[1]Dr. Aayushi Arya

[2]Dr Sunita Dixit

[3]Ashok Kumar Sahoo

[4]Y. Rajesh Babu

[5]Priyank Purohit

[6]Ibrahim Abdullah

Investigation into designing VLSI of a flexible architecture for a deep neural network accelerator

Abstract: - Deep learning, a branch of AI, makes use of specialised neural networks. Computational acceleration of high-performance deep neural network algorithms continues to necessitate efficient architecture, high memory bandwidth, parallel processing, and resources, despite decades of study on such algorithms. Excessive space requirements are a problem for DNN implementations caused by resource-intensive components like activation functions (AF) and multiply-and-accumulate (MAC) units. In addition, edge-AI applications necessitate a densely packed, power-hungry, high-throughput DNN accelerator. If we want to build a DNN accelerator that uses little power and occupies little space, we need to optimise the MAC architecture, the AF, and the network complexity so that data flows efficiently. In addition, providing functional configurability while working with restricted chip surface is a difficulty for DNN hardware designs based on ASICs. This dissertation explores the efficient and low-power VLSI architecture of DNN accelerators, addressing the hardware implementation of DNN and targeting applications with limited resources. In order to assess MAC and non-linear AF operations, we investigate and enhance the CORDIC architecture. The poor throughput is a major downside of CORDIC-based architectures, even though they are area and power efficient. Consequently, we suggest a pipelined design for CORDIC-based MAC and AF that focuses on performance. Due to the increased hardware resource consumption that comes with pipeline stages, this study investigates the mutual exclusivity of CORDIC stages and investigates in depth the accuracy variation related to the number of stages needed to achieve high throughput.

Keywords: Deep Learning, Artificial Intelligence, Deep Neural Networks, DNN Accelerators, Computational Efficiency, Parallel Processing, High Memory Bandwidth, Multiply-and-Accumulate (MAC) Units, Activation Functions,

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

Email: aayushi.arya@outlook.com

[2]Associate Professor SAITM Gurgaon

Bhatdwajsunita23@gmail.com

[3]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

[4]Assistant Professor, Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Andhra Pradesh, India , 522302 yrajeshbabu7@gmail.com

[5]Associate Professor, School of Pharmacy, Graphic Era Hill University, Dehradun; Adjunct Professor, Graphic Era Deemed to be University, Dehradun, Uttarakhand-248002, India

Mail id- prpurohit@gehu.ac.in

[6]Computer Science Department, Al-Turath University College, Baghdad, Iraq.

ibrahim.najim@turath.edu.iq

ORCID ID: 0000-0002-3070-2041

Copyright © JES 2024 on-line : journal.esrgroups.org

Introduction

Recent years have seen a meteoric rise in the popularity of artificial intelligence (AI). The advent of AI and ML has revolutionised contemporary society. Over the previous decade, this area has seen tremendous advancements. Recent developments such as autonomous cars, digital assistants, robots in manufacturing, and smart cities have demonstrated the feasibility of creating intelligent machines. Artificial intelligence has already revolutionised numerous industries, including retail, manufacturing, banking, healthcare, and journalism, and it is still expanding into uncharted regions. Machine learning is an AI subfield that enables computers to acquire new skills and knowledge without human intervention in the form of code. To create autonomous computer programmes that can get data and learn new things, machine learning is crucial. Machine learning, similar to the human brain, uses inputs like knowledge graphs or training data to comprehend domains, entities, and their relationships. Once entities have been established, deep learning can commence

