Hardware Acceleration Techniques for Convolutional Layers in Convolutional Neural Networks

Jieshen Cai[1]

[1]Dundee International Institute, Central South University, Changsha, Hunan, 410083, China
7802220130@csu.edu.cn

Abstract. Convolutional Neural Networks (CNNs) are widely used in deep learning because of the power of convolutional operations. However, the convolutional layer often becomes a performance bottleneck when performing complex computations, especially in resource-constrained environments. To address this problem, hardware accelerators have emerged to improve the convolutional layer's performance and efficiency. This review provides an overview of the main hardware acceleration solutions: Application-Specific Integrated Circuits (ASICs), Field-Programmable Gate Arrays (FPGAs), and Graphics Processing Units (GPUs). In addition, it explores different optimization techniques for hardware accelerators such as parallelization, quantization, and sparsity exploitation. Finally, this review looks at the future direction of convolutional layer hardware accelerators, including the co-optimization of hardware and software, reducing the power consumption for edge computing, enhancing the versatility, and overcoming the bottleneck of memory access, and highlights key challenges and possible priorities for future research. Through an in-depth analysis of current research results and future trends, this review aims to provide a comprehensive reference for the design and optimization of convolutional layer hardware accelerators.

Keywords: Convolutional Neural Networks, Convolutional Layers, Hardware Acceleration, Optimization

1 Introduction

Convolutional Neural Networks (CNNs) have become a cornerstone in deep learning, excelling at processing and interpreting visual data. Their architecture enables them to learn spatial hierarchies of features through convolutions. This capability allows machines to recognize patterns and make informed decisions across various applications, including image and video recognition, natural language processing (NLP), and autonomous driving.

At the heart of CNNs lies the convolutional layer, which is responsible for detecting patterns within data. This layer performs convolutions using learnable kernels. It traverses input data to produce feature maps, highlighting unique spatial attributes while reducing data complexity. The effectiveness of these convolutions is crucial, as they directly impact the accuracy and efficiency of the entire neural network.

J. Iqbal (ed.), *Proceedings of the 2025 2nd International Conference on Mechanics, Electronics Engineering and Automation (ICMEEA 2025)*, Advances in Engineering Research 272,
https://doi.org/10.2991/978-94-6463-821-9_46

However, the convolution process involves extensive computations, often engaging millions of pixels or nodes through multi-dimensional matrix operations. This high computational demand can become a bottleneck, especially in real-time applications with limited resources. Therefore, optimizing the convolutional layer is essential for deploying CNNs effectively in such environments.

In order to address the challenges targeting convolution in CNNs, hardware acceleration has emerged as an important solution to enhance the efficiency and performance of convolutional layers. Tailored hardware accelerators can address the computational limitations of CNN operations. This can be particularly helpful in resource-constrained settings, like edge computing, mobile devices, Internet of Things (IoT) devices, and embedded systems. These accelerators not only boost computational throughput but also reduce energy consumption.

Despite the advantages, designing efficient hardware accelerators presents challenges. Balancing acceleration speeds with precision and accuracy requirements is complex, especially when computational resources are limited. As CNNs grow in size and complexity, particularly with deeper convolutional layers, developing optimization strategies becomes more crucial. These strategies must effectively handle high-volume data processing, manage memory bandwidth, and minimize power consumption, all while maintaining precision. Such optimization is important for systems constrained by power and size limitations, such as wearables or IoT devices.

This review explores various strategies to enhance CNN performance through hardware accelerators for convolutional layers. It examines three primary types of hardware accelerators: Application-Specific Integrated Circuits (ASICs), Field-Programmable Gate Arrays (FPGAs), and Graphics Processing Units (GPUs). Each type offers unique attributes, and the discussion synthesizes current research to demonstrate the practical effectiveness of these solutions. Factors such as performance capability, energy efficiency, and scalability are considered, highlighting the advantages and challenges associated with each type of accelerator.

Furthermore, the review looks at key optimization techniques for enhancing convolutional layers: parallelism, quantization, and exploiting sparsity. These advanced approaches aim to balance performance improvements with minimal compromises on precision and quality. The effectiveness of these techniques is evaluated based on current research, outlining their respective strengths and weaknesses.

Finally, the review analyses technological development realities, identifies the related challenges, and explores potential future directions and emerging trends in optimizing convolutional layers using hardware accelerators.

