International Journal of Computational and Experimental Science and ENgineering (IJCESEN)

Vol. 10-No.4 (2024) pp. 1007-1014
http://www.ijcesen.com

ISSN: 2149-9144

Research Article

Reconfigurable Acceleration of Neural Networks: A Comprehensive Study of FPGA-based Systems

Chandanapriya Machireddy[1]*, Santhosh Chella[2]

[1]Department of Electronics and Communication Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram 522302, Guntur, Andhra Pradesh, India;
* **Corresponding Author Email:** Chandana.priya52@gmail.com- **ORCID:** 0009-0007-4415-7236

[2]Department of Electronics and Communication Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram 522302, Guntur, Andhra Pradesh, India;
Email: csanthosh@kluniversity.in - **ORCID:** 0000-0002-5301-2000

Article Info:

DOI: 10.22399/ijcesen.559
Received : 24 October 2024
Accepted : 28 October 2024

Keywords

FPGA,
Neural Network,
Deep Learning,
Hardware Acceleration,
Reconfigurable Computing.

Abstract:

This paper explores the potential of Field-Programmable Gate Arrays (FPGAs) for accelerating both neural network inference and training. We present a comprehensive analysis of FPGA-based systems, encompassing architecture design, hardware implementation strategies, and performance evaluation. Our study highlights the advantages of FPGAs over traditional CPUs and GPUs for neural network workloads, including their inherent parallelism, reconfigurability, and ability to tailor hardware to specific network needs. We delve into various hardware implementation strategies, from direct mapping to dataflow architectures and specialized hardware blocks, examining their impact on performance. Furthermore, we benchmark FPGA-based systems against traditional platforms, evaluating inference speed, energy efficiency, and memory bandwidth. Finally, we explore emerging trends in FPGA-based neural network acceleration, such as specialized architectures, efficient memory management techniques, and hybrid CPU-FPGA systems. Our analysis underscores the significant potential of FPGAs for accelerating deep learning applications, particularly those requiring high performance, low latency, and energy efficiency.

1. Introduction

Main Deep learning has experienced rapid advancements, revolutionizing various fields. This progress has resulted in significant breakthroughs in areas such as computer vision, natural language processing, and speech recognition [1]. These advancements demonstrate the power of deep learning to solve complex problems in diverse domains [2]. The training and deployment of large-scale neural networks require immense computational resources. This presents a significant challenge, as traditional hardware struggles to keep pace with the ever-increasing demands of these complex models [3].

The limitations of traditional CPUs and GPUs in handling the computational demands of large-scale neural networks have become increasingly evident. This has sparked a growing need for more efficient and powerful hardware solutions. To overcome the computational bottlenecks that hinder the development and deployment of these complex models, researchers and developers are actively exploring alternative hardware architectures capable of handling the massive computational workload associated with modern deep learning applications [4].

While conventional CPUs and GPUs offer significant processing power, they often fall short when it comes to meeting the ever-increasing computational demands of large-scale neural networks. The sheer volume of calculations required for training and deploying these complex models surpasses the capabilities of traditional hardware, creating a bottleneck that hinders progress in the field of deep learning. This challenge highlights the urgent need for alternative hardware solutions that can effectively handle the massive computational workload associated with modern neural network applications [5]. The limitations of traditional CPUs and GPUs in handling the computational demands of large-scale

neural networks have spurred a search for more efficient hardware solutions [6].

Field-Programmable Gate Arrays (FPGAs), with their inherent parallelism and reconfigurability, present a promising alternative for accelerating neural network workloads. This unique combination of features makes FPGAs particularly well-suited for tackling the complex computations involved in deep learning, offering the potential to significantly improve performance and efficiency [7]. FPGAs offer a distinct advantage over general-purpose processors in their ability to be tailored to the specific needs of a given neural network. This reconfigurability allows developers to optimize the hardware architecture to match the specific computational demands of the network, maximizing performance. By customizing the FPGA design to align with the unique characteristics of the neural network, developers can achieve significant performance gains over generic processors that are not optimized for these specific tasks. This flexibility and customization capability of FPGAs holds great potential for unlocking unprecedented performance levels in deep learning applications. [8].

