A Survey on Neural Network Hardware Accelerators

Tamador Mohaidat, Kasem Khalil, *Senior Member, IEEE*

***Abstract*—Artificial intelligence hardware accelerator is an emerging research for several applications and domains. The hardware accelerator's direction is to provide high computational speed with retaining low-cost and high learning performance. The main challenge is to design complex machine learning models on hardware with high performance. This paper presents a thorough investigation into machine learning accelerators and associated challenges. It describes a hardware implementation of different structures such as Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), and Artificial Neural Network (ANN). The challenges such as speed, area, resource consumption, and throughput are discussed. It also presents a comparison between the existing hardware design. Lastly, the paper describes the evaluation parameters for a machine learning accelerator in terms of learning & testing performance and hardware design.**

Impact statement– Neural networks have revolutionized the field of artificial intelligence, empowering machines to acquire knowledge from data and accomplish tasks that were previously deemed unachievable. This survey covers various types of accelerators, including custom ASICs, FPGAs, GPUs, and dedicated AI chips, and compares their performance, power efficiency, and scalability. The survey also discusses the design trade-offs involved in building neural network accelerators, such as memory hierarchy, dataflow architecture, and precision. It provides insights into the latest trends and advancements in hardware accelerators for neural networks. It helps researchers, engineers, and practitioners in the field choose the right hardware platform for their specific needs and optimize the performance and energy efficiency of their neural network models. Moreover, this survey can also inspire new research directions and advancements in the neural network hardware accelerator design, paving the way for the next generation of intelligent systems.

Keywords: Neural network, Artificial intelligence, hardware accelerator, Machine learning, machine learning on-chip, Recurrent neural network, Convolutional neural network, Artificial neural network, Machine learning design.

I. Introduction

MACHINE learning has emerged with a number of new issues alongside the growth of the Internet and multimedia technology. Machine learning has gained traction in various disciplines, with the goal of simulating human thought. Researchers from diverse fields collaborate to share their perspectives and techniques, contributing to the progression of the artificial intelligence field. Machine learning approaches are involved in several applications, and domains such as speech recognition [1, 2], image classification [3, 4], hardware and software fault prediction [5–7], text detection [8, 9], disease detection and prediction [4, 10, 11], nutrition monitoring[12], medical treatment [13, 14], and for defect inspection and metrology [15, 16]. The main challenges of current approaches are providing an optimized model of learning and hardware accelerator with low cost.

Machine learning serves as a crucial step towards achieving artificial intelligence. Functioning as a specialized branch of AI, machine learning involves utilizing data and algorithms to simulate human learning processes and continually improve its accuracy. Rather than relying on direct instruction, machine learning involves the use of mathematical models to facilitate computer learning. By utilizing historical data as input, machine learning algorithms can predict new output values. As the core component of artificial intelligence, machine learning enables computers to possess intelligent capabilities. Through the development of various theories and methodologies, research in machine learning aims to establish a specific application-based study system that utilizes insights from human physiology and cognitive science to conduct theoretical analysis and advance algorithmic models. Machine learning tracks complex human actions in multimedia streams. The field of machine learning comprises various types, including reinforcement learning, supervised learning, and unsupervised learning. Supervised learning involves training a model using data that is labeled, while unsupervised learning involves training a model using data that is unlabeled [17–19]. Reinforcement learning, on the other hand, requires trial and error to train a model [20, 21].

The importance of machine learning comes from its capacity to speed up the creation of new products and grant businesses insights into consumer behavior trends and operational patterns. Many of today's leading companies incorporate machine learning into their operations. Machine learning has become a critical factor in setting businesses apart from their competitors. Machine learning is reshaping every industry, from healthcare and education to transportation and the food and entertainment industries to manufacturing and more. The fields of bioinformatics, physics, chemistry, material analysis, and other related disciplines utilize intelligent methods that enhance their content development. These methods leverage machine learning as a core technology. It will significantly influence nearly every facet of individuals' lives. Both cloud computing and the Internet of Things (IoT) are driving the

Tamador Mohaidat is with the Electrical and Computer Engineering Department, University of Mississippi, Oxford, MS 38677 USA

Kasem Khalil is with the Electrical and Computer Engineering Department, University of Mississippi, Oxford, MS 38677 USA, and also with the Electrical Engineering Department, Assiut University, Asyut 71515, Egypt (e-mail: kmkhalil@olemiss.edu).

expanded utilization of machine learning to enable objects and gadgets to become ”smart” on their own [22–24].

Deep learning typically employs multiple layered structures, such as Convolutional Neural Networks (CNN) [25, 26], Recurrent Neural Networks (RNN) [27, 28], and Artificial Neural Networks (ANN) [29, 30], to process large-scale and unstructured data. Each structure has its own model of learning. ANN is based on multiple layers connected in a chain. Each layer has multiple nodes that perform the computation process. The nodes in each layer are interconnected with the nodes in the following layer until they reach the output layer. The architecture of a CNN consists of convolutional layers and pooling layers, with a pooling layer following each convolutional layer. A convolutional layer is used to run the computation of the input data with the stored weights. To reduce complexity in the subsequent layers, a pooling layer is utilized to diminish the data size. The RNN is based on memory for the learning process which makes it suitable for time series data.

Hardware implementation of machine learning has a significant role in current applications with low cost [31–36]. The main challenge is to provide a machine learning accelerator with high speed for problem classification with low hardware costs, without compromising desired performance in terms of area and power [37–42]. The objective of this paper is to examine the current machine-learning hardware accelerator approaches with their advantages and limitations discussion. It also defines the general challenges for any hardware accelerator design to be considered in future methods. Furthermore, it presents the evaluation parameters for a hardware accelerator. This study delves into a comprehensive examination of various hardware accelerators, without being limited to specific neural network architectures. It conducts an extensive comparative analysis of recent advancements, highlighting their respective strengths and shortcomings. This is accomplished by presenting numerical data for key performance metrics, including power consumption, area, and accuracy. By providing this detailed information, the reader gains a precise understanding of the distinctive attributes of each examined work. This paper's main contributions are summarized as follows:

- A list of the challenges on machine learning hardware accelerator.
- A comprehensive study on hardware accelerator systems.
- A comprehensive review on machine learning hardware accelerator.
- A comparison between the existing hardware accelerators
- An evaluation framework for machine learning hardware accelerators.

The subsequent sections of this paper are organized in the following manner. Section II presents unique challenges of machine learning hardware accelerator. Section III investigates multiple hardware accelerator systems. The models and datasets covered by the existing methods are presented in Section IV. Section V presents a review of machine learning accelerator with a comparison between the existing methods. Section VI presents an evaluation framework of a hardware accelerator, followed by the conclusion and a discussion of future work in Section VII.

II. Accelerator Challenges

The existing machine learning hardware accelerator faces some challenges in providing a design with the desired performance and cost. The machine learning models are complex, so their hardware implementation is complex and slow. Thus the research direction is to propose a design with less complexity while saving performance and increasing speed. Hardware accelerator challenges are power/energy consumption, throughput, area, speed, learning performance, and resource consumption. Each one is described as follows:

A. Power consumption

In cloud-based Deep Neural Network (DNN) processing, power consumption is a critical factor due to the strict power limits in data centers caused by cooling costs. Additionally, data movement consumes more energy than arithmetic operations like MAC, as capacitance is much higher. Hence, it is crucial to provide comprehensive reporting on not only the energy efficiency and power consumption of the chip but also both the energy efficiency and power consumption associated with off-chip memory. This includes considerations such as DRAM or the frequency of off-chip accesses. By evaluating the energy efficiency and power consumption of the entire system, regardless of the specific memory technology employed, a more holistic assessment can be achieved [43]. Embedded system designers face an increasing challenge in reducing hardware resources and power consumption while maintaining the computational complexity of real-time applications. The weights and intermediate results can be stored in on-chip buffers in some designs to cut down on time spent retrieving data from off-chip memory and the amount of power required to keep the system running. The main issue is to design a hardware accelerator with a light structure to reduce power consumption.

B. Throughput

Obtaining high throughput and low latency concurrently can be challenging depending on the approach taken. Expanding the amount of Process Elements (PEs) can enhance the overall throughput, resulting in an increased number of parallel MAC operations. However, the system's area cost and the area of the PE determine the number of PEs. If the area cost of the system remains constant, Expanding the amount of PEs results in a decrease in the area per PE or a reduction of on-chip storage. This could affect how PEs are utilized. Reducing the logic necessary to send operands to a MAC by using a single piece of logic can decrease the area per PE. The maximum throughput is determined by The amount of PEs and the maximum throughput achievable by a single PE. However, the actual throughput depends on several factors, such as the network architecture, weight and activation sparsity in the DNN model, and batch size. Increasing the batch size can enhance the reuse of data and increase throughput. The hardware's ability to support these approaches while maintaining PE utilization, the number of PEs, or cycles per second determines the overall impact of the DNN model on throughput [43].

C. Area

Machine learning accelerators face a multitude of challenges that can lead to area overhead, such as the need to perform both forward and backward passes without sharing any hardware resources between the two processes. Additionally, implementing the hardware accelerator on the chip can come at a high cost. To address these challenges, it is necessary to simplify complex machine learning models in hardware designs while also optimizing hardware components without sacrificing performance, making them more efficient and cost-effective.

D. Performance

Neural networks face difficulties with throughput due to waiting for the processing unit to finish reading data. To address this issue, improved activation functions are proposed in machine learning accelerator designs to enhance accuracy and performance. Strategies such as pooling, convolutional or kernel processing are used to further improve accuracy. To achieve low latency and high efficiency, the neural network is accelerated using pipeline design and multi-channel parallel processing. The main challenge is to maintain high performance in terms of sensitivity, accuracy, and specificity while avoiding the addition of complex hardware components.

E. Resource Consumption

Reducing hardware resources poses a significant challenge due to the increased computational complexity of real-time applications. To address this challenge, some innovative architectures have been proposed that use a convolutional PE array. This PE array can reuse pixel and weight data effectively. thereby reducing the number of resources consumed while maintaining performance in learning and testing. The basic concept is to reduce the hardware resources without compromising the learning and testing performance of the system. The convolutional PE array architecture exploits the fact that the convolution operation is both data and weight reuse friendly. The array can perform multiple convolution operations simultaneously, and the weights for each convolution operation are stored in a weight buffer. The input pixels are stored in a buffer, which can be accessed multiple times during the computation. The PE array can also incorporate multiple output channels and multiple input channels to handle different types of convolution operations. By efficiently reusing the weight data and input data, the convolutional PE array architecture reduces the number of memory accesses, which results in a decrease in power consumption and hardware resources. This technique enables the hardware designer to attain a balance between performance and hardware resources, which is a critical aspect of designing hardware accelerators for machine learning applications.

F. Speed

The design of neural networks with both high speed and energy efficiency has been a challenging task. This has prompted researchers to explore alternatives to Graphics Processing Units (GPUs) and Central Processing Units (CPUs) for efficient acceleration of the algorithms used in neural network models. Due to the high energy cost per read and write operation and the long access time associated with external memory, a number of systems continue to experience difficulties handling their data loads. Alternate methods first adjust the memory to allow for a bigger data bus or make use of several memories distributed across the system in order to cut down on the overhead of this data movement. Parallel access makes it possible to handle many data streams during a single clock cycle, which both accelerates the system's overall speed and makes better use of its available hardware resources.

III. Hardware Accelerator Systems

Hardware Accelerator systems are specialized hardware devices designed to accelerate the performance of specific tasks. These systems use dedicated hardware components such as FPGAs, ASICs, and GPUs to perform complex computations much faster than traditional CPUs. Hardware accelerator systems are widely used in a variety of industries due to their ability to perform complex computations faster and more efficiently than traditional computing systems. In the finance industry, hardware accelerators are used for a variety of purposes, including high-frequency trading, risk management, and fraud detection. High-frequency trading relies on the ability to make trades within fractions of a second, and hardware accelerators can process vast amounts of data in real time, making them a valuable tool. In healthcare, they can be used to accelerate medical imaging tasks such as MRI and CT scans, drug discovery and development, and genomics research. In scientific research, they can be used to accelerate simulations, modeling, and data analysis tasks. They also find applications in autonomous vehicles, aerospace, and defense industries for tasks such as image processing, sensor data analysis, and control systems. Hardware Accelerator systems are also used in high-performance computing applications such as machine learning, data analytics, and virtual reality [44, 45]. They can greatly improve the performance of computing devices, allowing for faster and more efficient processing of tasks. This can lead to improved productivity and reduced wait times for users. Additionally, hardware accelerator systems can help reduce energy consumption and lower costs. By offloading certain tasks from the CPU and GPU, these systems can operate more efficiently and require less power. This can result in significant savings for both individuals and organizations. The possibilities for the use of Hardware Accelerator systems are endless and are only limited by our imagination. There are various types of hardware accelerator systems available in the market today, each designed to accelerate specific types of tasks. Some of the common types include FPGA-based systems, CPU-based systems, GPU-based systems, and ASIC-based systems. Each type has its unique advantages and disadvantages. FPGA-based systems are highly flexible and can be programmed to perform different tasks. GPU-based systems are ideal for parallel processing and are commonly used in gaming and graphics applications. ASIC-based systems

This article has been accepted for publication in IEEE Transactions on Artificial Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAI.2024.3377147

are highly optimized for specific tasks and offer the highest performance but are expensive to design and manufacture. Fig. 1 shows the hardware accelerators used by all the works examined for this study. As can be shown, 64% of the works used FPGA to implement the DNN networks. While the CPU accounts for 16%, the GPU and ASIC account for 12% and 8%, respectively. Some common characteristics of hardware accelerators include:

A. Central Processing Unit (CPU)

The current CPU can execute SIMD (Single Instruction, Multiple Data) instructions by utilizing multiple ALUs simultaneously. This feature is particularly useful in image processing, as it allows the same instruction to be performed on a continuous stream of data. In computer vision, most operations occur across the image [44, 46].