Inspired by the way the human brain works, deep learning is a branch of machine learning that makes use of massive datasets to refine its algorithms. Similarly, the Deep Learning algorithm would iteratively finish a task, each time tweaking it slightly to improve the outcome, similar to how humans learn from experience. "Deep learning" is the name we give to neural networks that include several layers for the purpose of learning. Deep neural networks (DNNs) are the foundation of many of the latest AI applications. The goal of deep neural networks (DNNs), also known as deep learning, according to computer scientist John McCarthy, is to build intelligent machines capable of accomplishing human-made objectives. DNN's exceptional performance is a result of its ability to efficiently represent the input space by extracting high-level features from raw sensory data following statistical training on massive amounts of data. Additionally, DNN is widely used for a variety of non-linear detection issues, including lane detection, pattern identification, fault detection, and industry monitoring [8, 9]. Nevertheless, a DNN and real-time processing are frequently necessary for these applications, which necessitate substantial computational capacity. As a result of deep learning, many platforms are required to speed up the DNN algorithm's computation, such as GPU, CPU, ASIC, or FPGA. A number of processes, notably Multiply-Accumulate (MAC), are supported by the general-purpose platforms used by CPU and GPU-based DNN implementations. A major issue with CPUs and GPUs is the excessive amount of power they consume due to the underutilization of their resources [10]. When compared to streaming pixels and learning features, ASIC and FPGA devices offer quick multiplication operations with little resource utilisation and lower power consumption [11]. Additionally, FPGAs have a dedicated hardware architecture for MAC operation, but their design is tailored to DNN implementation, which allows them to achieve lower power consumption and high resource utilisation. The quickest and most energy-efficient hardware accelerators are those based on application-specific integrated circuits (ASICs). But they have limitations, such as a small inferred network size and an inability to re-configure neural networks with new features [12]. One 224× 224 input image requires 15.5G MACs and 138 million weights for convolutional neural networks (CNNs) like VGG-16. However, in the overall computation paradigm, MAC operations account for more than 90%. Our comparison of GPU, CPU, FPGA, and ASIC implementations is shown here. Although it requires more time and money to create, the NeuroFlow chip outperforms the GPU and CPU in terms of throughput. In addition, ASICs and FPGAs use less power and provide better performance per watt during operations. On top of that, FPGAs are very fast and cheap, thus it's easy to prototype the DNN with them. Despite a drop in performance, the CPU and GPU drastically cut down on development time and made deploying the DNN accelerator a breeze. Hardware should have low power consumption and high throughput to tackle edge-AI devices. For this reason, solutions based on field-programmable gate arrays or application-specific integrated circuits are necessary for these devices.

In different contexts, different kinds of activation functions are more or less useful. The multi-activation function's on-chip adjustable architecture, however, is not compatible with traditional architecture. Activation function (AF) optimisations and design strategies for a DNN architecture are explored in this chapter. Providing functional configurability while staying within the constraints of restricted chip size is a significant difficulty when designing activation functions (AF) in neural networks using ASICs. In order to fully utilise the ASIC's parallel processing capacity, as opposed to a general-purpose processor, an area-efficient customisable architecture for an AF is essential. Our proposed adaptable AF, the Coordinate Rotation Digital Computer (CORDIC) method, is based on the shift-and-add technique and aims to solve this problem. The suggested flexible activation function incorporates tan hyperbolic and sigmoid functions into its CORDIC design.

Computational models such as pattern recognition, prediction, and classification are well-known to use deep neural networks (DNNs). Computational unit, activation function (AF), mathematical precision, data formats, and design platform are the main components of a DNN implementation. There are three main categories into which hardware implementations of DNNs fall: application-specific integrated circuits (ASICs), digital signal processing (DSPs), and field programmable gate arrays (FPGAs). The alternatives do not compare to the performance and area advantages of an ASIC implementation [29]. Nevertheless, the ASIC design is immutable and cannot be altered to suit alternative uses. Furthermore, a big deep neural network implemented on an ASIC uses a lot of resources. Therefore, it is typically not done for neural networks to achieve high accuracy multiple activation functions. In addition, power-efficient applications require an effective design method to reduce the supply voltage [30, 31]. To further optimise performance, hardware implementation of a DNN might take advantage of the network's inherent parallelism. Various DNN-based applications have inspired the development of efficient VLSI designs [32], [15]. The accuracy and performance of a neural network, however, are heavily dependent on the precision of the data that is needed, from an application standpoint [32] [15]. In particular, when neural networks are implemented in hardware, increased precision typically comes at the expense of significant power and area overheads. The need for flexible architectures further complicates this trade-off. Compared to ASICs, FPGAs typically take up more space on the device, but they provide more customisable hardware designs [33]. Neural networks necessitate implementations of adjustable activation functions from a design perspective. Sigmoid, hyperbolic tangent (tanh), and exponential non-linear transformations are among its most appealing features [34]. Figure 1 shows the typical design of a neuron with several activation functions, as suggested in [35]. Among its many significant downsides are the following: higher data propagation time (as a result of MUX), power dissipation (static power dissipation) because of the unused hardware, and area overhead (path-A, path-B, and path-C) to separate activation functions. Since only one activation function can be enabled at a time, the separate hardware for each activation function is not the ideal option. To tackle this trade-off, we offer an optimised CORDIC-based architecture that allows neural networks to perform efficient yet adjustable computations.