2. FPGA Architecture and Design

This To fully understand the potential of FPGAs for accelerating neural networks, we must delve into their architecture. Examining the key components of an FPGA and how they interact reveals the unique capabilities that make them well-suited for handling the complex computations involved in deep learning. This section explores the fundamental architecture of FPGAs and analyzes their suitability for optimizing neural network operations [9].

Field-Programmable Gate Arrays (FPGAs) are highly customizable hardware platforms that offer a unique approach to accelerating neural network workloads. Understanding their key components is essential for appreciating their potential for deep learning applications.At the heart of an FPGA lies the configurable logic block (CLB), which acts as the fundamental building block for implementing custom circuits. Each CLB comprises logic gates and memory elements, allowing for the creation of a wide range of digital circuits tailored to specific tasks. These CLBs are highly versatile, enabling the implementation of complex logic functions and data processing operations. Beyond the CLBs, FPGAs also feature dedicated memory blocks. These blocks provide high-bandwidth storage for data and parameters used in neural network calculations. By providing fast access to critical data, these memory blocks significantly enhance the efficiency of neural network operations, particularly those involving large datasets. Finally, connecting these CLBs and memory blocks is the interconnect fabric, a network of wires and switches that allows for flexible routing and communication between different components of the FPGA. This fabric enables the dynamic configuration of data paths and control signals, allowing for efficient communication and data flow within the hardware. This intricate network of CLBs, memory blocks, and interconnect fabric provides the foundation for the customization and adaptability that make FPGAs so powerful for accelerating neural networks.

The unique architecture of FPGAs, with its configurable logic blocks (CLBs), dedicated memory blocks, and flexible interconnect fabric, empowers developers to tailor the hardware to the specific needs of neural networks, achieving significant performance gains. This adaptability allows for the efficient implementation of fundamental neural network operations, such as convolution, matrix multiplication, and activation functions, which form the backbone of deep learning models.

Convolutional operations, essential for image recognition and processing, involve sliding a filter across an input image, performing element-wise multiplication and summation. By configuring CLBs to perform these multiplications and summations in parallel, FPGAs can dramatically accelerate convolutional operations. The dedicated memory blocks can efficiently store the input image and filter data, while the interconnect fabric enables the efficient movement of data between CLBs and memory blocks, optimizing data flow for efficient computation.

Matrix multiplication, another crucial operation in neural networks, involves multiplying two matrices to produce a resulting matrix. FPGAs can be configured to perform matrix multiplication in a highly parallel manner, taking advantage of the CLBs' ability to perform multiple multiplications and additions simultaneously. The interconnect fabric allows for efficient data transfer between CLBs, enabling the parallel computation of matrix elements, significantly accelerating the overall process [10].

This section delves into the architecture of FPGAs and their suitability for neural network acceleration. We discuss the key components of an FPGA, including configurable logic blocks, memory blocks, and interconnect fabric. We explain how these components can be configured to implement specific neural network operations, such as convolution, matrix multiplication, and activation functions.

Activation functions, applied to the output of neurons, introduce non-linearity into the network, enabling the learning of complex patterns. FPGAs can efficiently implement various activation functions, such as ReLU (Rectified Linear Unit), sigmoid, and tanh, by configuring CLBs to perform the necessary calculations. The memory blocks can

store the activation function parameters, and the interconnect fabric enables efficient data movement between CLBs and memory blocks, optimizing the activation function application [11].

By leveraging the flexibility and parallelism of FPGAs, developers can implement these critical operations, convolution, matrix multiplication, and activation functions, in a highly optimized manner. This customized hardware implementation allows FPGAs to outperform traditional processors in terms of both speed and efficiency, making them ideal for accelerating neural network workloads. The ability to tailor the FPGA architecture to the specific needs of the neural network, coupled with the efficient implementation of these fundamental operations, positions FPGAs as a powerful tool for driving advancements in deep learning applications.

3. Hardware implementation strategies

Deploying neural networks on FPGAs requires careful consideration of hardware implementation strategies to maximize performance and efficiency. This involves choosing the most suitable approach to map the network's structure and operations onto the FPGA's unique architecture. Various strategies have been developed to leverage the FPGA's capabilities, each with its own strengths and weaknesses [12].

