iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 12 **Issue:** V **Month of publication:** May 2024

DOI: https://doi.org/10.22214/ijraset.2024.62595

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 12 Issue V May 2024- Available at www.ijraset.com

Review of Advanced Methods in Hardware Acceleration for Deep Neural Networks

Pramod Kumar[1], Hemant Sirohi[2], Prof (Dr) RM Bodade[3]
Faculty of Communication Engineering, Military College of Telecommunication Engineering, Indore, Madhya Pradesh, India

Abstract: ***Convolutional neural networks have become very efficient in performing tasks like Object Detection providing human like accuracy. However, their practical implementation needs significant hardware resources and memory bandwidth. In recent past a lot of research is being carried out for achieving higher efficiency in implementing such neural networks in hardware. We talk about FPGAs for hardware implementation due to their flexibility for customisation for such neural network architectures. In this paper we will discuss the metrics for efficient hardware accelerator and general methods available for achieving an efficient design. Further, we will discuss the actual methods used by recent research for implementation of deep neural networks particularly for object detection related applications. These methods range from actual ASIC design like TPUs [1] for on chip acceleration, state of the art open source designs like Gemini to methods like hardware reuse, re-configurable nodes and approximation in computations as a trade-off between speed and accuracy. This paper will be a valuable summary for the researchers starting in the field of hardware accelerators design for neural networks.***
Keywords: Deep Neural Network, FPGA, Quantization, Precision, Object Detection, YOLO.

I. INTRODUCTION

Artificial Intelligence (AI) including Machine Learning and Deep Learning has evolved as a universal field with applications in every aspect of life. Deep learning is based on the working of human brain and utilizes deep neural networks for achieving high degree of accuracy and is being widely used in computer vision. The main advantage of the DNN over other prediction techniques is its capability to learn hidden relationships in data with unequal variability Hardware accelerator term is used for devices which are specifically designed for a particular application such as object detection. On one hand, we have general-purpose systems such as CPUs and GPUs that are capable of performing any task but are not very efficient, and on the other hand, we have ASICs that can run with high efficiency for the task at hand but require lots of effort, time, and money to build. In this trade-off between efficiency and generalizability, FPGAs provide a very attractive middle ground. FPGAs are a set of logic gates that can be programmed to perform any function. In this review we examine the research in the field of hardware accelerators using FPGAs particularly in the field of object detection using Convolutional neural networks. We cover the various methods of optimising the efficiency and throughput without significant loss of accuracy.

II. KEY PERFORMANCE METRICS FOR HARDWARE ACCELERATORS

A. Accuracy

Accuracy [2] is employed as a metric to signify the level of quality in achieving a desired outcome for a certain task. The popularity and extensive utilisation of Deep Neural Networks (DNNs) in contemporary times can be attributed to their ability to attain exceptional accuracy across a diverse array of jobs. The choice of units for measuring precision is contingent upon the specific activity at hand. In the context of object detection, accuracy is commonly quantified as the proportion of images accurately predicted. On the other hand, in the domain of object detection, accuracy is typically measured using the mean average precision. Complexity in terms of the number of MAC operations and different weights, as well as increased size of the model) is sometimes necessary to attain high accuracy on challenging tasks or datasets. The variation in layer shapes and other factors can have an impact on the hardware's efficiency in processing the DNN model.

B. Throughput and Latency

Throughput is a metric employed to quantify the volume of data that can be effectively processed or the frequency of task executions that can be successfully accomplished within a specified duration. The need of high throughput is frequently emphasised in various applications. For example, the requirement of processing video at a rate of 30 frames per second is essential in order to achieve real-time performance.