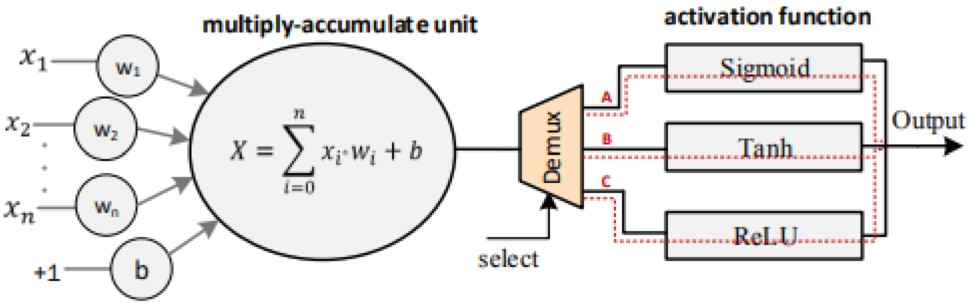

Figure 1: Typical design architecture of single neuron with configurable activation function

Types of Activation Functions and Design Techniques.

The non-linear transformation function of AF takes MAC output as input in DNN. More hardware resources are required for AF implementation because of its nonlinear nature, and resource utilisation grows exponentially with increasing precision. The PWL implementation also necessitates extra memory components for the AF with 16-bit and better precision. Since the number of memory components needed grows exponentially with increasing precision, hardware utilisation becomes prohibitively expensive. The DNN application makes advantage of many non-linear transformations.

There are various types of activation functions, including linear, sigmoid, tanth, regression linear units, parameterized regression linear units, exponential, linear unit, swish, and softmax [36]. In order to back-propagate from the output layer weights to the input layer weights, AF needs to be differentiable in order to calculate gradients. The ideal characteristics of an activation function are differentiability, non-linearity, and tractability. When it comes to activation functions, you can choose between linear and nonlinear varieties. A non-linear activation function should be chosen. Nonlinear activation functions like Sigmoid, Tanh, ReLU, and Exponential are more commonly utilised. Nevertheless, several nonlinear functions are utilised depending on the application; they include Leaky ReLU, SeLU, Softplus, and others. Several nonlinear activation functions with customisable architectures were covered, including sigmoid, tanh, and ReLU. In this chapter, we have created a flexible architecture based on the Co-ordinate Rotation Digital Computer (CORDIC) algorithm. It activates sigmoid, tanh, and ReLU using the same hardware.

Exploration of hardware implementation of activation functions

Popular AFs like sigmoid, Tanh, and ReLU are covered and explained in detail here. All of these functions are variants on the same basic concept, but their emphasis on specific mathematical processes makes them distinct. When it comes to local storage of AF parameters, it can make use of various on-chip memory resources as Lookup Tables (LUTs), Block RAM (BRAM), Distributed FPGA memory, or external DRAM. The following is a synopsis of a few of the implementation strategies:

1. LUTs based implementation by storing function

2. LUTs based implementation by storing parameters

3. Approximation in calculation into base-2 exploration

4. Coordinate Rotation Digital Computer (CORDIC) algorithm

5. Combinational logic based implementation The non-linear and continuous character of the AF makes a correct computation utilising hardware implementation problematic. Consequently, these functions are implemented using the Piece-Wise Linear (PWL) approach. As the number of quantization states grows exponentially with increasing precision (8-bit, 12-bit, 16-bit), the number of memory items required also grows exponentially. In [4], we can see the error contribution as a function of the various precisions of PWL activation functions. Even with combinational logic, non-linear AFs like sigmoid/tanh cannot be approximated efficiently [10]. In contrast to traditional ROM-based methods, however, utilising combinational logic alone results in reduced latency with minimal space overhead. To delve further into sigmoid function evaluation, an approximation approach for tanh AF implementation has been suggested in [4] employing combinational logic design.

