Deep Learning in Reconfigurable Hardware: A Survey

Mauricio A Dias, Daniel A P Ferreira
Intelligent Systems and Autonomous Vehicles Group (FHO-SIVA)
Engineering Department
Herminio Ometto Foundation (FHO),Araras, Brazil
{macdias,danielferreira}@fho.edu.br

***Abstract*—Deep Learning has been applied successfully to solve complex problems that involves analysis of large data sets and these good results can be directly related to the size and complexity of the networks and training algorithms. However, these structures are considerably resource-consuming and demand extra effort to be used on embedded systems. Researchers have chosen different alternatives to execute deep learning algorithms as servers, clusters, GPUs and FPGAs. Specific hardware structures have been designed to solve these problems in different platforms such as FPGAs and even ASICs. Although there are surveys about this subject, they do not present a clear criteria to real implementations, detailed explanation of design techniques and applied metrics. Presenting a different approach, this work analyzes reconfigurable hardware based structures designed to optimize deep learning algorithms. This work includes results of hardware accelerators implemented and tested using commercially available development boards avoiding simulation-based results.Results of the proposed analysis showed that a lot of effort is directed to this subject but presented results are far from what is expected for the performance of a hardware structure for deep learning.**

***Index Terms*—Deep Learning, Reconfigurable Computing, FPGA, GPU**

I. Introduction

Deep learning has been growing in several applications in recent years such as speech recognition, image classification, handwriting transcription among others. These were possible due to advances in hardware, datasets and algorithms, even though the mathematical background from deep learning was known before the years 2000. Deep learning derives from artificial neural networks and it is considered a subclass of the machine learning field [1]. Multi Layer Perceptron (MLP) networks were able to achieve good results and deal with the size of datasets until big data rise in both industrial and academic applications. One of the solutions to deal with big data are deep learning structures based on convolutional neural networks (CNN or convnets). Deep learning emerged as artificial neural networks architecture became larger and increased its processing capability, specially in the object recognition challenge ImageNet among others [3]. These large structures achieved interesting results but brought the need for more processing power.

There are several deep learning algorithms in the literature such as CNNs and deep belief networks (DBN). CNN, frequently used in image processing and classification, is a deep learning architecture with a convolutional layer responsible to convolve the input data with a filter thus resulting in a feature map as output. Subsequently, a pooling layer reduces the dimension of the image by some criteria such as the maximum value (known as maxpooling) or averaging nearby pixels. After the pooling layer, the data is now flattened and passes throw a fully connected layer, similar to those founded in MLP, to be processed and to return a output value [1], [3].

Among the possible options for implementing hardware structures for deep learning, this work focus in Field Programmable Gate Arrays (FPGAs). Results of deep learning hardware accelerators designed using reconfigurable computing show that FPGAs are a promising platform. The main contribution of this work is to consider previous works that present hardware implementation results and not only simulation results, showing the real state-of-the-art in this field. Simulation results can be considered important as ongoing work, but cannot be considered as real advance for the field since they cannot be used on real systems.

II. FPGA-based Hardware for Deep Learning

Considering that deep learning networks have a high number of connections and neurons, they can be considered both CPU and memory bound applications. FPGAs are interesting choices to implement hardware accelerators for deep learning networks due to their flexibility and low power consumption.

Surveys are important for researchers that need to understand how research evolved in determinate area until it's state-of-the-art. Considering hardware accelerators for deep learning designed for FPGAs, there are surveys available which analyses a huge number of research works with respect to different aspects of design, implementation techniques and their results [6]–[9]. A problem that can be found in most of them is the lack of commitment to ascertain whether the work actually implements what it proposes and whether the results are valid and reproducible. Most of analyzed research works fail to present the complete information about the results of their proposed FPGA-based solution [45]. Research works that present hardware solutions for neural network models used in deep learning were also included. The decision of analyzing only functional hardware designs also excludes works that present modelling and proofs about proposed techniques without testing them using networks implemented in FPGAs [46].

978-1-5386-5555-9/19/$31.00 ©2019 IEEE
DOI 10.1109/IPDPSW.2019.00026

Studies about how to design hardware structures for processing units, activation functions or any other specific feature of neural network models were only considered if tested with other parts of the network.