B. Graphic Processing Unit (GPU):

Compared to general-purpose CPUs, GPUs have developed into a specialized architecture designed for parallel processing, with SIMD instruction extensions that allow for concurrent execution of multiple tasks, including image processing workloads [47]. Fig. 2 shows the GPU architecture, GPUs have processing cores that are far simpler than those of standard high-performance CPUs [46]. These often have simpler control logic because they do not need to predict branching or prefetch data, and constrained memory per core. GPUs can accommodate a much larger number of cores on a single chip than CPUs due to simpler computing cores. GPU architectures work very well in situations where there aren't many or any branching conditions. In addition, GPU architectures include a memory architecture designed specifically for high-speed data streaming for image processing.

C. Field Programmable Gate Array (FPGA)

FPGA is a programmable integrated circuit capable of being reconfigured to perform different circuit functions multiple times. Instead of having a fixed design like a processor, The composition of an FPGA includes Digital Signal Processors (DSPs), Configurable Logic Blocks (CLBs), I/O pads, on-chip Block RAMs (BRAMs), and routing channels as shown in Fig. 3. In FPGA, you can set up your data tracks so that pixels can be directly transferred between computing units and external memory without any intermediary steps. Furthermore, distributed BRAMs can be used to utilize data locality in vision kernels by storing pixels on the chip. Programmers can change the structure of FPGA's hardware to make it into any shape they need. FPGA's fine-grained parallel architecture gives it advantages over GPU. Once the computation clock cycle time has been calculated, the designer can optimize the output mode to minimize the demand for data storage in main memory, thereby decreasing memory reading delays. FPGA programming is powerful, FPGA provides dynamic algorithm reconfiguration with robust reconfigurability. Also, FPGA uses much less power than GPU and works better with the same amount of power, which can help a lot with the processor's problem of getting rid of heat [45, 48].

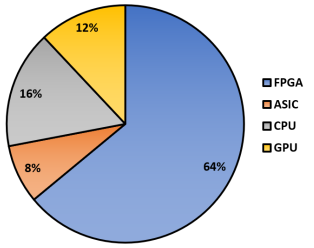

Fig. 1: Hardware Accelerator Systems.

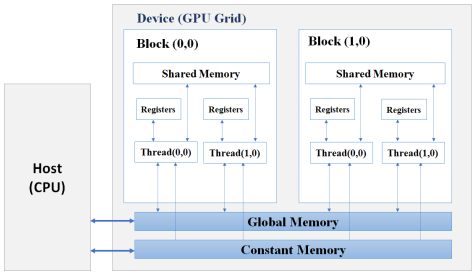

Fig. 2: GPU architecture block diagram.

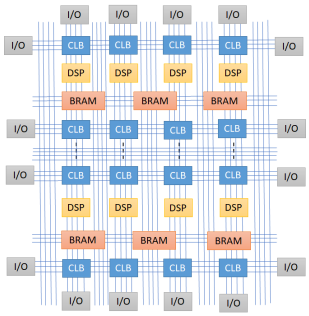

Fig. 3: FPGA architecture block diagram.

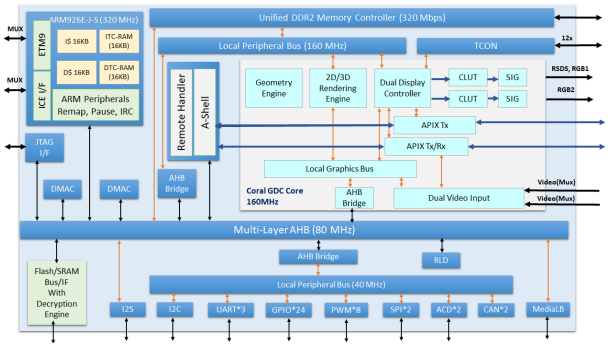

Fig. 4: ASIC block diagram.

D. Application-Specific Integrated Circuit (ASIC)

ASIC is another type of integrated circuit that is designed to perform specific tasks, as opposed to a CPU, which is a general-purpose processor, as shown in Fig. 4 [44]. ASICs are more specialized than GPUs since an ASIC is a processor built to perform a relatively small set of computations, whereas a GPU is still a massively parallel processor with thousands of processing units that can run multiple algorithms [49]. In contrast to FPGA, you cannot reprogram ASIC to do something different once it is required. Its logic has been fixed since it was made, but on FPGA you can make a different design that fits your needs better. ASICs are usually substantially more energy efficient as a result of this specialization.

This article has been accepted for publication in IEEE Transactions on Artificial Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAI.2024.3377147

IV. Models and Datasets

Datasets are essential for determining the accuracy of a DNN. Significant research effort has been expended over the decades to increase the performance of DNNs through innovative architectures. However, the constant need for more accuracy increased new, deeper, and incredibly complex models [37, 47]. In Fig. 5, we demonstrate the most datasets from all the papers examined in our survey; various datasets were utilized to assess the accuracy of the suggested DNN algorithms. There could be multiple datasets for the same work. MNIST, ResNet, CIFAR, and ImageNet are the most popular datasets, as shown. In general, there is a well-balanced distribution of research efforts among CIFAR, ResNet, and ImageNet. While many DNN hardware works are focusing on the MNIST dataset. It is apparent that a significant portion (27%) of the accuracy assessments are carried out on the basic MNIST network. Nonetheless, considerable attention is devoted to sophisticated networks such as the CIFAR (18%) and ImageNet (18%) networks.

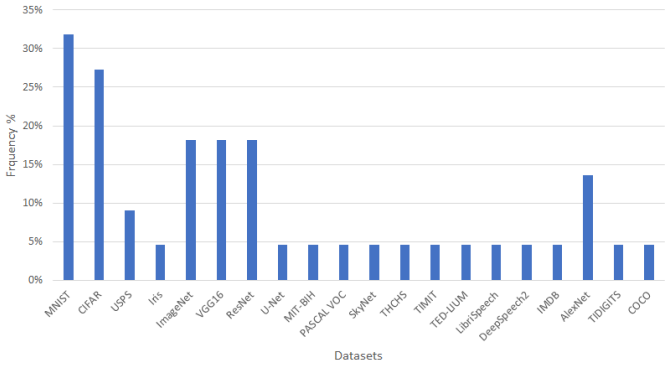

Fig. 5: DNNs models used in the works we reviewed.

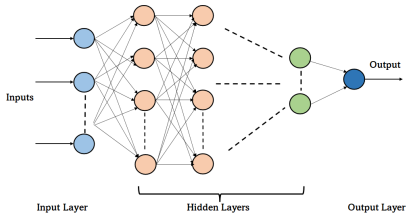

Fig. 6: Block diagram of ANN.

V. Accelerator Approaches

Several machine learning approaches, such as ANN, CNN, and RNN, are implemented on hardware. This section discusses the different machine learning accelerator approaches for each category as follows:

A. Artificial Neural Network

Like the biological neural network in the human body, ANN features a layered architecture in which each network node can process input and forward output to other nodes in the network. The nodes are known as neurons. ANN is comprised of three or more interconnected layers. The first layer contains input neurons that transfer data to the subsequent layers.The output layer produces the final output data. The layers between the input and output layers are hidden and made up of units that adaptively transform the information received from the previous layer through a sequence of transformations. The ANN can understand more complicated objects since each layer works as an input and output layer. The neural layer is the term used to refer to these inner layers together. The units within a neural layer aim to learn from the gathered information by assigning weights based on the internal architecture of the ANN [50–52]. These principles enable units to provide a changed result, which is delivered as an output to the next layer. Fig. 6 presents the general block diagram of the ANN. An adder accumulator receives the product of the input from each node after it has been multiplied by a weight. The result of the adder accumulator is sent to an activation function, which returns the final result. The final output is expressed by equation (1).

$$y_k = f\Big(\sum_{l=0}^{n} W_{lk} * X_l + b_k\Big) \tag{1}$$

Equation (1) represents the final output, where n denotes the total number of neurons. Here, X_l represents the output of the l^{th} neuron in the previous layer, W_{lk} represents the synaptic weight from the l^{th} neuron to the k^{th} neuron in the current layer. Additionally, f denotes the activation function and b represents the bias. ANNs have extensive practical applications, such as speech and image recognition, business intelligence predictive analysis, natural language processing, and many more. ANNs have several advantages, including processing speed, scalability, and accuracy. The most significant advantage of ANNs is their high-speed processing, which can be achieved through parallel implementation.

Khalil *et al.* [53] present an area-efficient ANN implementation. The proposed method utilizes certain layers called hidden layers in a novel way to reduce the number of layers in the ANN by nearly half. The study proposes non-conventional layers called hidden layers. Each of these layers serves two distinct functions through the intelligent use of various weights, which are adaptable. Consequently, each hidden layer performs the tasks of two regular ANN layers. Fixed layers are the other kind of layers that are traditionally used. The fixed layers act as a single layer and are not flexible. The proposed architecture minimizes the number of fixed layers. In addition to the adaptable layers proposed, specific applications may still require one or more fixed layers. The implementation of the proposed method using Verilog HDL and Altera Arria 10 GX FPGA resulted in a 41% reduction in the area compared to the state-of-the-art method, with low overhead in terms of power consumption and speed. The benefits in terms of area reduction and accuracy are substantial.

Huynh *et al.* [54] present a feedforward DNNs accelerator on FPGAs for an efficient hardware implementation architecture. The proposed neural network architecture performs fully connected feedforward DNNs with customizable layers, neurons per layer, and inputs, using only one physical processing layer. The network represents inputs, weights, and outputs in half-precision floating-point number format using 16-bit operands and employs the MNIST database for handwritten digit recognition applications. The performance

evaluation is conducted on two Xilinx FPGA boards, the Virtex-5 XC5VLX-110T and ZynQ-7000 7Z045. The accuracy achieved for the 784-40-40-10 network on the Virtex-5 XC5VLX-110T chip is 97.20%, while for the 784-126-126-10 network on the ZynQ-7000 7Z045 chip, it is 98.16%. The peak performance achieved is 15.81 and 15.90 thousand handwritten image frames per second (kFPS), respectively. Medus *et al.* [55] present novel hardware for any FeedForward Neural Network (FFNN) implementation, including logistic regression, multilayer perceptrons, and autoencoders. The architecture allows for any number of layers, units of layers, and input and output numbers. The hardware utilizes matrix algebra techniques and employs serial-parallel computation. It utilizes a systolic ring of Neural Processing Elements (NPEs) that only requires as many NPEs as neuron units in the largest layer, regardless of the number of layers. The utilization of resources increases linearly as the number of NPEs increases. This adaptable design works as a real-time application accelerator and does not have an impact on the system clock frequency due to its size. In contrast to most existing methods, the proposed approach only utilizes a single Activation Function Block (AFB) for the entire FFNN. Performance, accuracy, and resource usage are evaluated for several activation functions and network topologies. In a Virtex7 FPGA, the architecture operates at 550 MHz clock speed. The proposed approach achieves classification performance comparable to a floating-point approach using an 18-bit fixed point. A smaller weight bit size has no effect on accuracy and allows for more weights in the same memory. A x 256 acceleration was attained for a real-time application of abnormal cardiac detection, and different FFNNs for Iris and MNIST datasets were evaluated.

Khalil *et al.* [56] present a NoC-based adaptive neural network. NoC is composed of routers and PEs, where each router is connected to its PE, consisting of m nodes used for constructing neural network layers. A configuration packet is utilized to specify the number of layers and nodes per layer. The suggested method can allocate multiple routers to represent a layer based on the required configuration. Thus, the proposed solution enables the creation of several layers with flexible node configurations (as required by the application). The method is implemented on FPGA Altera 10 GX, achieving an accuracy of 98.18% with the MNIST dataset. Multiple datasets are used for testing, and the results demonstrate that The suggested approach uses comparable resources as the traditional method. Xiao *et al.* [57] present intrachip and interchip communication strategies for neural network accelerators named NeuronLink. In terms of intrachip communication, this study suggests techniques for route computation parallelization, arbitration interception, and scoring crossbar arbitration to improve virtual-channel routing. These techniques lead to a high-throughput Network on a Chip (NoC) for multicast-based traffic while keeping the hardware cost low. Additionally, this work proposes a lightweight and NoC-aware chip-to-chip interconnection architecture to enable efficient interconnection for neural network processors that use NoCs for interchip communication. B.Zhang *et al.* [58] introduce a Programmable in-Memory Computing Accelerator (PIMCA) for DNN inference with low precision (1-2 b). PIMCA integrates 108 IMC SRAM macros with custom 10T1C bit-cell technology in 28-nm. These macros can store all weights of the targeted 1-b VGG-9 model, eliminating off-chip data movement during DNN inference. The IMC SRAM macros also improve speed and energy efficiency by eliminating row-by-row memory accesses. Additionally, a flexible SIMD processor is integrated to handle non-MAC operations, such as max-pooling, element-wise addition, and residual operations, eliminating data movement energy consumption and latency between the accelerator and host. The test chip prototyped in a 28-nm technology achieves a system-level (macro-level) peak energy efficiency of 437 TOPS/W and a peak throughput of 49 TOPS at 42-MHz clock frequency.

L. Song *et al.* [59] propose HYPAR, a method proposed to ascertain layer-specific parallelism in the training of deep neural networks using an array of DNN accelerators. HYPAR divides the feature map tensors (input and output), kernel tensor, gradient tensor, and error tensors among the DNN accelerators, with each division representing the chosen parallelism for weighted layers. The primary optimization goal is to identify a partition that minimizes overall communication during the complete training of a DNN. Notably, HYPAR demonstrates practicality with a linear time complexity for the partition search. This approach is implemented within the HYPAR architecture, an HMC-based DNN training framework designed to reduce data movement. The results show an average performance increase of 3.39× and an energy efficiency improvement of 1.51× when compared to the default data parallelism. Wei *et al.* [60] introduce a novel approach called Layer-Conscious Memory Hierarchy (LCMH) for DNN accelerators. LCMH dynamically determines the appropriate memory levels for each layer based on their specific requirements for off-chip memory bandwidth and on-chip buffer size. This allows for the avoidance of off-chip memory usage in layers with high memory demands, keeping their data on-chip instead. Experimental results demonstrate that designs implementing layer-conscious memory management achieve a significant speedup of up to 36% compared to designs using Uniform Memory Hierarchy, and a 5% improvement over current state-of-the-art designs.

