# Hardware/Software Co-design for Convolutional Neural Networks Acceleration: A Survey and Open Issues

Cuong Pham-Quoc[1,2(✉)], Xuan-Quang Nguyen[1,2], and Tran Ngoc Thinh[1,2]

[1] Ho Chi Minh City University of Technology (HCMUT), Ho Chi Minh City, Vietnam
{cuongpham,nxquang,tnthinh}@hcmut.edu.vn
[2] Vietnam National University - Ho Chi Minh City (VNU-HCM), Ho Chi Minh City, Vietnam

**Abstract.** In this paper, we survey hardware/software co-design approaches in the literature to accelerate Convolutional neural networks, one of the two successful forms of Deep Neural Networks. We classify these approaches according to target platforms used to accelerate CNNs, including FPGA-based and ASIC-based. Due to the flexibility of FPGAs, we mainly focus on FPGA-based designs. These designs are categorized into sub-classes according to optimization techniques used. We then analyze in detail to compare FPGA-based hardware accelerator systems for CNNs regarding working frequency and performance efficiency. Through the survey, we also identify open issues for this research topic. These challenges can be research directions for optimizing performance, accuracy, and energy consumption of hardware accelerator systems for CNN using the hardware/software co-design approach.

**Keywords:** Hardware/software co-design · Convolutional neural networks · FPGA · ASIC

## 1 Introduction

Convolutional Neural Networks (CNNs), one of the two most successful forms of Deep Neural Networks (DNNs) along with Recurrent Neural Networks [14] (RNNs), are becoming a dominant approach in machine learning for different applications such as image classification, voice recognition, or natural languages processing. In recent years, many researchers focus on improving the accuracy of DNN models that make these models require high computing power, consume much energy, and use a large amount of memory [15]. For example, the VGG19 CNN model image classification with $224 \times 224$ input pixels may take up to


P. Cong Vinh and A. Rakib (Eds.): ICCASA 2021, LNICST 409, pp. 164–178, 2021.
https://doi.org/10.1007/978-3-030-93179-7_13

39B+ floating-point operations (FLOPs) and need more than 500 MB of memory storage parameters [35]. Therefore, executing the inference phase of CNN models in suitable platforms is non-trivial work. With the current technology, traditional CPUs can offer up to ∼100GFLOPs per second while using more than 1 J (J) of energy per 1GOP. Hence, traditional CPUs cannot meet the requirements of high-performance and low power for CNNs-based applications.

Meanwhile, to continue improving the performance of computing systems, including embedded and edge computing, at the end of Moore's Law and Dennard scaling [42], system architects develop a new paradigm that concurrently optimizes both software and hardware for application domains. The most promising approach in this paradigm is to design specialized high-performance and energy-efficient hardware-based computing cores to perform a hefty workload. Two well-known instances of this approach are general-purpose graphic processor units (GPGPUs) and hardware accelerator systems. Although GPGPUs offer tremendous potential for the training and inference phases of DNNs [36], they suffer from energy inefficiency. Therefore, both academia and industry have already applied fully specialized hardware accelerators for DNNs such as neural processing units (NPUs), in which the Tensor processing unit is an excellent example [6]. It is also crucial to develop software support when developing specialized hardware accelerators because systems usually cannot fully utilize them without appropriate software support. These software supports include operating-system-level supports such as direct memory access supports and compiler-level supports such as forming new instructions in ISA (instruction set architecture). Therefore, to accelerate DNNs, we need a hardware/software co-design approach to develop both hardware and software for better optimization.

This paper surveys the cutting-edge systems designed and implemented with the hardware-software co-design approach for accelerating CNNs-based applications. We categorize these systems according to the hardware technologies, including Field Programmable Gate Arrays (FPGAs) and Application Specific Integrated Circuits (ASICs). Although in the literature, there exist some related surveys recently [10,23,43], they mainly focus on analyzing techniques used to optimized hardware accelerator cores such as pruning, quantization, or data reuse. In contrast, we focus on the co-design approach to eventually resulting in system performance and energy consumption. We also identify open issues for accelerations of CNNs with the hardware/software co-design approach so that researchers can consider them as future work.

The rest of the paper is organized as follows. Section 2 gives an overview of the CNN architecture and the background of the hardware/software co-design approach. A survey of CNN accelerations with HW/SW co-design is presented and analyzed carefully in Sect. 3. Based on the study, we deliver open issues for future work of the research topic in Sect. 4. Finally, we conclude our paper in Sect. 5.

## 2 Preliminary

This section presents an overview of the CNN architecture upon which many studies in the literature try to accelerate with the hardware/software (HW/SW) co-design approach. To clarify the co-design approach, we also introduce the HW/SW co-design approach background in this section.