This review aims to deepen the understanding of hardware acceleration in CNNs and expects to serve as a guide for ongoing research and development efforts.

2 Hardware Acceleration Solutions for Convolutional Layers

This section will explore three main hardware acceleration solutions for convolutional layers: Application-Specific Integrated Circuits (ASICs), Field-Programmable Gate Arrays (FPGAs), and Graphics Processing Units (GPUs). Each platform is unique and offers different advantages and challenges for convolutional layer acceleration. By analysing the characteristics of each hardware, this section aims to provide a

comprehensive perspective on the research and application of convolutional layer hardware acceleration.

2.1 Application-Specific Integrated Circuit (ASIC)

Application-Specific Integrated Circuits (ASICs) are integrated circuits designed for a particular application instead of general-purpose use. Since ASICs employ customized hardware architectures, they provide significant advantages in computational capabilities that fulfill high bandwidth and low latency requirements. This means they are well suited for convolutional layer acceleration. ASICs can also be designed to be highly integrated. This means that they can fit into space-critical applications.

2.1.1 Related Research.

Several remarkable achievements utilizing ASICs for convolutional layer acceleration have emerged in recent years. Google's Tensor Processing Unit (TPU) serves as a prime example. TPU leverages optimized matrix multiplication units and high-bandwidth memory architectures. It is specifically developed to accelerate algorithms of machine learning, particularly for executing training and inference of deep learning models [1]. Research has proven that TPUs have demonstrated an advantage in terms of performance, price/performance and energy ratio over traditional CPUs and GPUs [2].

In addition, studies have explored hardware optimizations to further improve the performance of ASICs in convolutional layer acceleration. Lee et al. proposed a convolutional block architecture based on a bit-level multiply-accumulator (BLMAC) paired with an advanced Booth encoder, specifically designed to enhance the computational process within convolutional layers [3]. This innovative architecture optimizes the critical path of convolution operation, resulting in a remarkable reduction in latency by 53%. The reduction in latency not only reveals the efficacy of the proposed approach but also facilitates higher throughput. This enables real-time processing of complex neural network operations, which is important for energy-constrained applications.

2.1.2 Advantages and Limitations.

ASICs have significant advantages in accelerating convolutional layers, primarily characterized by high performance and low power consumption. They ensure high throughput and low latency during convolution operations, making them highly efficient in large-scale deep learning computations.

However, the development costs for custom ASICs are typically high, creating barriers for small enterprises to afford such solutions. Because ASICs are designed for specific tasks, they lack flexibility and may struggle to adapt to the evolving algorithms or network structures as well [1]. Another limitation of ASICs is that their development cycle tends to be longer compared with other platforms, which may impact rapid iteration and responsiveness to market needs.

2.2 Field-Programmable Gate Array (FPGA)

Field-Programmable Gate Arrays (FPGAs) are reconfigurable hardware devices that offer significant versatility in their application. Unlike ASICs, FPGAs can be programmed to meet the specific requirements of different applications, which allows for adaptations to varying convolution algorithms and application requirements. Their capability to perform parallel processing also supports the effective implementation of pipelining techniques, leading to substantial performance improvements.

2.2.1 Related Research.

In recent years, there has been a significant focus on developing FPGA-based hardware accelerators to enhance the performance of convolutional layers. Several studies have presented advances in FPGA-based hardware accelerators for convolutional layers. They demonstrate substantial improvements in factors such as computational efficiency, speed, and energy consumption. For instance, Yan et al. introduced a specifically designed FPGA-based accelerator [4]. They implemented a deep flow mechanism and leveraged FPGA's high parallelism to improve computational efficiency. By optimizing the image data input format, their accelerator reduced cache requirements across network layers, conserving substantial RAM resources. The on-board testing based on a yolov5s neural network showed that the designed architecture has an acceleration ratio of up to 142x, while consuming extremely low power. The experimental results highlight the accelerator's effectiveness in real-time applications.

2.2.2 Advantages and Limitations.

The adaptability of FPGAs makes them an ideal choice for many convolution acceleration tasks. Due to FPGAs' inherent parallelism and reconfigurability, they can offer notable advantages in accelerating convolutional layers. Their flexibility allows for the customization of hardware architectures to efficiently execute convolution operations, leading to significant improvements in computational speed and energy efficiency. FPGAs also provide a good balance between cost and energy efficiency [5].