Research works that choose Xilinx FPGAs presented results using:

- Virtex®-7 FPGAs that have more than $480k$ logic cells, $2.8k$ DSP Slices, 700 IO pins and on-chip memory of $37,000$ Kb;
- Zynq-7000 System on Chip that has a dual-core ARM cortex A9 processor hard connected to a Artix-7 FPGA with $215k$ logic cells, low power consumption that is the chip of ZedBoard Kit;
- Kintex®UltraScale FPGAs that are design using $20nm$ technology up to $1,000K$ logic cells, $4,100$ DSP slices, 56.9 Mb of Block Ram and 676 IO pins;
- Spartan-6 FPGAs that are up to $147K$ logic cells, $4,800$ Kb of memory, 180 DSP slices and 540 IO pins are still used and also.

Similarly, research works that choose Intel FPGA presented results using:

- Stratix®V FPGAs built with 28nm technology up to $622K$ logic elements, 50 Mbits M20K memory blocks, variable precision DSP blocks;
- Stratix®10 FPGAs up to 5.5 million logic elements, 229 Mbits M20K memory blocks, variable precision DSP blocks;
- Arria 10®FPGAs up to $480K$ logic elements, 28 Mbits M20K memory blocks, variable precision DSP blocks;
- Stratix®III devices up to $338K$ logic elements, 20 Mbits of enhanced TriMatrix memory and High-speed DSP blocks that provide dedicated implementation of $9x9$, $12x12$, $18x18$, and $36x36$ multipliers.

The best metric to evaluate hardware design depends on the platform and design decisions. Thus researchers should choose wisely which metrics are more suitable for each hardware. In addition, at least one metric should become universal, such as Giga Operations per Second (GOPS), a metric widely used by deep learning researchers. However, when the GOPS metric shows a result not as good as expected other metrics come up to state that certain design is better than other. Metrics from analyzed works are described below:

- *GOPS* - Giga Operations Per Second, the number of MAC operations realized by the hardware accelerator in one second. [7] [10] [12] [14] [20] [23] [24] [29] [33] [35] [36] [37] [39] [42] [43].
- *GOPS/W* - GOPS per Watt is an energy efficiency metric used to benefit hardware with low energy consumption [14] [17] [18] [20] [27] [29] [32] [36] [37].
- *GOPS/Slice* - GOPS per Slice represents the relation between the number of operations and area consumption, used when the final area of the proposed hardware achieved interesting results [10] [12] [39] [42].
- *GOPS/DSP* - GOPS per used Digital Signal Processors. Simmilar to GOPS/Slice metric that can be used when DSP consumption is available to be compared [29] [36] [39].
- *Speedup* - this metric is widely used in different situations and is the standard metric to justify the design of hardware structures for deep learning. Usually hardware designs achieve speedup of more than $4x$ when compared to CPU implementations [5] [13] [15] [25] [27] [28] [35].
- *Accuracy* - this metric is used when numeric representation precision is compromised in order to achieve better execution time [7] [11] [18] [22] [23] [25] [28] [30] [32] [34].
- *Energy consumption (Watts)* - this is a key feature of hardware accelerators used to justify their use in embedded applications [7] [13] [15] [16] [29] [40] [41].
- *Execution time (seconds)* - this metric is used when proposed hardware structures can process a high number of frames per second, also called *performance*. This is a poor metric that should not be used to compare hardware structures due to its subjectivity [13] [19] [22] [23] [26] [31] [33] [34] [38] [40] [43].
- *GFLOPS* - Giga Floating-Point Operations Per Second measures hardware performance of floating-point based hardware accelerators. Since most of hardware implementations used fixed-point architectures, this metric should not be used [12] [41].
- *Resources* - FPGAs are resource-limited hardware. Hardware designs that are able to achieve acceptable results without consuming a huge amount of hardware resources are preferable. This metric is also called *Area* [11] [15] [17] [21] [23] [30] [38] [41] [43].

Specific metrics can be created to allow better design decisions. An interesting new metric was proposed by Shah [37] called *Nop*, which is calculated as the division between the number of used MACs and the number of pixels in an input image's row. The result of this metric is used to configure the hardware design during execution time.

