches. For Zynq Ultrascale+, we developed an application that calls the NN hardware accelerator on programmable logic (PL). The operating system is *Petalinux*, a Linux-based system provided by Xilinx.

Classification: AlexNet Figure 2 show the frames-per-second achieved by the AlexNet CNN on Tegra X2. We see how interference done on TX2 is faster than UC+ by a factor of $10\times$. As shown in Figure 2, we astonishingly experience up to 600 FPS for the TX2, while the FPGA-based implementation running on xfDNN engine achieves slightly more than 60 FPS, as reported in Table I. In Figures 3 and 4, and in Table I, it is possible to see a comparison of the power consumption (P_{board} and P_{inf}) during inference on UC+. Looking at Table I, we see a huge performance gap between the same network running on two NN engines (namely, PipeCNN [17] and xfDNN [11]) tested on Zynq board. This is because the PipeCNN OpenCL framework is not optimized nor fully integrated in Xilinx SDSoC development environment. This highlights the importance also of the engine framework, and not only of the network itself, on performance. Power consumption is still comparable.

TABLE I
AlexNet on Zynq Ultrascale+

Framework	PS Freq. [MHz]	PL Freq. [MHz]	T.put [FPS]	P_{board} [Watt]	P_{inf} [Watt]
PipeCNN	1110	100	0,2462	20,2231	0,4603
xfDNN	2400	200	61,554	23,1165	0,5097

Figure 2 shows that in the TX2 platform, performance scales with the CPU clock, but not with the GPU clock. This happens because the application is heavily memory bound, and the data transfers are managed by the CPU, which becomes sensitive to clock scaling. The compute part on the GPU is insensitive to clock frequency for a relatively "simple" problem such as image classification. This does not happen for more complex tasks, such as object detection, where the GPU processes higher workloads. Figure 5, which refers to YOLO, shows how performance scales both with CPU and GPU clock.

Looking at power consumption, we see how it greatly varies for the TX2 and XU+ boards, i.e., respectively 8W and 20W. This is due to the big difference on the I/O hardware modules that we can see on both boards. If we consider the sole SoC, however (P_{inf}), XU+ performs slightly better. As explained, P_{inf} captures the power consumed by the SoC while doing actual computation, and results confirm the highest power efficiency of FPGA technologies when compared to GPUs.

Detection: YOLO We explore three variants of YOLO, namely full *YOLO*, *Small-YOLO* (slightly tinier than YOLO, with smaller memory footprint), and *Tiny-YOLO*, a minimal version of YOLO with only nine convolutional layers. Table II reports the performance on Zynq Ultrascale+. For Tegra X2 implementations, Figures 5, 6 and 7 report results for the "original" YOLO, while Figures 8, 9 and 10 show the ones

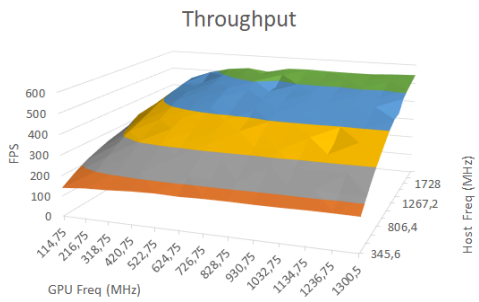

Fig. 2. AlexNet on TX2 (Throughput)

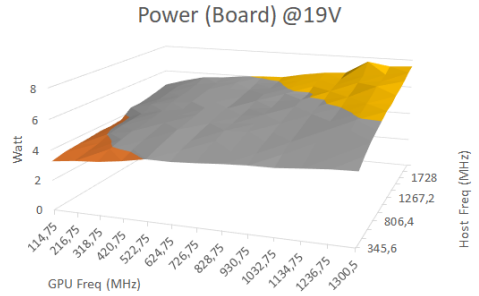

Fig. 3. AlexNet on TX2 (P_{board})

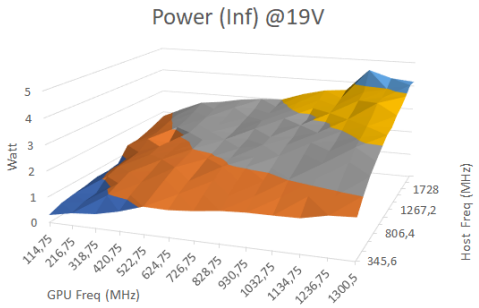

Fig. 4. AlexNet on TX2 (P_{inf})

of Tiny-YOLO . In this case, performance in terms of frames-per-second are similar. We experience 6.6 FPS for YOLO on UC+, and about 8 FPS on TX2. Tiny-YOLO achieves up to 22.6 FPS on UC+, and about 30 FPS on TX2. Power consumptions are almost comparable, but, counter-intuitively, Tiny-YOLO consumes slightly more than its "bigger" variants. It is probably due to some approximation/error in our measurement methodology, and we will investigate this further.