One approach involves direct mapping, Direct mapping is one approach to deploying neural networks on FPGAs, where the individual layers of the network are directly mapped onto the FPGA fabric. This strategy leverages the inherent parallelism of the hardware, enabling simultaneous processing of multiple operations within the network. The FPGA's configurable logic blocks (CLBs) are utilized to implement the computational units of each layer, while the dedicated memory blocks store the weights and activations associated with the network. The interconnect fabric facilitates efficient data transfer between these components, allowing for smooth data flow between layers. This direct mapping approach allows for high-performance execution of the neural network, as the operations within each layer can be processed in parallel, taking advantage of the FPGA's massively parallel architecture. This strategy can be particularly effective for smaller neural networks with simpler architectures, where the mapping process is relatively straightforward. However, direct mapping can become complex and resource-intensive for larger and more intricate networks, requiring careful optimization and resource allocation [13].

Dataflow architectures offer an alternative approach to deploying neural networks on FPGAs, where the network is implemented as a series of interconnected processing units, each specialized for a specific operation. This modular design breaks down the complex network into smaller, more manageable units, simplifying implementation and increasing flexibility. Each processing unit is responsible for performing a particular operation, such as convolution, matrix multiplication, or activation function application. These units are then interconnected via data paths, allowing for the efficient flow of data through the network. This modular structure enables easy adaptation and scalability, as new units can be added or modified without affecting the overall network architecture. Dataflow architectures offer a more flexible approach compared to direct mapping, particularly for larger and more complex networks. This strategy allows for the optimization of individual units, maximizing performance for specific operations. Additionally, the modular design makes it easier to manage and debug the network, facilitating efficient development and deployment.

A third approach to implementing neural networks on FPGAs involves designing and incorporating specialized hardware blocks tailored for specific operations. This strategy offers the potential for maximum performance optimization, but it demands careful planning and potentially increased development time. Instead of relying on general-purpose CLBs, this approach utilizes custom hardware blocks designed specifically for operations like convolution kernels or activation functions. These blocks are highly optimized for their specific task, leveraging the FPGA's resources to achieve maximum efficiency. For example, a convolution kernel block could be designed to efficiently perform the multiplications and summations involved in convolutional operations, while an activation function block could be optimized for a specific activation function, like ReLU or sigmoid. This approach allows for significant performance gains for the specific operations handled by the specialized blocks. However, designing and implementing these custom blocks requires specialized knowledge and engineering expertise, potentially increasing development time. This strategy is often employed when performance is paramount, justifying the additional effort and complexity in design [14].

By exploring these diverse hardware implementation strategies, researchers and developers can choose the best approach for a particular neural network, maximizing performance while balancing design complexity and development time.

4. Performance Evaluation and Comparison

Comparing hardware platforms like GPUs, ASICs, and FPGAs can be tricky because performance often hinges on the specific application. For instance, a GPU optimized for server tasks is difficult to directly

compare to an ASIC or FPGA designed for embedded systems, as their power requirements and data processing volumes differ significantly. Despite these challenges, researchers often rely on standard metrics like area, power consumption, and operations per second to define hardware performance. These metrics offer a common ground for comparing diverse platforms, even though application-specific optimizations can still significantly impact real-world performance [15-18].Inference speed is a key performance metric that measures how many inferences a system can perform per second. It indicates the system's efficiency in processing data and generating predictions. Energy efficiency, measured as power consumption per inference, is a critical factor, especially for battery-powered devices. It determines how long a system can operate on a single charge while performing inferences. Memory bandwidth signifies the speed at which a system can transfer data between its memory and processing units. This is crucial for efficient handling of large datasets, as it directly impacts the time required to load and process data.Beyond core performance metrics, an accelerator's flexibility and scalability are also crucial. This includes its adaptability to new network models and its ability to handle variable bitwidths for data processing. Since accelerators are often tailored for specific applications, direct comparisons can be complex. Evaluations are best conducted using relevant datasets and common models to understand their performance in practical scenarios.