Enhancing Performance of AF using Pipelining

The hyperbolic rotation mode of the CORDIC algorithm has been used in the hardware designs to implement adjustable activation functions. Because the CORDIC algorithm is inherently iterative, these systems have lower power and area overheads than traditional memory-based devices, but they have low throughput. In order to implement neural network applications with adjustable non-linear activation functions at high throughput, this section presents an improved CORDIC-based hardware solution that handles this trade-off. In brief, the following are the main contributions of the work:

This paper presents a Pareto analysis of the relationship between accuracy and the two independent CORDIC phases. For non-linear activation functions, this paper sheds light on how to determine the optimal number of pipeline stages to use in a CORDIC-based architecture, taking into account the space and power requirements of each step. According to our research, increasing throughput without decreasing accuracy is possible with a four-stage pipeline.

• In order to examine the effects of computational approximation, which results in a decreased number of pipeline stages, the error cost functions for the AF model are extracted.

• Using the CMOS 45nm technology node and comparing it to state-of-the-art designs, we examine and talk about the circuit's physical properties such as area, power, and critical delay for derived Pareto points.

Hardware Efficient and Configurable Design of Neuron using CORDIC Architecture

Area and performance suffer as computational complexity rises in a hardware implementation of an ANN. Because ASIC designs are fixed, there is additional effort required to configure them for features like bit precision flexibility or activation function types (sigmoid, hyperbolic tangent, tanh, etc.). Power and space are also used up by calculations with higher bit precision, such as 32-bit or 64-bit [5]. Quicker reaction times and simpler technology (bit accuracy) are therefore highly desired [6]. The classic design for ASIC-based neurons, as suggested in [8], has several activation functions and a multiply-accumulate unit. A multi-adder tree and many multipliers constitute a MAC unit. The activation function that receives the MAC unit's output is chosen by the multiplexer based on the application.

An Empirical Approach to Enhance the Performance of MAC Unit using Pipelining

A key challenge to the effective use of contemporary DNNs is the resource-intensive Multiply-Accumulate (MAC) unit. If we want Deep Neural Network (DNN) accelerators to operate better, we need to optimise computation efficiency and throughput. This project aims to create a MAC unit that is based on the CORDIC architecture. Low throughput is a key drawback of CORDIC-based devices, despite their size and power efficiency. Our performance-centric pipelined architecture for MAC solves the problem of low throughput. In this study, we use a Pareto analysis to look closely at the accuracy variation across various precision levels and the critical pipeline steps for optimum performance.

Introduction to MAC and its Performance Enhancing Techniques.

For calculations requiring greater bit-precision, DNN's resource-intensive MAC block becomes even more taxing due to its power-hungry multiplication operation. In the past, researchers have suggested effective ways to personalise ASIC and FPGA architectures. But there are compromises in physical performance metrics, throughput, and accuracy with state-of-the-art methods. Additionally, bandwidth constraint is a barrier while implementing the accelerator in hardware. Accordingly, in [7, 8], performance metrics for hardware-based acceleration with multi-bit precision (8, 16, 24, or 32 bits) have been examined. Less hardware resources were used by the approach of iterative computation shift-and-add based multiplication, which reduced the logic arithmetic complexity [6]. In [8], the CORDIC architecture is suggested as a means to efficiently develop MAC. Nevertheless, the throughput is reduced for the iterative CORDIC-based calculations. Additionally, in order to compute with n-bit accuracy, an n-bit barrel shifter and n-1 additions are needed. Presented in [75] is the efficient design for MAC based on Vedic multipliers, which operates at low precision. But when the algorithm's critical path and propagation time increase, it becomes unscalable for high-precision systems.