V. Related Works

Nakahara et al. [10] developed an FPGA version of YOLO where inputs and weights are binarized [5], this network is implemented with SDSoC, an high level synthesis tool provided by Xilinx. They also carry an extensive benchmarking of their network running on Zynq Ultrascale+ and Tegra X2, achieving 40 FPS at 4.5 Watts for the UC+. Thile the TX2 implementation only reaches 2 FPS at 7 Watt. In both cases, while power consumption is acceptable, performance can certainly be improved.

Work in [14] shows an highly optimized FPGA implementation of YOLOv2, but their benchmarking does not take into account other network typologies and other more efficient GP-GPU implementations of YOLOv2.

Xilinx engineers proposed a reduced/quantized version of

TABLE II
YOLO on Zynq Ultrascale+

Network	PS Freq. [MHz]	PL Freq. [MHz]	T.put [FPS]	P_{board} [Watt]	P_{inf} [Watt]
YOLO	2400	200	6,6728	23,7292	0,6959
Small-YOLO	2400	200	7,9311	23,7137	0,6804
Tiny-YOLO	2400	200	22,6807	24,1798	1,1465

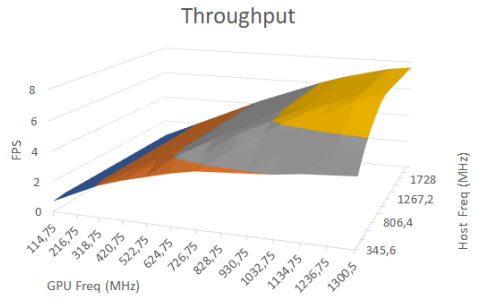

Fig. 5. YOLO on TX2 (Throughput)

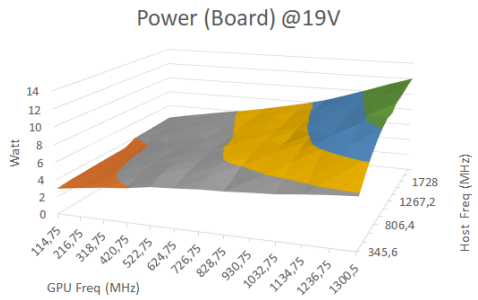

Fig. 6. YOLO on TX2 (P_{board})

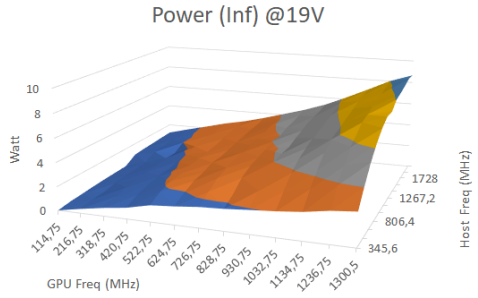

Fig. 7. YOLO on TX2 (P_{inf})

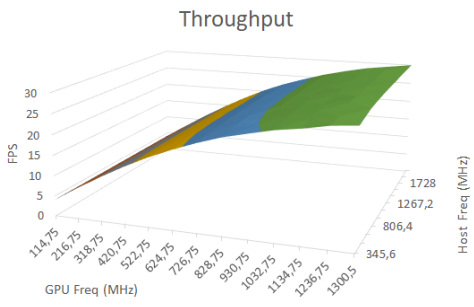

Fig. 8. Tiny-YOLO on TX2 (Throughput)

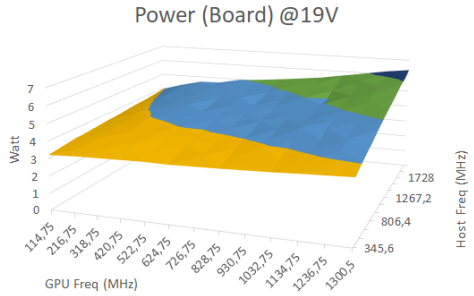

Fig. 9. Tiny-YOLO on TX2 (P_{board})

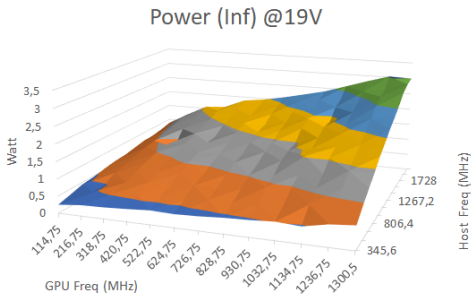

Fig. 10. Tiny-YOLO on TX2 (P_{inf})

Tiny-YOLO called Tincy-YOLO [1], which operates at 16 FPS, consuming 6 Watt for the FPGA accelerator. In their work, they compare the accuracy of Tincy-YOLO with respect to other versions of YOLO, but they don't compare against GP-GPUs implementation. Our work completes their analysis.