Table 1 provides a comparison of various hardware platforms and accelerator models, highlighting key metrics like area, power consumption, and inference speed. As anticipated, general-purpose architectures tend to have larger area and higher power consumption compared to specialized architectures due to their lack of application-specific optimizations. This table summarizes the hardware characteristics discussed earlier, providing insights into the strengths and weaknesses of each architecture. A comprehensive comparison of various hardware platforms and accelerator models is crucial for understanding the trade-offs involved in choosing the best solution for a specific deep learning application. A graph illustrating this comparison would be invaluable, with key metrics like area, power consumption, and inference speed plotted against each other. The following Figure 1, Figure 2 and Figure 3 represents the inference speed v/s energy efficiency, Energy efficiency v/s Memory bandwidth and Inference speed v/s Memory bandwidth.

5. Emerging trends in fpga-based neural network acceleration

This section highlights emerging trends in the field of FPGA-based neural network acceleration.

Table 1. *Comparison between accelerations implemented on different hardware platforms.*

Name	Platform	Reference	Technology(nm)	Inference speed (mm^2)	Energy efficiency (mW)	Memory bandwidth (GB/s)
Cambricon-X	ASIC	[19]	65	6.38	954	256
SCNN	ASIC	[20]	16	7.9	-	200
EIE	ASIC	[21]	45	40	600	200
NullHop	ASIC	[22]	28	8.1	155	256
NullHop	FPGA Xilinx Zynq 7100	[22]	28	-	2300	256
SqueezeFlow	ASIC	[23]	65	4.80	536	200
UNPU	ASIC	[24]	65	16	297	-
FlexFlow	ASIC	[25]	65	3.89	1000	200
DNA	ASIC	[26]	65	16	479	-
SIGMA	ASIC	[27]	28	65.10	22,300	-
DNPU	ASIC	-	65	16	279	-
Nvidia V100	GPU	-	12	815	250,000	900
Nvidia A100	GPU	-	7	826	400,000	1500
Intel Xeon Platinum 9282	CPU	-	14	-	400,000	750
AMD Ryzen Threadripper 3970x	CPU	-	7	-	280,000	900

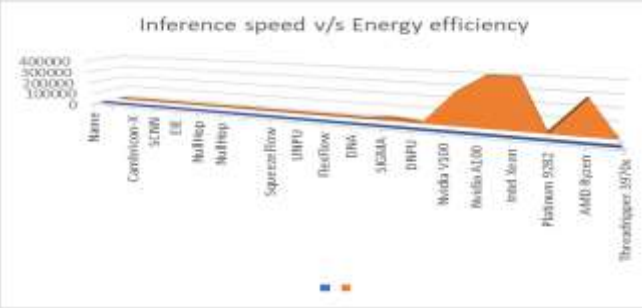

Figure 1. *Inference speed v/s Energy efficiency*

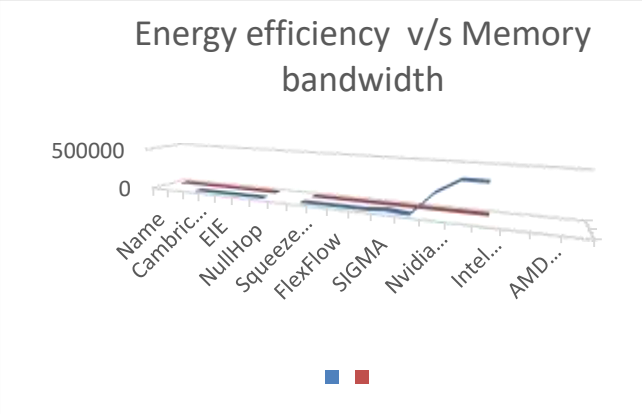

Figure 2. *Energy efficiency v/s Memory bandwidth*

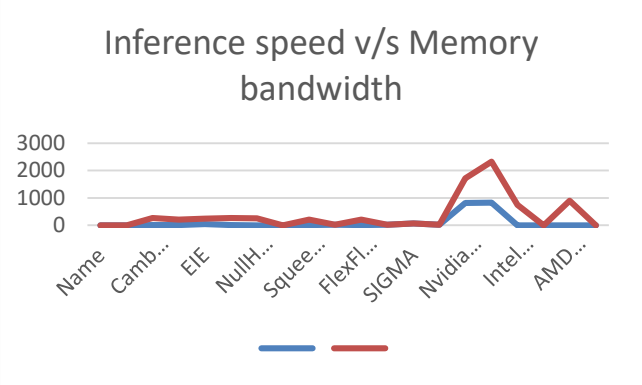

Figure 3. *Inference speed v/s Memory bandwidth*

The development of specialized hardware architectures for FPGA-based neural network acceleration is a key driver in pushing the boundaries of deep learning performance. Unlike general-purpose computing platforms, these custom architectures are meticulously designed to optimize for the specific computational demands of neural networks, resulting in substantial speedups. Traditional CPUs and GPUs, while powerful, are often hampered by their general-purpose design. This means they are optimized for a wide range of tasks, not specifically tailored to the highly parallel nature of neural network operations like matrix multiplications and convolutions. FPGAs, on the other hand, offer a unique advantage: their reconfigurable nature allows them to be custom-designed for specific applications, essentially creating dedicated hardware for neural network acceleration. Efficient memory management is a crucial aspect of FPGA-based neural network acceleration, directly impacting performance and energy consumption. Neural networks often require massive amounts of data for training and inference, and how this data is accessed and moved through the system significantly affects performance. Optimizing memory access patterns and minimizing data movement overhead is a key challenge in maximizing the potential of FPGA acceleration. Traditional memory architectures, often designed for general-purpose computing, may not be ideal for the unique demands of neural networks. These demands include High Bandwidth, data locality, and data re-use. Neural networks require constant access to large amounts of data, demanding high-bandwidth memory interfaces. Optimal performance requires data to be readily available in close proximity to the processing units, minimizing data movement across the system. Neural networks often reuse data multiple times, necessitating efficient caching mechanisms to avoid redundant data fetching. Techniques for optimizing memory management in FPGA-based neural network acceleration include Data Prefetching, On-Chip Data Buffering, Specialized Memory Hierarchies, Data Compression and Optimized Data Layouts. Anticipating future data requirements and prefetching data into local memory buffers allows for faster access and reduces latency. Implementing on-chip buffers within the FPGA architecture allows for storing frequently accessed data close to processing units, reducing off-chip memory access and latency. Designing custom memory hierarchies tailored to the specific needs of neural networks, such as multiple levels of caches and specialized memory blocks for different data types. Compressing the data before transferring it across the system reduces memory bandwidth requirements and speeds up data movement. Arranging data in memory in a way that minimizes memory access conflicts and maximizes data locality, further reducing access times. The quest for optimal performance in neural network acceleration often necessitates a balance between specialized hardware and general-purpose computing capabilities. Hybrid CPU-FPGA systems offer a compelling solution by combining the strengths of both worlds, creating a synergistic platform for efficient and flexible deep learning deployment. While FPGAs excel at accelerating computationally intensive tasks like neural network inference, CPUs offer versatility in handling control logic, data preprocessing, and general-purpose operations. This inherent complementary nature makes hybrid systems particularly well-suited for diverse deep

learning applications. Hybrid systems effectively leverage these strengths are Computationally intensive tasks, like neural network inference, are offloaded to the FPGA for acceleration, while the CPU handles tasks that require flexibility and general-purpose capabilities. The modular nature of hybrid systems allows for scalability by adding more FPGAs or CPUs as needed, depending on the specific application requirements. By combining the specialized capabilities of FPGAs with the cost-effectiveness of CPUs, hybrid systems offer a balance between performance and affordability. Examples of hybrid CPU-FPGA systems include Embedded Systems, Edge Computing and High-Performance Computing. For real-time applications like autonomous vehicles or robotics, hybrid systems combine the high performance of FPGAs for image processing with the control and versatility of CPUs for decision-making and system management. Hybrid systems can be deployed at the edge of the network to enable on-device inference, reducing latency and data transmission costs for applications like facial recognition or speech recognition. Hybrid systems are becoming increasingly popular in high-performance computing clusters for tasks like scientific simulations and large-scale data analysis. The development of hybrid CPU-FPGA systems is a testament to the ongoing pursuit of efficient and flexible deep learning solutions. These systems offer a promising path forward, balancing the advantages of both specialized hardware and general-purpose computing, enabling a wider range of deep learning applications across diverse industries.