In the context of data analytics, the term “high throughput” refers to the ability to analyse a larger volume of data within a specified time frame. The significance of high-throughput big data analytics is escalating due to the exponential growth of visual data. This is especially crucial when prompt action is required based on the analysis, such as in the fields of security or terrorism prevention, medical diagnostics, or drug discovery. The metric commonly used to quantify throughput is the frequency of operations executed within a given time frame. When it comes to inference, the measurement of throughput is typically expressed as the number of inferences performed per second, or alternatively, as the duration required to execute a single inference, measured in seconds. Latency refers to the temporal interval between the arrival of input data to a given system and the generation of the corresponding output. The requirement of low latency is crucial for the seamless operation of real-time interactive applications, including augmented reality, autonomous navigation, and robots. The measurement of latency is commonly in units of seconds. The impact of the DNN model on the number of MAC operations per inference is dependent on network architecture (specifically, layer shapes) and the sparsity of weights and activations. However, the overall effect of the DNN model on throughput and latency is contingent upon the hardware’s capability to incorporate these approaches without significantly diminishing the utilisation of processing elements (PEs), the number of PEs, or the cycles per second. This is the reason why the quantity of MAC operations may not serve as an adequate indicator for evaluating both throughput and delay. Likewise, the quantity of processing elements (PEs) present in the hardware and their maximum throughput do not provide an adequate basis for assessing the overall throughput and latency. Reporting the real runtime of deep neural network (DNN) models on hardware is of utmost importance in order to consider additional factors such as the utilisation of processing elements (PEs).

C. Energy and Power

Energy efficiency refers to the measurement of the quantity of data processing per unit of energy. The significance of high energy economy becomes evident when considering the processing of deep neural networks (DNNs) on embedded devices situated at the edge, which possess limited battery capacity. Edge processing is often favoured over cloud computing in specific applications because of factors such as latency, privacy concerns, and constraints in connection bandwidth. The quantification of energy efficiency is commonly expressed as the ratio of activities performed to the amount of energy consumed, typically measured in joules. Power consumption” is a metric employed to quantify the quantity of energy utilised within a given timeframe. The relationship between power consumption and heat dissipation is such that an increase in power consumption leads to a corresponding increase in heat produced. Consequently, the maximum power consumption is determined by a design criterion commonly referred to as the thermal design power. The thermal design power is the power level that the cooling system is specifically designed to handle. The consideration of power consumption is crucial in processing Deep Neural Networks (DNNs) in cloud environments. This is primarily due to the fact that data centres impose strict power limitations, primarily driven by the associated cooling expenses. Additionally, handheld/wearable products face stringent power constraints. This is because users of such devices are typically highly sensitive to heat, and the physical design of these devices restricts the implementation of effective cooling mechanisms, such as the absence of fans. Power consumption is commonly expressed in units of watts (W) or joules per second (J/s).

D. Hardware Cost

To assess the desirability of a specific architecture or technique, it is imperative to take into account the associated hardware expenses.

The term”hardware cost” refers to the financial expenditure associated with constructing a system. Determining the financial viability of a system holds significance from both an industrial and research standpoint. From a business standpoint, the limitations on costs are influenced by market dynamics. For instance, embedded devices are subject to more rigorous cost constraints compared to processor utilised in cloud computing. The chip area, measured in square millimetres, and the process technology, such as 45 nm CMOS, are significant factors that influence cost. These factors determine the limitations on on-chip storage capacity and computational capabilities, such as the number of processing elements (PEs) for custom deep neural network (DNN) accelerators, the number of cores for central processing units (CPUs) and graphics processing units (GPUs), and the number of digital signal processing

(DSP) engines for field programmable gate arrays (FPGAs), among others. In order to convey data pertaining to area without explicitly mentioning a particular process technology, one can utilise the on-chip memory capacity (such as the storage capacity of the global buffer) and computational resources (such as the number of processing elements) as indicators of area.

E. Flexibility

The adaptability of a deep neural network (DNN) accelerator is also a determining factor in its merit. Flexibility pertains to the extent of DNN models that can be accommodated on the DNN processor, as well as the software environment's capacity (such as the mapper) to fully use the hardware's capabilities for any given DNN model. In light of the rapid progress in research and implementation of deep neural networks (DNNs), it is becoming more crucial for DNN processors to possess the capability to accommodate a diverse array of DNN models and jobs.