### 2.1 Convolutional Neural Networks

Figure 1 illustrates a traditional simple CNN. Typically, a CNN consists of Convolution layers and Fully connected layers. Convolution layers usually calculate two main operators, including 2D convolutions (2D Conv) and subsampling ($X-$pooling). The input, e.g., a $32 \times 32$ pixels color image with three red, green, and blue channels, is convoluted with convolutions kernels, e.g., $5 \times 5$ to create $C1$ feature maps, e.g., 9 $28 \times 28$ feature maps. Consequently, these feature maps are sampled with different pooling functions (max-pooling or average-pooling) to reduce the redundancy of feature maps. For example, a $2 \times 2$ max-pooling function in Fig. 1 reduces 9 $28 \times 28$ feature maps to 9 $14 \times 14$ feature maps. The two operations can be repeated several times depending on application domains and experts. Outputs of the Convolution layers (2D feature maps) are converted to a 1D vector. Each vector element is connected to a neuron of a traditional neural network in the Fully connected layers for classification.

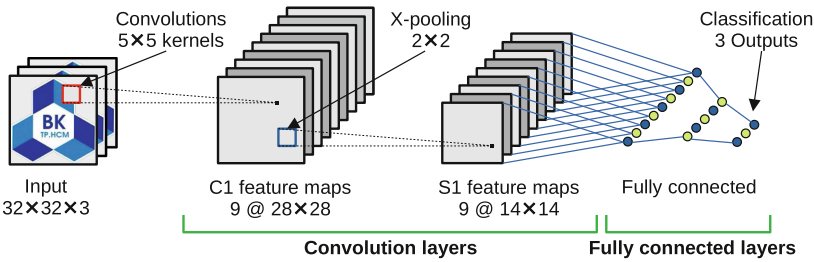



**Fig. 1.** An overview of convolution neural network

In CNN models, the Convolution layers are the most critical operations and contribute more than 99% of processing time [10]. The 2D Conv operation is given by Eq. 1.

$$F(x,y) = f(\sum_{j=0}^{k-1}\sum_{i=0}^{k-1} I(x - \frac{k-1}{2} + i, y - \frac{k-1}{2} + j) \times W(i,j) + b) \quad (1)$$

where $F(x,y)$ is the convolution value of the point $(x,y)$ in the output feature maps, $f$ is the activation function used for CNN (e.g., Tanh, Sigmoid, etc.), $k$ is the size of the kernel $W$ (odd number), $I(x - \frac{k-1}{2} + i, y - \frac{k-1}{2} + j)$ is a pixel value in the input feature maps, and $b$ is the bias.

### 2.2 Hardware Software Co-design

Figure 2 presents the flow of the hardware/software co-design approach. Hardware/software co-design exploits trade-offs between hardware and software to achieve system-level goals such as performance, time-to-market, or faster and better integration by designing both hardware and software concurrently [3]. Hardware/software co-design approaches can be classified into three categories. Those are co-design of embedded systems, co-design of application-specific instruction set processors (ASIPs), and co-design for reconfigurable computing (FPGAs). Compared to the first former one, the two latter ones offer more performance and suitable for accelerating computationally-intensive applications like CNNs. The co-design of ASIPs targets ASIC technologies that are not flexible and able to optimize after manufactured. In contrast, the co-design of reconfigurable computing offers more rooms to personalize for a particular application. Therefore, many studies in the literature mainly exploit the latest approach.

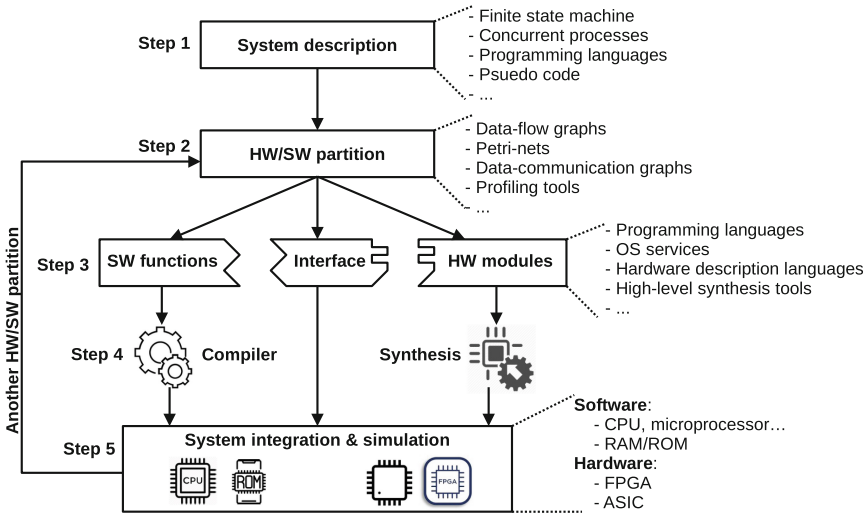



**Fig. 2.** The hardware/software co-design flow

## 3 CNN Accelerations with the HW/SW Co-design Approach

This section analyzes hardware accelerator systems for CNNs based on the hardware/software co-design in the literature. We categorize these proposals into classes FPGA-based acceleration and ASIC-based acceleration.