FPGAs are devices with specific features so the hardware accelerator can be designed in a way that these features can be well explored. In [10] the number representation system was designed to use a small amount of MACs so the FPGA internal DSPs can be used for other functions. On-chip memory in FPGAs are too small to store an entire deep learning structure but if hardware tasks are planned to optimize memory usage and the number of MACs used is reduced, the final design can achieve better resource consumption as presented in [35]. Hardware/Software co-design is a well known solution for complex systems and good results using this technique were achieved in [11] [12] [21] [27] [31] [40], when processors were used to execute part of the algorithms or only to manage memory operations.

As CNNs are implemented with a high number of chained for-loops, the most used optimization is the loop unrolling. When its possible to execute more than one iteration of the loop concurrently, loop unrolling is able to achieve high speedups. Results of specific hardware designed for unrolling convolutional loops are presented in [12] [23] [24] [31]. Also these loops can be implemented without optimizations and still achieve good results [13] [26].

FPGA-based hardware designs usually allow a change on numeric representation without compromising the design accuracy. Fixed-point representations were well explored in [5] [18] [21] [32] [36] [37] [42] [43]. However, they do not check how and if the precision was compromised is provided and it is unlikely that the structure achieved almost the same results using 32-bit floating point and 8-bit fixed point representations as reported in some of these works. Changing the numeric representation is highly recommended to save memory and execution time but there is a trade-off with accuracy that should be considered.

OpenCL can be used to generate hardware and allow RTL, VHDL and Verilog modules to be used. However, since it was originally designed to program CPUs, it will produce less optimized FPGA-based designs. Results are usually compared to GPU implementations and sometimes the FPGA design is unable to achieve good results as discussed in [25] [28] [41].

Instruction-based co-processors can be used as hardware accelerators, allowing better throughput and execution time. It is possible to change the co-processor structure based on CNN parameters before generating the hardware so the instruction set can be reduced and memory use can also be optimized [17]. Also the numeric representation of the co-processor can also be set to specific cases where accuracy is not a problem [32]. One of the most interesting co-processor design is presented in [37] where the instructions represent the co-processor settings and can be modified during execution to change the network's behaviour.

Layer multiplexing [22] [14] is an interesting strategy to design hardware accelerators for neural networks. This approach can lead to a complex control unit and some time consuming operations. Another option is to pipeline matrix operations [15] [33] [36] in order to save execution time and decrease the number of memory accesses because the calculated values are used in the next stage of the pipeline while in the pipeline registers.

Presented strategies does not consider changing the CNN implementation to achieve better results with hardware acceleration. Nonetheless, there are some changes that can be applied to the original CNN execution flow in order to find a more hardware-friendly algorithm. The most common approach is pruning, which cuts connections between layers and neurons to decrease the number of stored weighs [14] [34]. A better exploration of the pruning search space can be performed by artificial intelligence algorithms as proposed in [30] where NSGAII, an Estimation Distribution Algorithm (EDA), was used to find the best network to be implemented in hardware.

Interactive Stencil Loops (ISLs) are loops whose data dependencies span across multiple iterations. An interesting way to explore ISL optimizations is considering that a certain value of an element k in a certain iteration $n+1$ depends on a small number of elements in k neighbourhood at iteration n. This fact was explored in [19] to create a hardware structure in which calculated values are stored and controlled in a way that a step doesn't need to end before the next one starts. Residual learning is a technique that allows a better approximation of a complex function $H(x)$ using a residual function $F(x)$ that can be represented as $F(x) = H(x) + x$. In CNNs this idea is represented by adding bypass connections to the network in order to improve the network results. In [23] a residual network is implemented in hardware using also loop unrolling optimizations resulting in an interesting alternative to regular CNN.

Matrix multiplications are the core of CNNs. Alternatives to improve this operation can result in good solutions for hardware implementation. Coppersmith and Winogard's matrix multiplication algorithm was used in [29] to improve the way convolution is executed achieving good results. However, this algorithm is suitable for very large matrices and is not indicated for all deep learning networks. Another method to optimize the number of multiplications is to use depth-wise convolution [36] where the 2D kernel ($3x3$ for example) is broken into two new 1D kernels ($3x1$ and $1x3$). This operation reduces the number of multiplications by more than 70%. A kernel used for convolution can be represented by linear superposition of a binary mutually orthogonal basis vectors set [39]. Using this relation the operations are faster and considerably less complex than the convolution of the original filter. Results showed that the precision loss is not a problem so this technique can be used for deep learning hardware implementations without restrictions.