In order to save resources, researchers looked at the modified booths method for multiplication in [6]. The physical performance metrics were examined after implementing a MAC design based on Wallace trees [7]. Its architecture makes use of the efficient lower-precision AND and OR planes. Nevertheless, the complexity of such design increases as bit accuracy increases. The use of an approximation multiplier in MAC calculation helps to decrease the power consumption and circuit latency on the device. Approximate computation is preferred by error-tolerant applications. The article [9] introduced a partial product accumulation tree that is roughly accurate. Additionally, the multiplier's erroneous logic compressors provide encouraging outcomes for hardware implementation [8]. Consequently, power and energy efficiency are both enhanced.

Sharing weights simplifies MAC calculation and maintains a consistent storage capacity [11].Research on a reversible logic structure high-performance array multiplier that improves throughput performance has been conducted [8]. On the other hand, multi-precision signed/unsigned computing is outside the scope of current enhancement strategies. This study evaluates a high-performance CORDIC-based MAC unit for DNN accelerators, which addresses the area, power, and throughput trade-off.

To fix the throughput problem, this workflow uses CORDIC phases. The matter is handled by assessing the required pipeline steps, which come with an overhead in terms of area. Since accuracy and the number of necessary pipeline stages are inherently incompatible, we have examined several stages of the CORDIC-based architecture to comprehend the trade-off between the two. In addition, we have tested the accuracy of performance using various levels of mathematical precision. Although there is a little loss in accuracy, 8-bit precision computing is much faster than 16-bit precision and uses 4 times less memory bandwidth. On top of that, we were able to reduce space utilisation without sacrificing throughput performance or accuracy thanks to the Pareto research that evaluated the necessary number of phases. Using a 45 nm technology node, the next section synthesises the suggested design and describes the physical properties of the circuit, including area, power, and critical delay.

Iterative Architecture Benefits and their Limitations in MAC

Over the last ten years, researchers have concentrated on developing more efficient design strategies to enhance the MAC's performance. Particularly in the domain of DNNs, approximation computation approaches have been the main emphasis for design improvement. At a tolerable loss of accuracy, the technique enables the DNN accelerator to reduce its space and power consumption. Using a recursive Coordinate Rotation DIgital Computer (CORDIC) based architecture, researchers have examined the trade-offs between throughput, on-chip size, and power saving in [18]. The MAC implementation repeatedly conducts computing using the recursive CORDIC architecture. With its little power and space requirements, the CORDIC algorithm makes advantage of the shift-and-add operation. Each iteration of the CORDIC architecture's calculations is mutually exclusive, thus N-bit precision computing takes N+1 clocks. Therefore, instead of using iterative design in CORDIC, pipeline architecture may improve throughput performance. Because of their high computing requirements, DNNs have a resource-intensive design. One of the most important benefits of parallelism is the improvement in DNN's throughput performance. Space and energy are limited resources in AI and mobile apps running on the edge. As a result, we have improved the performance of the pipeline and suggested an efficient design of MAC based on the CORDIC architecture. There is area overhead with MAC, even if it includes pipeline steps that increase throughput. In situations when an approximation may cause errors, the neural network is robust. Thus, to determine the most efficient number of CORDIC pipeline stages in terms of both space and power consumption, a performance-centric study must be conducted. We mainly looked at the accuracy performance variance across pipeline stages in the assessment procedure. We were able to save a lot of space by reducing the number of pipeline stages needed. We have completed the design of the LeNet architecture using the suggested CORDIC-based MAC.

Various stages of the pipeline have verified the correctness. We continued training the network until we reached the maximum accuracy, and when compared to the correct Tensor libraries for MNIST and CIFAR-10, we saw an accuracy loss of around 1%. In comparison to computations based on Xilinx MAC, fully connected NN applied to Vertex-7 demonstrates performance that is about two times greater. Furthermore, at the 45 nm node, the physical properties of the design are retrieved and compared to prior studies.