### 3.1 FPGA-Based Acceleration

Nowadays, FPGAs play an essential role in data sampling and processing industries due to their flexibility in custom hardware, highly parallel architecture, and low power consumption. Most FPGA-based accelerations are based on the general architecture of the FPGA-based HW accelerator, as shown in Fig. 3. At the same time, there is a soaring demand for high-energy efficiency hardware implementations in the artificial intelligence field and massively parallel computing capacity for training and inference CNNs. Some advantages of FPGAs can be listed, such as good power and performance, efficiency in parallel processing, supporting customized architecture, high on-chip memory bandwidth, low latency, high reliability, high accuracy, and a relatively short time to market. However, FPGAs usually suffer from low working frequency compared to ASIC or GPU designs. Therefore, hardware system architects can use several optimization techniques to improve the throughput of proposed systems. The surveys below are categorized according to optimization techniques used, including **increasing parallelism**, **reducing the complexity of computing**, and **exploiting data reuse**. Although a single proposal may deploy multiple optimization techniques, we classify them according to the method that contributes most to performance improvement.

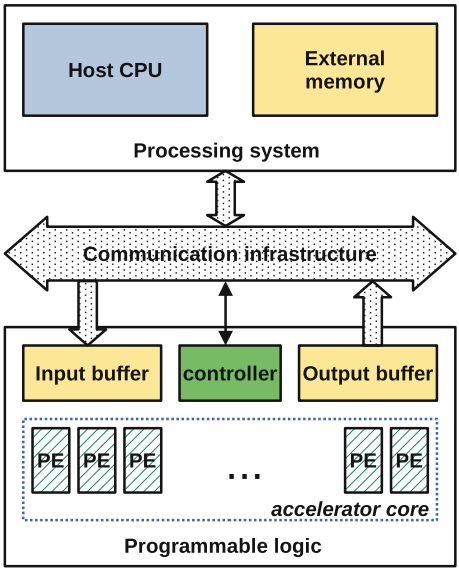
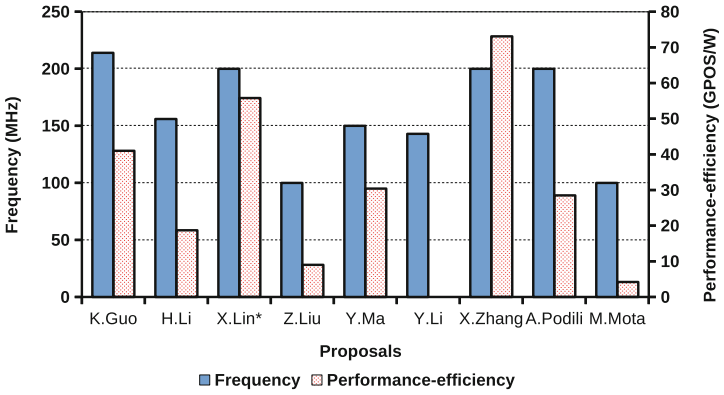



**Fig. 3.** The generic architecture of FPGA-based HW accelerator

**Increasing Parallelism:** Due to many hardware resources, FPGAs offer a high level of parallelism. This optimization technique builds several processing

elements (PEs) or exploits the loop unrolling approach to accelerate the system performance. Kaiyuan Guo et al. [8] (**K. Guo**) proposed an FPGA-based Aristotle architecture for accelerating CNNs. In this proposed architecture, an array of PEs deployed in reconfigurable fabrics calculates multiple 2D convolution operations. The PEs can process convolutions in a pipeline model to achieve the 1 pixel per cycle throughput. The entire computing systems include a CPU, external memory, and FPGA fabrics for convolution operations. A shared memory performs data communication between the host processor and PEs in the FPGA fabrics. Other works exploiting this optimization technique in the literature as follows. Research in [17] (**H. Li**) introduced a hardware accelerator CNN with all layers concurrently working in a pipeline model. Xinhan Lin et al. proposed an FPGA-based CNN with the layer clusters paralleling mapping method [19] (**X. Lin**). Parallel structures for CNNs on FPGA were suggested in [20] (**Z. Liu**). Yufei Ma et al. introduced their FPGA-based CNN architecture, fully utilizing hardware resources to achieve higher performance [22] (**Y. Ma**). The novel parallel convolution binarized architecture was proposed in [44] (**Y. Li**). Research in [48] (X. Zhang) suggested a fine-grained layer-based pipeline architecture for FPGA-based CNN's acceleration. A highly parallelized Winograd convolution computing engine was developed in [31] (**A. Podili**). Mohammad Motamedi et al. exploited exploiting all sources of parallelism in a deep CNN [26] (**M. Mota**). There may exist other works in the literature that apply the increasing parallelism technique. However, we only focus on research published in high-ranked conferences and journals and reporting detailed parameters for comparison. Figure 4 compares the proposals for maximum working frequency (MHz) and performance efficiency (Giga-operations-per-second per Watt - GOPS/W).