III. Conclusion

Hardware designs for deep learning are interesting alternatives due to the high execution time of deep learning algorithm in general purpose computers. In this work a survey of how hardware accelerator are being designed and implemented considering reconfigurable computing was presented. Presented metrics were all used to evaluate hardware structures implemented and testes in FPGAs but there is no consensus on which one is the best metric. Analyzed works suggested a tendency to consider GOPS and GOPS/W good metrics. Among different hardware techniques used to implement deep learning algorithms in hardware there is also not a pattern. Different structures for different optimization goals were proposed achieving good results. From presented analysis is possible to say that there is a tendency to build hardware generation frameworks that are able to change the final hardware according to the application requirements. The main question is when these frameworks will became available to be used and where researchers can find them to use in their works.

References

[1] Wason, R. "Deep learning: Evolution and expansion." Cognitive Systems Research, v. 52, p. 701708, 2018.

[2] Haykin, S. Neural networks and learning machines. New York: Prentice Hall, 2009.

[3] Moons, B. Bankman, D. and Verhelst, M. Embedded Deep Learning. Springer, 2019.

[4] Goodfellow, I. Bengio, Y. and Courville, A. Deep learning. [S.l: s.n.], 2017.

[5] I.Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, Y. Bengio, ”Binarized Neural Networks”, Advances in Neural Information Processing Systems, n. 29, pp. 4107–4115, 2016.

[6] Yiran Chen, Hai (Helen) Li, Chunpeng Wu, Chang Song, Sicheng Li, Chuhan Min, Hsin-Pai Cheng, Wei Wen, Xiaoxiao Liu, ”Neuromorphic computing’s yesterday, today, and tomorrow an evolutional view”, Integration, vol. 61, pp. 49-61, 2018.

[7] T. Wang, C. Wang, X. Zhou, H. Chen, ”A Survey of FPGA Based Deep Learning Accelerators: Challenges and Opportunities”, arXiv : 1901.04988v1, 2018.

[8] A. Shawahna, S. M. Sait, A. El-Maleh, ”FPGA-Based Accelerators of Deep Learning Networks for Learning and Classification: A Review,” in IEEE Access, vol. 7, pp. 7823-7859, 2019.

[9] A.G. Blaiech, K. B. Khalifa, C. A. V. Sakuyama, M .A.C. Fernandes, M. H. Bedoui, ”Taxonomy of FPGA-Based Topologies for Future Deep Learning Architectures”, J. of Systems Architecture, 2019.

[10] H. Nakahara, T. Sasao, ”A deep convolutional neural network based on nested residue number system,” 2015 25th Int. Conference on Field Programmable Logic and Applications (FPL), London, 2015, pp. 1-6.

[11] L. B. Saldanha, C. Bobda, ”An embedded system for handwritten digit recognition”, J. of Systems Architecture, vol. 61, no. 10, 2015, pp. 693-699.

[12] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, J. Cong, ”Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural Networks”, ACM/SIGDA Int. Symposium on Field-Programmable Gate Arrays, ACM, New York, NY, USA, pp. 161-170, 2015.

[13] A. Alhamali et al., ”FPGA-Accelerated Hadoop Cluster for Deep Learning Computations”, IEEE Int. Conference on Data Mining Workshop (ICDMW), Atlantic City, NJ, 2015, pp. 565-574.

[14] T. Posewsky, D. Ziener, ”Throughput optimizations for FPGA-based deep neural network inference”, Microprocessors and Microsystems, vol. 60, 2018, pp. 151-161.

[15] C. Wang, L. Gong, Q. Yu, X. Li, Y. Xie, X. Zhou, ”DLAU: A Scalable Deep Learning Accelerator Unit on FPGA,” in IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol. 36, no. 3, pp. 513-517, March 2017.