FPGA implementations also face certain challenges. The resource constraints of FPGA platforms can restrict the size and complexity of the convolutional networks that can be effectively implemented. Debugging FPGA designs is typically more complex than for ASICs or GPUs as well, potentially leading to longer time-to-market durations for systems leveraging FPGA technologies.

2.3 Neural Processing Unit (GPU)

Graphics Processing Units (GPUs) are widely recognized for their exceptional parallel computation capabilities. GPUs have a large number of cores designed to handle multiple tasks simultaneously, which allows them to execute the various operations required in convolutional computations efficiently. GPUs usually have extensive software support and development tools as well. This makes deployment easier.

Libraries such as CUDA and cuDNN can further enhance GPU performance by providing optimized algorithms specifically designed for deep learning applications.

2.3.1 Related Research.

In a study that conducted a comprehensive comparison between GPU and CPU performance for training different CNN models, including FCN-5, FCN-8, AlexNet, ResNet-50, and LSTM variants, the authors evaluated the performance based on the iteration duration [2]. The study shows that using GPU-based computing resources resulted in substantial speedups for all deep learning tools evaluated. GPUs outperformed CPUs in training various CNN architectures, demonstrating the advantages of GPU's parallel computing capabilities in accelerating deep learning tasks. This performance boost is crucial, especially for complex models where the computational demands are high. The general consensus in the field is that GPUs are capable of handling the massive parallelism and computational intensity of convolutional operations, resulting in significantly shorter training times and improved model efficiency.

2.3.2 Advantages and Limitations.

GPUs offer significant advantages in accelerating convolutional layers mainly due to their massively parallel processing capabilities and high computational throughput. With thousands of cores designed for parallel execution, GPUs can handle large-scale convolution operations efficiently, making them highly effective in processing complex deep learning models. With widely available support for frameworks such as TensorFlow and PyTorch, GPU-based implementations also benefit from a mature ecosystem and software stack. This makes it easier for researchers to leverage GPU hardware for convolutional layer acceleration.

Nevertheless, GPUs have certain drawbacks despite their powerful computational abilities. One key challenge is the high power consumption of GPUs, especially when operating at full capacity for large-scale convolutional operations. This can make GPUs less suitable for mobile or embedded systems. Moreover, while GPUs excel at parallel computation, they may not be as efficient as specialized accelerators like ASICs or FPGAs when it comes to fine-tuning hardware for specific tasks. This can result in suboptimal performance in certain scenarios, particularly for applications with unique hardware requirements. Additionally, GPUs tend to be more expensive than alternatives.

3 Optimization Techniques for Convolutional Layer Acceleration

In this section, three optimization techniques for accelerating convolutional layers in hardware accelerators will be discussed: parallelism, quantization, and exploiting sparsity. They enhance performance from both hardware and algorithm perspectives. Parallelism leverages hardware accelerator’s ability to execute multiple operations

concurrently. Quantization is primarily an algorithmic move that depends on hardware support. It reduces numerical precision to ease computational loads. Exploiting Sparsity bridges both worlds by identifying redundant computations and skipping them through algorithmic pruning and hardware-level support. Collectively, these strategies form a robust framework for optimizing convolutional computations in hardware accelerators.

3.1 Parallelism

Parallelism is a cornerstone of convolutional layer hardware acceleration. By dividing the convolution workload into smaller independent tasks, this approach allows concurrent processing of data elements or channels through techniques such as data parallelism, task parallelism, and pipelining. Although it is inherently hardware-focused optimization, strategies like tiling and workload partitioning can further enhance its effectiveness.

3.1.1 Related Research.

Parallelization techniques have been widely applied in hardware acceleration of convolutional layers in recent years. Many studies have explored the application of this technique and demonstrated their effectiveness and criticality. Research by Xiong proposed an FPGA-based hardware accelerator for CNNs [6]. It explored the parallelism of convolutional layer operation by deeply analyzing the forward operation principle of convolutional layer. The research also designed a hardware architecture with parallel input channels, parallel output channels and convolutional window depth pipeline. In this architecture, a fully parallel multiply-add-tree module is designed to accelerate the convolution operation, and the pipelined operation of the convolution window is achieved by an efficient window buffer module. Experimental results show that the accelerator achieves an energy efficiency ratio of 32.73 GOPS/W and a performance of 317.86 GOPS.