Deep learning is a sophisticated AI approach that has been used to solve complicated issues in many different disciplines, including medical diagnostics, autonomous driving, picture identification, and natural language processing. Deep neural networks (DNNs) are the backbone of deep learning algorithms. DNNs are networks of artificial neurons that can learn complicated patterns and representations from input via linked layers of neurons. Despite DNNs' impressive performance in several domains, specialised hardware accelerators are often needed for their broad implementation to fulfil the rigorous computational demands of speed, power efficiency, and resource utilisation.

Designing efficient hardware designs for deep learning is still a demanding job, even after decades of study and development in high-performance DNN accelerators. Data neural networks (DNNs) need efficient hardware architectures that can perform computationally demanding operations with fast throughput and low power consumption. These operations include multiply-and-accumulate (MAC) and non-linear activation functions (AF). In addition, there is a growing need for economical and versatile VLSI designs for DNN accelerators due to the abundance of edge-AI applications that prioritise power economy above computing resources.

Hardware Implementation Layer-Multiplexed High-performance DNN Accelerator

In order to apply deep neural networks (DNN) to practical issues, substantial amounts of memory bandwidth, processing power, and other hardware resources are required. Efficient DNN engines are necessary for the AI-powered Internet of Things applications. As the depth of the neural network increases, so does the amount of hardware resources needed to construct a DNN. Our work here suggests a new architecture for the DNN engine that is both more economical and has better performance. A layer-multiplexed DNN accelerator has been developed and deployed at the system level. Implementing the suggested pipelined MAC design overcomes the throughput limits that occur with the architecture's reuse of the single layer inside the DNN.

When it comes to hardware cost, the multiply-accumulate (MAC) unit is where deep neural networks really shine [74]. Also, it uses well over 90% of the processing power in DNN accelerators [76]. The problem has been resolved by using the CORDIC method [18]. Invented in 1959 by Jack E. Volder, it is capable of doing many mathematical operations, including multiplication, trigonometric functions, and many more [87]. John Walter later in 1971 enhanced an effective algorithm for a modifiable mode of operation and its uses [8]. The hardware design process may make use of algorithms to simulate a neuron's mathematical processes [13]. The recursive-CORDIC architecture has been suggested as a new and efficient neuron computational unit [18]. Though it has poor throughput because of its recursive design, CORDIC-based neuron architecture beats the state-of-the-art in area and power reports when compared to earlier work [18, 19]. Due to its reliance on the recursive technique, the CORDIC-based design consistently delivers subpar performance. For architectures using layer multiplexing, this problem will become critical. For n-bit precision computation evaluations, the conventional CORDIC procedure calls for n clock cycles before the final results evaluation. The most recent, cutting-edge designs have tackled optimisation of designs at many levels of abstraction. Prior work effectively computes the essential mathematical action multiplication using several methods such as Vedic multiplier, Wallace tree adder, and booth's algorithm. In addition, methods including computational approximation in error-resistant applications, bit-serial computing bit rounding using CORDIC arithmetic, and reused hardware have been used to optimise designs [9]. Every cutting-edge method has its own set of trade-offs when it comes to performance metrics. Implementations of the MAC unit in DNN that are based on the recursive CORDIC engine provide the most efficient use of both space and power [18]. There are several benefits to the CORDIC-based architecture. Its primary benefit is the drastic reduction in both space and energy consumption. Furthermore, the design's computational simplicity allows it to reap greater advantages at higher precision arithmetic. Recursive CORDIC-based MAC's numerical precision is n iteration-dependent; more iterations lead to improved accuracy but significantly reduce throughput (n times). Second, a barrel shifter, which is a component of the hardware design of the generic CORDIC architecture, has a high critical latency and additional space overhead.

Conclusion

This work proposes a modular VLSI architecture that would allow fast and scalable deep learning solutions for edge-AI applications, which is a big step forward. The key to solving social problems and driving technical progress via the use of artificial intelligence lies in expanding the frontiers of innovation in very large scale integration (VLSI) design. To further evaluate the suggested architecture's practicality and performance in real-world settings, it would be beneficial to conduct empirical validation via hardware prototyping and real-world deployment in edge-AI applications.