B. Convolutional Neural Network

CNN is a deep learning model that is used to process data. Its architecture is intended to learn spatial feature hierarchies automatically and adaptively., starting from low-level features and building up to high-level ones. CNNs are made up of a variety of building blocks that help them extract relevant features from input data. [61–64]. As shown in Fig. 7 CNN structures are typically made up of three layers: convolution layers, pooling layers, and fully connected layers. Convolution and pooling layers are used to extract features from input data. These features are then mapped to the final output, which is typically a classification result, using a fully connected layer. A crucial component of CNN is the convolution layer, which comprises a stack of mathematical operations, including the convolution linear operation. Pixel values are recorded in digital images as an array of numbers in a two-dimensional (2D) grid. CNNs use a small parameter grid called

a kernel as an optimizable feature extractor, which is applied at each position in an image. This makes CNNs particularly efficient for image processing. The extracted features can grow hierarchically and become increasingly complicated as the output of one layer is fed into the next. Using an optimization process called backpropagation and gradient descent, training involves improving parameters such as kernels to reduce the disparity between outputs and ground truth labels. CNNs have become widely used in various applications, including image classification, image captioning, object detection, facial recognition, semantic segmentation, and other applications.

To perform a convolution operation in a CNN, we need to find a local receptive field in the input feature map that has the same size as the convolution kernel. Next, multiply the corresponding points by the convolution kernel. Finally, add the offset coefficient to the final result given by equation (2).

$$f = bais + \sum m \sum n(pixel_{mn} \times weight_{mn}) \quad (2)$$

Equation (2) uses m and n to represent the height and width of the feature window, f is a pixel of the feature image generated by the convolutional layer, and bias is a constant value added to each feature window. The resulting values are typically passed through a nonlinear activation function such as the hyperbolic tangent (tanh), Rectified Linear Unit (ReLU), and Sigmoid. When convolution is performed without padding, the output dimension is less than the input dimension. Output size is given by equation (3), which is a function of filter size and stride:

$$D = \frac{H - F + 2P}{S} + 1 \quad (3)$$

In equation (3), H denotes the input size, F denotes the filter size, S denotes the stride size, and P denotes the padding size. The pooling layer is responsible for reducing network complexity, compressing features, and removing redundant information. The size of the final feature map after pooling is determined by the kernels' movement step, which is given by equation (3). The fully connected layer, or Multilayer Perceptron (MLP), is the last layer of a CNN and is composed of layers, each of which has many neurons (nodes). Nodes in one layer are directly connected to nodes in the preceding and subsequent layers. Finally, the fully connected layer is linked to the final output node where classification results are received. A classification function, such as SoftMax, can be utilized at this point. The SoftMax function can be obtained by equation (4).

$$f_x(X_j) = \frac{e^{(x_j)}}{\sum_1^i e^{(x_i)}} \quad for\ j = 1, 2, 3, \ldots\ldots, k. \quad (4)$$

In equation (4), x is the input signal and k is the number of output classes.

Tang *et al.* [65] create an improved CNN image classification model. Maximum pooling is used in the model network structure. The accuracy of the four activation functions of Sigmoid, Tanh, ReLU, and T-ReLU is compared in this paper to improve neural network performance and image classification

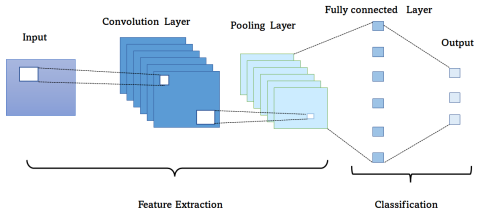

Fig. 7: Block diagram of CNN.

accuracy. The T-ReLU activation function improves the model, raising image classification accuracy from 62% to 76.52%. Khalil *et al.* [66] propose a hardware implementation of a new pooling method, Absolute Average Deviation (AAD), for use in a CNN accelerator. AAD makes use of the spatial proximity of pixels by computing vertical and horizontal deviations, resulting in higher accuracy and lower area and power consumption compared to other pooling methods. The AAD pooling method achieves over 98% accuracy without increasing computational complexity. It was tested using various neural network structures and datasets, including EEG, ImageNet, COCO, and USPS. VHDL was used to implement AAD on Altera Arria10 GX FPGA with 45-nm technology, using Synopsys Design Compiler. Song *et al.* [67] propose a multi-die-based CNN accelerator. The VU9P chip involves three accelerators connected to an independent Super Logic Region (SLR). The host computer manages the three accelerators under the control of the accelerator, which installs one accelerator in each SLR and uses on-chip resources. This system utilizes an 8-bit quantization method to enhance the throughput and computational efficiency of a single Digital Signal Processor (DSP) for accelerating the YOLOv4-tiny algorithm. The design employs a full reuse of feature maps and weights during the calculation process and stores intermediate results in the on-chip buffer to minimize off-chip access, reduce bandwidth pressure, and decrease power consumption. Moreover, a designed instruction group enables the host computer to control the accelerator. This architecture achieves a frame rate of 148.14 Frames Per Second (FPS) and a peak throughput of 2.76 tera operations per second (TOPS) at a frequency of 200MHz with an energy efficiency ratio of 93.15 GOPS/W. It delivers promising results in real-time target detection applications.

Ting *et al.* [68] propose a Batch Normalization (BN) processor that supports training and inference processes. To accelerate CNN training, The proposed work develops an efficient dataflow that incorporates a novel BN processor design as well as processing elements for convolution acceleration. By sharing hardware elements between the two passes, this study took use of the comparable calculations necessary for the BN forward and backward passes, reducing the area overhead. The method completed Automatic Placement & Routing (APR) and post-APR simulation on the training of neural network and functional verification of the BN processor. The method implemented the BN processor in a CMOS technology process. The proposed solution accelerates the CNN training process while saving hardware. The proposed architecture can reduce the total area by 40.13%.

Khabbazan *et al.* [69] describe optimized hardware for

CNNs for use in embedded vision systems. This design method is intended to be applied to low-end hardware with the least resources needed. This hardware proposes a Z-turn evaluation board architecture with a Xilinx Zynq-7000 System-on-Chip (SoC). This architecture optimizes all computations to be 8-bit. Moreover, due to its high-speed performance, low power consumption, and compact size, the architecture is a suitable option for CNN applications that require portability and embedded systems.

Xiao *et al.* [70] present a neural network acceleration architecture that is efficient, scalable, and has low latency and low error rates. The architecture achieves acceleration by utilizing multi-channel parallel computing methods between layers and employing a pipeline design that prioritizes high efficiency and low latency performance requirements. The addition of a line buffer to accommodate varying image widths and the implementation of a selectable convolution kernel size mechanism enhance the network's flexibility and scalability. This proposed neural network performs 32-bit floating-point operations. Since CNNs are based on floating point operations, there will be a loss of precision and time-consuming transformation work if the algorithm's FPGA implementation involves the conversion of floating-point values to fixed-point values. The MNIST dataset is used to perform handwritten number recognition to perform an experimental evaluation of the solution. The acceleration strategy is implemented using the Xilinx zynq-7000 FPGA, and the results of calculating 28*28 handwritten images at a clock frequency of 200M in 25.95us are examined. 98.43% accuracy rate is obtained.

Lee *et al.* [71] present S3NAS, a quick hardware-aware NAS approach. The process is broken down into three stages: supernet design, Single-Path NAS for quick architectural exploration, and scaling and postprocessing. The initial stage involves creating a supernet, which is a set of candidate networks with two main features. Firstly, it allows for varying numbers of blocks in the stages, and secondly, it permits blocks to have parallel layers with different kernel sizes (MixConv). To minimize the hyperparameter search overhead, a differential search can be carried out by extending the Single-Path NAS method to include the MixConv layer and incorporating a loss term that takes into account the latency. The network is scaled to its maximum within the latency constraint using compound scaling as the last step. In the post-processing step, SE blocks and h-swish activation functions are incorporated if they are found to be advantageous. The efficiency of the proposed methodology is demonstrated by tests conducted on four different hardware platforms. Using TPUv3, The search process can be completed within 4 hours, resulting in the discovery of networks that offer superior trade-offs between latency and accuracy compared to state-of-the-art networks. Moreover, this model outperforms other models by 0.6% in terms of accuracy and 14% in terms of speed compared to EfficientNet-B2.

Liu *et al.* [72] propose hardware architecture tailored for streaming applications, with a strong emphasis on increasing computation efficiency by fully accelerating CNNs on FPGAs. To support an inference of CNNs with varied topologies, the architecture integrates most computational functions, convolutional and deconvolutional layers into a single unified module. It efficiently handles concatenative and residual connections between the functions, resulting in highly customized acceleration. This design is further enhanced by utilizing various levels of parallelism, layer fusion, and completely utilizing DSPs. The suggested accelerator has been tested using a variety of benchmark models and implemented on Intel's Arria 10 GX1150 hardware. The results show a high performance of over 1.3 TOP/s throughputs and up to 97% computation efficiency.

Wang *et al.* [73] propose a double buffer memory access structure, which considerably increases the computing unit's memory access efficiency; Furthermore, the proposed architecture utilizes a "ping-pong" buffer structure and employs calculation delay to overlap with memory access delay, resulting in improved acceleration performance. To improve the computation performance of the computing unit, an accelerator structure with a multilevel cache is proposed to execute data preparation reading. To prevent waiting for the processing unit when reading data, a double buffer method is used to wait for the processing unit when reading data; a double buffer method is used to perform calculation and data reading alternately. Based on the experimental results, the proposed accelerator in this paper achieved a detection speed of 15 frames per second (FPS) when processing an input image of size $3 \times 160 \times 320$, while maintaining the same test accuracy as the original design. This signifies a 1.5 times enhancement in acceleration when compared to the original design.

Achararit *et al.* [74] offer an Accuracy-and-Performance-Aware NAS (APNAS) that can efficiently create DNNs. APNAS is based on a weight-sharing and reinforcement learning-based exploration method. First, provide a technique for calculating the cycle count in an RNN such that the network search does not require running a time-consuming hardware simulator. Additionally, they use analytical models for cycle count estimates to speed up the DNN creation process even further. The accuracy of these analytical models is demonstrated by the fact that they provide cycle count estimates that are comparable to those generated by a cycle-accurate hardware simulator. Then, in the RL, establish a reward function by including a configurabexcellentrameter for configuring the trade-off between the performance and accuracy of the generated DNNs. The study showed that APNAS could construct neural network models in 0.55 GPU days on an Nvidia GTX 1080Ti GPU, resulting in an average of 53% fewer cycles when compared to a manually developed neural network model (ResNet) and a state-of-the-art NAS. They generated CNNs by APNAS for two different image classification datasets (CIFAR-10 and CIFAR-100) that required 52.78% and 53.57% fewer cycles compared to a manually designed CNN.

Yuan *et al.* [75] propose hardware-oriented compression and hybrid quantization techniques to reduce the memory requirements of CNNs. They classified all layers as either "no-pruning layers (NP-layers)" or "pruning layers (P-layers)" based on their processing features. The former uses parallel computation for high performance with a regular weights distribution, while the latter has a high compression ratio but is asymmetric due to

pruning. The approach aimed to balance compression ratio and processing efficiency while maintaining reasonable accuracy by using uniform and incremental quantization techniques, as well as a distributed convolutional architecture with multiple parallel Finite Impulse Response (FIR) filters for the regular model in the NP-layers. They introduced a shift-accumulator-based processing element with Activation-driven Data Flow (ADF) for handling the irregular sparse model in the P-layers. They also proposed a Hardware/Algorithm Co-Optimization (HACO) method based on the compression strategy and hardware architecture to implement an NP-P hybrid compressed CNN model on FPGAs. They implemented the compressed VGG-16 model on a Xilinx VCU118 evaluation board for image applications and achieved a compression ratio of 27.5x for a hardware accelerator on a single FPGA chip without the use of off-chip memory, processing 83.0 FPS.

Huang *et al.* [76] propose FPGA-based CNN hardware accelerator design, which utilizes a row-level pipelined streaming technique to calculate Convolution Layers (CLs) using a multi-CE architecture. They also presented a mapping mechanism to optimize the computational resource utilization ratio of the PE Array, achieving up to 98.15%. Additionally, an effective data storage system was implemented to improve the work efficiency of the Computing Engine (CE) by continuously feeding input data. A weighted data allocation technique was proposed to reduce the need for off-chip bandwidth while sacrificing some on-chip storage capacity. The design was tested on XC7VX980T FPGA, achieving 1 TOPS at 150 MHz, which is approximately 98.15% of the theoretical throughput. Moreover, a ResNet-101 accelerator was implemented, achieving 600 GOPS at 100 MHZ with up to 96.12% throughput efficiency. Kim *et al.* [77] present an ASIC accelerator for deep CNNs that uses a novel conditional computing technique to significantly reduce the number of redundant computations and external memory accesses. By combining subsequent max-pooling processes, Precision Cascading (PC) is a novel conditional computing technique that reduces redundant convolution operations. In addition, combining precision-cascading with zero-skipping greatly reduced energy and external memory access. For VGG- 16 CNN for ImageNet, The accelerator achieved peak/average energy efficiency of 8.85/1.22 TOPS/W at a voltage of 0.9V, and low external memory access of 55.31MB or it can be defined as 0.0018 access/MAC. Cheng *et al.* [78] introduce a low-power sparse CNN accelerator featuring a pre-encoding radix-4 Booth multiplier. Leveraging the properties of the radix-4 Booth algorithm, the accelerator reduces the number and bit width of Partial Products (PPs) and encoder power consumption. it incorporates an activation selector module that chooses activations corresponding to non-zero weights for subsequent multiple-add operations after offline encoding of non-zero weights. Additionally, it consolidates eight encoders from relevant multipliers into a single pre-encoding module to save area. The proposed work is developed using the Verilog HDL language and implemented in a 28nm process. The proposed accelerator achieves a performance of 7.0325 TOPS/W with 50% sparsity and scales up to 14.3720 TOPS/W at 87.5% sparsity.