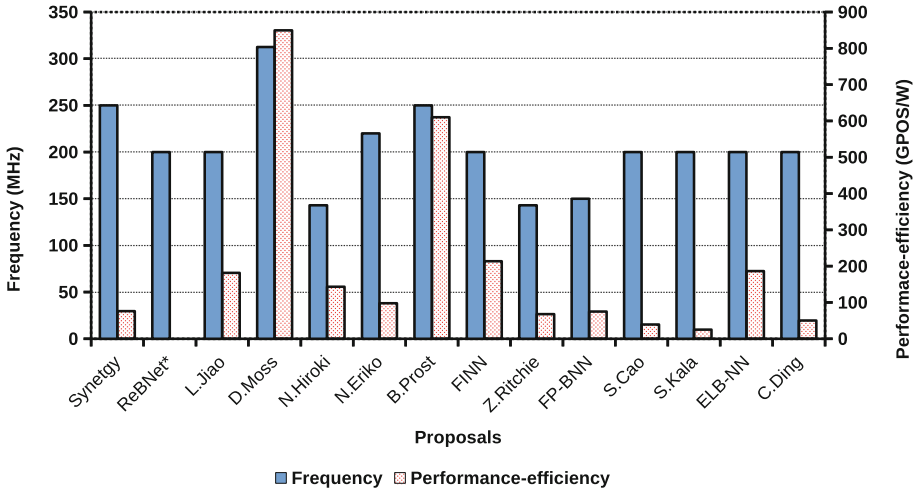



* this work did not provide power consumption; we used the experiment FPGA device to estimate. Y.Ma's proposal mentioned performance in term of frames per second instead of GOPS

**Fig. 4.** Maximum working frequency and performance-efficiency comparison of the parallelism proposals

**Reducing the Complexity of Computing:** By reducing the complexity of convolution operations, for example, pruning, multiplexer-free, shift-operator, or Binarized Neural Network (BNN), this optimization can improve the performance. Yifan Yang et al. [45] proposed the **Synetgy** FPGA-based accelerator and a novel ConvNet model for accelerating $1 \times 1$ convolutions. Due to 1-bit weight and 4-bit input, output, and activation function, the proposed system achieved an impressive improvement in performance when compared to other works. M. Ghasemzadeh et al. proposed **ReBNet** [5] for the FPGA design that uses Binary CNN with the standard `Xnor` operation. The low-bit-width computing module for FPGA was introduced in [12] (**L. Jiao**). Authors in [25] (**D. Moss**) presented their FPGA-based accelerators with both binary weights and binary activation functions. The FPGA-based BNN architecture with the XNOR-MAC circuit streaming operation was proposed in [27] (**N. Hiroki**). Eriko Nurvitadhi et al. [28] (**N. Eriko**) proposed a hardware accelerator architecture for BNNs. Research in [32] (**B. Prost**) introduced the architecture of an FPGA-based accelerator for convolutional ternary neural networks. **FINN**, a novel optimizations for mapping BNNs onto FPGAs, was presented in [38]. FPGA-based accelerators for very low precision were proposed in [49] (**Z. Ritchie**). S. Liang et al. introduced their **FP-BNN** architecture [18] with multipliers replaced by XNOR and shift operators. The Bank-Balanced Sparsity, which is a novel and efficient FPGA accelerator implementation, was presented in [1] (**S. Cao**). S. Kala et al. introduced CNN FPGA-based processing units for computing both general element-wise matrix multiplications and Winograd filtering algorithm [13] (**S. Kala**). Research in [40] proposed the FPGA-based acceleration of hybrid extremely low-bit-width CNNs called (**ELB-NN**). Caiwen Ding et al. presented their REQ-YOLO framework to implement efficient FPGA-based object detection [4] (**C. Ding**). Figure 5 compares the working frequencies and performance efficiency of the proposals.

**Exploiting Data Reuse:** Due to the limitation of on-chip buffer size (Block RAM), the data reuse optimization technique helps the system to reduce communication overhead when transferring data between the on-chip buffer and external memory. In an FPGA-based hardware accelerator system, data communication overhead usually contributes up to 50% of the entire application's execution time [30]. Jichen Wang et al. [39] (**J. Wang**) proposed a data reuse scheme called ping-pong to avoid data transfer between external memory and buffers. Research in [21] (**L. Lu**) introduced an FPGA-based architecture for employing line-buffer structure to accelerate matrix multiplication in CNNs. The FP-DNN framework was designed in [7] (**Y. Guan**) to optimize communication bandwidth used for FPGA-based CNN implementations. Qiu Jiantao et al. proposed an FPGA-based design for CNNs that reduced memory footprint to improve the performance [33] (**Q. Jiantao**). Research in [41] (**X. Wei**) implemented a systolic array architecture for low global data transfer to accelerate CNNs. Yongming Shen et al. proposed the **Escher** architecture [34] for FPGA-based CNNs acceleration focusing on data buffered on-chip. Kaiyuan Guo et al. introduced