Wang et al. proposed a new model parallelization approach to decouple the structure of CNN through group convolution and channel rearrangement procedures, aiming to reduce the memory footprint of each device and eliminate the cost of synchronization between devices [7]. They designed a parallel FPGA accelerator optimized for ShuffleNet, the classical CNN model. Experimental results show that their approach significantly increased the speed of ShuffleNet (1.42 times faster) and reduced the memory footprint, while maintaining accuracy.

3.1.2 Analysis.

Studies have shown that the performance of hardware accelerators for convolutional layers can be effectively improved by parallelization techniques. However, parallelization approaches are facing some challenges. Hardware resource constraints may affect the extent of parallelization and pipelining, resulting in limited performance enhancement. Increased design complexities may lead to longer development cycles as well. In addition, the co-optimization of hardware and algorithms still requires further research to achieve more efficient acceleration.

3.2 Quantization

Quantization is a technique that reduces the precision of weights and activations in convolutional layers. This approach enables faster arithmetic operations and lower power consumption by representing data with fewer bits, typically using formats like INT8. While quantization is fundamentally an algorithmic optimization that involves altering numerical representations, it depends on hardware support for low-precision arithmetic. Methods such as quantization-aware training and post-training quantization help mitigate accuracy loss while reaping efficiency benefits. Consequently, quantization is a powerful strategy to enhance efficiency and throughput in convolutional accelerators and facilitate the deployment of CNNs on resource-constrained hardware platforms.

3.2.1 Related Research.

Studies have proven the effectiveness of quantization approaches in optimizing convolutional performance for hardware acceleration. For instance, Bao et al. introduced a learnable parameter soft clipping full integer quantization (LSFQ) method [8]. It employs a learnable clipping parameter to achieve efficient quantization of both weights and activations, enabling integer-only arithmetic operations. The evaluation of the quantization approach based on several CNN models demonstrated that the LSFQ method had a less than 1% accuracy loss even with 3- or 4-bit quantization. This highlights its effectiveness in maintaining model performance while reducing computational requirements.

3.2.2 Analysis.

It is clear that through reducing the bit-width of weights and activations, various quantization approaches contribute to significant improvements in various aspects for convolution layer hardware accelerator, such as computational efficiency and memory usage. But challenges remain in balancing the trade-off between reduced precision and model accuracy, as aggressive quantization can lead to a drop in performance. Future research is needed to develop adaptive quantization strategies that dynamically adjust precision levels to maintain accuracy while achieving hardware efficiency.

3.3 Exploiting Sparsity

In the field of hardware acceleration of convolutional layers, making full use of sparsity has become one of the key techniques. Sparsity refers to the presence of a large number of zeros in the weights and activation values of convolutional layers. They do not contribute valid information and can therefore be ignored, thus reducing computation and memory access. Exploiting sparsity reduces the number of effective computations by selectively processing only non-zero values, thereby lowering memory bandwidth requirements and improving overall efficiency. On the algorithmic side, approaches like pruning and low-rank approximations intentionally create sparse models, which subsequently reduce computational demands. As a complement to these algorithmic

approaches, hardware accelerator designs can incorporate specialized data formats and control logic to bypass unnecessary operations. The combination of algorithm design and hardware support not only accelerates convolutional computations but also contributes to reduced power consumption and enhanced scalability.

3.3.1 Related Research.

In recent years, academics have conducted research on the application of sparsity in convolutional layer hardware acceleration. Liu et al. proposed a structured sparsity-based hardware accelerator, S2TA, aimed at improving the energy efficiency of quantized convolution inference on mobile devices [9]. The accelerator exploits the density-bounded block (DBB) sparsity of weights and activation values and achieves efficient use of sparsity through hardware primitives. Experimental results show that S2TA achieves more than double speedup and reduced energy consumption compared to conventional system arrays with zero-value clock gating.

Xu et al. on the other hand proposed a new parallel strategy that aims to take full advantage of the sparsity of the input feature maps on GPU platforms [10]. They designed a new storage format that fuses convolutional operations with subsequent pooling operations to reduce the data transfer between CPU and GPU and the loading time of GPU memory. In experiments with several main CNN models such as VGG-19, ResNet-50, DenseNet-121, and RegNetX-16GF, the method achieves around double speedups on top of cuDNN and cuSPARSE.