[16] G. Hegde, Siddhartha, N. Ramasamy, V. Buddha, N. Kapre, ”Evaluating Embedded FPGA Accelerators for Deep Learning Applications,” 2016 IEEE 24th Int. Symposium on Field-Programmable Custom Computing Machines (FCCM), Washington, DC, 2016, pp. 25-25.

[17] H. Sharma et al., ”From high-level deep neural models to FPGAs,” 2016 49th Annual IEEE/ACM Int. Symposium on Microarchitecture (MICRO), Taipei, 2016, pp. 1-12.

[18] Y. Wang et al., ”Low power Convolutional Neural Networks on a chip,” 2016 IEEE Int. Symposium on Circuits and Systems (ISCAS), Montreal, QC, 2016, pp. 129-132.

[19] M. Bacis, G. Natale, E. Del Sozzo, M. D. Santambrogio, ”A pipelined and scalable dataflow implementation of convolutional neural networks on FPGA,” 2017 IEEE Int. Parallel and Distributed Processing Symposium Workshops (IPDPSW), Lake Buena Vista, FL, 2017, pp. 90-97.

[20] E. Nurvitadhi et Al, ”Can FPGAs Beat GPUs in Accelerating Next-Generation Deep Neural Networks?”, Proceedings of the 2017 ACM SIGDA Int. Symposium on Field Programmable Gate Arrays (FPGA). ACM, New York, NY, USA, 5-14, 2017.

[21] X. Zhou, S. Li, F. Tang, S. Hu, Z. Lin, L. Zhang, ”DANoC: An Efficient Algorithm and Hardware Codesign of Deep Neural Networks on Chip,” in IEEE Transactions on Neural Networks and Learning Systems, vol. 29, no. 7, pp. 3176-3187, July 2018.

[22] T. V. Huynh, ”Deep neural network accelerator based on FPGA,” 2017 4th NAFOSTED Conf. on Information and Computer Science, Hanoi, 2017, pp. 254-257.

[23] Y. Ma, M. Kim, Y. Cao, S. Vrudhula, J. Seo, ”End-to-end scalable FPGA accelerator for deep residual networks,” 2017 IEEE Int. Symposium on Circuits and Systems (ISCAS), Baltimore, MD, 2017, pp. 1-4.

[24] Y. Ma, Y. Cao, S. Vrudhula, J. Seo, ”Optimizing Loop Operation and Dataflow in FPGA Acceleration of Deep Convolutional Neural Networks”, Proceedings of the 2017 ACM/SIGDA Int. Symp. on Field-Programmable Gate Arrays (FPGA). ACM, New York, NY, USA, 45-54, 2017.

[25] P. R. Gankidi, J. Thangavelautham, ”FPGA architecture for deep learning and its application to planetary robotics,” 2017 IEEE Aerospace Conference, Big Sky, MT, 2017, pp. 1-9.

[26] M. Vstias, R. P. Duarte, J. T. de Sousa, H. Neto, ”Parallel dot-products for deep learning on FPGA,” 2017 27th Int. Conf. on Field Programmable Logic and Applications (FPL), Ghent, 2017, pp. 1-4.

[27] K. Guo, S. Han, S. Yao, Y. Wang, Y. Xie, H. Yang, ”Software-Hardware Codesign for Efficient Neural Network Acceleration,” in IEEE Micro, vol. 37, no. 2, pp. 18-25, Mar.-Apr. 2017.

[28] X. Liu, H. A. Ounifi, A. Gherbi, Y. Lemieux, W. Li, ”A Hybrid GPU-FPGA-based Computing Platform for Machine Learning”, Procedia Computer Science, vol. 141, 2018, pp. 104-111.

[29] Wu, Di. A Novel Low-Communication Energy-Efficient Reconfigurable CNN Acceleration Architecture., 2018.

[30] M. Loni, M. Daneshtalab, M. Sjdin, ”ADONN: Adaptive Design of Optimized Deep Neural Networks for Embedded Systems,” 2018 21st Euromicro Conference on Digital System Design (DSD), Prague, 2018, pp. 397-404.

[31] Y. Ma, N. Suda, Y. Cao, S. Vrudhula, J. Seo, ”ALAMO: FPGA acceleration of deep learning algorithms with a modularized RTL compiler”, Integration, vol. 62, 2018, pp. 14-23.