*ReBNet mentioned performance in term of frames per second instead of GOPS

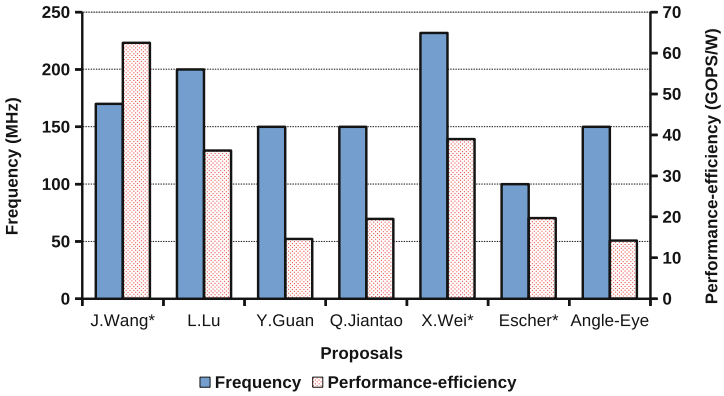
**Fig. 5.** Maximum working frequency and performance-efficiency comparison of the reducing complexity proposals

the **Angle-Eye** architecture [9] for FPGA-based CNNs to fully utilize the on-chip buffer. Similar to other optimization techniques, there may exist some other related work in the literature. However, we only focus on studies published by top conference publications and reporting detailed parameters for comparison. We compare proposals that use this technique in Fig. 6.

Figure 7 compares the three above optimization techniques in terms of average operational frequency and performance efficiency. Due to the reduction of complexity, computing cores in this technique usually require fewer hardware resources. Therefore, they often function at a higher frequency and provide better performance than computing cores in other techniques. However, they typically suffer from lower accuracy. However, they usually suffer from lower accuracy. The two approaches, exploiting data reuse and increasing parallelism, offer the same operating frequency and performance. Although they are not as good as the reducing complexity technique, they provide higher accurate results.

### 3.2 ASIC-Based Acceleration

Compared to FPGA design, ASIC-based acceleration suffers from longer time-to-market, higher design effort, and implementation cost. However, ASIC-based acceleration systems offer higher performance and energy efficiency than FPGA accelerators. Different from FPGA-based HW accelerator architecture, ASIC-based accelerations for CNNs can exploit different architectures. Moreover, instead of developing only hardware-based convolution cores, ASIC-based approaches usually implement entire systems, including general-purpose host processors and convolution cores.




* These works did not provide power consumption; we used the experiment FPGA devices to estimate.

**Fig. 6.** Maximum working frequency and Performance-efficiency comparison of the data-reuse technique proposals

Hokchhay Tann et al. [37] presented their MF-DFP accelerator for CNN models. The architecture includes an array of neural processing units (NPUs) and three independent buffers for inputs, outputs, and kernels. Each NPU contains some processing units for processing 16 neurons. 8-bit quantizations are used to eliminate multipliers resulting in only ∼1.1% accuracy reduction. Table 1 summarize details of some well-known and latest ASIC-based CNN hardware accelerator systems in the literature.

**Table 1.** Comparison of the ASIC-based accelerators for CNNs

| Proposals | Throughput | Max. Freq. | Technology | Area | Power |
|---|---|---|---|---|---|
| MF-DFP [37] | N/A | 250 MHz | 65 nm | 1.99 $mm^2$ | 138.96 mW |
| Eyeriss [2] | 42 GOPS | 250 MHz | 65 nm | 12.25 $mm^2$ | 332 mW |
| Envision [24] | 408 GOPS | 200 MHz | 28 nm | 1.87 $mm^2$ | 300 mW |
| UNPU [16] | 345.6 GOPS | 200 MHz | 65 nm | 16 $mm^2$ | 297 mW |
| Sticker [46] | 5,638 GOPS | 200 MHz | 65 nm | 7.8 $mm^2$ | 248.4 mW |
| D.Han [11] | 51.2 GOPS | 200 MHz | 65 nm | 3.52 $mm^2$ | 126 mW |
| SNAP [47] | 7,884 GOPS | 480 MHz | 16 nm | 2.4 $mm^2$ | 364 mW |

## 4 Open Issues for FPGA-Based Hardware/Software Co-design

As mentioned above, although ASIC-based designs offer higher performance and energy efficiency, they suffer from several obstacles, exceptionally long time-to-

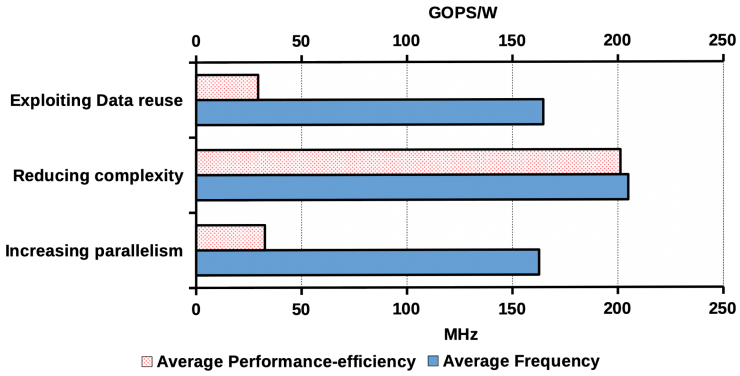



**Fig. 7.** Comparison of the three optimizations mentioned above

market. Therefore, we mainly focus on the FPGA-based approach only because there exists more room for optimization with different methods. Although many have proposed several FPGA-based hardware accelerator systems in the literature and industry, future research should consider multiple open issues when working with this approach. To the best of our knowledge, we recognize five different possible problems compared to other accelerator approaches.