X. Yu *et al.* [79] introduce an FPGA-based acceleration platform utilizing supertile methods tailored for general-purpose CNNs in data center applications. The design of a dispatching-assembling buffering model incorporating broadcast cache sets, tailored for a multi-SU architecture, significantly enhances both reading and writing bandwidth. Additionally, the paper discusses a two-dimensional filter processing unit capable of handling a class of filter-like and pointwise operations, striking a balance between design complexity and performance. The experiment demonstrates that the suggested architecture implemented on KU115 attains the highest peak performance and throughput on FPGAs. Furthermore, it operates at a comparable level to cutting-edge GPUs, but with a latency more than 50 times lower. When compared to the total cost of ownership, the FPGA improves server throughput by 149.2%, albeit with a modest 31.5% increase in costs. R. Hwang *et al.* [80] propose GROW, a Graph convolutional neural network (GCN)accelerator utilizing a Row-stationary SpDeGEMM dataflow. Unlike previous SpDeGEMM accelerators, GROW combines software/hardware architecture co-design to minimize data movements, notably enhancing memory-bound SpDeGEMM performance. It employs a row-stationary dataflow based on Gustavson's algorithm, enabling flexible adaptation to heterogeneous sparsity patterns. Compared to GCNAX, GROW's dataflow significantly reduces memory bandwidth waste, particularly during the aggregation phase. While it introduces a more irregular reuse of dense matrices, GROW employs a graph partitioning algorithm to improve dataflow locality. This is coupled with a multi-row stationary run-ahead execution model, enhancing memory-level parallelism and overall throughput.

C. Recurrent Neural Network

RNNs are a particular class of neural network that function well for processing time series and other sequential data. An RNN has loops that allow information to be stored inside the network. Fig. 8 presents the general block diagram of the RNN. RNNs leverage prior knowledge to make predictions about future events, making them valuable tools for complex task performance by enabling sequence modeling of vectors within the API. RNN can be viewed as a series of interconnected networks. RNNs are an extension of feedforward networks to handle variable-length sequences. Some of the most commonly used recurrent architectures include Gated Recurrent Units (GRUs) and Long Short-Term Memory (LSTM). RNNs are often designed with a chain-like structure, making them well-suited for tasks in natural language processing such as language translation, speech recognition, sentiment analysis, and automatic speech recognition. However, traditional RNNs experienced poor network performance due to vanishing and exploding gradients in applications requiring long-term input data. LSTM is a variation of RNN that has been proposed to address this problem. However, LSTM introduces gating units and many extra parameters, making direct implementation difficult on resource-limited platforms like FPGAs. LSTM block diagram comprises three gates and two activation functions, as shown in Fig. 9. The first gate, known as the "forget gate," is responsible for deciding which portions of the cell state

should be disregarded. The next step involves determining which new data should be stored in the cell. The "input gate" determines the values that need updating, while the "tanh" function generates new potential values. The final layer is the output layer, which determines which data will be sent out. Each part's equation is calculated by equations (5) through (10) [81].

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{5}$$

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{6}$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{7}$$

$$\tilde{c}_t = \tanh(W_c[h_{t-1}, x_t] + b_c) \tag{8}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{9}$$

$$h_t = o_t \odot \tanh(c_t) \tag{10}$$

Where f_t is the result of the forget gate, i_t is the result of the input gate, and o_t is the result of the output gate. Whereas the matrix multiplication $W_f[h_{t-1}, x_t] = W_h h_{t-1} + W_x x_t$. h_t is the cell output, whereas $\tilde{c}$ and c_t denote the new and final states, respectively. The weights of the forget gate, the input gate, and the output gate are, in order, W_f, W_i, W_o. Biases are bf for the forget layer, b_i for the input layer, and b_o for the output layer. The symbol of $\odot$ represents the elementwise (Hadamard) multiplication, σ is the logistic sigmoid function, and $tanh$ is the hyperbolic tangent function.

RNN has several hardware implementations for low cost [82–84]. Khalil *et al.* [28] propose Economic LSTM (ELSTM) is a new LSTM architecture requiring only a single gate. ELSTM requires fewer processing units. Using a single gate, the proposed method can retain memory and learn data sequences. The proposed method offers the advantage of faster learning by utilizing fewer computation units. The accuracy of the proposed method is comparable to that of the other method, while having fewer units, and Compared to LSTM, the proposed method offers a reduction in area and power consumption by 34% and 35%, respectively. This design exhibits notable attractiveness for low-cost hardware applications. The method proposed in this study is evaluated using three datasets: ImageNet, IMDB, and MNIST. The testing and implementation of the proposed method are performed using Altera Arria 10 GX FPGA. Wu *et al.* [85] introduce an energy-efficient scalable processor that leverages the data locality of compressed RNNs. By eliminating redundant connections and sharing quantized values among several weights, the RNN models are significantly compressed. Adopting the quantified sparse matrix encoding significantly reduces repeated calculations and memory operations. Both approaches ensure the suggested design has a high level of energy efficiency. Scalable architecture and network cross-division approach enable hardware parallelism and flexibility. More than 80% of the weight fetching and matrix-vector multiplications for applications like natural language and keyword spotting can be further decreased when using compressed RNNs compared to traditional processors. The peak energy efficiency reaches 3.89 GOPS/mW. It achieves a peak performance of 24 GOPS and dissipates 6.16-mW power with a 1.1 V supply and 200 MHz.

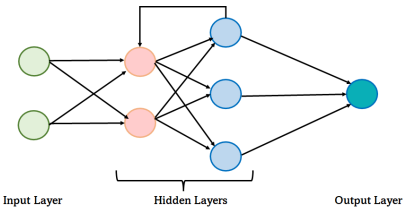

Fig. 8: Block diagram of RNN.

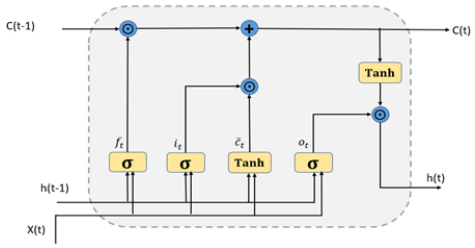

Fig. 9: Block diagram of LSTM.

Kadetotad *et al.* [86] propose LSTM RNN accelerator featuring an accelerator that is based on using a memory compression method known as hierarchical coarse-grain sparsity that was algorithm-hardware co-optimized (HCGS). HCGS offers considerable compression (16x) of LSTM weights with gentle error rate degradation while minimizing index memory cost. The suggested LSTM accelerator utilizes a combination of hierarchical blockwise sparsity and low-precision quantization to store the compressed weights of LSTMs consisting of three layers and 512 cells in only 288 kB of on-chip SRAM. This method effectively reduces the necessary computation by up to 16 times. The prototype chip, fabricated using 65-nm LP CMOS technology, achieves a remarkable energy efficiency of up to 8.93 TOPS/W for real-time speech recognition. Experimental evaluations conducted on TIMIT, TED-LIUM, and LibriSpeech datasets provide solid evidence of the effectiveness and suitability of HCGS across multiple LSTM RNNs. Nan *et al.* [87] present a Hybrid-Iterative Compression (HIC) technique for LSTM/GRU, which separates gating units into error-sensitive and error-insensitive groups and compresses them using different techniques, leveraging the error sensitivity of RNNs. Additionally, a proposed energy-efficient accelerator for bidirectional RNNs is made. In this accelerator, weights are rearranged to optimize data flow in the matrix operation unit based on the block structure matrix (MOU-S). A fine-grained parallelism configuration of matrix-vector multiplications is used to improve BRAM utilization (MVMs). The challenge of load imbalance between MOU-S and the matrix operation unit based on top-k pruning (MOU-P) is effectively addressed through the implementation of the timing matching technique. The architecture of the compressed LSTM/GRU, as proposed, has been thoroughly assessed on the Xilinx ADM-PCIE-7V3 platform. Gao *et al.* [88] propose EdgeDRNN, an RNN accelerator based on the Gated Recurrent Unit (GRU) is optimized for low-latency edge RNN inference with a batch size of 1 while maintaining a lightweight design. To utilize temporal sparsity in RNNs, EdgeDRNN employs a delta network technique inspired by spiking algorithms. The weight

storage of EdgeDRNN is implemented using low-cost off-chip DRAM, and it employs temporal sparsity to decrease memory bandwidth requirements during RNN updates. By employing sparse updates, the memory access to DRAM weight can be reduced by a factor of up to 10. Furthermore, the delta value can be dynamically adjusted to strike a balance between latency and accuracy requirements. This helps optimize EdgeDRNN for efficient edge RNN inference with low latency.

Shan *et al.* [89] introduces Dynamic Recurrent Routing Neural Networks (DRRNets) as a solution to typical RNN problems such as complicated dependencies and gradient vanishing. The suggested DRRNets use the routing pointer matrix's low-rank attribute to construct adaptive routes for diverse dependencies and drastically decrease redundant parameters by discovering low-rank approximations for fully connected layers based on the inner structure of the cell state. The article contains an optimization algorithm for training the network and assesses the model's performance in a variety of tasks, including image classification, language modeling, and speaker recognition. J. Chen *et al.* [90] introduce a specialized hardware accelerator called "Eciton" designed for implementing LSTM neural networks. Eciton showcases the ability to conduct real-time inference for LSTM neural network models of practical size, all while operating within a power constraint of 17 mW. In comparison to FPGA implementations that demand higher power consumption, Eciton delivers competitive performance. This is achieved through the utilization of 8-bit fixed-point weight quantization, hard sigmoid activation functions, and meticulously optimized microarchitecture, effectively minimizing chip resource and memory demands. Although these quantization techniques lead to a slight accuracy reduction of approximately 5% when assessed on real-world predictive maintenance LSTM models consisting of 3 to 4 layers, the advantage of low resource requisites permits Eciton to be accommodated within a cost-effective, low-power Lattice iCE40 UP5K FPGA.

D. Transformer-based and diffusion-based models

Transformer-based models have gained a significant amount of attention in recent years because of their outstanding results on Natural Language Processing (NLP) problems. The transformer architecture was first described in [91] by Vaswani et al. It uses self-attention mechanisms to capture dependencies between different input data elements, enabling parallel processing of sequences and reducing the sequential nature of conventional RNNs. For tasks like Accelerator Optimization, Automatic Machine Learning, and Compiler Optimization [92], transformer-based models can be applied. Diffusion-based models are a type of probabilistic model that propagates information across data points through repetitive processes. These models have found use in a variety of domains, such as image denoising, data imputation, and generation tasks, data-driven accelerator design, and neural architecture search [93, 94]. Zhao *et al.* [95] introduce a transformer accelerator utilizing an Output Block Stationary (OBS) dataflow to optimize memory access and improve DSP utilization, resulting in higher energy efficiency. By minimizing repeated memory access and employing block-level and vector-level broadcasting, the accelerator achieves reduced memory access bandwidth for input and output. The FPGA-based verification of the proposed accelerator demonstrates impressive performance, with a throughput of 728.3 GOPs and an energy efficiency of 58.31 GOPs/W when evaluating a Transformer-in-Transformer (TNT) model. Z. Cheng*et al.* [96] present a novel Transformer-based model which is proposed for signal detection in a Multi-user Molecular Communication (MMC) system. The model is trained using received data generated with varying initial distances between transmitters and receivers. The numerical results demonstrate that the trained Transformer-based model exhibits excellent convergence and outperforms the traditional DNN in terms of signal detection, achieving a lower Bit Error Rate.

E. Large Language Models

A Large Language Model (LLM) is a specialized hardware or software component designed to enhance the performance of large language models in natural language processing tasks. LLMs, such as OpenAI's GPT-3 and BERT, have demonstrated remarkable capabilities in understanding and generating human-like text, but they come with substantial computational requirements, making them resource-intensive and time-consuming to run on standard hardware [97]. These accelerators leverage techniques like parallel processing, optimized memory access, and specialized circuit designs to improve the overall efficiency of language model computations. The LLMs accelerator has become crucial in a wide range of applications, including chatbots, language translation, text summarization, and sentiment analysis [98]. Maddigan*et a.l* [99] introduce an innovative system called Chat2VIS, which harnesses the capabilities of LLMs. Through effective prompt engineering, Chat2VIS demonstrates a more efficient solution for language understanding, resulting in simpler and more accurate end-to-end outcomes compared to previous methods. The research reveals that Chat2VIS, utilizing LLMs and proposed prompts, offers a reliable approach to generating visualizations from natural language queries, even when queries are imprecise or insufficiently specified. Moreover, this solution significantly reduces development costs for Natural Language Interface systems while achieving superior visualization inference abilities compared to traditional NLP approaches that rely on handcrafted grammar rules and tailored models.

F. Performance comparison of different methods

Table I provides a comprehensive summary of various machine learning hardware accelerators, highlighting their key features, performance metrics, and targeted applications. It aims to offer an overview of the latest advancements in ML hardware acceleration, assisting researchers, developers, and technology enthusiasts in understanding the landscape of available solutions and their respective strengths. By analyzing the characteristics and capabilities of different accelerators, readers can make informed decisions regarding the most suitable hardware for their specific ML requirements.