3.3.2 Analysis.

It can be seen that exploiting sparsity can significantly improve the performance and energy efficiency of hardware accelerators for convolutional layers. Through the co-design of hardware and algorithms, the amount of computation and memory access can be effectively reduced, thus accelerating the inference process of CNNs. Challenges that sparsity exploitation shows still exist despite the great potential in convolutional layer hardware acceleration it has shown. First, the predictability and controllability of sparsity patterns are crucial for hardware design. For example, sparsity exploitation requires accurate modelling of the sparsity of the feature graph for efficient hardware implementation [10]. Secondly, the complexity and energy overhead of the hardware implementation are also issues that need to be taken into account and will continue to be optimized in the future.

4 Future Directions and Challenges

Currently, there are still many issues and pain points facing the research and design of hardware accelerators for convolutional layers, but they also potentially point to major future directions for related development. This section will explore these trends, discuss the inherent challenges, and propose potential solutions that may pave the way for future-generation convolutional layer hardware accelerators.

4.1 Co-optimization of Hardware and Software

In hardware accelerator design, pure hardware modifications cannot provide as much performance improvement as algorithmic level optimizations, and hardware and software co-designs can further increase the improvement dramatically [11]. This technique can help achieve better balance in terms of performance, power consumption and adaptability.

Hardware-software co-design will be a key principle in the future design of hardware acceleration for convolutional layers, and indeed for almost all neural network architectures and algorithms. Hardware-software co-design makes sure that the specific demand for mobile devices and embedded systems can be fully met through reducing the time spent on development and enhancing resource utilization [12]. This means new challenges, including increased design complexity and the necessity for adaptable, evolving solutions that can keep pace with rapid advances in deep learning algorithms. Overcoming these challenges will require innovative co-design frameworks and cross-disciplinary collaboration.

4.2 Reducing Power Consumption for Edge Computing

With the development of edge computing, the need for low-power convolutional layer hardware accelerators has risen rapidly. Several algorithm-level optimization strategies, like quantization and sparsity exploitation that are discussed in this review, can help reduce the power consumption. Additionally, innovations in hardware technologies, such as power gating and dynamic voltage and frequency scaling (DVFS), are expected to play important roles. Research has demonstrated that these techniques can effectively reduce power consumption while maintaining hardware computing performance [13]. A number of hardware accelerator solution providers have also actively launched low-power accelerator solutions for edge devices, such as the edge TPU from Google [14]. The primary challenge remains in achieving an optimal balance between performance and power efficiency, particularly when deploying in environments with limited energy resources.

4.3 Enhancing Versatility

With the gradual diversification of CNN algorithm architectures and application scenarios, the need for generalization of hardware accelerators is increasing. Hardware accelerators in the future need to be more versatile to support various convolution algorithms, like standard convolution, depth-separable convolution, etc. Several recent research have been committed to developing hardware designs that can support a wide range of convolutional algorithms to meet different needs. For example, Sedaghatgoo et al. proposed ARMAN, a reconfigurable monolithic 3D accelerator architecture capable of reconfiguring based on the neural network structure [15].

What similar types of research reveal is that the future development of convolutional layer hardware accelerators needs to be designed to achieve higher configurability and scalability, in order to meet the demands of various convolutional algorithms. This

requires hardware designs that can flexibly adapt to different algorithm operations and their requirements. Therefore, finding efficient approaches to hardware architectures that are more configurable and scalable will be a main challenge for the future development of hardware accelerators.

4.4 Overcoming Memory Access Bottlenecks

As the demand for deep learning applications grows, the memory access speed has become a main bottleneck limiting the performance of hardware accelerators [16]. The computation speed of hardware accelerators keeps increasing rapidly, while the memory access speed is still relatively lagging behind. This phenomenon results in failure of hardware accelerators to fully utilize their performance. Therefore, overcoming this bottleneck is a key direction to improve the efficiency of hardware accelerators.

To address this issue, researchers are exploring the use of novel memory technologies that can be applied to hardware accelerators, in order to help improve data transfer speed. For instance, the development of High Bandwidth Memory (HBM) has been a significant advancement. HBM offers higher data transfer rates compared to conventional memory types, thereby reducing latency and improving throughput for memory access [17]. In addition, optimizing data flow and memory access patterns is a key strategy. For example, adopting data prefetching and caching strategies can effectively reduce memory access latency.

All these approaches still face challenges such as cost and power consumption practically. Future research should continue focusing on developing cost-effective and energy-efficient solutions to memory access bottlenecks. This helps to further enhance the overall performance of convolutional layer hardware accelerators.

