[1]Dr. Aayushi Arya

[2]Huzaifa Umar

[3]Indrajeet Kumar

[4]Dr. Vuda Sreenivasa Rao

[5]Ashok Kumar Sahoo

[6]Dr. Shital Kakad

Multilayer Neural Network Hardware Designs for Faster Convolutional Processing

Abstract: - The demand for efficient hardware implementations of convolutional neural networks (CNNs) has surged with the proliferation of deep learning applications in various domains such as computer vision, natural language processing, and autonomous systems. Convolutional layers, being the fundamental building blocks of CNNs, are computationally intensive, requiring optimized hardware architectures for real-time inference tasks. This paper presents novel hardware designs targeting faster convolutional processing within multilayer neural networks. We propose a multi-faceted approach that leverages parallelism, pipelining, and hardware acceleration techniques to enhance the efficiency of convolution operations. Our design optimally exploits the inherent data-level and model-level parallelism present in CNNs to achieve high throughput while minimizing latency.

Keywords: Efficient hardware implementations, Convolutional Neural Networks (CNNs), Deep learning applications, Computer vision, Natural Language Processing (NLP), Autonomous systems, Convolutional layers

Introduction

In recent years, the widespread adoption of deep learning across various domains, including computer vision, natural language processing (NLP), and autonomous systems, has fueled the demand for efficient hardware implementations of convolutional neural networks (CNNs). CNNs have proven to be highly effective in handling complex data and extracting meaningful features, making them indispensable in many real-world applications. However, the computational demands of convolutional layers, which serve as the cornerstone of

[1] School of Technology, Woxsen University, Kamkole, Sangareddy District, Greater Hyderabad, Telangana, 502345.

Email: aayushi.arya@outlook.com

[2]Operational Research Center in Healthcare, Near East University, TRNC Mersin 10, Nicosia 99138, Turkey, huzaifa.umar@neu.edu.tr

[3]Associate Professor, Computer Science and Engineering, Graphic Era Hill University, Dehradun; Adjunct Professor, Graphic Era Deemed to be University, Dehradun, Uttarakhand-248002, India

Mail id- ikumar@gehu.ac.in

[4]Associate Professor, Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Green Fileds, Vaddeswaram, A.P., 522302.

vsreenivasarao@kluniversity.in

[5]Professor, Computer Science and Engineering, Graphic Era Hill University, Dehradun; Adjunct Professor, Graphic Era Deemed to be University, Dehradun, Uttarakhand-248002, India

Mail id- ashok.sahoo@gehu.ac.in

[6]Assistant Professor Vishwakarma Institute of Technology, Pune

shitalkakad2604@gmail.com

CNNs, pose significant challenges for real-time inference tasks, particularly in resource-constrained environments. This paper addresses the need for faster convolutional processing within multilayer neural networks by presenting novel hardware designs. Our approach is multi-faceted, drawing on principles of parallelism, pipelining, and hardware acceleration to enhance the efficiency of convolution operations. By exploiting both data-level and model-level parallelism inherent in CNNs, our designs aim to achieve high throughput while minimizing latency, thereby enabling accelerated inference performance.

The key contributions of our proposed hardware designs can be summarized as follows:

Parallelism Utilization: We leverage parallelism at both the data and model levels to distribute computation across multiple processing units efficiently. This includes parallelizing convolution operations across input feature maps and exploiting parallelism across multiple convolutional layers within the network.

Pipelined Processing: To further boost throughput, we adopt a pipelined architecture that enables continuous data flow through successive stages of computation. By overlapping computation and minimizing idle cycles, pipelining helps maximize hardware utilization and accelerate convolutional processing.

Hardware Acceleration Techniques: Our designs incorporate dedicated hardware accelerators tailored specifically for convolution operations. These accelerators feature specialized arithmetic units optimized for convolutional arithmetic and memory structures designed to facilitate efficient data access, thereby accelerating convolutional processing.

Efficient Memory Management: We employ a hierarchical memory organization strategy to minimize data movement overhead and optimize memory access patterns. This includes on-chip memory buffers for storing intermediate results and reducing reliance on off-chip memory accesses, which tend to be slower and more power-intensive.