[32] K. Guo et al., ”Angel-Eye: A Complete Design Flow for Mapping CNN Onto Embedded FPGA,” in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 37, no. 1, pp. 35-47, Jan. 2018.

[33] J. Xu, Z. Liu, J. Jiang, Y. Dou, S. Li, ”CaFPGA: An automatic generation model for CNN accelerator”, Microprocessors and Microsystems, vol. 60,2018, pp 196-206.

[34] J. Faraone, G. Gambardella, N. Fraser, M. Blott, P. Leong, D. Boland, ”Customizing Low-Precision Deep Neural Networks for FPGAs,” 2018 28th International Conference on Field Programmable Logic and Applications (FPL), Dublin, 2018, pp. 97-973.

[35] S. Liang, S. Yin, L. Liu, W. Luk, S. Wei, ”FP-BNN: Binarized neural network on FPGA”, Neurocomputing, vol. 275, 2018, pp 1072-1086.

[36] W. Ding, Z. Huang, Z. Huang, L. Tian, H. Wang, S. Feng,”Designing efficient accelerator of depthwise separable convolutional neural network on FPGA”, Journal of Systems Architecture, 2018.

[37] N. Shah, P. Chaudhari and K. Varghese, ”Runtime Programmable and Memory Bandwidth Optimized FPGA-Based Coprocessor for Deep Convolutional Neural Network,” in IEEE Transactions on Neural Networks and Learning Systems, vol. 29, no. 12, pp. 5922-5934, Dec. 2018.

[38] A. Boutros, S. Yazdanshenas, V. Betz, ”You Cannot Improve What You Do not Measure: FPGA vs. ASIC Efficiency Gaps for Convolutional Neural Network Inference”. ACM Trans. Reconfigurable Technol. Syst., vol. 11, n. 3, 2018.

[39] A. Zhakatayev, J. Lee, ”Efficient FPGA implementation of local binary convolutional neural network”. Proceedings of the 24th Asia and South Pacific Design Automation Conference (ASPDAC ’19). ACM, New York, NY, USA, pp 699-704, 2019.

[40] L. Li, C. Sau, T. Fanni, J. Li, T. Viitanen, F. Christophe, F. Palumbo, L. Raffo, H. Huttunen, J. Takala, S. S. Bhattacharyya, ”An integrated hardware/software design methodology for signal processing systems”, Journal of Systems Architecture, vol. 93, 2019, pp. 1-19.

[41] G. J. Lacey, ”Deep Learning on FPGAs”, Master Thesis, School of Engineering, University of Guelph, Canada, Aug. 2016, 94p.

[42] J. Wang, J. Lin and Z. Wang, ”Efficient Hardware Architectures for Deep Convolutional Neural Network,” in IEEE Trans. on Circuits and Systems I: Regular Papers, vol. 65, no. 6, pp. 1941-1953, June 2018.

[43] J. Wang, Q. Lou, X. Zhang, C. Zhu, Y. Lin, D. Chen, ”Design Flow of Accelerating Hybrid Extremely Low Bit-width Neural Network in Embedded FPGA”. arXiv preprint arXiv : 1808.04311, 2018.

[44] A. HajiRassouliha, A. J. Taberner, M. P. Nash, P. M.F. Nielsen, ”Suitability of recent hardware accelerators (DSPs, FPGAs, and GPUs) for computer vision and image processing algorithms”, Signal Processing: Image Communication, vol. 68, 2018, pp. 101-119.

[45] Q. Yu, C. Wang, X. Ma, X. Li, X. Zhou, ”A Deep Learning Prediction Process Accelerator Based FPGA,” 2015 15th IEEE/ACM Int. Symposium on Cluster, Cloud and Grid Computing, Shenzhen, 2015, pp. 1159-1162.

[46] C. Tsai, Y. Chih, W. H. Wong, C. Lee, ”A Hardware-Efficient Sigmoid Function With Adjustable Precision for a Neural Network System,” in IEEE Trans. on Circuits and Systems II: Express Briefs, vol. 62, no. 11, pp. 1073-1077, Nov. 2015.