F. Scalability

The significance of scalability has grown in light of the extensive applications of deep neural networks (DNNs) and the emergence of technologies aimed at expanding not just the chip's dimensions but also constructing systems comprising many chips, commonly known as chiplets, or even wafer scale processors. The term "scalability" pertains to the "extent to which a design can be expanded" in order to "achieve greater throughput and energy efficiency" by using number of "processing elements (PEs) and on-chip storage

III. TECHNIQUES TO IMPROVE EFFICIENCY OF HARDWARE ACCELERATOR FOR DNNs.

A. Energy Efficient Data Flows

In the context of DNNs, the main performance bottleneck arises from memory access. Each MAC operation necessitates three memory reads (for filter weight, feature map activation, and partial sum) and one memory write (for updating the partial sum). In the worst-case scenario, all these memory accesses must go through the off-chip DRAM, significantly impacting both processing throughput and energy efficiency. Accelerators, such as spatial architectures, present an opportunity to mitigate the energy cost associated with data movement by introducing multiple tiers of local memory hierarchy, each with varying energy costs. This hierarchy includes a substantial global buffer, several hundred kilobytes in size, connected to DRAM; an inter-PE network enabling direct data exchange between ALUs; and a register file (RF) within each processing element (PE), typically a few kilobytes or smaller. The inclusion of these diverse memory levels helps enhance energy efficiency by offering cost-effective data access solutions. For instance, retrieving data from the RF or neighbouring PEs incurs energy costs one or two orders of magnitude lower than fetching from DRAM. Accelerators can be tailored to support

Specialized processing data flows that leverage this memory hierarchy. The dataflow determines which data are read into which level of the memory hierarchy and when they are processed. Given the deterministic nature of DNN processing, it's possible to design a fixed dataflow that can adapt to various DNN shapes and sizes, optimizing for optimal energy efficiency. This optimized dataflow minimizes accesses to the more energy-intensive levels of the memory hierarchy, recognizing that larger memories capable of storing substantial data consume more energy than smaller counterparts.

B. Data Reuse Opportunities in DNNs

Few properties of DNN lend them to be leveraged for data reuse. Firstly, MAC operations do not have to be in a particular order within a layer which lends it for parallelism to improve throughput. Secondly, the "data movement cost can be addressed by the DNNs property that same piece of data is often used for multiple MAC operations" [3]. This property manifests in following forms:

1) *Input Feature Map Reuse:* Reuse of input feature maps occurs when various filters (with a dimension of M) are applied to the same input feature map, leading to the generation of multiple output feature maps. Consequently, each input activation is utilized multiple times, precisely M times.
2) *Filter Reuse:* Filter reuse manifests when processing a batch of input feature maps (with a size of N), as the same filter is applied to all inputs within the batch. Consequently, each filter weight is employed repeatedly, specifically N times.
3) *Convolutional Reuse:* Convolutional reuse pertains to convolutional layers, where the dimensions of a filter (R x S) are smaller than those of an input feature map (H x W). The filter traverses various positions within the input feature map, often with overlapping regions, to produce an output feature map. As a result, each weight and input activation are additionally reused P x Q and R x S times, respectively, to generate distinct output activations.

C. Reducing Precision

Data movement consumes huge energy and determines the throughput. The bit width for weight representation can be reduced to improve latency. "Using fewer bits per weight and/or activation effectively reduces the number of unique values that they can take on and thus is often referred to as reducing precision [4]. It results in reduced data movement, reduced storage cost, reduced energy per memory access, and reduced energy and time per MAC operation".

Benefits of Reduction in Precision: -

1) Reduces data movement and thereby consuming less energy and improving throughput. This keeps the PEs busy and reduces idle cycles:
2) Reduces storage required for "weights, activations, and/or partial sums" by reducing on chip memory and keeping more weights in same memory by data reuse:

D. Quantisation

A strategy to lower computational expenses involves minimizing the bit count necessary to represent the weights and activations of a DNN model." This reduction in bits allocated per weight and/or activation results in fewer distinct values they can represent, commonly known as precision reduction. The bit width of a specific data type is determined by the number of unique values it can encompass. This transformation, which involves mapping data from a wide range of potential values (full precision) to a more limited set of values (reduced precision), is referred to as quantization.

E. Mixed Precision

The most suitable quantization approach depends on the data distribution. To further reduce the necessary bit width while upholding accuracy, a tailored quantization method can be employed for each distinct data type, each of which possesses varying distributions. Employing diverse levels of precision for different data types is commonly known as mixed precision. In the context of inference, the pertinent data types encompass weights, activations, and partial sums. When it comes to training, the relevant data types comprise weights, activations, partial sums, gradients, and weight updates. Typically, training necessitates higher precision than inference due to the wider range exhibited by gradients and weight updates compared to weights and activations. Consequently, this is why inference can be executed using 8-bit fixed-point representation, whereas training requires 16-bit floating-point precision.