Flexibility and Scalability: Our hardware designs are configurable and scalable, allowing them to adapt to various CNN architectures and convolutional layer configurations. This flexibility ensures compatibility with diverse network requirements and sizes, making our designs versatile and widely applicable.

Through extensive simulations and comparisons with state-of-the-art implementations, we demonstrate the superior performance of our proposed hardware designs across a range of CNN workloads. The results highlight significant speedup and efficiency gains, underscoring the potential of our designs to accelerate real-time inference tasks in resource-constrained environments.

Convolutional neural networks (CNNs) have made tremendous strides in accuracy and architecture, which is great news for many AI applications. Specifically, convolutional neural networks (CNNs) stand out as a promising approach to picture processing problems including recognition and classification [4, 13, 14]. CNN models that can process images in real-time while using little power are essential for embedded device image processing applications. Nevertheless, the data-intensive processing's better precision comes at a price: very high computational complexity and energy consumption. Adding more network layers to CNN improves its accuracy and provides better visual comprehension. Furthermore, overall CNN performance is hindered by the expensive transfer of data between on-chip or off-chip memory and processing components. Hence, performance optimisation is a primary goal while developing hardware accelerators such as GPUs, FPGAs, and ASICs. Although there are a number of hardware-based accelerators for convolutional neural network (CNN) acceleration in the literature, progress in areas like as computational complexity, resource utilisation, and energy efficiency is still in its early stages. While there are a few of designs that aim to drastically cut down on the amount of computation cycles needed for convolution operations, the majority of previous work has focused on creating memory and processor architectures with high levels of parallelism. The implementation of low-level convolution operations has not changed, despite the fast evolution of CNN architectures. Since most of the calculations in a CNN model occur during the convolution operation, fully-connected layers and convolutions together account for the majority of the network's computational power usage. The computing demands of implementing these low-level operations on an embedded device are high. In addition, there are a lot of data transfers between the memory and processor components required for data-intensive CNN processing, including input, output, and intermediate data transfers. High latency overhead is produced by this enormous data transfer.

This study suggests an accelerator design that uses the computing mechanism of convolution operations to reduce computational complexity and total processing delay. The accelerator architecture may run more efficiently with loading units that use the two important network parameters, weight and pixel. Furthermore, for new computing techniques, a redesigned kernel structure is also created. For a number of input parameters, the suggested accelerator expedites convolution processing and improves latency reduction. The primary benefits of this work are as follows:

• A new computing technique that reduces the amount of processing cycles needed to produce new pixels is suggested using a 3x1 kernel structure. To reduce data transfer, we take use of data that overlaps with neighbouring pixels in the input pixel matrix. The suggested kernel architecture allows for efficient and inexpensive data transfer between the on-chip memory buffer and processing components.

• The novel kernel loading technique and the parallelization of convolution processing are made possible by two loading units, which increase the overall performance of the CNN. Implementing convolutional neural network (CNN) models on embedded devices becomes much easier with this design.

• The total number of cycles needed to compute a convolution operation has been the subject of our theoretical investigation. In addition, we used Xilinx Vivado HLS 17.4 to simulate the convolution layers of the AlexNet [17] and VGG-16 [19] architectures on the FPGA xcvu9p-flgb2104-2-i device, and we were able to accelerate the convolution processes.

Reducing Data Movement Cost

In recent times, many FPGA-based accelerator designs have been suggested for use in deep convolutional neural network (CNN) applications. Using big data processing, we may examine the accelerator's performance. There is a lot of data transfer required to operate hundreds of kernels and channels in convolution operations simultaneously. Data transfer costs are higher than calculation costs in this highly parallel computing paradigm. The two main factors that determine the cost of data transportation are the storage location and the retrieval location. Due to the large amount of data sent between the on-chip memory and the PEs, there is a significant amount of delay. Data transfers from off-chip memory to PEs also use more power than data transfers from on-chip memory. Additionally, the total performance of the CNN is diminished due to the discrepancy between the compute throughput of the PEs and the memory bandwidth of the FPGA device. Designing dataflow systems that accomplish CNN processing while preserving processing performance is of utmost importance. There is a hierarchical structure to the data transportation cost that takes into account power consumption and data access rate. At each level, various dataflow designs take use of distinct kinds of data movement. Given that convolution operations comprise more than 90% of CNN operations and overall execution time, this study employs dataflow architecture to minimise superfluous data movement in convolution layers. The main limitations of applications are power consumption, latency, and resource utilisation.