5 Conclusion

This review summarizes the mainstream hardware accelerator solutions, including ASIC, FPGA and GPU. Several current studies are reviewed to demonstrate the potential of each option, and their strengths and weaknesses are also analysed. Focusing on how to solve the bottlenecks posed by convolutional layers, this review also looks at some key optimization techniques of hardware accelerators. Through systematic analysis, the characteristics of each optimization technique and the corresponding issues yet to be addressed in the future are pointed out. Finally, this review looks ahead and identifies some of the key challenges in the development of hardware accelerators for convolutional layers, as well as the main directions for future studies. Hardware accelerators for convolutional layers will continue to evolve around these directions, further expanding the widespread applications of hardware accelerators in deep learning.

References

1. Ravikumar, A., Sriraman, H., Saketh, P.M.S., et al.: 'Effect of neural network structure in accelerating performance and accuracy of a convolutional neural network with GPU/TPU for image analytics', PeerJ Computer Science, 2022, 8, p.e909
2. Hu, Y., Liu, Y., Liu, Z.: 'A survey on convolutional neural network accelerators: GPU, FPGA and ASIC', 14th International Conference on Computer Research and Development, Shenzhen, China, 2022, pp. 100-107
3. Lee, S.S., Nguyen, T.D., Meher, P.K., et al.: 'Energy-efficient high-speed ASIC implementation of convolutional neural network using novel reduced critical-path design', IEEE Access, 2022, 10, pp. 34032-34045
4. Yan, R., Yi, J., He, J., et al.: 'FPGA-based convolutional neural network design and implementation', 3rd Asia-Pacific Conference on Communications Technology and Computer Science, Shenyang, China, 2023, pp. 456-460.
5. Yan, F., Koch, A., Sinnen, O.: 'A survey on FPGA-based accelerator for ML models', 2024, arXiv preprint arXiv:2412.15666
6. Jun, X.: 'FPGA deep learning acceleration based on convolutional neural network', 2020, arXiv preprint arXiv:2012.03672
7. Wang, J., Tong, W., Zhi, X.: 'Model parallelism optimization for CNN FPGA accelerator', Algorithms, 2023, 16, (2), pp. 110
8. Bao, Z., Zhan, K., Zhang, W., et al.: 'LSFQ: A low precision full integer quantization for high-performance FPGA-based CNN acceleration', IEEE Symposium in Low-Power and High-Speed Chips, Tokyo, Japan, 2021, pp. 1-6
9. Liu, Z.G., Whatmough, P.N., Zhu, Y., et al.: 'S2TA: Exploiting structured sparsity for energy-efficient mobile CNN acceleration', IEEE International Symposium on High-Performance Computer Architecture, Seoul, Korea, Republic of, 2022, pp. 573-586
10. Xu, W., Sun, Y., Fan, S., et al.: 'Accelerating convolutional neural network by exploiting sparsity on GPUs', ACM Transactions on Architecture and Code Optimization, 2023, 20, (3), pp. 1-26
11. Zhao, Y.: 'Software and hardware co-design for efficient neural networks'. PhD thesis, University of Cambridge, 2022
12. Vaithianathan, M.: 'Hardware-software co-design for performance optimization in embedded systems', International Journal of Emerging Research in Engineering and Technology, 2025, 1, (1), pp. 30-37
13. Vaithianathan, M., Patil, M., Ng, S.F., et al.: 'Energy-Efficient FPGA design for wearable and implantable devices', ESP Int. J. Adv. Sci. Technol, 2024, 2, pp. 37-51
14. Ghimire, D., Kil, D., Kim, S.H.: 'A survey on efficient convolutional neural networks and hardware acceleration', Electronics, 2022, 11, (6), p. 945
15. Sedaghatgoo, A., Hajisadeghi, A.M., Momtazpour, M., et al.: 'ARMAN: A reconfigurable monolithic 3D accelerator architecture for convolutional neural networks', 2024, arXiv preprint arXiv:2402.04431
16. Manasi, S.D., Sapatnekar, S.S.: 'DeepOpt: Optimized scheduling of CNN workloads for ASIC-based systolic deep learning accelerators', Proceedings of the 26th Asia and South Pacific Design Automation Conference, Tokyo, Japan, 2021, pp. 235-241
17. Asifuzzaman, K., Abuelala, M., Hassan, M., et al.: 'Demystifying the characteristics of high bandwidth memory for real-time systems', IEEE/ACM International Conference On Computer Aided Design, Munich, Germany, 2021, pp. 1-9