References

1. V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, “Efficient processing of deep neural networks: A tutorial and survey,” Proceedings of the IEEE, vol. 105, no. 12, pp. 2295–2329, 2017.
2. A. Rodr´ıguez-V´azquez, R. Dom´ınguez-Castro, F. Medeiro, and M. Delgado-Restituto, “High resolution cmos current comparators: Design and applications to current-mode function generation,” Analog integrated circuits and signal processing, vol. 7, no. 2, pp. 149–165, 1995.
3. M. Kulkarni, V. Sridhar, and G. H. Kulkarni, “4-bit flash analog to digital converter design using cmos-lte comparator,” in 2010 IEEE Asia Pacific Conference on Circuits and Systems. IEEE, 2010, pp. 772–775
4. W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, “A survey of deep neural network architectures and their applications,” Neurocomputing, vol. 234, pp. 11–26, 2017.
5. E. Monmasson, L. Idkhajine, M. N. Cirstea, I. Bahri, A. Tisan, and M. W. Naouar, “Fpgas in industrial control applications,” IEEE Transactions on Industrial informatics, vol. 7, no. 2, pp. 224–243, 2011.
6. G. Raut, S. Rai, S. K. Vishvakarma, and A. Kumar, “A CORDIC based configurable activation function for ANN applications,” in 2020 IEEE Computer Society Annual Symposium on VLSI (ISVLSI). IEEE, 2020, pp. 78–83

7. E. Wu, X. Zhang, D. Berman, I. Cho, and J. Thendean, "Compute-efficient neural-network acceleration," in Proceedings of the 2019 ACM/SIGDA International Symposium on FieldProgrammable Gate Arrays, 2019, pp. 191–200.
8. C. Farabet, B. Martini, B. Corda, P. Akselrod, E. Culurciello, and Y. LeCun, "Neuflow: A runtime reconfigurable dataflow processor for vision," in CVPR 2011 workshops. IEEE, 2011, pp. 109–116.
9. S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," in International conference on machine learning. PMLR, 2015, pp. 1737–1746.
10. M. Blott, T. B. Preußer, N. J. Fraser, G. Gambardella, K. O'brien, Y. Umuroglu, M. Leeser, and K. Vissers, "Finn-r: An end-to-end deep-learning framework for fast exploration of quantized neural networks," ACM Transactions on Reconfigurable Technology and Systems (TRETS), vol. 11, no. 3, pp. 1–23, 2018.
11. E. Nurvitadhi, D. Sheffield, J. Sim, A. Mishra, G. Venkatesh, and D. Marr, "Accelerating binarized neural networks: comparison of FPGA, CPU, GPU, and ASIC," in FieldProgrammable Technology (FPT), 2016 International Conference on. IEEE, 2016, pp. 77– 84.
12. S. Himavathi, D. Anitha, and A. Muthuramalingam, "Feedforward neural network implementation in FPGA using layer multiplexing for effective resource utilization," IEEE Transactions on Neural Networks, vol. 18, no. 3, pp. 880–888, 2007.
13. G. Raut, S. Rai, S. K. Vishvakarma, and A. Kumar, "Recon: Resource-efficient cordic-based neuron architecture," IEEE Open Journal of Circuits and Systems (OJCAS), vol. 2, pp. 170–181, 2021. [Online]. Available: doi:10.1109/OJCAS.2020.3042743
14. K. Khalil, O. Eldash, A. Kumar, and M. Bayoumi, "An efficient approach for neural network architecture," in 2018 25th IEEE International Conference on Electronics, Circuits and Systems (ICECS). IEEE, 2018, pp. 745–748
15. Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," IEEE journal of solid-state circuits, vol. 52, no. 1, pp. 127–138, 2016.
16. G. Raut, A. Biasizzo, N. Dhakad, N. Gupta, G. Papa, and S. K. Vishvakarma, "Data multiplexed and hardware reused architecture for deep neural network accelerator," Neurocomputing, vol. 486, pp. 147–159, 2022.