Optimizing Convolution Operation

CNN brings solutions for a wide range of tasks in smart devices despite of their lowarea footprint and energy budget. Superior accuracy of CNN comes at the cost of increased computational complexity. Increase in the number of network layers and kernels enhance the performance of the CNN model that demands high design space and power consumption. In real-time embedded systems, power efficiency and resource utilization are the primary design concern while deploying the computational model of the CNN as an execution model on FPGA. As discussed above, the computational workload of a CNN inference basically comes from convolution layer that involves an intensive use of MAC operations. As a consequence, these high number of MAC computations create challenges for low-energy embedded devices. Increasing the computational power of PEs can alleviate this problems, however, use of resource constraint embedded devices make it infeasible. Hence, the feasible solution to this challenge is to take advantage of the approximate computing (AC) that accelerates the execution of CNN on FPGAs. This approach employs optimization strategies in different levels so that the state-of-the-art high accuracy CNNs can easily be deployed on resource constraint FPGA devices. AC obtains significant gains in computational throughput and power efficiency by maintaining an acceptable CNN accuracy. In this thesis, overall CNN's computational cost is reduced by using approximation in fixed point arithmetic and implementing approximate adders and multipliers without affecting their predictive performance.

Machine Learning

Machine learning (ML) is a statistical approach that enables computer systems to learn—that is, to become better at using data—over time, independently of how they were originally constructed. A mountain of data, both organised and unstructured, is necessary for machine learning algorithms to acquire knowledge and make predictions. The original coiner of the phrase "machine learning" was Arthur Samuel in 1959. The study and development of algorithms with the ability to learn from data and generate predictions is the focus of machine learning (ML), a branch of AI. The areas of artificial intelligence that gave rise to it include pattern recognition and computational learning theory. Among the many conceivable uses for ML is the development of high-performance algorithms, which would be very difficult, if not impossible, to do without it. Some of the many uses for ML include detection in computer vision, object categorization, email and network filtering, and generalised classification. ML has many commonalities with computational statistics, another field that employs computers to make predictions, and the two fields often intersect. The area receives techniques, theory, and application fields from mathematical optimisation, which has significant linkages to it. Furthermore, ML is used in conjunction with data mining, a subfield of unsupervised learning that places greater emphasis on exploratory data analysis. ML may also be trained under supervision to identify significant abnormalities after learning and establishing baseline behaviour for different tasks and entities. In data analytics, ML is a way to handle complicated algorithms and models that aim at making predictions. A large number of commercial applications that made use of ML principles were predictive analytics. By learning from patterns and correlations in the data over time, these analytical models enable researchers and analysts to discover previously unknown information and structures, as well as make choices with a high degree of confidence. The availability of more data and powerful computers has led to the rise in popularity of Artificial Neural Networks (ANNs), a subset of machine learning algorithms.