TABLE I: SUMMARY OF MACHINE LEARNING HARDWARE ACCELERATORS

Ref.	Method	Power	Area	Performance	Dataset	Accuracy	Hardware Device
[54]	A fully connected feedforward DNN with a customizable number of layers, neurons per layer, and inputs are performed by the neural network architecture using just one physical processing layer. • Advantages: Adequate recognition performance can be achieved with relatively modest network sizes, resulting in an increased performance while consuming fewer hardware resources and power. • Limitations: Compared to other related works, the performance of floating-point DNN in this architecture on The MNIST dataset exhibits a comparatively lower when compared to fixed-point and binary-based neural networks.	N/A	N/A	• 15.90 Thousand handwritten image Frames Per Second (kFPS)	MNIST	98.16%	FPGA
[69]	An optimized hardware approach for CNNs for embedded vision systems. • Advantages: The presented design exhibits superior performance with significantly fewer DSP48 and BRAM resources and only half of the LUTs in comparison to previous CNN implementations. This accelerator operates at a frequency of 160 MHz and consumes 1.77 watts of power, achieving a performance of 40.96 GOP/s while utilizing only 134 processing units and 601 KB of internal memory. • Limitations: This architecture has less performance, and energy efficiency compared to other related works	1.77 W	N/A	• 40.96 GOPs	AlexNet	N/A	FPGA
[56]	adaptive hardware architecture for neural-network-on-chip • Advantages: With the MNIST dataset, the proposed method achieves an accuracy of 98.18% and can construct networks of various sizes to suit different applications. • Limitations: The proposed method incurs an area overhead of 13% when compared to the state-of-the-art method.	0.97 W	13% reduction	N/A	MNIST and CIFAR	98.18%	FPGA
[66]	Designing a novel hardware implementation AAD pooling for a CNN accelerator • Advantages: AAD pooling technique, which takes into account pixel variations to obtain a highly accurate representation. The hardware implementation achieves excellent separability and high precision. The AAD pooling achieved 98.51% accuracy without increasing computational complexity. • Limitations: Compared to the max and average methods, the proposed method incurs only a minor increase in power consumption and execution time.	0.31 mW	244.46 nm^2	N/A	EEG, ImageNet, Common Objects in Context (COCO), and USPS	98.51%	FPGA
[53]	Reconfigurable hardware design approach for economic neural network • Advantages: The suggested approach's hardware structure consists of a neural network with fewer layers than the state-of-the-art method, resulting in a 41% reduction in area while preserving performance efficiency. Furthermore, the suggested method allows for the configuration and updating of the number of layers in the on-chip design. It can be easily adapted for complex speech recognition and image classification problems. • Limitations: The power consumption of the proposed method is 44 mW, which is slightly higher (9%) than the power consumption of the state-of-the-art methods.	44 mW	41% reduction	N/A	MNIST, CIFAR-10, and USPS	96.8%	FPGA
[71]	Fast hardware-aware neural architecture search methodology • Advantages: The performance of this model exceeds that of the other models, achieving 0.6% higher accuracy and 14 • Limitations: This design has a higher number of parameters and a larger number of FLOPS compared to EfficientNet-B1.	N/A	N/A	• 14% faster than EfficientNet-B2	ImageNet	0.6% higher accuracy than EfficientNet-B2.	CPU, and GPU
[72]	Full-stack acceleration of deep CNNs • Advantages: The proposed method achieves a high level of performance with over 1.3 TOP/s throughput and up to 97% computation efficiency. • Limitations: The proposed method exhibits high resource utilization and compute density, which may lead to a reduced working clock frequency.	VGG16: 17.2W ResNet-50: 19.1W U-Net: 21.5 W	N/A	• 1.3-1.59 TOP/s throughputs and up to 97% computation efficiency	VGG16, ResNet-50, and U-Net	N/A	FPGA

[55]	Systolic parallel hardware architecture for the FPGA acceleration of FFNNs • Advantages: The proposed architecture enables the implementation of a multilayer FFNN with up to 3600 neurons per layer on a single chip, without the need for external memory, achieving a maximum performance of 1980 GOPS. The architecture is designed in a way that it can be easily scaled to larger capacity devices or multi-chip configurations using a simple NPE ring extension. The proposed architecture can adopt any type of FFNN. • Limitations: Compared to some related works, the proposed architecture employs a higher number of DSP blocks and registers.	N/A	N/A	• 1980 Giga operations per second (GOPS)	Iris, MNIST, and MIT-BIH&AHA	98.53%	FPGA
[67]	Design and implementation of CNNs accelerator based on multi-die • Advantages: At a clock frequency of 200MHz, the accelerator architecture can achieve a peak throughput of 2.76 (TOPS) and a frame rate of 148.14 frames per second (FPS). The proposed method has achieved significant improvements in performance, with a threefold increase, and energy efficiency, with a ratio of 93.15 GOPS/W, resulting in excellent real-time target detection results. • Limitations: The proposed method exhibits comparable power consumption to existing accelerators designed for the YOLOv4-tiny algorithm.	12.689 W	N/A	• 148.14 frames per second (FPS) • A peak throughput of 2.76 tera operations per second (TOPS)	PASCAL VOC	N/A	FPGA
[65]	CNN image classification model • Advantages: This model improves the image classification accuracy by 14.52% using the T-ReLU activation function. • Limitations: The model training requires much time. It also requires additional learning and improvement. To improve accuracy, additional fully connected layers can be added to the network or the number of nodes in the existing fully connected layer can be increased. However, the current paper only utilizes a single fully connected layer with 128 nodes and evaluates the network's performance on the small CIFAR-10 dataset.	N/A	N/A	N/A	CIFAR-10	76.52%	N/A
[68]	Batch normalization processor design for CNN training and inference • Advantages: This method eliminates the need for a divider and reduces the area required for the original divider. Also, It can finish normalizing each data set in a single cycle.The proposed architecture offers a significant reduction in the time required for the original division operation while also achieving a 40.13% reduction in the total area. • Limitations: It is not tested using datasets for accuracy verification	N/A	40.13% reduction	N/A	MNIST	N/A	FPGA
[73]	Accelerator structure with a double buffer memory access structure significantly improves memory access efficiency. • Advantages: The design proposed in this paper accelerates the processing by approximately 1.4 times compared to the original design, as measured by the number of cycles required to complete the task. • Limitations: The proposed design shows an increase in resource utilization for BRAM, LUT, and FF compared to the original design.	N/A	N/A	• Detection speed of 15 FPS when the input image is $3 \times 160 \times 320$. • It achieves an acceleration effect of approximately 1.4 times the original design.	SkyNet	N/A	FPGA
[70]	Multi-channel parallel processing across layers and a pipeline design make the neural network acceleration architecture efficient, scalable, low-latency, and low-error. • Advantages: The proposed neural network acceleration architecture results in low latency, high accuracy, and scalability. With a clock frequency of 200M, it can process 28*28 handwritten images in only 25.9us, and the DSP consumption is reduced through a reasonable multiplex method. The network achieves a 98.43% accuracy rate with minimal errors, making it suitable for a wide range of applications. • Limitations: Compared to other previous CNN implementation methods, the utilization of FF in this method is higher and requires more resources.	N/A	N/A	• It takes 25.9us to calculate 28×28 handwritten images at a clock frequency of 200M.	MNIST	98.43%	FPGA

[85]	A novel RNN processor has been proposed that prioritizes energy efficiency by leveraging data locality and network compression techniques through a new quantified sparse matrix encoding format. • Advantages: The proposed processor design utilizes the network cross-division method, allowing for a high degree of flexibility and parallelism in handling various sizes of embedded RNN applications while maintaining scalability. • Limitations: The number of MACs and SRAM units in this design exceeds that of some related works.	6.16 mW	0.65 mm2	• The peak performance is 24 GOPS • The peak energy efficiency reaches 3.89 GOPS/mW	THCHS-30 Chinese speech corpus and a command word library	N/A	ASIC
[86]	An energy-efficient LSTM RNN accelerator with hierarchical coarse-grain sparsity memory compression, an algorithm-hardware co-optimized memory compression method (HCGS). • Advantages: Comparing the hierarchical blockwise sparsity technique to earlier research shows advantageous error rate and memory compression trade-offs. It has a high MAC efficiency reaching 99.66%. • Limitations: It has higher power consumption than some existing methods.	67.3/1.85 mW	7.74 mm2	• 8.93 TOPS/W for two-layer LSTM for TIMIT data set • 7.22/7.24 TOPS/W for three-layer LSTM for TED-LIUM/LibriSpeech data sets	TIMIT, TED-LIUM, and LibriSpeech data sets.	20.6% PER for TIMIT, 21.3% WER for TED-LIUM, and 11.4% WER for LibriSpeech data sets.	N/A
[74]	The new DNN design framework APNAS emphasizes accuracy and efficiency during neural architecture search. • Advantages: APNAS is capable of generating DNNs with fewer parameters (i.e., cycle count) while maintaining relatively high accuracy compared to state-of-the-art NAS techniques. This is achieved by adjusting the weight of the RNN to account for cycle count, allowing APNAS to successfully trade off accuracy and cycle count. • Limitations: This model has less accuracy than other state-of-the-art NAS techniques	N/A	N/A	• It offers an average of 53% fewer cycles than state-of-the-art techniques.	CIFAR-10 and CIFAR-100.	93.75%	FPGA
[75]	Hardware-oriented compression and hybrid quantization techniques require less memory for CNN accelerator. • Advantages: The compressed VGG-16 architecture proposed in the paper achieves high performance without the need for off-chip memory. It can process images at a rate of 83.0 FPS while maintaining the same level of accuracy. The hardware design can be used in various real-time image processing applications that have limited resources. • Limitations: The proposed method exhibits a slightly lower accuracy compared to other state-of-the-art FPGA designs, also it comes with high resource utilization.	N/A	N/A	• 83.0 frames per second (FPS), and a compression ratio of 27.5x is achieved for a hardware accelerator on a single FPGA chip without using off-chip memory.	VGG-16, ResNet-50, and ResNet-152	N/A	GPU
[76]	A novel multi-CE-based row-level pipelined streaming method for calculating convolution layers (CLs). • Advantages: The PE Array exhibits exceptional work efficiency, reaching up to 99.83%, with a remarkable resource utilization ratio of 98.15%. The VGG16 and ResNet-101 accelerators, which utilize this PE Array, achieve throughput efficiencies of 98.15% and 96.12%, respectively, outperforming other existing works. • Limitations: It has a higher number of multiplication than the state-of-the-art methods.	14.36 mW	N/A	• The work efficiency of the PE Array is up to 99.83%. • The VGG16 accelerator on XC7VX980T FPGA, achieves 1 TOPS at 150 MHz. • A ResNet-101 accelerator is also implemented, achieving 600 GOPS at 100 MHZ.	VGG-16, and ResNet-101	N/A	FPGA
[77]	ASIC accelerator for deep CNNs. • Advantages: This work demonstrates significant improvements in DCNN inference throughput and overall system-level energy efficiency, which includes both the accelerator chip and off-chip memory. • Limitations: It has higher on-chip SRAM than the state-of-the-art methods.	203 mW	N/A	• The peak energy efficiency of 8.85 TOPS/W at 0.9V supply • low external memory access of 55.31 MB (or 0.0018 access/MAC) for ImageNet classification with VGG-16 CNN.	ImageNet	N/A	ASIC
[87]	A hybrid-iterative compression (HIC) technique for LSTM/GRU • Advantages: This method achieves a significant compression ratio of 37.1/32.3 for LSTM/GRU with negligible accuracy loss. It also shows improved energy efficiency for LSTM networks (5%-237%) and a 58% improvement for GRU networks. • Limitations: It has lower FPS computation than the state-of-the-art methods.	GRU: 14.906 W Vanilla LSTM: 17.106 W	N/A	• The improvement in the energy efficiency of the LSTM networks is (5%-237%) • A 58% improvement in GRU networks. • It achieves 379,507 FPS • It achieves a latency of 2.635 μs.	DeepSpeech2 (AN4)	N/A	FPGA

[28]	ELSTM architecture, which only requires a single gate, is more suitable for low-cost hardware design due to its simplicity in terms of components. The gate is responsible for data deletion and updating, and its output is fed to three components: the memory layer, the update layer, and the output layer. ELSTM requires fewer processing units compared to traditional LSTM architectures. • Advantages: ELSTM has a low error and faster training, allowing it to achieve high accuracy faster. Compared to LSTM, the proposed method saves 34% of area and 35% of power consumption.	1.192 W	34% reduction	• The proposed method has a latency of 23 ms and a throughput of 258.4 MOPS.	MNIST, IMDB, and ImageNet.	90.89%	FPGA
[57]	NeuronLink is a neural network accelerator communication mechanism that combines intrachip and interchip communication techniques. • Advantages: The Neuron Link-based DNN accelerator proposed in this work outperforms previous NoC-based DNN accelerators in terms of power efficiency, with a 1.27×–1.34× improvement, and area efficiency, with a 2.01×–9.12× improvement. • Limitations: The design requires a significant number of FIFOs to store various packets with different priorities and routers, which increases the demand for BRAMs. [leftmargin=*]	15.4 W	41.2 mm2	• Neuron Link-based DNN accelerator outperforms the previous NoC-based DNN accelerators 1.27×–1.34× in terms of power efficiency and 2.01×–9.12× in terms of area efficiency. [leftmargin=*]	ResNet-18, and AlexNet	N/A	FPGA
[88]	EdgeDRNN is a RNN accelerator optimized for low-latency edge inference and based on a lightweight implementation of Gated Recurrent Unit (GRU) networks. • Advantages: EdgeDRNN achieves an average effective throughput of 20.2GOp/s and a wall plug power efficiency over 4X higher than the commercial edge platforms of AI. • Limitations: It uses off-chip weigh storage	2.3 W	N/A	• An effective throughput is 20.2GOp/s and a wall plug power efficiency is over 4X higher than the commercial edge AI platforms.	TIDIGITS, and SensorsGas.	99%	FPGA
[78]	A low-power sparse CNN accelerator featuring a pre-encoding radix-4 Booth multiplier. • Advantages:This design surpasses others in terms of both area and energy efficiency. • Limitations: The accelerator struggles to achieve high MAC utilization in small-size convolutional layers and fully connected layers.	160.17 mW	0.4839 mm2	• 7.0325 TOPS/W at 50% sparsity • 14.3720 TOPS/W at 87.5%	VGG16, and AlexNet	N/A	FPGA
[89]	Dynamic recurrent routing neural networks (DRRNets) are proposed as a remedy for traditional RNN challenges like intricate dependencies and gradient vanishing. • Advantages: DRRNets utilize fewer parameters than other models in the same domain. This model is the first to use the low-rank property in terms of input structure, and both the decoupling and low-rank features are imposed on fully connected layers.	N/A	N/A	N/A	MNIST and CIFAR10	99%	CPU, and GPU
[58]	A Programmable in-Memory Computing Accelerator (PIMCA) for DNN inference with low precision (1-2 b). • Advantages: By employing this method, a significant reduction of up to 73% in the total program size is achieved, resulting in fewer cycle counts and ultimately leading to improved energy efficiency.	124mW	20.9 mm2	The peak energy efficiency of 437 TOPS/W and a peak throughput of 49 TOPS at 42-MHz clock frequency.	CIFAR-10	1-/1-b VGG-9: 83.20% 1-/1-b ResNet-18: 83.48% 2-/2-b ResNet-18: 86.48%	FPGA
[95]	A transformer accelerator utilizing an OBS dataflow resulting in higher energy efficiency. • Advantages: The proposed OBS dataflow reduces the power consumption of the BRAM, which leads to an overall power reduction of 33%. OBS also lowers the input reading and output writing bandwidth.	80 mW	N/A	Throughput of 728.3 GOPs and an energy efficiency of 58.31 GOPs/W	ImageNet	79.5%	FPGA

VI. Evaluation

An evaluation of a machine learning accelerator is significant for any design for validation. The evaluation is divided into training & testing and hardware evaluations. Each evaluation parameter is described as follows:

A. Training and Testing Evaluation

In machine learning classification models, performance measures are used to evaluate how well the models perform in a specific context. This evaluation helps to improve machine learning classification models. Some of the performance metrics are accuracy, specificity, precision, tension, F1 score (tension), and loss function. Model performance is critical for machine learning because it allows us to understand

the strengths and limitations of these models when making predictions in new situations. True Positives (TPs), True Negatives (TNs), False Positives (FPs), and False Negatives (FNs) are commonly used performance measures for evaluating the performance of classification models. TP refers to the number of correctly predicted positive cases, while TN is the number of correctly predicted negative cases. False Positives (FPs) are the number of negative cases that were incorrectly predicted as positive, and False Negatives (FNs) are the number of positive cases that were incorrectly predicted as negative. These metrics are typically used to calculate other performance measures, such as accuracy, precision, recall, and F1 score. [66, 100]. The evaluation parameters are given by equations (11) – (20).