1. **Low-frequency:** As shown in the previous section, FPGA-based accelerator cores function at a lower frequency than ASIC. Hence, peak throughputs of these cores are also worst than ASIC. Therefore, future research should focus on pipeline design or critical datapath optimization to enhance FPGA-based computing core working frequency. Besides, modern FPGA architecture should offer more complex configurable logic blocks and equip higher performance multiplication units inside.
2. **Design time:** Although design time and time-to-market of FPGA-based accelerator cores generally are better than ASIC, these periods are much more extended than GPU. Therefore, high-level synthesis design tools that directly synthesize high-level programming language CNN-based applications to hardware description language modules should be used more frequently. In addition, FPGA vendors and researchers should focus on the performance optimization of these tools and support more high-level programming languages instead of only C/C++ and Python.
3. **Low buffer size:** One of the most critical issues of FPGA-based convolution computing cores is the limited size of on-chip buffers. It, in turn, increases processing time when processing more and more data. While designers or architects can configure the size of memory in ASIC and GPU's internal memory is enormous, FPGAs suffer from a low amount of on-chip memory. Therefore, FPGA-based CNN computing cores should be equipped with a better memory hierarchy. Distributed memory can also be a solution for the low buffer issue. However, this may increase the complexity of the design. This problem, in turn, can reduce the working frequency of the system.

4. **Data communication overhead:** As discussed in [30], the communication overhead of a generic FPGA-based accelerator system contributes to half of system processing time. Therefore reducing data communication overhead is an essential demand for future research, especially in CNN-based applications due to enormous data. Furthermore, researchers should investigate high-performance communication infrastructure for host and cores in CNNs such as hybrid interconnect or reconfigurable on-chip interconnection networks [29].
5. **Design space exploration:** Finally, one of the most critical requirements to achieve to most optimized systems in terms of performance and energy efficiency is design space exploration. Compared to GPU designs, FPGA-based designs suffer from longer design time. Therefore, it is much difficult for us to explore the design space of FPGA-based systems than GPU. Hence, researchers should develop estimation tools or design space exploration tools for CNNs-based applications for FPGAs. Researchers must have much knowledge in both deep neural networks and FPGA design to cope with this issue.

Figure 8 compares the difficulty levels of the five open issues in hardware/software co-design for CNNs. To the best of our experience, the **design space exploration** is the most challenging issue since researchers need to be experts in both deep neural networks and FPGA design. Meanwhile, **design time** is the most straightforward issue due to many high-level synthesis tools released in the past years.

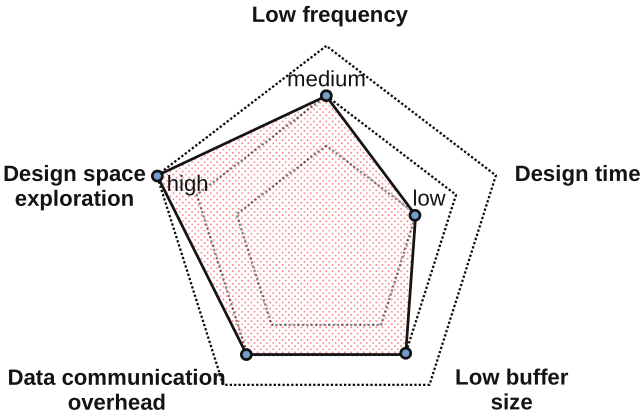



**Fig. 8.** The difficulty level of the five open issues

## 5 Conclusion

In this paper, we present one of the most promising approaches for accelerating the performance of convolutional neural networks computing, the hardware/software co-design. We survey proposals submitted in high-ranking journals and conferences for this approach. We categorize these proposals into two classes, FPGA-based and ASIC-based accelerators. The former accelerators offer more room to optimize than the latter one. Therefore, we mainly analyze and compare proposed systems with the first category. We finally introduce five different research open issues for future work on this topic. These hardware/software co-design challenges for accelerating CNNs are research directions for further optimizing the performance and energy of CNN-based applications.

**Acknowledgment.** This research is funded by Vietnam National University - Ho Chi Minh City under grant number B2021-20-02.