Convolution Neural Network

CNN is a popular feed-forward deep ANN for image analysis because it mimics human perception. The need for pre-processing the input data is minimised when CNNs use different multi-layer perceptron techniques. Convolutional neural networks (CNNs) are very popular for high-dimensional data, such as photos and movies. CNNs are composed of neurons with learnable biases and weights. Convolutional neural networks (CNNs) function similarly to regular neural networks. There are inputs to each neuron, and each neuron does a dot product and, if desired, a non-linearity follows. From the original picture pixels at one end of the network to the class scores at the other, the whole thing is still an expression of a single differentiable scoring function. Also, in the last (fully-connected) layer of a CNN, you'll find a loss function, such SVM or Softmax. A CNN layer differs significantly from other types of neural networks in that, rather of using standard matrix multiplication, each unit is a two- or high-dimensional filter that is convolved with the layer's input. When learning patterns from high-dimensional input material, like pictures or films, this is crucial. Voice recognition, picture categorization, and segmentation are just a few of the many current applications of convolutional neural networks (CNNs). The lowest mistake rate recorded on the dataset was 0.23%, which they attained on the digit recognition test. Their use extends to the identification and categorization of objects as well. One example is the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), which uses hundreds of item classes and millions of photos to serve as a benchmark for object identification and classification. Almost single top-performing team in the ILSVRC-2014 relied on CNN. The winning model, GoogLeNet, used a CNN with almost 30 layers and got results that are almost human-level accurate. Further applications of CNNs include face detection and video analysis. Through a hierarchical feature learning procedure, CNNs may autonomously identify critical object attributes. There is a need for specialised and customised hardware support methods because to the high processing demands of CNNs in all of the aforementioned applications.

Hardware Accelerator

Utilising MAC operations extensively in the convolution and FC layers constitutes a significant portion of a CNN's computing effort. Efficient execution of CNN layers by avoiding external memory accesses and utilising parallelism requires a specific hardware architecture for CNN workloads. Temporal and spatial architectures are two major types of hardware that speed up convolutional neural network computation. Processors and graphics processing units (GPUs) are examples of temporal hardware design, which allows for a great deal of programmability. The memory and centralised control mechanism are shared by all processor elements (PE) in a temporal architecture. When applied to regular parallelism patterns, this design's highly parallel structure provides efficient processing. Alternatively, low-power spatial designs enable fine-grain parallelism on asymmetrical patterns. Each physical element (PE) in the geographical framework has its own memory as well as a common global memory. Reducing access to external memory is achieved only by global memory communicating with external memory. Inside the on-chip networks (NoC), PEs are linked to one another. Figure 1 shows the fundamental spatial and temporal topologies of the convolutional neural network (CNN) computing paradigm. when GPU-based accelerators do a better job of processing data when training NNs, they aren't the best choice for convolutional neural network (CNN) inference in embedded or mobile systems that are concerned about power consumption. NN Accelerators built on the PIM Memory Cube (HMC) technique. These designs take digital data, transform it into analogue form, and then use crossbar memory to calculate matrix multiplication. In order to do calculations that decrease data transport costs, PIM employs processing logics inside memory. While memory technology grows exponentially, the current PIM-based accelerators rely on analogue, digital, and mixed-signal circuits, which use up most of the chip power. Also, CNN models with FP precision provide excellent classification accuracy, whereas PIM designs are limited to computations with fixed-point precision. It is common practice to use ASIC and FPGA devices when deploying CNN designs as spatial hardware. The main drawbacks of ASIC design are its high manufacturing cost and lack of reconfigurability. The goal of this effort is to develop CNN accelerator designs that are both space and power efficient. Platforms that are well-suited for cutting-edge CNN models include FPGAs because to their hardware flexibility and exceptional energy efficiency. Processing performance is increased and memory accesses to external devices are reduced because to FPGA's in-situ on-chip memories and high-density Digital Signal Processing (DSP) blocks. The functional requirements of CNN applications may be re-programmed even after manufacture using FPGA. In order to reduce execution latency, the spatial design of an FPGA may unroll a loop execution and make advantage of extra hardware resources. The FPGA's pipeline technique reorganises the distribution of computer resources to speed up processing without adding unnecessary space overhead. Therefore, when space and power are limited, FPGA is seen as a practical method for convolutional neural network (CNN) acceleration in design. The restricted processing power and memory resources of FPGA devices make it difficult to deploy deep CNN models for use in embedded applications and High-Performance Computing (HPC) data centres. For this reason, achieving maximum energy efficiency in the design of hardware accelerators is of paramount importance. To speed up the execution of CNN, there are a number of hardware designs in the literature that are based on field-programmable gate arrays (FPGAs).