1) Accuracy: A test's accuracy is measured by its ability to differentiate classes accurately [55, 66, 100]. It indicates the quality of the result for a given task. Accuracy can be calculated using the following equation:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{11}$$

More complex DNN models typically require more computations and more memory resources to process the input data, which can lead to slower processing times and higher resource utilization. This is especially true for hardware implementations of DNNs, where the processing capabilities and resources are more limited compared to software implementations. As a result, there is often a trade-off between model complexity, accuracy, hardware performance, and efficiency.[43].

2) Sensitivity: It can be defined as a true positive rate that measures the ratio between the number of classes that were correctly identified and the total number of TPs and false negatives [55, 66, 100]. Sensitivity is by equation (12).

$$Sensitivity = \frac{TP}{TP + FN} \tag{12}$$

3) Precision: It is a positive predictive value. It measures the proportion of true positive predictions to total positive predictions produced by the model. A high precision indicates that the model is good at avoiding false positives. [55, 66, 100]. Precision is by equation (13).

$$Precision = \frac{TP}{TP + FP} \tag{13}$$

4) Specificity: It calculates the percentage of actual negative cases that the classifier correctly classifies as negative. It is often referred to as the true negative rate. [30][34]. Specificity is calculated by equation (14).

$$Specificity = \frac{TN}{TN + FP} \tag{14}$$

5) F1 Score or Tension: It measures the balanced relationship between sensitivity and precision [34]. F1 Score is calculated by equation (15).

$$F1\ Score\ (Tension) = \frac{2 * Sensitivity * Precision}{Sensitivity + Precision} \tag{15}$$

6) Loss Function: The evaluation of how well an algorithm models a dataset involves a mathematical function that depends on the machine learning algorithm's parameters. This function, known as a loss function, plays a crucial role in the training process and the results obtained from any deep learning methodology. Loss functions are typically categorized as either Regression loss or Classification Loss. Regression Loss Functions used in regression neural networks predict an output value from an input value rather than pre-selected labels such as Mean Squared Error and Mean Absolute Error. On the other hand, classification neural networks use classification loss functions, which allow selecting a category with the highest probability of the input belonging to it, such as Binary Cross-Entropy and Categorical Cross-Entropy. Each one is described as follows:

- **Mean Squared Error (MSE):** It is also known as $L2$ Loss. MSE calculates the average of the squared differences between the predicted and actual values across the entire dataset. MSE is calculated as follows:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{16}$$

  MSE is sensitive towards outliers; given multiple examples with the same input feature values, the ideal prediction is the mean target value. This function is ideal for calculating loss due to its many features. The difference is squared. Thus the predicted value might be above or below the target value, but big errors are penalized. MSE is a convex function with a global minimum, making gradient descent optimization easier to use to select weight values.
- **Mean Absolute Error (MAE):** MAE is also known as $L1$ loss. MAE represents the difference between the target and predicted values extracted by averaging the absolute difference over the data set. MAE is calculated as follows:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{17}$$

  MAE is a robust metric that is not significantly affected by outliers. In cases where multiple samples have the same input feature values, MAE chooses the median target value as the best prediction. Compare this to MSE, where the mean represents the best prediction. MAE's limitation is that its gradient magnitude depends only on the sign of the difference between the predicted and actual values, not the error size. This results in large gradient magnitudes even for small errors, which can lead to convergence problems. Because of this, a loss function is known as a Huber Loss was developed. This loss function combines the benefits of MSE and MAE into a single package. We can define it using the following function:

$$Huber\ Loss = \begin{cases} \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 & \text{if } |y_i - \hat{y}_i| \leq \delta \\ \frac{1}{n}\sum_{i=1}^{n}\delta(|y_i - \hat{y}_i| - \frac{1}{2}\delta) & \text{if } |y_i - \hat{y}_i| > \delta \end{cases} \tag{18}$$

In equation (18), (δ) delta hyperparameter defines the range for MAE and MSE.

- **Binary Cross-Entropy (Log Loss):** Cross-Entropy loss is also called logarithmic loss, log loss, or logistic loss. This is the loss function used in binary classification models, which takes in an input and should classify it into one of two pre-defined categories. Classification neural networks output a vector of probabilities, the probability that the input fits into each pre-set category, and pick the category with the highest probability as the final output.

$$CE\ Loss = -\frac{1}{n}\sum_{i=1}^{N}(y_i.log(p_i)) + (1 - y_i).log(1 - p_i) \tag{19}$$

In binary classification, the actual value of y can only be 0 or 1. To accurately determine the loss between actual and predicted values, it is necessary to compare the actual value $(0or1)$ to the probability that the input aligns with that category ($p(i)$ = probability that the category is 1; $1 - p(i)$ = probability that the category is 0).

- **Categorical Cross-Entropy:** In multi-class classification tasks, where an example can only belong to one of several possible categories, a categorical cross-entropy is commonly used. This function is designed to measure the difference between two probability distributions. We use categorical cross-entropy when the number of classes is more than two. Binary cross-entropy is a special case of categorical cross-entropy, where $M = 2$, and M is the number of categories.

$$CE = -\frac{1}{n}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}.log(p_i j) \tag{20}$$

B. Hardware Evaluation

Various metrics can be used to evaluate hardware systems, such as power consumption, area, throughput, and latency. These metrics are helpful in comparing and assessing the advantages and limitations of different designs.

1) Energy Efficiency and Power Consumption: The efficiency of energy usage is a measure of the amount of data that can be processed or the number of tasks that can be performed per unit of energy. This is particularly important when processing DNNs on embedded devices at the edge. Power consumption is the amount of energy consumed during a given period. The Thermal Design Power (TDP) is a design criterion that determines the maximum power consumption, which is the amount of power that the cooling system can dissipate due to increased power consumption.

2) Area: The size of each PE and the total area cost of the system together determine the optimal number of PEs. If the area cost of the system remains the same, increasing the number of PEs will necessitate either decreasing the amount of space required for each PE or exchanging some of the on-chip storage areas for additional PEs. However, decreasing the amount of storage on-chip can have an impact on how PEs are utilized. You can also reduce the area per PE by reducing the logic needed to send operands to a MAC [43].

3) Throughput: Throughput refers to the amount of data that can be transmitted or processed within a specific time frame. It is a key performance metric used to evaluate the efficiency and performance of network connections or data processing systems, as it indicates how many packets or messages can successfully reach their destination. Throughput is commonly measured in bits per second (bps) and is often expressed in units of megabits per second (Mbps) or gigabits per second (Gbps). A higher throughput indicates a more efficient network or system, while a lower throughput can indicate performance issues or bottlenecks. [32] [44].

4) Latency: It indicates how long it takes for packets to reach their destination. In a network, the way throughput and latency work are directly related. Applications that require real-time interaction, such as augmented reality, autonomous navigation, and robotics, require low latency in order to work correctly. Throughput and latency frequently tend to dissipate due to the maximum throughput of a conversation being determined by the level of latency. Conversations are data exchanges from one point to another. Thus, depending on the approach, achieving high throughput and low latency simultaneously can sometimes be incompatible, and both metrics should be reported [43]. Latency is measured in milliseconds (ms).

5) Analysis: Our AI accelerator survey begins with power usage and throughput comparisons. In Fig. 10, a comprehensive examination of power consumption, quantified in watts, is juxtaposed against the frequency of operations executed per second, measured in Giga Operations per Second (GOPS). As part of our investigation, we derived throughput figures by the multiplication of power and power efficiency for specific articles. The observed trend reveals that contemporary accelerators predominantly align with the throughput trendline situated at 1 Tera operations per second (TOPS). Notably, accelerators with a low-power design exhibit a discernible pattern: their power consumption typically exceeds the threshold of 0.1 watts, while simultaneously showcasing a throughput surpassing 1 GOPS. It's worth highlighting that only a limited number of accelerators fall beneath these specified benchmarks.

Fig. 11 presents each accelerator's power efficiency together with the year that it was first published. We calculate the power efficiency of those papers that did not achieve it by dividing throughput by power. In the previous two years, the power efficiency of AI accelerators has ranged from a minimum of more than 50 GOPS/W to a maximum of more than 70 TOPS/W. In the surveyed accelerators, we observe that FPGA implementations have higher power efficiency than other implementations. According to the data, no major new developments have been produced that significantly affect power, power efficiency, or throughput when compared to previous years.

This article has been accepted for publication in IEEE Transactions on Artificial Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAI.2024.3377147

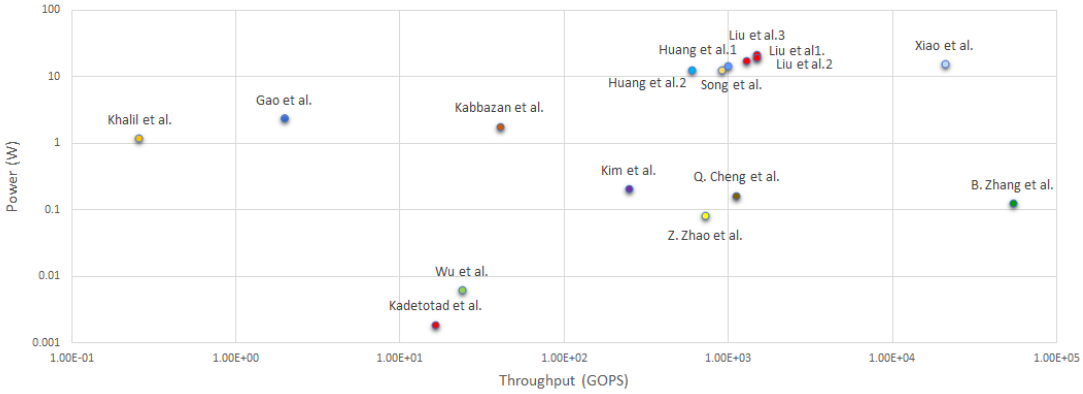

Fig. 10: Performance analysis of the existing methods.

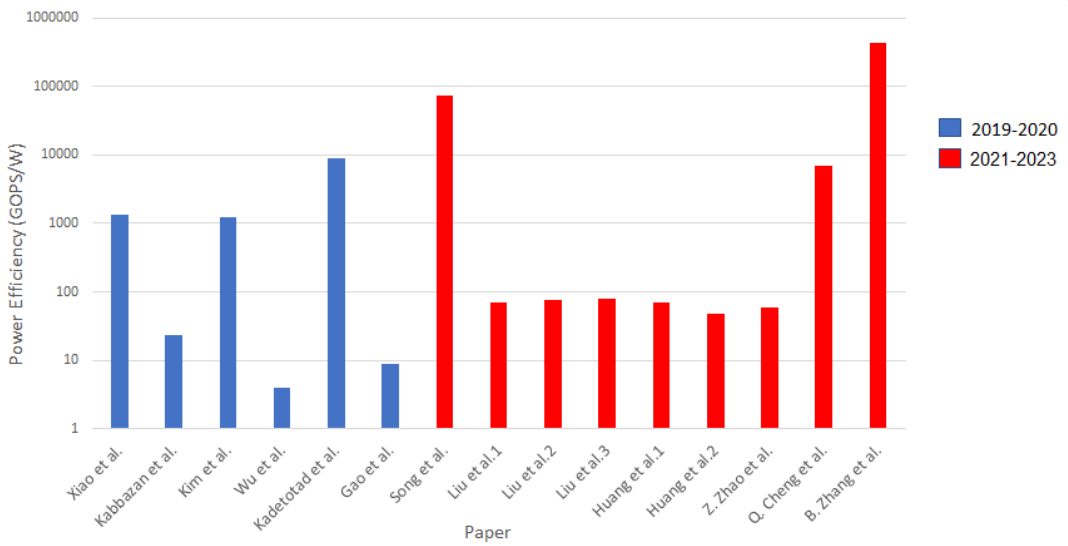

Fig. 11: Comparison between the recent methods.

C. Future Machine Learning Accelerator Designs

Future machine learning accelerator designs face several challenges as AI applications continue to grow in complexity and scale. Here are some insights and suggestions to address these challenges:

- **Leveraging reconfigurable designs:** Reconfigurable designs with optimization strategies such as parallel processing, dynamic resource allocation, and area optimization, it becomes possible to increase the speed of machine learning accelerators while minimizing costs and maintaining flexibility to adapt to varying workloads and applications. The reconfigurable designs proposed encompass nodes that possess the ability to seamlessly transition between different layers, thereby heightening network speed and achieving specific performance objectives. This adaptability empowers optimization by enabling the configuration and updating of on-chip layer quantities, offering a versatile approach to resource allocation. Furthermore, a reduction in the number of adders and multipliers is integrated, leading to a decrease in computational operations. The successful integration of these design elements yields a solution that excels in both efficiency and resource utilization.
- **Power Efficiency:** Energy consumption is a major concern for AI systems, especially in mobile and edge computing scenarios. Improving power efficiency through techniques like quantization, parsity, and specialized memory architectures will be vital. Data reuse is an effective approach for reducing the energy consumption of data transfer. This requires moving data once from a remote, large memory source (such as an off-chip DRAM) and then using it for multiple operations from a nearby, smaller memory location (such as an on-chip buffer or a PE's scratchpad). The optimization of data movement holds substantial importance in the overall design of DNN processors. Furthermore, by reducing the number of adders and multipliers, the system executes fewer computational operations, leading to decreased energy consumption.
- **Model Size and Complexity:** State-of-the-art AI models are becoming larger and more complex, demanding significant computational power and memory resources. Future accelerators need to be scalable to handle these large models efficiently. The optimization involves condensing layers, such as combining two layers to function as effectively as four, thereby enhancing performance.