F. Varying Precision

Much like how distinct data type's exhibit varying distributions, these distributions can also differ across different segments of the DNN, such as layers, filters, or channels. Consequently, for a more pronounced reduction in bit width, the quantization approach can adjust to accommodate these varying distributions. Allowing the level of precision to fluctuate across the diverse sections of the DNN model is commonly termed as varying precision.

G. Binary Nets

There have been studies focused on reducing the number of bits used for weights and/or activations to as few as one bit, essentially limiting each input value to just two possible values. Such types of DNNs are commonly referred to as binary nets. In addition to lowering memory requirements, binary nets also substantially decrease computational costs. When one of the operands is binary, multiplication simplifies to addition. If both operands are binary, the multiplication operation can be transformed into an XNOR operation, and multiple multiplications can be executed using bitwise XNOR operations. Following these bitwise operations, accumulation is performed using a hop count. However, one of the main drawbacks of binary nets is their impact on accuracy. Using binary weights and activations directly leads to a significant drop in accuracy, with a 29.8 percentage point decrease observed when applying them to AlexNet for image classification on the Image Net dataset. Consequently, binary nets are often combined with many of the techniques discussed earlier to regain lost accuracy. For example, in many cases, the first and last layers in the DNN are retained at full precision (32-bit float), which is a form of varying precision that necessitates adaptable hardware.

H. Exploiting Sparsity

An important attribute of the data employed in DNN calculations is its potential sparsity. When we mention data sparsity, we are indicating the presence of numerous recurring values within the data, with zero being the most frequent repeated value. This sparsity gives an opportunity to reduce MAC operations. This may result in saving of energy or time or both." The following sources can be exploiting sparsity: -

1) *Activation Sparsity:* Sparsity in activation data can arise from various sources, with the most apparent and frequently leveraged source being the use of ReLU as a nonlinearity. Additional sources include capitalizing on correlations within input data or the up sampling process in up-convolution layers. ReLU, a commonly employed nonlinearity in DNNs, functions by setting all negative values to zero. Consequently, the resulting output activations of feature maps post- ReLU exhibit fewer non-zero values, essentially becoming sparse. For instance, experimental results with AlexNet's feature maps indicate a density of non-

zero values ranging from 19 to 63This activation sparsity gives ReLU an advantage over other nonlinearities like sigmoid, among others. Further reduction in activation density can be achieved by setting values below a specific threshold to zero, often referred to as pruning, a technique that can also be applied to weights. The fact that ReLU effectively eliminates negative output activations introduces the possibility of employing approximate computing to reduce the number of MAC operations. To be precise, we can halt the computation of the output activation value if we can predict in advance that the output will be negative.

2) *Weight Sparsity:* Sparsity in weights can originate from various sources. One naturally occurring instance of repeated weights within a filter arises when the number of weights in the filter surpasses the count of unique weights possible, a limit often determined by the number of bits per weight. For instance, if each weight in a 3 x 3 x 128 filter is represented using 8 bits, it will exhibit repeated weights since 8 bits can only represent 256 distinct values. Additionally, the introduction of repeated weights in the DNN model can be intentional and controlled. This can be achieved through filter design or implemented during the training process by techniques such as weight pruning (where weights are set to zero) or by reducing the number of distinct weights available

IV. RECENT WORKS ON EFFICIENT HARDWARE IMPLEMENTATION OF DNNS

A. *Binarized Neural Networks*

According to S liang et al [5] DNN accelerators which are based on FPGAs suffer problems of limited resources and memory bandwidth can be a bottleneck during data loading. To mitigate this problem, they have use Binarization to compress a neural network. This technique can directly shrink the weights from 32 bit (single precision floating point) to a single bit. Analytical resource aware model analysis with XNOR, pop count and shifting operations for BNN to replace multipliers has been used for this method.

B. *Tiny YOLO*

Object detection architecture, for the purpose of implementing it on an FPGA. To optimise computational efficiency, a strategy is employed wherein a highly efficient and versatile computing engine is utilised. This computing engine consists of 64 replicated Processing Elements (PEs) that operate concurrently. Within each processing element (PE), a total of 32 multiply accumulate (MAC) operations are carried out in a pipelined fashion to enhance parallelism. In order to minimise memory usage, the technique of data reusing and data sharing has been employed. This involves parallel processing elements (PEs) sharing the same input data, and utilising on-chip buffers to store and cache data and weights for subsequent reuse.