Figure 1 shows the optimisation methods that deal with the problems of implementing CNNs on FPGAs. To speed up CNN on FPGAs, you may use one or all three of these optimisations. To minimise the amount of mathematical operations, computational optimisation makes use of approximation computing and algorithmic optimisation using feature maps and kernels. To make the convolution and FC layers' matrix multiplication faster, algorithmic optimisation applies computational transformations to the input data, such as the Generalised Envelope Multiplications (GEMMs), Winograd Transform, and Fast Fourier Transform (FFT). To map CNNs on FPGAs effectively, we need to reduce the amount of multiplication operations. To boost the processing speed and energy efficiency of the CNN's accelerator, the approximation computing technique trades low accuracy for increased efficiency. While running CNN layers, approximate computation decreases the accuracy of the operand and the amount of mathematical operations needed to do so without impacting the prediction performance. Quantization, pruning, and the use of approximate hardware (adders and multipliers) are just a few examples of how computational approximation may be useful in accelerator design. The processing parallelism in FPGA-based CNN accelerators is used via data-path optimisation. Architectural designs for convolutional neural network (CNN) mapping on field-programmable gate arrays (FPGAs) are optimised using loop optimisation methods such as loop unrolling, loop tiling, or loop exchange to maximise computational performance and external bandwidth efficiency. Furthermore, a plethora of data-flow designs, including

Synchronous Dataflow (SDF) and Dataflow Process Network (DPN), enhance the hardware mapping of deep convolutional neural network (CNN) models on FPGA devices with limited resources. Various forms of fixed-point and floating-point arithmetic are used in the construction of accelerator designs for FPGA implementations by means of the High Level Synthesis (HLS) tool and the Register-Transfer Level (RTL) compiler. FPGA devices may be effectively used to map convolutional neural networks (CNNs), thanks to architectural-level fine-grain optimisation in addition to algorithmic optimisation.

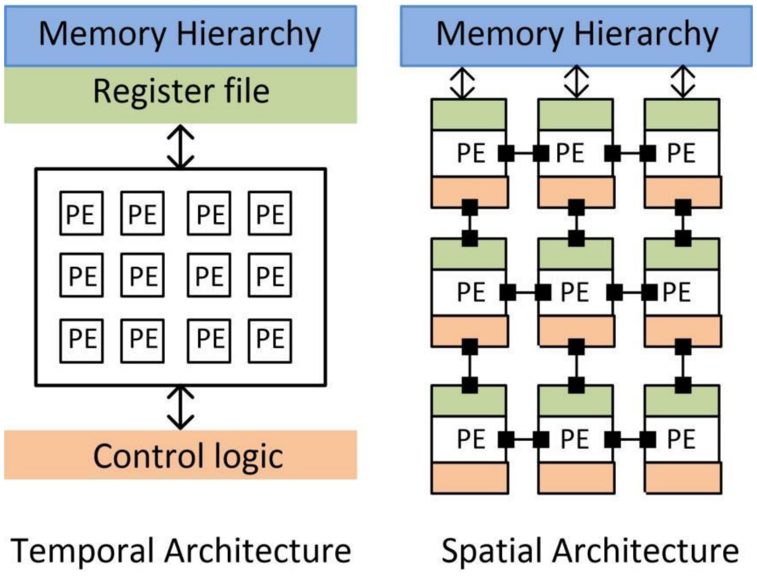

Figure 1: Computing architectures for CNN.

In this work, data path optimization and approximate computing are jointly implemented in the accelerator architecture for additive hardware performance gains. Also, Vivado HLS and Synopsys tools are used to port CNN inference onto FPGAs. HLS tool generates a synthesis report mentioning the performance metrics of the synthesized hardware design. The report includes mainly area, latency, loop latency, and iteration interval. Area refers to the number of hardware resources such as Look-Up Tables (LUT), Flip Flops (FF), Block RAMs (BRAMs), and DSP48s required to implement the hardware architecture. The pipeline, Array Partition, Unroll, and Dataflow directives are used in the proposed accelerator architectures to improve throughput and reduce area and latency