## References

1. Cao, S., et al.: Efficient and effective sparse LSTM on FPGA with bank-balanced sparsity. In: Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, FPGA 2019, pp. 63–72. Association for Computing Machinery (2019)
2. Chen, Y.H., Krishna, T., Emer, J.S., Sze, V.: Eyeriss: an energy-efficient reconfigurable accelerator for deep convolutional neural networks. IEEE J. Solid-State Circ. **52**(1), 127–138 (2017)
3. DeMicheli, G., Sami, M.: Hardware/Software Co-design. Nato Science Series E. Springer, Netherlands (1996). https://www.springer.com/gp/book/9780792338833
4. Ding, C., Wang, S., Liu, N., Xu, K., Wang, Y., Liang, Y.: REQ-YOLO: a resource-aware, efficient quantization framework for object detection on FPGAs. In: Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, FPGA 2019, pp. 33–42. Association for Computing Machinery, New York (2019). https://doi.org/10.1145/3289602.3293904
5. Ghasemzadeh, M., Samragh, M., Koushanfar, F.: ReBNet: residual binarized neural network. In: 2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), pp. 57–64. IEEE Computer Society, Los Alamitos, May 2018
6. Google Inc.: Cloud tensor processing units. https://cloud.google.com/tpu/docs/tpus. Accessed 2 June 2021
7. Guan, Y., et al.: FP-DNN: an automated framework for mapping deep neural networks onto FPGAs with RTL-HLS hybrid templates. In: 2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), pp. 152–159 (2017)
8. Guo, K., Han, S., Yao, S., Wang, Y., Xie, Y., Yang, H.: Software-hardware codesign for efficient neural network acceleration. IEEE Micro **37**(2), 18–25 (2017)
9. Guo, K., et al.: Angel-eye: a complete design flow for mapping CNN onto embedded FPGA. IEEE Trans. Comput. Aided Des. Integr. Circ. Syst. **37**(1), 35–47 (2018)
10. Guo, K., Zeng, S., Yu, J., Wang, Y., Yang, H.: [DL] a survey of FPGA-based neural network inference accelerators. ACM Trans. Reconfigurable Technol. Syst. **12**(1), 1–26 (2019)

11. Han, D., Lee, J., Lee, J., Yoo, H.J.: A low-power deep neural network online learning processor for real-time object tracking application. IEEE Trans. Circ. Syst. I Regul. Pap. **66**(5), 1794–1804 (2019)
12. Jiao, L., Luo, C., Cao, W., Zhou, X., Wang, L.: Accelerating low bit-width convolutional neural networks with embedded FPGA. In: 2017 27th International Conference on Field Programmable Logic and Applications (FPL), pp. 1–4 (2017)
13. Kala, S., Jose, B.R., Mathew, J., Nalesh, S.: High-performance CNN accelerator on FPGA using unified winograd-GEMM architecture. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **27**(12), 2816–2828 (2019). https://doi.org/10.1109/TVLSI.2019.2941250
14. Khan, A., Sohail, A., Zahoora, U., Qureshi, A.S.: A survey of the recent architectures of deep convolutional neural networks. Artif. Intell. Rev. **53**(8), 5455–5516 (2020). https://doi.org/10.1007/s10462-020-09825-6
15. Lacey, G., Taylor, G.W., Areibi, S.: Deep learning on FPGAs: past, present, and future (2016). https://arxiv.org/abs/1602.04283
16. Lee, J., Kim, C., Kang, S., Shin, D., Kim, S., Yoo, H.J.: UNPU: a 50.6 TOPS/W unified deep neural network accelerator with 1b-to-16b fully-variable weight bit-precision. In: 2018 IEEE International Solid - State Circuits Conference - (ISSCC), pp. 218–220 (2018)
17. Li, H., Fan, X., Jiao, L., Cao, W., Zhou, X., Wang, L.: A high performance FPGA-based accelerator for large-scale convolutional neural networks. In: 2016 26th International Conference on Field Programmable Logic and Applications (FPL), pp. 1–9 (2016)
18. Liang, S., Yin, S., Liu, L., Luk, W., Wei, S.: FP-BNN: binarized neural network on FPGA. Neurocomputing **275**, 1072–1086 (2018)
19. Lin, X., Yin, S., Tu, F., Liu, L., Li, X., Wei, S.: LCP: a layer clusters paralleling mapping method for accelerating inception and residual networks on FPGA. In: 2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC), pp. 1–6 (2018)
20. Liu, Z., Dou, Y., Jiang, J., Xu, J.: Automatic code generation of convolutional neural networks in FPGA implementation. In: 2016 International Conference on Field-Programmable Technology (FPT), pp. 61–68 (2016)
21. Lu, L., Liang, Y., Xiao, Q., Yan, S.: Evaluating fast algorithms for convolutional neural networks on FPGAs. In: 2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), pp. 101–108 (2017)
22. Ma, Y., Cao, Y., Vrudhula, S., Seo, J.S.: 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 2017, pp. 45–54. ACM, New York (2017)
23. Mittal, S.: A survey of FPGA-based accelerators for convolutional neural networks. Neural Comput. Appl. **32**(4), 1109–1139 (2020)
24. Moons, B., Uytterhoeven, R., Dehaene, W., Verhelst, M.: 14.5 envision: a 0.26-to-10TOPS/W subword-parallel dynamic-voltage-accuracy-frequency-scalable convolutional neural network processor in 28nm FDSOI. In: 2017 IEEE International Solid-State Circuits Conference (ISSCC), pp. 246–247 (2017)
25. Moss, D.J.M., et al.: High performance binary neural networks on the xeon+FPGA$^{TM}$ platform. In: 2017 27th International Conference on Field Programmable Logic and Applications (FPL), pp. 1–4 (2017)