C. *YOLO Object Detection*

In a related work on YOLO [6] streaming hardware accelerator implemented a YOLO CNN whose parameters are retrained and quantized with PASCAL VOC dataset using binary weight and flexible low-bit activation. The binary weight enables storing the entire network model in Block RAMs of a field-programmable gate array (FPGA) to reduce off-chip accesses aggressively and thereby achieve significant performance enhancement.

D. *Gemmini Generator*

Gemmini [7] is a comprehensive, open-source tool that generates deep neural network (DNN) accelerators. It supports a wide range of hardware architectures, programming interfaces, and system interaction options. Gemmini provides users with the capability to construct a wide range of accelerators, ranging from low-power edge accelerators to high performance cloud accelerators that are equipped with out of- order CPUs. Subsequently, users have the opportunity to examine the interplay between hardware components, system on- a-chip (SoC) architecture, operating system (OS), and software overhead in order to assess their collective impact on overall performance and efficiency.

E. *Spiking Neural Networks*

Unlike ANNs where the major operations are matrix multiplications of weights and activations of network layers, the spiking nature of SNNs avoid complex computations thereby reducing resource and energy requirements. J Han et al [8], [9] proposed a FPGA based SNN hardware implementation using a hybrid of time stamped and event driven updating algorithms. The model was evaluated on MNIST dataset and showed a classification accuracy of 97.06 percent with .477 W power consumption and achieved a frame rate of 161 frames per second. The issues of memory bandwidth bottleneck can be reduced by reducing bit width of weights and network pruning.

F. Efficient Activation Function Implementation

Activation functions using purely combinational circuits. They focussed on two widely used activation functions namely SELU and Tanh. The results of their work show that neural networks are generally insensitive to precision of the activation functions and combinational circuits-based approach is very efficient in terms of speed and area with negligible accuracy loss. The results were validated using MNIST, CIFAR and ImageNet datasets. On similar concept different ways can be used to approximate the activation functions [10], [11], [12], [13], [4].

G. Approximation of transcendental functions

Xin Fan et al [10] proposed leveraged pareto front optimisation at three hierarchies from the parameter layer upto the structure and algorithmic layer to approximate the design space of activation functions. They achieved an accuracy of 99 percent on MNIST dataset. In this work a compact shift-based approximation is proposed that applies to 2's compliment numbers which achieves 1.5 times area reduction and 3.4 times energy consumption reduction. H. Approximate Computing Approximate computing is predicated on the capacity of several systems and applications to endure a certain degree of degradation in the quality or optimality of the calculated outcome. Approximate computing approaches offer a significant enhancement in energy efficiency by alleviating the requirement for absolute precision or deterministic computations. Jie Han et al [9] explored various design of approximation arithmetic blocks, as well as the relevant error and quality measurements. Few approximate arithmetic circuits are given below: -

1) Approximate Full Adders

In several approximate implementations, multiple-bit adders are partitioned into two modules: the upper module, which handles the more significant bits with accuracy, and the bottom module, which handles the less significant bits with approximation. A modified and therefore imprecise function of addition is implemented by a single-bit approximate adder for each lower bit. The process of simplifying a full adder design at the circuit level involves modifying some entries in the truth table of the full adder at the functional level.

2) Approximate Multipliers

The utilisation of speculative adders to calculate the sum of partial products is a common approach when considering approximate multipliers. Nevertheless, employing approximate adders directly in a multiplier may not be optimal in terms of balancing accuracy with energy and area savings. Reducing the critical route of combining the partial products is a crucial design aspect for achieving an approximation multiplier. Due to the typical implementation of multiplication using a cascaded array of adders, it is common practise to omit certain less significant bits in the partial products.