Shift and Accumulate based accelerator architecture for CNN

Multiplying and adding the overlapping values of two input pictures is the convolution operation in image processing. Convolution is often calculated by adding up the dot products of all the pixels in the input picture that have kernels applied to them across all of the dimensions. A kernel is a tiny matrix of varying sizes and patterns of integers that moves throughout the whole input picture. The kernel is selected according to the image's intended functionalities, such as blur, smooth, etc. An array of MAC processors calculates MAC operations. The paper explains how a 3 × 3 kernel K and an image matrix I perform a convolution operation. For the output picture O, the first pixel O11 is determined as

$$O11 = p11 \times 1 + p12 \times 0 + p13 \times 1 + p21 \times 0 + p22 \times 1 + p23 \times 0 + p31 \times 1 + p32 \times 0 + p33 \times 1.$$

Hardware Accelerator Architecture for CNN based on Approximate Computing Units

Among all the deep neural networks (DNNs), Convolutional Neural Networks (CNNs) are gaining attention and popularity in many computer vision (CV) applications [9]. The computationally intensive CNN has achieved state-of-the-art performance in image classification, face detection, and speech recognition. High computation complexity in CNN inference needs specific hardware to accelerate. Recent advancements in deep CNN architectures and dedicated hardware accelerators provide satisfying processing throughput and energy efficiency. Development of the real-time devices relies on high performance CNNs implemented on embedded devices. The diverse shape and size of CNN architectures, along with their dedicated accelerators, create challenges to the deployment on embedded devices that have limited computational resources. In addition, high energy efficiency and low area overhead of CNN's hardware are the primary performance metrics for real-time applications on embedded systems. Despite the challenges and constraints, researchers have developed many efficient custom hardware for deep neural networks Rapid progress in the hardware architecture of the CNN based inference accelerators achieved higher computing efficiency than CPUs and GPUs. However, optimization in the computation of CNNs and their related hardware architectures may improve the overall performance for embedded applications. To store and run a deeper neural network (NN), FPGA devices require a considerable amount of on-chip memory and computational units. Computing approximation helps to fit such neural models on-chip. In general, neural network accelerators focus on optimizing either the network's algorithm or the hardware architecture. The first category of accelerators treats the computational primitives of the network model. In contrast, the second category of accelerators optimized the data transfer and memory design. Both sets of accelerators have improved their performance while implementing DNNs.

Model	**Number of Layers**		**Size (M)**	**Parameter (M)**
	Convolution	**FC**		
AlexNet	7	3	240	64
VGG-16	15	3	542	150
GoogleNet	23	1	42	12.8
ResNet-50	52	1	102	20.2
ResNet-152	157	1	237	70.2

Table 1: The comparison of different CNN architectures

Traditional convolutional neural network (CNN) algorithms generate output feature maps by combining input feature maps with convolutional kernels. The multiply-and-accumulate (MAC) processes, which make up 99% of a CNN model, are housed in a sequence of convolutional and fully-connected (FC) layers. The size of the feature maps produced by each convolution layer is reduced by pooling layers as well. Computation in convolutional neural networks (CNNs) revolves on the convolution operation, which may include hundreds of millions of MAC operations per layer. As an added complication, convolution processing sometimes involves substantial data transfer between on-chip memory and MAC units. The total processing speed, energy efficiency, and resource utilisation of CNNs are significantly affected by the implementation of the convolution layers on hardware. The processing of higher-resolution, bigger input photos increases the data and computing demands.

Conclusion

In this paper, we have presented novel hardware designs aimed at achieving faster convolutional processing within multilayer neural networks. Leveraging principles of parallelism, pipelining, and hardware acceleration, our designs offer significant improvements in efficiency, throughput, and latency reduction, addressing the growing demand for accelerated inference tasks in deep learning applications. Through a multi-faceted approach, we optimized our hardware architectures to exploit both data-level and model-level parallelism inherent in convolutional neural networks (CNNs). By parallelizing convolution operations across input feature maps and adopting a pipelined architecture for continuous data flow, we effectively maximized hardware utilization and minimized idle cycles. Additionally, the integration of dedicated hardware accelerators tailored for convolution operations further accelerated processing, with specialized arithmetic units and optimized memory structures contributing to enhanced performance.