6. Conclusion

This paper provides a comprehensive overview of the potential of FPGAs for accelerating neural networks. We have demonstrated that FPGAs offer a compelling alternative to traditional hardware solutions, particularly for applications requiring high performance, low latency, and energy efficiency. We believe that the ongoing research and development in this area will lead to even more powerful and efficient FPGA-based systems for neural network processing in the future as previous works have done [28-37].

Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper
- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.
- **Author contributions:** The authors declare that they have equal right on this paper.
- **Funding information:** The authors declare that there is no funding to be acknowledged.
- **Data availability statement:** The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.

References

[1] Junyi Chai, Hao Zeng, Anming Li, Eric W.T. Ngai, (2021). Deep learning in computer vision: A critical review of emerging techniques and application scenarios, *Machine Learning with Application*s, 6;100134 https://doi.org/10.1016/j.mlwa.2021.100134.

[2] Sarker, I.H. (2021). Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions. *SN COMPUT. SCI.* 2;420. https://doi.org/10.1007/s42979-021-00815-1

[3] Yuan, X., Wang, Y., Xu, Z. et al. (2023). Training large-scale optoelectronic neural networks with dual-neuron optical-artificial learning. *Nat Commu*n 14; 7110. https://doi.org/10.1038/s41467-023-42984-y

[4] Tufail S, Riggs H, Tariq M, Sarwat (2023). AI. Advancements and Challenges in Machine Learning: A Comprehensive Review of Models, Libraries, Applications, and Algorithms. *Electronics.* 12(8):1789. https://doi.org/10.3390/electronics12081789

[5] Alzubaidi, L., Zhang, J., Humaidi, A.J. et al. (2021) Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *J Big Data* 8;53. https://doi.org/10.1186/s40537-021-00444-8

[6] Martin Wisniewski L, Bec J-M, Boguszewski G, Gamatié A. (2022). Hardware Solutions for Low-Power Smart Edge Computing. *Journal of Low Power Electronics and Applications*. 12(4):61. https://doi.org/10.3390/jlpea12040061

[7] Wu R, Guo X, Du J, Li J. (2021). Accelerating Neural Network Inference on FPGA-Based Platforms—A Survey. *Electronics.* 10(9):1025. https://doi.org/10.3390/electronics10091025

[8] Martín-Martín, A., Padial-Allué, R., Castillo, E., Parrilla, L., Parellada-Serrano, I., Morán, A., & García, A. (2024). Hardware Implementations of a Deep Learning Approach to Optimal Configuration of Reconfigurable Intelligence Surfaces. *Sensors (Basel, Switzerland),* 24(3). https://doi.org/10.3390/s24030899

[9] A. Shawahna, S. M. Sait and A. El-Maleh, (2019). FPGA-Based Accelerators of Deep Learning Networks for Learning and Classification: A Review,

in *IEEE Access*, 7;7823-7859, doi: 10.1109/ACCESS.2018.2890150.

[10] Boutros, A., Arora, A., & Betz, V. (2024). Field-Programmable Gate Array Architecture for Deep Learning: Survey & Future Directions. *ArXiv*. /abs/2404.10076

[11] Li, Zhengjie & Zhang, Yufan & Wang, Jian & Lai, Jinmei. (2020). A survey of FPGA design for AI era. *Journal of Semiconductors.* 41; 021402. 10.1088/1674-4926/41/2/021402.

[12] Zhiqiang Que, Hongxiang Fan, Marcus Loo, He Li, Michaela Blott, Maurizio Pierini, Alexander Tapper, and Wayne Luk. (2024). LL-GNN: Low Latency Graph Neural Networks on FPGAs for High Energy Physics. ACM Trans. *Embed. Comput. Syst.* 23(2);17-28 pages. https://doi.org/10.1145/3640464

[13] Neu, M., Becker, J., Dorwarth, P. et al. (2024). Real-Time Graph Building on FPGAs for Machine Learning Trigger Applications in Particle Physics. *Comput Softw Big Sci* 8;8. https://doi.org/10.1007/s41781-024-00117-0

[14] Morteza Babaee Altman, Wenbin Wan, Amineh Sadat Hosseini, Saber Arabi Nowdeh, Masoumeh Alizadeh, Machine learning algorithms for FPGA Implementation in biomedical engineering applications: A review, *Heliyon*, 10(4);e26652, https://doi.org/10.1016/j.heliyon.2024.e26652

[15] Joo-Young Kim, (2021). Chapter Five - FPGA based neural network accelerators, Editor(s): Shiho Kim, Ganesh Chandra Deka, Advances in Computers, Elsevier,122;35-165, ISBN 9780128231234, https://doi.org/10.1016/bs.adcom.2020.11.002

[16] Mittal, S. (2020). A survey of FPGA-based accelerators for convolutional neural networks. *Neural Comput & Applic* 32; 1109–1139. https://doi.org/10.1007/s00521-018-3761-1

[17] Wang C, Luo Z. (2022). A Review of the Optimal Design of Neural Networks Based on FPGA. *Applied Sciences.* 12(21):10771. https://doi.org/10.3390/app122110771

[18] Capra M, Bussolino B, Marchisio A, Shafique M, Masera G, Martina M. (2020). An Updated Survey of Efficient Hardware Architectures for Accelerating Deep Convolutional Neural Networks. *Future Internet.* 12(7):113. https://doi.org/10.3390/fi12070113

[19] Zhang, S.; Du, Z.; Zhang, L.; Lan, H.; Liu, S.; Li, L.; Guo, Q.; Chen, T.; Chen, Y. Cambricon-X (2016) An accelerator for sparse neural networks. In Proceedings of the 2016 *49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), Taipei, Taiwan,* 15–19 October 2016; pp. 1–12

[20] Parashar, A.; Rhu, M.; Mukkara, A.; Puglielli, A.; Venkatesan, R.; Khailany, B.; Emer, J.; Keckler, S.W.; Dally, W.J. (2017). SCNN: An accelerator for compressed-sparse convolutional neural networks. In Proceedings of the 2017 ACM/IEEE *44th Annual International Symposium on Computer Architecture (ISCA), Toronto, ON*, Canada, 24–28 June 2017; pp. 27–40.

[21] Han, S.; Liu, X.; Mao, H.; Pu, J.; Pedram, A.; Horowitz, M.A.; Dally, W.J. EIE: Effcient Inference Engine on Compressed Deep Neural Network. *In Proceedings of the 43rd ACM/IEEE Annual International Symposium on Computer Architecture, ISCA 2016, Seoul, Korea, 18–22 June 2016;* IEEE Computer Society: Washington, DC, USA, 2016; pp. 243–254.

[22] Aimar, A.; Mostafa, H.; Calabrese, E.; Rios-Navarro, A.; Tapiador-Morales, R.; Lungu, I.; Milde, M.B.; Corradi, F.; Linares-Barranco, A.; Liu, S.; et al. (2019). NullHop: A Flexible Convolutional Neural Network Accelerator Based on Sparse Representations of Feature Maps. *IEEE Trans. Neural Netw. Learn. Syst*. 30;644–656.

[23] Li, J.; Jiang, S.; Gong, S.; Wu, J.; Yan, J.; Yan, G.; Li, X. (2019). SqueezeFlow: A Sparse CNN Accelerator Exploiting Concise Convolution Rules. *IEEE Trans. Comput.* 68;1663–1677

[24] Lee, J.; Kim, C.; Kang, S.; Shin, D.; Kim, S.; Yoo, H. (2019). UNPU: An Energy-Effcient Deep Neural Network Accelerator With Fully Variable Weight Bit Precision. *IEEE J. Solid-State Circuits* 54;173–185.

[25] Lu, W.; Yan, G.; Li, J.; Gong, S.; Han, Y.; Li, X.(2017). FlexFlow: A Flexible Datafow Accelerator Architecture for Convolutional Neural Networks. In Proceedings of the 2017 *IEEE International Symposium on High Performance Computer Architecture (HPCA),* Austin, TX, USA, 4–8 February 2017; pp. 553–564.