Additionally, simplifying units and reducing the number of pooling layers results in a reduced overall area footprint.

- **Diverse Workloads:** With the increasing diversity of AI workloads, designing accelerators that can efficiently handle various tasks is essential. Addressing the diversity of AI workloads requires a multifaceted approach that encompasses various strategies and methodologies like quantization (reducing the precision of weights and activations) and pruning (removing less significant connections) to reduce the computational requirements of AI models. This can make them more versatile and adaptable to different workloads.
- **Real-time Inference:** Many AI applications require real-time or low-latency replies. Future accelerators must face the challenge of offering rapid inference while maintaining high accuracy, especially in time-sensitive fields such as autonomous vehicles and robotics. Implementing parallel processing techniques to execute multiple tasks simultaneously. This can lead to substantial reductions in response times for AI applications. Also, utilizing edge devices with processing capabilities to perform computations locally reduces the need for data to be sent back and forth to a centralized server.
- **Bottlenecks in Data transfer:** Addressing memory access and data transfer bottlenecks in the accelerator can be achieved through various strategies. Reusing data in calculations helps minimize the need for frequent data transfers. Processing data in batches reduces the frequency of memory access and transfers, thereby enhancing overall efficiency. Additionally, employing cache memory to store frequently accessed data mitigates the impact of slow memory access. Data compression algorithms can also be employed to reduce the volume of data transferred, leading to improved performance. Utilizing Direct Memory Access (DMA) controllers allows for the offloading of data transfer tasks from the CPU, enabling it to focus on computation. Furthermore, structuring algorithms and data layouts to enhance spatial locality can further reduce the frequency of memory accesses. These combined approaches can effectively alleviate memory access and data transfer bottlenecks, ultimately enhancing the performance of the accelerator.
- **Hardware-Software Co-Design:** Tight collaboration between hardware and software teams is required to extract maximum performance from accelerators. Co-design efforts can result in improved hardware-software integration and targeted optimizations. Future AI accelerators may increasingly adopt neuromorphic computing principles, mimicking the brain's architecture. Also, quantum computing advances, Algorithms, and software frameworks will need to be tailored for quantum hardware.
- **Heterogeneous Computing:** To strike a balance between performance and energy efficiency, heterogeneous computing designs incorporating several types of accelerators (e.g., CPUs, GPUs, TPUs) may become more common. Each type of processor can be optimized for specific types of computations.

Addressing these challenges would necessitate ongoing research and development in both the hardware and software areas. Collaboration between academia, industry, and the open-source community will be critical to advancing machine learning accelerator designs that match the needs of tomorrow's AI landscape.

VII. Conclusion

Machine learning is involved in most of the current domains such as IoT environment and biomedical systems. The main challenge is to design a machine learning hardware accelerator with high speed and performance at a low cost. This paper investigated different hardware accelerator structures: ANN, CNN, and RNN. It described the existing approaches with a comparison that shows the features and limitations of each method. This paper also presented the current challenges for designing machine learning accelerators. We highlighted the evaluation parameters of both the learning and hardware sides such as accuracy, sensitivity, area, speed, throughput, and energy consumption. Thus, this paper presented a complete survey on machine learning hardware accelerators to help new researchers and designers in the field. For future research, the hardware accelerator can have the reconfiguration features to be suitable for multiple applications. The reconfiguration process can be done online based on application criteria. Also, a hardware accelerator might be implemented using mixed circuits to have both benefits of analog and digital designs. Furthermore, some hardware components can be shared to support multiple operations to save area on a chip.

References

[1] A. B. Nassif, I. Shahin, I. Attili, M. Azzeh, and K. Shaalan, "Speech recognition using deep neural networks: A systematic review," *IEEE access*, vol. 7, pp. 19 143–19 165, 2019.

[2] S. Dua, S. S. Kumar, Y. Albagory, R. Ramalingam, A. Dumka, R. Singh, M. Rashid, A. Gehlot, S. S. Alshamrani, and A. S. AlGhamdi, "Developing a speech recognition system for recognizing tonal speech signals using a convolutional neural network," *Applied Sciences*, vol. 12, no. 12, p. 6223, 2022.

[3] M. Chun, H. Jeong, H. Lee, T. Yoo, and H. Jung, "Development of korean food image classification model using public food image dataset and deep learning methods," *IEEE Access*, vol. 10, pp. 128 732–128 741, 2022.

[4] C. T. Sari and C. Gunduz-Demir, "Unsupervised feature extraction via deep learning for histopathological classification of colon tissue images," *IEEE transactions on medical imaging*, vol. 38, no. 5, pp. 1139–1149, 2018.

[5] K. Khalil, O. Eldash, A. Kumar, and M. Bayoumi, "Machine learning-based approach for hardware faults prediction," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 11, pp. 3880–3892, 2020.

[6] R. Malhotra, "A systematic review of machine learning techniques for software fault prediction," *Applied Soft Computing*, vol. 27, pp. 504–518, 2015.

[7] K. Khalil, O. Eldash, A. Kumar, and M. Bayoumi, "Intelligent fault-prediction assisted self-healing for embryonic hardware," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 14, no. 4, pp. 852–866, 2020.

[8] L.-Q. Zuo, H.-M. Sun, Q.-C. Mao, R. Qi, and R.-S. Jia, "Natural scene text recognition based on encoder-decoder framework," *IEEE Access*, vol. 7, pp. 62 616–62 623, 2019.

[9] Y. Xu, Y. Wang, W. Zhou, Y. Wang, Z. Yang, and X. Bai, "Textfield: Learning a deep direction field for irregular scene text detection," *IEEE Transactions on Image Processing*, vol. 28, no. 11, pp. 5566–5579, 2019.

[10] U. P. Singh, S. S. Chouhan, S. Jain, and S. Jain, "Multilayer convolution neural network for the classification of mango leaves infected by anthracnose disease," *IEEE Access*, vol. 7, pp. 43 721–43 729, 2019.

[11] K. Li, J. Daniels, C. Liu, P. Herrero, and P. Georgiou, "Convolutional recurrent neural networks for glucose prediction," *IEEE journal of biomedical and health informatics*, vol. 24, no. 2, pp. 603–613, 2019.

[12] C. N. Freitas, F. R. Cordeiro, and V. Macario, "Myfood: A food segmentation and classification system to aid nutritional monitoring," in *2020 33rd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*. IEEE, 2020, pp. 234–239.

[13] H. Jelodar, Y. Wang, R. Orji, and S. Huang, "Deep sentiment classification and topic discovery on novel coronavirus or covid-19 online discussions: Nlp using

lstm recurrent neural network approach," *IEEE Journal of Biomedical and Health Informatics*, vol. 24, no. 10, pp. 2733–2742, 2020.

[14] M. Li, W. Hsu, X. Xie, J. Cong, and W. Gao, "Sacnn: Self-attention convolutional neural network for low-dose ct denoising with self-supervised perceptual loss network," *IEEE transactions on medical imaging*, vol. 39, no. 7, pp. 2289–2301, 2020.

[15] B. Dey, S. Halder, K. Khalil, G. Lorusso, J. Severi, P. Leray, and M. A. Bayoumi, "Sem image denoising with unsupervised machine learning for better defect inspection and metrology," in *Metrology, Inspection, and Process Control for Semiconductor Manufacturing XXXV*, vol. 11611. SPIE, 2021, pp. 245–254.

[16] B. Dey, S. Wu, S. Das, K. Khalil, S. Halder, P. Leray, S. Bhamidipati, K. Ahi, M. Pereira, G. Fenger *et al.*, "Unsupervised machine learning based sem image denoising for robust contour detection," in *International Conference on Extreme Ultraviolet Lithography 2021*, vol. 11854. SPIE, 2021, pp. 88–102.

[17] Y. Liu, M. Jin, S. Pan, C. Zhou, Y. Zheng, F. Xia, and P. Yu, "Graph self-supervised learning: A survey," *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[18] X. Wang, D. Kihara, J. Luo, and G.-J. Qi, "Enaet: A self-trained framework for semi-supervised and supervised learning with ensemble transformations," *IEEE Transactions on Image Processing*, vol. 30, pp. 1639–1647, 2020.

[19] S. Ahmed, Y. Lee, S.-H. Hyun, and I. Koo, "Unsupervised machine learning-based detection of covert data integrity assault in smart grid networks utilizing isolation forest," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 10, pp. 2765–2777, 2019.

[20] A. Uprety and D. B. Rawat, "Reinforcement learning for iot security: A comprehensive survey," *IEEE Internet of Things Journal*, vol. 8, no. 11, pp. 8693–8706, 2020.

[21] H. Xu, A. D. Domínguez-García, and P. W. Sauer, "Optimal tap setting of voltage regulation transformers using batch reinforcement learning," *IEEE Transactions on Power Systems*, vol. 35, no. 3, pp. 1990–2001, 2019.

[22] M. Saharkhizan, A. Azmoodeh, A. Dehghantanha, K.-K. R. Choo, and R. M. Parizi, "An ensemble of deep recurrent neural networks for detecting iot cyber attacks using network traffic," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8852–8859, 2020.

[23] P. Goswami, A. Mukherjee, M. Maiti, S. K. S. Tyagi, and L. Yang, "A neural-network-based optimal resource allocation method for secure iiot network," *IEEE Internet of Things Journal*, vol. 9, no. 4, pp. 2538–2544, 2021.

[24] M. Woźniak, J. Siłka, M. Wieczorek, and M. Alrashoud, "Recurrent neural network model for iot and networking malware threat detection," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 8, pp. 5583–5594, 2020.

[25] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: analysis, applications, and prospects," *IEEE transactions on neural networks and learning systems*, 2021.

[26] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, and D. J. Inman, "1d convolutional neural networks and applications: A survey," *Mechanical systems and signal processing*, vol. 151, p. 107398, 2021.

[27] V. Veerasamy, N. I. A. Wahab, M. L. Othman, S. Padmanaban, K. Sekar, R. Ramachandran, H. Hizam, A. Vinayagam, and M. Z. Islam, "Lstm recurrent neural network classifier for high impedance fault detection in solar pv integrated power system," *IEEE Access*, vol. 9, pp. 32 672–32 687, 2021.

[28] K. Khalil, O. Eldash, A. Kumar, and M. Bayoumi, "Economic lstm approach for recurrent neural networks," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 66, no. 11, pp. 1885–1889, 2019.

[29] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, A. M. Umar, O. U. Linus, H. Arshad, A. A. Kazaure, U. Gana, and M. U. Kiru, "Comprehensive review of artificial neural network applications to pattern recognition," *IEEE Access*, vol. 7, pp. 158 820–158 846, 2019.

[30] 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.

[31] K. Khalil, O. Eldash, B. Dey, A. Kumar, and M. Bayoumi, "Architecture of a novel low-cost hardware neural network," in *2020 IEEE 63rd International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2020, pp. 1060–1063.

[32] E. Wang, J. J. Davis, R. Zhao, H.-C. Ng, X. Niu, W. Luk, P. Y. Cheung, and G. A. Constantinides, "Deep neural network approximation for custom hardware: Where we've been, where we're going," *ACM Computing Surveys (CSUR)*, vol. 52, no. 2, pp. 1–39, 2019.

[33] K. Khalil, B. Dey, M. Abdelrehim, A. Kumar, and M. Bayoumi, "An efficient reconfigurable neural network on chip," in *2021 28th IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*. IEEE, 2021, pp. 1–4.

[34] K. Khalil, O. Eldash, A. Kumar, and M. Bayoumi, "N 2 oc: Neural-network-on-chip architecture," in *2019 32nd IEEE International System-on-Chip Conference (SOCC)*. IEEE, 2019, pp. 272–277.

[35] 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.

[36] M. A. Rajput, S. Alyami, Q. A. Ahmed, H. Alshahrani, Y. Asiri, and A. Shaikh, "Improved learning-based design space exploration for approximate instance generation," *IEEE Access*, vol. 11, pp. 18 291–18 299, 2023.

[37] G. Armeniakos, G. Zervakis, D. Soudris, and J. Henkel, "Hardware approximate techniques for deep neural network accelerators: A survey," *ACM Computing Surveys*, vol. 55, no. 4, pp. 1–36, 2022.

[38] K. Khalil, A. Kumar, and M. Bayoumi, "Low-power convolutional neural network accelerator on fpga," in *2023 IEEE 5th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. IEEE, 2023, pp. 1–5.

[39] C. Åleskog, H. Grahn, and A. Borg, "Recent developments in low-power ai accelerators: A survey," *Algorithms*, vol. 15, no. 11, p. 419, 2022.

[40] M. Giordano, L. Piccinelli, and M. Magno, "Survey and comparison of milliwatts micro controllers for tiny machine learning at the edge," in *2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. IEEE, 2022, pp. 94–97.

[41] S. S. Saha, S. S. Sandha, and M. Srivastava, "Machine learning for microcontroller-class hardware-a review," *IEEE Sensors Journal*, 2022.

[42] K. Khalil, T. Mohaidat, and M. Bayoumi, "Low-cost hardware design approach for long short-term memory (lstm)," in *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2023, pp. 1–5.

[43] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "How to evaluate deep neural network processors: Tops/w (alone) considered harmful," *IEEE Solid-State Circuits Magazine*, vol. 12, no. 3, pp. 28–41, 2020.

[44] N. Gupta, "Chapter one - introduction to hardware accelerator systems for artificial intelligence and machine learning," in *Hardware Accelerator Systems for Artificial Intelligence and Machine Learning*, ser. Advances in Computers, S. Kim and G. C. Deka, Eds. Elsevier, 2021, vol. 122, pp. 1–21. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0065245820300541

[45] A. Reuther, P. Michaleas, M. Jones, V. Gadepally, S. Samsi, and J. Kepner, "Survey of machine learning accelerators," in *2020 IEEE high performance extreme computing conference (HPEC)*. IEEE, 2020, pp. 1–12.