References

1. K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in IEEE Proceedings of the Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
2. T. Luong, H. Pham, and C. D. Manning, "Effective Approaches to Attention-based Neural Machine Translation," in Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2015, pp. 1412–1421.

3. D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen et al., “Deep speech 2: End-to-end speech recognition in english and mandarin,” in International Conference on Machine Learning, 2016, pp. 173–182.
4. D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” in International Conference on Learning Representations (ICLR), 2014, pp. 1–15.
5. Y. Chen, Y. Xie, L. Song, F. Chen, and T. Tang, “A Survey of Accelerator Architectures for Deep Neural Networks,” Engineering, vol. 6, no. 3, pp. 264–274, 2020
6. K. He, X. Zhang, S. Ren, and J. Sun, “Identity mappings in deep residual networks.” Lecture Notes in Computer Science., B. Leibe, J. Matas, N. Sebe, M. Welling, Eds.(Springer International Publishing, 2016), pp. 630–645, 2020
7. T. Yuan, W. Liu, J. Han, and F. Lombardi, “High performance CNN accelerators based on hardware and algorithm co-optimization,” Transactions on Circuits and Systems I: Regular Papers, vol. 68, no. 1, pp. 250–263, 2020
8. D. Wang, K. Xu, J. Guo, and S. Ghiasi, “DSP-efficient hardware acceleration of convolutional neural network inference on FPGAs,” Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 39, no. 12, pp. 4867–4880, 2020
9. C. Zhu, K. Huang, S. Yang, Z. Zhu, H. Zhang, and H. Shen, “An efficient hardware accelerator for structured sparse convolutional neural networks on FPGAs,” Transactions on Very Large Scale Integration (VLSI) Systems, vol. 28, no. 9, pp. 1953–1965, 2020.
10. M. S. Kim, A. A. Del Barrio, H. Kim, and N. Bagherzadeh, “Effects of Approximate Multiplication on Convolutional Neural Networks,” arXiv preprint arXiv:2007.10500, 2020
11. H. Pourmeidani, S. Sheikhfaal, R. Zand, and R. F. DeMara, “Probabilistic interpolation recoder for energy-error-product efficient DBNs with p-bit devices,” Transaction on Emerging Topics in Computing, 2020
12. A. Ahmad and M. A. Pasha, “FFConv: an fpga-based accelerator for fast convolution layers in convolutional neural networks,” Transaction on Embedded Computing Systems (TECS), vol. 19, no. 2, pp. 1–24, 2020
13. B. Asgari, R. Hadidi, T. Krishna, H. Kim, and S. Yalamanchili, “ALRESCHA:A Lightweight Reconfigurable Sparse-Computation Accelerator,” in International Symposium on High Performance Computer Architecture (HPCA). IEEE, 2020, pp. 249–260.
14. E. Qin, A. Samajdar, H. Kwon, V. Nadella, S. Srinivasan, D. Das, B. Kaul, and T. Krishna, “Sigma: A sparse and irregular GEMM accelerator with flexible interconnects for DNN training,” in International Symposium on High Performance Computer Architecture (HPCA)). IEEE, 2020, pp. 58–70
15. G. Li, P. Wang, Z. Liu, C. Leng, and J. Cheng, “Hardware acceleration of CNN with one-hot quantization of weights and activations,” in Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2020, pp. 971–974
16. Z.-G. Liu, P. N. Whatmough, and M. Mattina, “Systolic Tensor Array: An Efficient Structured-Sparse GEMM Accelerator for Mobile CNN Inference,” Computer Architecture Letters, vol. 19, no. 1, pp. 34–37, 2020.
17. W. Zhang, M. Zhai, Z. Huang, C. Liu, W. Li, and Y. Cao, “Towards end-to-end speech recognition with deep multipath convolutional neural networks,” in International Conference on Intelligent Robotics and Applications. Springer, 2019, pp. 332–341
18. C. Szegedy, W. Liu, J. Yangqing, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going Deeper with Convolutions,” in IEEE Proceedings of Conference on Computer Vision and Pattern Recognition, 2015, pp. 1–9.