[26] Tu, F.; Yin, S.; Ouyang, P.; Tang, S.; Liu, L.; Wei, S. (2017). Deep Convolutional Neural Network Architecture With Reconfgurable Computation Patterns. IEEE Trans. *Very Large Scale Integr. (VLSI) Syst*. 25;2220–2233.

[27] Qin, E.; Samajdar, A.; Kwon, H.; Nadella, V.; Srinivasan, S.; Das, D.; Kaul, B.; Krishna, T. (2020). SIGMA: A Sparse and Irregular GEMM Accelerator with Flexible Interconnects for DNN Training. In Proceedings of the 2020 *IEEE International Symposium on High Performance Computer Architecture (HPCA), San Diego*, CA, USA, 22–26 February 2020; pp. 58–70.

[28] Agnihotri, A., & Kohli, N. (2024). A novel lightweight deep learning model based on SqueezeNet architecture for viral lung disease classification in X-ray and CT images. *International Journal of Computational and Experimental Science and Engineering*, 10(4);592-613. https://doi.org/10.22399/ijcesen.425

[29] Priti Parag Gaikwad, & Mithra Venkatesan. (2024). KWHO-CNN: A Hybrid Metaheuristic Algorithm Based Optimzed Attention-Driven CNN for Automatic Clinical Depression Recognition . *International Journal of Computational and Experimental Science and Engineering*, 10(3)491-506. https://doi.org/10.22399/ijcesen.359

[30] Polatoglu, A. (2024). Observation of the Long-Term Relationship Between Cosmic Rays and Solar Activity Parameters and Analysis of Cosmic Ray Data with Machine Learning. *International Journal of Computational and Experimental Science and Engineering*, 10(2);189-199. https://doi.org/10.22399/ijcesen.324

[31] Rama Lakshmi BOYAPATI, & Radhika YALAVARTHI. (2024). RESNET-53 for Extraction of Alzheimer's Features Using Enhanced Learning Models. *International Journal of Computational and Experimental Science and Engineering*, 10(4)879-889. https://doi.org/10.22399/ijcesen.519

[32] ÇOŞGUN, A. (2024). Estimation Of Turkey's Carbon Dioxide Emission with Machine Learning. International *Journal of Computational and Experimental Science and Engineering*, 10(1)95-101. https://doi.org/10.22399/ijcesen.302

[33] Nagalapuram, J., & S. Samundeeswari. (2024). Genetic-Based Neural Network for Enhanced Soil Texture Analysis: Integrating Soil Sensor Data for Optimized Agricultural Management. *International Journal of Computational and Experimental Science and Engineering*, 10(4);962-970. https://doi.org/10.22399/ijcesen.572

[34] S.D.Govardhan, Pushpavalli, R., Tatiraju.V.Rajani Kanth, & Ponmurugan Panneer Selvam. (2024). Advanced Computational Intelligence Techniques for Real-Time Decision-Making in Autonomous Systems. *International Journal of Computational and Experimental Science and Engineering*, 10(4);928-937. https://doi.org/10.22399/ijcesen.591

[35] Paç, A. B., & Yakut, B. (2024). Assessing the Profit Impact of ARIMA and Neural Network Demand Forecasts in Retail Inventory Replenishment. *International Journal of Computational and Experimental Science and Engineering*, 10(4);811-826. https://doi.org/10.22399/ijcesen.439

[36] PATHAPATI, S., N. J. NALINI, & Mahesh GADIRAJU. (2024). Comparative Evaluation of EEG signals for Mild Cognitive Impairment using Scalograms and Spectrograms with Deep Learning Models. *International Journal of Computational and Experimental Science and Engineering*, 10(4)859-866. https://doi.org/10.22399/ijcesen.534

[37] Radhi, M., & Tahseen, I. (2024). An Enhancement for Wireless Body Area Network Using Adaptive Algorithms. *International Journal of Computational and Experimental Science and Engineering*, 10(3)388-396. https://doi.org/10.22399/ijcesen.409