[46] M. F. Hashmi, R. Pal, R. Saxena, and A. G. Keskar, "A new approach for real time object detection and tracking on high resolution and multi-camera surveillance videos using gpu," *Journal of Central South University*, vol. 23, pp. 130–144, 2016.

[47] M. Capra, B. Bussolino, A. Marchisio, G. Masera, M. Martina, and M. Shafique, "Hardware and software optimizations for accelerating deep neural networks: Survey of current trends, challenges, and the road ahead," *IEEE Access*, vol. 8, pp. 225 134–225 180, 2020.

[48] Z. Qi, W. Chen, R. A. Naqvi, and K. Siddique, "Designing deep learning hardware accelerator and efficiency evaluation," *Computational Intelligence and Neuroscience*, vol. 2022, 2022.

[49] S. Bavikadi, A. Dhavlle, A. Ganguly, A. Haridass, H. Hendy, C. Merkel, V. J. Reddi, P. R. Sutradhar, A. Joseph, and S. M. P. Dinakarrao, "A survey on machine learning accelerators and evolutionary hardware platforms," *IEEE Design & Test*, vol. 39, no. 3, pp. 91–116, 2022.

[50] Z. Zhang, K. Zhang, and A. Khelifi, *Multivariate time series analysis in climate and environmental research*. Springer, 2018.

[51] B. Dey, K. Khalil, A. Kumar, and M. Bayoumi, "A reversible-logic based architecture for artificial neural network," in *2020 IEEE 63rd International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2020, pp. 505–508.

[52] K. Khalil, O. Eldash, A. Kumar, and M. Bayoumi, "Self-healing approach for hardware neural network architecture," in *2019 IEEE 62nd International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2019, pp. 622–625.

[53] K. Khalil, A. Kumar, and M. Bayoumi, "Reconfigurable hardware design approach for economic neural network," *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2022.

[54] T. V. Huynh, "Deep neural network accelerator based on fpga," in *2017 4th NAFOSTED Conference on Information and Computer Science*. IEEE, 2017, pp. 254–257.

[55] L. D. Medus, T. Iakymchuk, J. V. Frances-Villora, M. Bataller-Mompeán, and A. Rosado-Muñoz, "A novel systolic parallel hardware architecture for the fpga acceleration of feedforward neural networks," *IEEE Access*, vol. 7, pp. 76 084–76 103, 2019.

[56] K. Khalil, B. Dey, A. Kumar, and M. Bayoumi, "Adaptive hardware architecture for neural-network-on-chip," in *2022 IEEE 65th International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2022, pp. 1–4.

[57] S. Xiao, Y. Guo, W. Liao, H. Deng, Y. Luo, H. Zheng, J. Wang, C. Li, G. Li, and Z. Yu, "Neuronlink: An efficient chip-to-chip interconnect for large-scale neural network accelerators," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 9, pp. 1966–1978, 2020.

[58] B. Zhang, S. Yin, M. Kim, J. Saikia, S. Kwon, S. Myung, H. Kim, S. J. Kim, J.-S. Seo, and M. Seok, "Pimca: A programmable in-memory computing accelerator for energy-efficient dnn inference," *IEEE Journal of Solid-State Circuits*, vol. 58, no. 5, pp. 1436–1449, 2023.

[59] L. Song, J. Mao, Y. Zhuo, X. Qian, H. Li, and Y. Chen, "Hypar: Towards hybrid parallelism for deep learning accelerator array," in *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2019, pp. 56–68.

[60] X. Wei, Y. Liang, P. Zhang, C. H. Yu, and J. Cong, "Overcoming data transfer bottlenecks in dnn accelerators via layer-conscious memory managment," in *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 120. [Online]. Available: https://doi.org/10.1145/3289602.3293947

[61] M. Capra, B. Bussolino, A. Marchisio, M. Shafique, G. Masera, and M. Martina, "An updated survey of efficient hardware architectures for accelerating deep convolutional neural networks," *Future Internet*, vol. 12, no. 7, p. 113, 2020.

[62] K. Khalil, B. Dey, A. Kumar, and M. Bayoumi, "A reversible-logic based architecture for convolutional neural network (cnn)," in *2021 IEEE International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2021, pp. 1070–1073.

[63] H. Li, X. Yue, Z. Wang, W. Wang, H. Tomiyama, and L. Meng, "A survey of convolutional neural networks—from software to hardware and the applications

in measurement," *Measurement: Sensors*, vol. 18, p. 100080, 2021.
[64] B. Dey, K. Khalil, A. Kumar, and M. Bayoumi, "A reversible-logic based architecture for vggnet," in *2021 28th IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*. IEEE, 2021, pp. 1–4.
[65] Y. Tang, L. Tian, Y. Liu, Y. Wen, K. Kang, and X. Zhao, "Design and implementation of improved cnn activation function," in *2022 3rd International Conference on Computer Vision, Image and Deep Learning & International Conference on Computer Engineering and Applications (CVIDL & ICCEA)*. IEEE, 2022, pp. 1166–1170.
[66] K. Khalil, O. Eldash, A. Kumar, and M. Bayoumi, "Designing novel aad pooling in hardware for a convolutional neural network accelerator," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 3, pp. 303–314, 2022.
[67] Q. Song, J. Zhang, L. Sun, and G. Jin, "Design and implementation of convolutional neural networks accelerator based on multidie," *IEEE Access*, vol. 10, pp. 91 497–91 508, 2022.
[68] Y.-S. Ting, Y.-F. Teng, and T.-D. Chiueh, "Batch normalization processor design for convolution neural network training and inference," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2021, pp. 1–4.
[69] B. Khabbazan and S. Mirzakuchaki, "Design and implementation of a low-power, embedded cnn accelerator on a low-end fpga," in *2019 22nd Euromicro Conference on Digital System Design (DSD)*. IEEE, 2019, pp. 647–650.
[70] H. Xiao, K. Li, and M. Zhu, "Fpga-based scalable and highly concurrent convolutional neural network acceleration," in *2021 IEEE International Conference on Power Electronics, Computer Applications (ICPECA)*. IEEE, 2021, pp. 367–370.
[71] J. Lee, J. Rhim, D. Kang, and S. Ha, "S3nas: Fast hardware-aware neural architecture search methodology," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2021.
[72] S. Liu, H. Fan, M. Ferianc, X. Niu, H. Shi, and W. Luk, "Toward full-stack acceleration of deep convolutional neural networks on fpgas," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.
[73] H. Wang, Y. Zhao, and F. Gao, "A convolutional neural network accelerator based on fpga for buffer optimization," in *2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, vol. 5. IEEE, 2021, pp. 2362–2367.
[74] P. Achararit, M. A. Hanif, R. V. W. Putra, M. Shafique, and Y. Hara-Azumi, "Apnas: Accuracy-and-performance-aware neural architecture search for neural hardware accelerators," *IEEE Access*, vol. 8, pp. 165 319–165 334, 2020.
[75] T. Yuan, W. Liu, J. Han, and F. Lombardi, "High performance cnn accelerators based on hardware and algorithm co-optimization," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 1, pp. 250–263, 2020.
[76] W. Huang, H. Wu, Q. Chen, C. Luo, S. Zeng, T. Li, and Y. Huang, "Fpga-based high-throughput cnn hardware accelerator with high computing resource utilization ratio," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.
[77] M. Kim and J.-S. Seo, "Deep convolutional neural network accelerator featuring conditional computing and low external memory access," in *2020 IEEE Custom Integrated Circuits Conference (CICC)*. IEEE, 2020, pp. 1–4.
[78] Q. Cheng, L. Dai, M. Huang, A. Shen, W. Mao, M. Hashimoto, and H. Yu, "A low-power sparse convolutional neural network accelerator with pre-encoding radix-4 booth multiplier," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 70, no. 6, pp. 2246–2250, 2023.
[79] X. Yu, Y. Wang, J. Miao, E. Wu, H. Zhang, Y. Meng, B. Zhang, B. Min, D. Chen, and J. Gao, "A data-center fpga acceleration platform for convolutional neural networks," in *2019 29th International Conference on Field Programmable Logic and Applications (FPL)*, 2019, pp. 151–158.
[80] R. Hwang, M. Kang, J. Lee, D. Kam, Y. Lee, and M. Rhu, "Grow: A row-stationary sparse-dense gemm accelerator for memory-efficient graph convolutional neural networks," in *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2023, pp. 42–55.
[81] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm networks," in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, vol. 4. IEEE, 2005, pp. 2047–2052.
[82] K. Smagulova and A. P. James, "A survey on lstm memristive neural network architectures and applications," *The European Physical Journal Special Topics*, vol. 228, no. 10, pp. 2313–2324, 2019.
[83] K. Khalil, B. Dey, A. Kumar, and M. Bayoumi, "A reversible-logic based architecture for long short-term memory (lstm) network," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2021, pp. 1–5.
[84] Y. Wei, J. Zhou, Y. Wang, Y. Liu, Q. Liu, J. Luo, C. Wang, F. Ren, and L. Huang, "A review of algorithm & hardware design for ai-based biomedical applications," *IEEE transactions on biomedical circuits and systems*, vol. 14, no. 2, pp. 145–163, 2020.
[85] J. Wu, F. Li, Z. Chen, and X. Xiang, "A 3.89-gops/mw scalable recurrent neural network processor with improved efficiency on memory and computation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 12, pp. 2939–2943, 2019.
[86] D. Kadetotad, S. Yin, V. Berisha, C. Chakrabarti, and J.-s. Seo, "An 8.93 tops/w lstm recurrent neural network accelerator featuring hierarchical coarse-grain sparsity for on-device speech recognition," *IEEE Journal of Solid-State Circuits*, vol. 55, no. 7, pp. 1877–1887, 2020.
[87] G. Nan, Z. Wang, C. Wang, B. Wu, Z. Wang, W. Liu, and F. Lombardi, "An energy efficient accelerator for bidirectional recurrent neural networks (birnns) using hybrid-iterative compression with error sensitivity," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 9, pp. 3707–3718, 2021.
[88] C. Gao, A. Rios-Navarro, X. Chen, S.-C. Liu, and T. Delbruck, "Edgedrnn: Recurrent neural network accelerator for edge inference," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 10, no. 4, pp. 419–432, 2020.
[89] D. Shan, Y. Luo, X. Zhang, and C. Zhang, "Drrnets: Dynamic recurrent routing via low-rank regularization in recurrent neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 4, pp. 2057–2067, 2023.
[90] J. Chen, S. Hong, W. He, J. Moon, and S.-W. Jun, "Eciton: Very low-power lstm neural network accelerator for predictive maintenance at the edge," in *2021 31st International Conference on Field-Programmable Logic and Applications (FPL)*, 2021, pp. 1–8.
[91] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.
[92] W. Li, S. Wang, and G. Liu, "Transformer-based model for fmri data: Abide results," in *2022 7th International Conference on Computer and Communication Systems (ICCCS)*, 2022, pp. 162–167.
[93] S. Ansari and K. A. Alnajjar, "Multi-hop genetic-algorithm-optimized routing technique in diffusion-based molecular communication," *IEEE Access*, vol. 11, pp. 22 689–22 704, 2023.
[94] M. S. Rao, K. Venkata Rao, and M. H. M. Krishna Prasad, "Hybrid security approach for database security using diffusion based cryptography and diffie-hellman key exchange algorithm," in *2021 Fifth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, 2021, pp. 1608–1612.
[95] Z. Zhao, R. Cao, K.-F. Un, W.-H. Yu, P.-I. Mak, and R. P. Martins, "An fpga-based transformer accelerator using output block stationary dataflow for object recognition applications," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 70, no. 1, pp. 281–285, 2023.
[96] Z. Cheng, Z. Zhang, J. Jiang, and J. Sun, "Signal detection of mobile multi-user molecular communication system using transformer-based model," in *2023 8th International Conference on Computer and Communication Systems (ICCCS)*, 2023, pp. 85–90.
[97] Y. Yan, W. Du, D. Yang, and D. Yin, "Cipta: Contrastive-based iterative prompt-tuning using text annotation from large language models," in *2023 4th International Conference on Electronic Communication and Artificial Intelligence (ICECAI)*, 2023, pp. 174–178.
[98] Y. Ye, H. You, and J. Du, "Improved trust in human-robot collaboration with chatgpt," *IEEE Access*, vol. 11, pp. 55 748–55 754, 2023.
[99] P. Maddigan and T. Susnjak, "Chat2vis: Generating data visualizations via natural language using chatgpt, codex and gpt-3 large language models," *IEEE Access*, vol. 11, pp. 45 181–45 193, 2023.
[100] W. Zhu, N. Zeng, N. Wang *et al.*, "Sensitivity, specificity, accuracy, associated confidence interval and roc analysis with practical sas implementations," *NESUG proceedings: health care and life sciences, Baltimore, Maryland*, vol. 19, p. 67, 2010.

Tamador Mohaidat received the B.Sc. degree in computer engineering from Yarmouk University, Jordan, in 2010. She is currently pursuing an M.Sc. degree in computer engineering at the Department of Electrical and Computer Engineering, The University of Mississippi, MS, USA. She was a Lecturer in the Deanship of the preparatory year at Sattam bin Abdul-Aziz University, Saudi Arabia, for two years. She is currently a Research Assistant at the Department of Electrical and Computer Engineering, The University of Mississippi, MS, USA. Her research interests include very large-scale integration (VLSI), artificial intelligence, machine learning, and hardware accelerator.

Kasem Khalil (Senior Member, IEEE) received the B.Sc. and M.Sc. degrees in electrical engineering from Assiut University, Asyut, Egypt, in 2009 and 2014, respectively, and the Ph.D. degree in computer engineering from the Center of Advanced Computer Studies (CACS), University of Louisiana at Lafayette, Lafayette, LA, USA, in 2021. He has been serving as an associate editor at Elsevier Microelectronics Journal since 2022. His research interests include electronics, very large-scale integration (VLSI), microelectronics, reconfigurable hardware, self-healing hardware system, machine learning, hardware accelerators, network-on-chip, artificial intelligence, intelligent hardware system, and the Internet of Things.

Dr. Khalil was the recipient of the IEEE Transactions on Very Large Scale Integration Systems Prize Paper Award (IEEE Circuits and Systems Society VLSI Paper Award), 2023.