3) Limited Numerical Precision

Deep networks can be trained using only 16-bit wide fixed point number representation when using stochastic rounding, and incur little to no degradation in the classification accuracy according to Suyog Gupta et al [14] Also, an energy efficient hardware accelerator can be implemented using low precision fixed point arithmetic. The substitution of floating-point units with fixed-point arithmetic circuits comes with significant gains in the energy efficiency and computational throughput, while potentially risking the neural network's performance. For low-precision fixed-point computations, where conventional rounding schemes fail, adopting stochastic rounding during deep neural network training delivers results nearly identical as 32-bit floating-point computations

Configurable Layer Reuse Configurable Layer Reuse for minimalist Hardware Architecture. The work of E Wu et al [15] proposes a configurable layer reuse architecture with an aim to compute faster and keeping compute busy and simplify implementation. They used 3 parallel googlenets with their own weights and aggregated 3046 images per second. In this work 95 percent DSP cycles were utilized; all tensors were stored in SRAM (UltraRAM and BRAM).

Rout et al [4] proposed an approach of performance centric pipeline Coordinate Rotation Digital Computer (CORDIC)–based MAC unit that implements a scalable CORDIC-based DNN architecture that is area- and power efficient and has high throughput. The CORDIC-based neuron engine uses bit-rounding to maintain input-output precision and minimal hardware resource overhead.

V. CONCLUSION

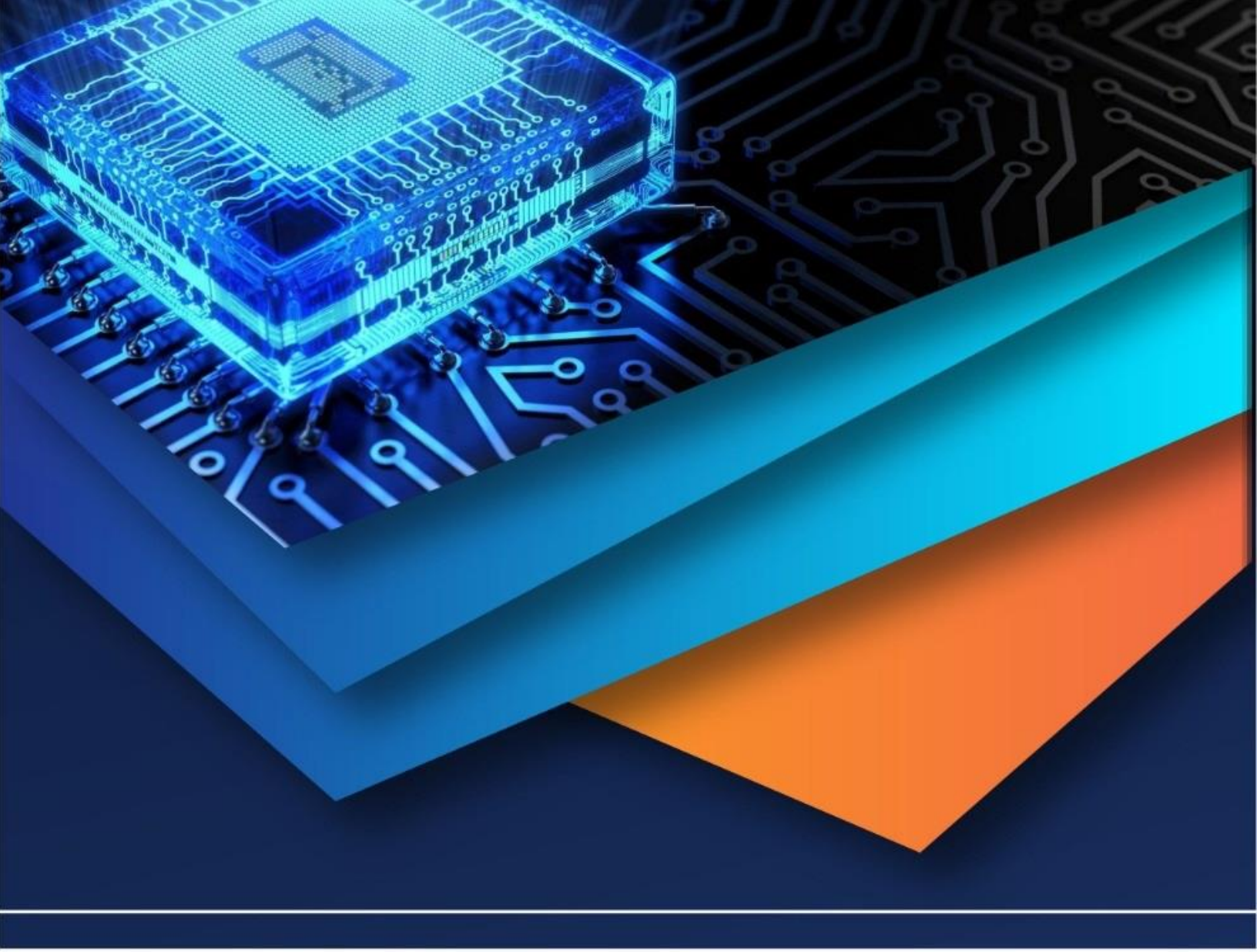
We have compiled the most recent research in the field of implementation of Deep Neural Networks on hardware (FPGA). It can be observed that due to limitation of memory bandwidth and compute resources there exist many problems in implementing full scale neural networks on FPGA.