The group of Don Wang [17] developed an FPGA framework for image classification and a comparison of the CNN models AlexNet and VGG-16 [7] on both Altera and Xilinx FPGAs. In this case, the shortest classification time is achieved by Altera DE5-net and is 23 FPS for AlexNet and 1.4 FPS for VGG-16, with a power consumption of 27.3 Watt and 29.8 Watt respectively.

VI. Conclusions

We evaluated the power consumption and performance of image classification and objects detection Convolutional Neural Networks, on representative embedded platforms based on GP-GPU and FPGA accelerators. We experience comparable performance for the object detection networks, due to their complexity, while for classification network, GP-GPUs still outperform (non-optimized) FPGAs by at least one order of magnitude. On the other hand, FPGAs are better from the power efficiency viewpoint.

It must be pointed out that, porting complex software like CNNs on programmable logics is cumbersome, mainly because High Level Synthesis still cannot be as efficient as traditional design, and the open-source drivers and frameworks are not yet optimized. Also, measuring power consumption of the SoC only without using expensive industrial tools is cumbersome, and we will study new and more accurate methodologies to do it.

Results in Table I show that also the network engine can play a significant role. In the future, we will test also other frameworks such as Caffeinated [2] or NEURaghe [9]. We are interested in exploring other network models with binarization, such as [5], [10] and [4]. We will also put in the picture other representative platforms, such as Kalray MPPA many-core [6], and application-specific circuits for NN.

This work is supported by the HERCULES project, funded by European Union's Horizon 2020 research and innovation program under grant agreements No. 688860.

References

[1] M. Blott (Xilinx, Inc.). Tincy YOLO: a real-time, low-latency, low-power object detection system running on a Zynq UltraScale+ MPSoC, 2017.
[2] R. DiCecco, G. Lacey, J. Vasiljevic, P. Chow, G. Taylor, and S. Areibi. Caffeinated fpgas: Fpga framework for convolutional neural networks. In *2016 International Conference on Field-Programmable Technology (FPT)*, pages 265–268, Dec 2016.
[3] T. Feist. Vivado Design Suite, 2012.
[4] H. N. Haruyoshi Yonekawa. GUINNESS: A GUI based binarized Neural NEtwork SyntheSizer toward an FPGA, 2017.
[5] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4107–4115. Curran Associates, Inc., 2016.
[6] Kalray Corporation. Many-core Kalray MPPA, 2012.
[7] A. Z. Karen Simonyan. Very deep convolutional networks for large-scale image recognition. 2015.
[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton. 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, USA, 2012. Curran Associates Inc.
[9] P. Meloni, A. Capotondi, G. Deriu, M. Brian, F. Conti, D. Rossi, L. Raffo, and L. Benini. Neuraghe: Exploiting CPU-FPGA synergies for efficient and flexible CNN inference acceleration on zynq socs. *CoRR*, abs/1712.00994, 2017.
[10] H. Nakahara, H. Yonekawa, T. Fujii, and S. Sato. A lightweight yolov2: A binarized cnn with a parallel support vector regression for an fpga. In *Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '18, pages 31–40, New York, NY, USA, 2018. ACM.
[11] N. Ni and V. Kathail (Xilinx, Inc.). Caffe to Zynq: State-of-the-Art Machine Learning Inference Performance in Less Than 5 Watts, 2017.
[12] NVIDIA. The Tegra X1 Platform, 2015.
[13] NVIDIA. NVIDIA DRIVE PX: scalable AI Supercomputer For Autonomous Driving, 2017.
[14] J. Redmon and A. Farhadi. YOLO9000: better, faster, stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 6517–6525, 2017.
[15] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Large scale visual recognition challenge 2012. http://image-net.org/challenges/LSVRC/2012/index, 2012.
[16] Shri Sundaram. Building autonomous vehicles using Drive PX 2, 2017.
[17] D. Wang, K. Xu, and D. Jiang. Pipecnn: An opencl-based open-source fpga accelerator for convolution neural networks. In *2017 International Conference on Field Programmable Technology (ICFPT)*, pages 279–282, Dec 2017.
[18] Xilinx, Inc., . The Xlinx Ultrascale Architecture.