26. Motamedi, M., Gysel, P., Akella, V., Ghiasi, S.: Design space exploration of FPGA-based deep convolutional neural networks. In: 2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 575–580 (2016)
27. Nakahara, H., Fujii, T., Sato, S.: A fully connected layer elimination for a binarizec convolutional neural network on an FPGA. In: 2017 27th International Conference on Field Programmable Logic and Applications (FPL), pp. 1–4 (2017)
28. Nurvitadhi, E., Sheffield, D., Sim, J., Mishra, A., Venkatesh, G., Marr, D.: Accelerating binarized neural networks: Comparison of FPGA, CPU, GPU, and ASIC. In: 2016 International Conference on Field-Programmable Technology (FPT), pp. 77–84 (2016)
29. Oveis-Gharan, M., Khan, G.N.: Reconfigurable on-chip interconnection networks for high performance embedded SOC design. J. Syst. Architect. **106**, 101711 (2020)
30. Pham-Quoc, C., Al-Ars, Z., Bertels, K.: Heterogeneous hardware accelerators interconnect: an overview. In: 2013 NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2013), pp. 189–197 (2013)
31. Podili, A., Zhang, C., Prasanna, V.: Fast and efficient implementation of convolutional neural networks on FPGA. In: 2017 IEEE 28th International Conference on Application-specific Systems, Architectures and Processors (ASAP), pp. 11–18 (2017)
32. Prost-Boucle, A., Bourge, A., Pétrot, F., Alemdar, H., Caldwell, N., Leroy, V.: Scalable high-performance architecture for convolutional ternary neural networks on FPGA. In: 2017 27th International Conference on Field Programmable Logic and Applications (FPL), pp. 1–7 (2017)
33. Qiu, J., et al.: Going deeper with embedded FPGA platform for convolutional neural network. In: Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, FPGA 2016, pp. 26–35. ACM, New York (2016)
34. Shen, Y., Ferdman, M., Milder, P.: Escher: A CNN accelerator with flexible buffering to minimize off-chip transfer. In: 2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), pp. 93–100 (2017)
35. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition (2015). https://arxiv.org/abs/1409.1556
36. Strigl, D., Kofler, K., Podlipnig, S.: Performance and scalability of GPU-based convolutional neural networks. In: 2010 18th Euromicro Conference on Parallel, Distributed and Network-based Processing, pp. 317–324 (2010)
37. Tann, H., Hashemi, S., Bahar, R.I., Reda, S.: Hardware-software codesign of accurate, multiplier-free deep neural networks. In: 2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC), pp. 1–6 (2017)
38. Umuroglu, Y., et al.: FINN: a framework for fast, scalable binarized neural network inference. In: Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, FPGA 2017, pp. 65–74. ACM, New York (2017)
39. Wang, J., Lin, J., Wang, Z.: Efficient hardware architectures for deep convolutional neural network. IEEE Trans. Circ. Syst. I Regul. Pap. **65**(6), 1941–1953 (2018)
40. Wang, J., Lou, Q., Zhang, X., Zhu, C., Lin, Y., Chen, D.: Design flow of accelerating hybrid extremely low bit-width neural network in embedded FPGA. In: 2018 28th International Conference on Field Programmable Logic and Applications (FPL), pp. 163–1636 (2018). https://doi.org/10.1109/FPL.2018.00035
41. Wei, X., et al.: Automated systolic array architecture synthesis for high throughput CNN inference on FPGAs. In: 2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC), pp. 1–6 (2017)

42. Williams, R.: What's next? [The end of Moore's law]. Comput. Sci. Eng. **19**(02), 7–13 (2017)
43. Wu, R., Guo, X., Du, J., Li, J.: Accelerating neural network inference on FPGA-based platforms-a survey. Electronics **10**(9), 1025 (2021)
44. Yang, L., He, Z., Fan, D.: A fully onchip binarized convolutional neural network FPGA impelmentation with accurate inference. In: Proceedings of the International Symposium on Low Power Electronics and Design, ISLPED 2018. ACM, New York (2018)
45. Yang, Y., et al.: Synetgy: Algorithm-hardware co-design for convnet accelerators on embedded FPGAs. In: Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, FPGA 2019, pp. 23–32. ACM, New York (2019)
46. Yuan, Z., et al.: Sticker: a 0.41-62.1 TOPS/W 8bit neural network processor with multi-sparsity compatible convolution arrays and online tuning acceleration for fully connected layers. In: 2018 IEEE Symposium on VLSI Circuits, pp. 33–34 (2018)
47. Zhang, J.F., Lee, C.E., Liu, C., Shao, Y.S., Keckler, S.W., Zhang, Z.: SNAP: a 1.67—21.55TOPS/W sparse neural acceleration processor for unstructured sparse deep neural network inference in 16nm CMOS. In: 2019 Symposium on VLSI Circuits, pp. C306–C307 (2019)
48. Zhang, X., et al.: DNNBuilder: an automated tool for building high-performance DNN hardware accelerators for FPGAs. In: Proceedings of the International Conference on Computer-Aided Design, ICCAD 2018. ACM, New York (2018)
49. Zhao, R., et al.: Accelerating binarized convolutional neural networks with software-programmable FPGAs. In: Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, FPGA 2017, pp. 15–24. ACM, New York (2017)