However, for application of deep neural networks like object detection these hardware (FPGA) can be exploited due to their advantages over ASICs and general purpose hardware (CPU/GPU). With the evolution of DNN Models starting from simple neural networks for edge identification to latest YOLO models the complexity of models is increasing and this demands to have a set of methods which are available with the researchers so as to use while implementing these models. This paper has summarized such methods which are effective and have been validated in various experiments by researchers and can be employed in further research in this field.

REFERENCES

Here are the references used in this comprehensive review of modern counter-drone technologies:

[1] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers et al., "In data centre performance analysis of a tensor processing unit," in Proceedings of the 44th annual international symposium on computer architecture, 2017, pp. 1–12.
[2] 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.
[3] K. Khalil, O. Eldash, B. Dey, A. Kumar, and M. Bayoumi, "A novel reconfigurable hardware architecture of neural network," in 2019 IEEE 62nd International Midwest Symposium on Circuits and Systems (MWSCAS). IEEE, 2019, pp. 618–621.
[4] 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.
[5] S. Liang, S. Yin, L. Liu, W. Luk, and S. Wei, "Fp-bnn: Binarized neural network on fpga," Neurocomputing, vol. 275, pp. 1072–1086, 2018.
[6] D. T. Nguyen, T. N. Nguyen, H. Kim, and H.-J. Lee, "A high-throughput and power-efficient fpga implementation of YOLO cnn for object detection," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 27, no. 8, pp. 1861–1873, 2019.
[7] H. Genc, S. Kim, A. Amid, A. Haj-Ali, V. Iyer, P. Prakash, J. Zhao, D. Grubb, H. Liew, H. Mao et al., "Gemmini: Enabling systematic deep-learning architecture evaluation via full-stack integration," in 2021 58th ACM/IEEE Design Automation Conference (DAC). IEEE, 2021, pp. 769–774.
[8] J. Han, Z. Li, W. Zheng, and Y. Zhang, "Hardware implementation of spiking neural networks on fpga," Tsinghua Science and Technology, vol. 25, no. 4, pp. 479–486, 2020.
[9] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," in 2013 18th IEEE European Test Symposium (ETS). IEEE, 2013, pp. 1–6.
[10] X. Fan, S. Zhang, and T. Gemmeke, "Approximation of transcendental functions with guaranteed algorithmic qos by multilayer pareto optimization," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 28, no. 12, pp. 2495–2508, 2020.
[11] S. Gomar, M. Mirhassani, and M. Ahmadi, "Precise digital implementations of hyperbolic tanh and sigmoid function," in 2016 50th Asilomar Conference on Signals, Systems and Computers. IEEE, 2016, pp. 1586–1589.
[12] K. Leboeuf, A. H. Namin, R. Muscedere, H. Wu, and M. Ahmadi, "High speed vlsi implementation of the hyperbolic tangent sigmoid function," in 2008 Third international conference on convergence and hybrid information technology, vol. 1. IEEE, 2008, pp. 1070–1073.
[13] G. Rajput, G. Raut, M. Chandra, and S. K. Vishvakarma, "Vlsi implementation of transcendental function hyperbolic tangent for deep neural network accelerators," Microprocessors and Microsystems, vol. 84, p. 104270, 2021.
[14] 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.
[15] I. Tsmots, O. Skorokhoda, and V. Rabyk, "Hardware implementation of sigmoid activation functions using fpga," in 2019 IEEE 15th International Conference on the Experience of Designing and Application of CAD Systems (CADSM). IEEE, 2019, pp. 34–38.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY