Bulletin of Electrical Engineering and Informatics
Vol. 15, No. 2, April 2026, pp. 1049~1060
ISSN: 2302-9285, DOI: 10.11591/eei.v15i2.11142

Resource optimization of approximate convolutional neural network-based accelerator on FPGA platform

Pooja Alana Puttaswamy[1], Kasaragod Poornima Kamath[1], Priyadarshini Jainapur[1], Karanam Sunil Kumar[2], Kumar Puttaswamy Gowda[3]
[1]Department of Electronics and Communication Engineering, B.M.S. College of Engineering, Bengaluru, India
[2]Department of Computer Science and Engineering, R.V. College of Engineering, Bengaluru, India
[3]Department of Electronics and Communication Engineering, Dayananda Sagar College of Engineering, Bengaluru

Article Info

Article history:

Received Aug 1, 2025
Revised Oct 24, 2025
Accepted Dec 6, 2025

Keywords:

Accelerator
Convolutional neural network
Field-programmable gate arrays
Pipeline mechanism
Resource optimization

ABSTRACT

A significant number of machine learning techniques use convolutional neural networks (CNN). The hardware acceleration is essential due to the tremendous computation demands of CNNs, as well as the need for improved energy performance and lower usage response time. This manuscript presents the resource-optimized CNN-based hardware accelerator on a field-programmable gate array (FPGA) platform. The design uses LeNet-5 architecture for handwritten digits classification using the MNIST dataset. The CNN accelerator uses three different optimization approaches in this work, including fixed-point (FP) data optimization with shortened bits, approximate multiply-accumulate (MAC) operations, and loop unrolling features with a pipelining mechanism to optimize the hardware resources. These approaches improve the latency and resources, and overall performance of the CNN architecture. The CNN-based accelerator is designed and implemented on the Xilinx Zynq platform with the high-level synthesis (HLS) tool. The proposed MAC unit obtains a latency of 3.58 ms, with a frequency of 120.69 MHz on chip. The design uses only 96 block random access memory (BRAMs), digital signal processing (DSP) units of 106 and obtains the accuracy of 99.01%. The proposed accelerator improves the performance over state-of-the-art accelerators in concerning the area, power and accuracy.

This is an open access article under the CC BY-SA license.

Corresponding Author:

Priyadarshini Jainapur
Department of Electronics and Communication Engineering, B.M.S. College of Engineering
P.O. Box No: 1908, Bull Temple Road, Bengaluru -560019, Karnataka, India
Email: priyadarshinijainapur.ece@bmsce.ac.in

1. INTRODUCTION

Artificial neural networks (ANNs) are used in deep learning (DL), a branch of machine learning (ML). Models developed by DL have a lot of combinations of learnt processes, in contrast to ML frameworks. With a layered structure, deeper visualizations are assembled using fewer conceptual notions, while difficult ideas are articulated using the framework of more basic ones. DL is at present providing state-of-the-art solutions for a number of computer vision and speech recognition applications. Several types of DL models are appropriate for distinct uses. Recurrent neural networks (RNNs), convolutional neural networks (CNNs), and autoencoders are a few of these [1]. Deep neural networks (DNNs) are now widely employed in a variety of programs, such as finding objects, semantic classification, recognition of things, and medical imaging, due to the increasing success and ongoing advancement of DL. Nowadays, DNNs can outperform humans in several of these kinds of tasks. Nevertheless, since the system has billions of factors, this

improved efficiency comes at the expense of high computational expense, storage utilization, and power usage. Numerous attempts have already been made to lower DNNs' storage and computational needs without sacrificing their ideal precision [2], [3].

A type of NN with an established depth and an extensive amount of convolution procedures is called a CNN. CNN's strong recognizing object recognition capabilities make it a popular choice for internet of things (IoT) applications like facial identification and home surveillance. Currently, general processing unit (GPU), application specific integrated circuit (ASIC), or field-programmable gate arrays (FPGAs) are the primary hardware accelerators used by CNN. Accelerating throughput while disregarding resource usage is the primary goal of the majority of investigations on this type of CNN hardware acceleration, which is focused on workstations or powerful data facilities [4]. Due to their relatively high effectiveness, rapid design, and software-assisted reconfigurability, FPGAs have been used as hardware architectures for performance assessment in many research investigations on CNN acceleration. Additionally, they can execute widespread mathematical calculations with the highest efficiency because of their integrated complicated processing elements, which include digital signal processing (DSP) units. CNNs' substantial memory dimensions, processing capacity, and power usage are their main drawbacks. The consequences of increased power and resources are necessary to attain high inference performance. Recently created FPGAs, on the other hand, have a lot of computing elements and power sources that are streamlined enough to accommodate fast connectivity and high-power CNN acceleration designs. This makes them excellent for exploring with several distinct CNN designs [5]. FPGAs, on the other hand, are huge arrays of pre-built hardware-mapped components. Because user-designed logic and procedures have been generated for placement inside these pieces, they might not make the best utilization of resources. Specifically, DSP units fail to be utilized whenever provided low-precision input and execute mathematical calculations with high levels of precision. Another difficulty in CNN accelerator development is making FPGA assets more efficient [6]–[12]. While using CNN accelerators, users can substitute more effective mathematical operators for DSP units to achieve improved utilization and usage.

Problem statement: recent FPGA-based CNN accelerators have enhanced performance through high-level synthesis (HLS) automation, sparsity strategies, and arithmetic optimizations. The majority of designs are application-specific or lack systematic approximation-aware techniques, which limits the amount of resource optimization that can be done for approximate CNN structures. Accuracy, power, and resource efficiency are also not balanced across a variety of CNN workloads by current frameworks. Therefore, an FPGA-based approximate CNN accelerator that maximizes resources while preserving a respectable level of accuracy and performance is required.

Contribution: the MNIST handwritten digit database was used to train and evaluate the hardware accelerator implementation for the LeNet-5 CNN framework for handwritten digit classification. The contribution of the work is as follows: this work's three-phase optimization strategy, which closely combines pragma-driven loop parallelism, approximation-aware multiply-accumulate (MAC) design, and FP data refinement for FPGA CNN acceleration, essentially makes it distinctive. In contrast to earlier accelerators that prioritize either architectural parallelism or quantization, this design uses lightweight approximate MAC units that lower latency and DSP utilization without sacrificing accuracy, and it methodically investigates fractional-bit precision control to reduce overflow and utilization of resources. Furthermore, a balanced hardware mapping method that maintains timing closure and maximizes parallelism is made possible by the incorporation of HLS pragmas (unroll, pipeline, and dataflow). The proposed LeNet-5 accelerator distinguishes itself from current FPGA CNN accelerators, which usually only handle isolated areas of optimization, by achieving an effective trade-off between area, power, and accuracy by integrating these granular approximations with dataflow-oriented hardware reuse. The organization of the paper is as follows: the proposed work is discussed in section 2. The results and discussion of the CNN accelerator, including synthesis results and comparative analysis, are highlighted in section 3. Lastly, the overall work is concluded in section 4.

This section describes the current work of CNN accelerators for various applications on hardware platforms. Nisar *et al.* [13] suggested digital single-lens reflex (DSLR-CNN) employing digit-serial left-to-right (LR) arithmetic according to a most-significant-digit-first (MSDF) fashion to reduce latency with digit-level parallelism, while Kurapati and Ramachandran [14] provided Siberian Tiger-based CNN hardware accelerator (ST-CHA) architecture with sparsity optimization, which achieved good accuracy and low power. Shafique and Lee [15] created an on-demand arithmetic (ONA)-based accelerator that significantly reduced power consumption for both ResNet and visual geometry group-16 (VGG-16) models. Syed *et al.* [16] created a shared-layer FPGA accelerator that allows for >90% accuracy in multitasking. A multiplication-free lookup-based CNN accelerator was introduced by Fuketa *et al.* [17] to remove digital signal processor (DSP) dependency, while Filippas *et al.* [18] used HLS libraries to generate CNN hardware flexibly and securely. Examples of application-oriented designs are Ghani *et al.* [19], who optimized a Basys-3 FPGA CNN for

healthcare IoT with ~91% accuracy. To speed up different kernel sizes, Lee *et al.* [20] suggested a flexible diagonal cyclic array (FDCA)-based system on chip (SoC), while He *et al.* [21] maximized resource usage for multipliers, achieving 102.3 giga operations per second (GOPS) on VGG16. Loop tiling techniques were emphasized by Park and Park [22] for resource-efficient performance. Thejaswini *et al.* [23] proposed approximate feedforward CHA (AF-CHA) and the flexible pipelined feedforward CHA (F-CHA), which achieved up to 1.42× speedup with accuracy benefits, whereas Neelam and Prince [24] introduced VCONV intellectual property (IP) optimized for AlexNet with 56 GOPS efficiency. Approximate designs are becoming more and more popular. Li *et al.* [25] used a unified Winograd-based architecture with pipelining and function reuse to increase DSP productivity. Zheng *et al.* [26] developed a low-power accelerator for rice leaf disease detection with 97.4% accuracy. The summary of recent works [13]–[26] on hardware-based CNN accelerators is tabulated in Table 1.

Table 1. Summary of recent works on hardware-based CNN accelerators

Author and year	Problem	Approach	Results	Limitations
Nisar *et al.* (2024) [13]	High latency in CNN arithmetic	Digit-serial LR arithmetic (MSDF)	Latency of 0.94 ms, and Power usage of 1249.42 mW	Complex design and limited scalability
Kurapati and Ramachandran (2024) [14]	Inefficient CNN for media/forecasting	ST-CHA with sparsity optimization	150 bps, 0.43 W, and 92.8% accuracy, 0.5 ns delay	Narrow application scope
Shafique and Lee (2024) [15]	Delay and power inefficiency	ONA-based CNN with reconfigurable power	Power savings: 19–34% across models,	Hardware complexity; accuracy trade-offs
Syed *et al.* (2024) [16]	Data instability and multiple accelerators needed	Shared-layer FPGA CNN with reconfiguration	0.599 W power, 120 K LUTs, and 15.63 µ latency	Limited model generalizability
Fuketa *et al.* (2024) [17]	Dependence on DSP multipliers	Lookup-based CNN with residual quantization	DSP-free FPGA CNN and comparable speed	Accuracy/scalability unproven
Filippas *et al.* (2024) [18]	Complex CNN hardware generation	HLS library with refinement knobs	Flexible and safe design exploration	Higher overhead vs RTL
Ghani *et al.* (2024) [19]	Costly and slow clinical diagnostics	Optimized Basys-3 FPGA CNN	~91% accuracy; low-cost healthcare IoT	Limited resources and small dataset
Lee *et al.* (2024) [20]	Inefficient kernel-size handling	FDCA-based CNN SoC	Faster inference; YOLOv5 accuracy 43.1%	Low accuracy and precision trade-offs
He *et al.* (2024) [21]	Multiplier underutilization	Data rearrangement and reuse (512 multipliers)	102.3 GOPS at 100 MHz (VGG16)	Model-specific optimization
Park and Park (2025) [22]	Suboptimal loop tiling	Resource-constrained tiling optimization	Near-optimal efficiency	Sensitive to tile-size heuristics
Thejaswini *et al.* (2025) [23]	Resource-constrained performance	AF-CHA and F-CHA accelerators	AF-CHA: 1.42× speedup; F-CHA: 1.07×	Approximation may reduce reliability
Neelam and Prince (2025) [24]	High power/cost CNN IPs	VCONV IP for AlexNet	56 GOPS and 1.64 Mb BRAM	Limited to AlexNet
Li *et al.* (2025) [25]	Low DSP efficiency	Winograd with unified input pipelining	5.8× improved GOPS/DSP efficiency	Complexity; Winograd-specific
Zheng *et al.* (2025) [26]	Slow agri-disease classification	FPGA CNN with pipelining and buffering	97.4% accuracy, 3.21 W, and 43 ms/frame	Application-specific design

Research gaps: although CNN accelerators have advanced significantly, there are still important gaps in the resource optimization of approximation FPGA designs. Although works that concentrate on arithmetic improvements ([13], [15], [17]) decrease latency and DSP reliance, they do not systematically integrate approximation approaches to combine economy and accuracy. Strong outcomes are shown by application-driven accelerators ([14], [19], [26]), but they are domain-specific and difficult to generalize. Productivity is increased by design automation initiatives ([18], [22], [25]), but approximation-aware optimization is ignored. The speedup and efficiency of new approximate CNN accelerators ([23]) are noteworthy, but their scalability, dependability, and resource usage across several FPGA platforms are not well studied. This emphasizes the way general-purpose FPGA-based CNN accelerators require a unified strategy that uses approximation computation to achieve performance, power, and area efficiency while maintaining acceptable accuracy.

2. METHOD

A convolution technique that employs a 2D array of sources is used by CNNs, a form of DNN. A CNN uses kernels, just like a conventional DNN, even if its outcome is the total of multiple inputs and weights. The groups of weights known as kernels use the input features to carry out sliding-window

convolution procedures. The combination of overlapping source feature maps and classifiers makes up every result node. As a result, a CNN can speed up network training and inference while drastically lowering the number of parameters to be trained. It is possible to use pooling layers (PL), also known as subsampling layers, to minimize the network's dimension and operations. PLs are designed to collect useful information across every part of a source to minimize the dimensions of the input. Additionally, attainable factors like weight and bias values may be present in PLs, contingent upon the network's architecture. Networks can be made nonlinear by introducing activation functions (AFs). The resulting data from nodes is processed by AFs before being transmitted to the subsequent layer. To create more intricate models, nonlinear AFs are utilized.

Lecun *et al.* [27] created the CNN framework known as LeNet-5. The MNIST handwriting digit datasets were employed to train and evaluate this handwriting digit classification framework [28]. Figure 1 illustrates the CNN structure of LeNet-5, whereas Figure 2 shows the internal structure of LeNet-5 CNN (data flow). It has two PL (Pool1 and Pool2), three convolutional layers (CLs) (Conv1, Conv2, and Conv3), and two layers that are fully connected (FC1 and FC2). Tanh AFs are used in each layer. Digital classifying outcomes (zero to nine) are the outcome, and the input is a 32×32 handwritten digit image.

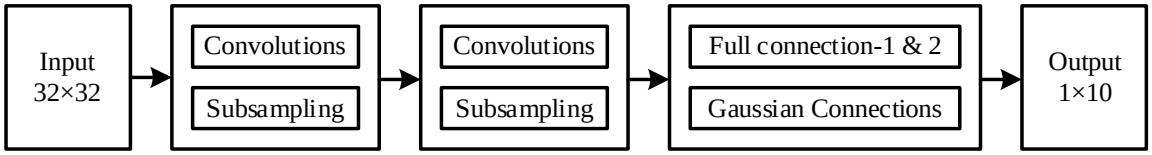


Figure 1. CNN architecture of LeNet-5

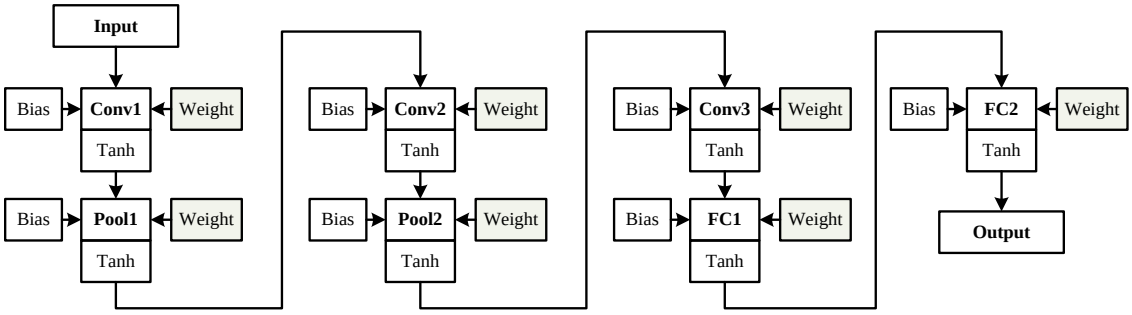


Figure 2. Internal architecture of LeNet-5 CNN (data flow)

A 5×5 kernel is used by the CLs to carry out convolutions. The result of the amount of input characteristic maps, result characteristic maps, and kernel size (5×5) yields the number of attainable weights. The quantity of result characteristic maps is equal to the quantity of biases. The input image's width and height are trimmed in half by PLs using 2×2 average pooling activities. Learned weights and biases are also used by LeNet-5. Characteristic maps are combined by weights and subsequently incorporated into biases following average analysis. Training weights and biases are thus equal to the input characteristic maps' number. The construction of the FC layers is identical to that of the conventional ANN layers. Every result is connected to every input. The sum of the quantities of intake and outlet neurons determines the quantity of weights. The quantity of output neurons equals the quantity of biases.

2.1. Proposed accelerator for LeNet-5 convolutional neural network

The LeNet-5 CNN framework serves as the foundation for the suggested accelerator. The accelerator transforms incoming information into fixed points (FP) from a 32-bit floating-point (FLP) information format. The network's FP results are transformed into 32-bit FLP data before being transferred as the finished outcomes. After completing the initial configuration once, the system can repeat the weights and biases that have been stored in its core cache for fresh incoming image set patterns. Three main optimization approaches are used by the suggested CNN accelerator: FP data optimization, approximation MAC procedures, and loop parallelism.

a. FP data optimization*:* in computation, there are two main formats for representing real numbers, as follows: FLP and FP. Figure 3 shows the construction of each of these forms for the identical 32-bit length. One sign bit, eight exponent bits, and twenty-three significant bits make up the IEEE-754 single-precision binary FLP form, which has a total of 32 bits. The 32-bit FP and FLP format is illustrated in Figures 3(a) and (b), respectively. The exponent scales information as a power of two and represents the information according to their relevance. The ability of a number's decimal point to shift with respect to its essential numbers is referred to as float. Consequently, a large range of integers can be represented in the FLP form. Both integer with sign and fractional bits make up the FP notation. In essence, binary information displaced by a specified constant fraction is what is meant by FP information. It differs from FLP information in that the bit positions remain static without further number movement. Even though the number of bits limits the precision of information, FP calculation is more cost-effective. In hardware applications, an FP data representation can lower computation engine size, power, and time.

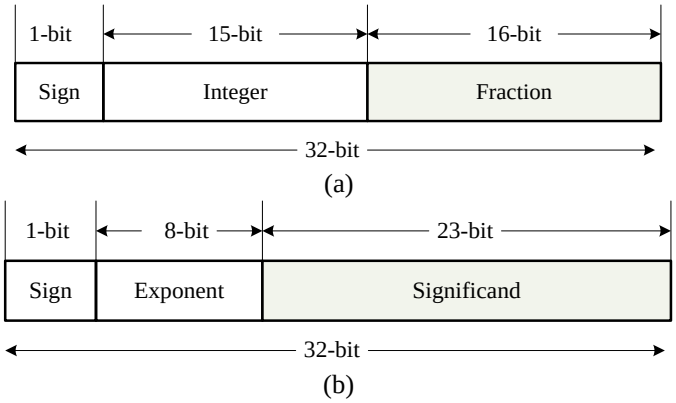


Figure 3. Representation of incoming informations in the form of; (a) 32-bit fixed-point (FP) format and (b) 32-bit floating-point (FLP) format

This work uses computational approaches with varying data sizes to find the ideal number of fractional bits. The system's structure, specifically the dimension of the convolution kernels, determines the number of integer bits used. Significant oversights in a system can arise from overflow, which occurs when the total results of MAC computations on inputs and weights exceed the total quantity of integer bits available from storage. This work utilizes six bits, which are the bare minimum of integer bits required in the CNN design to prevent overflow during accumulation. For a 5×5 kernel containing multiplied values between -1 and 1, this enables us to retain data between -32 and 31. The dimension of each layer's result varies from -1 to 1 because the LeNet-5 system analyzes each layer's response using the tanh AF. Thus, by shortening unwanted integer bits, this work additionally minimizes the RAM space for keeping information and the port width among layers. Eight-bit information can be obtained by shortening the 12-bit information produced by MAC operations by four integer bits. The following can also increase the speed of memory operations. For maintaining variables, the suggested accelerator employs low-precision signed FP data with two integer bits and six fractional bits. This has the advantage of employing our approximation MAC function units for CLs, which can drastically reduce memory capacity and resource usage with minimal loss of accuracy. The accelerator implements 12-bit FP computations for PLs and FC layers. In place of DSP units, functioning units are synthesized as flip-flops (FFs) and LUTs. Whenever the multiplication outcome is longer than 12 bits, the information is shortened. The investigation found that, in comparison to the CLs, the classification accuracy is less affected by the accuracy of the PLs and FC layers.

b. Approximation MAC procedures*:* Figure 4 shows the two versions in the accelerator structure that use two distinct approximation MAC operation units for the CLs. The rounded MAC unit is shown in Figure 4(a), while the carry MAC unit is shown in Figure 4(b). The precision of the multiplication is 18 bits; however, the smallest six fractional bits are eliminated by rounding or a carry bit when information is passed to the accumulator. These techniques simplify the accumulation process without significantly altering the mathematical outcomes. Since the information size is modest, it would be impractical to use high-precision DSP structures; hence, the suggested MAC functions are carried out on FPGAs as logic components and LUTs. The FP data format is quantized using the around-to-zero technique by the rounded MAC unit. Information rounded to the floor is the outcome of shortening FP information. This work showed that only shortening the smaller six fractional bits drastically decreases the classification

accuracy. The rounded result consequently becomes balanced for both positive and negative numbers by applying a round-to-zero procedure. The HLS resources describe the procedure as a hardware component that is dynamically constructed. Rather than rounding, and so the carry MAC unit sends an additional carry bit to the adder. All six of the lowest bits have been shortened. The sign bit of the final signed FP data is called the carry bit. Unlike positive numbers, which have been rounded to zero, negative ones are shown as two's complement, hence shortening the lower bits, and rounding the number to minus infinity. A negative number is drawn closer to an infinite positive number by the additional carry bit.

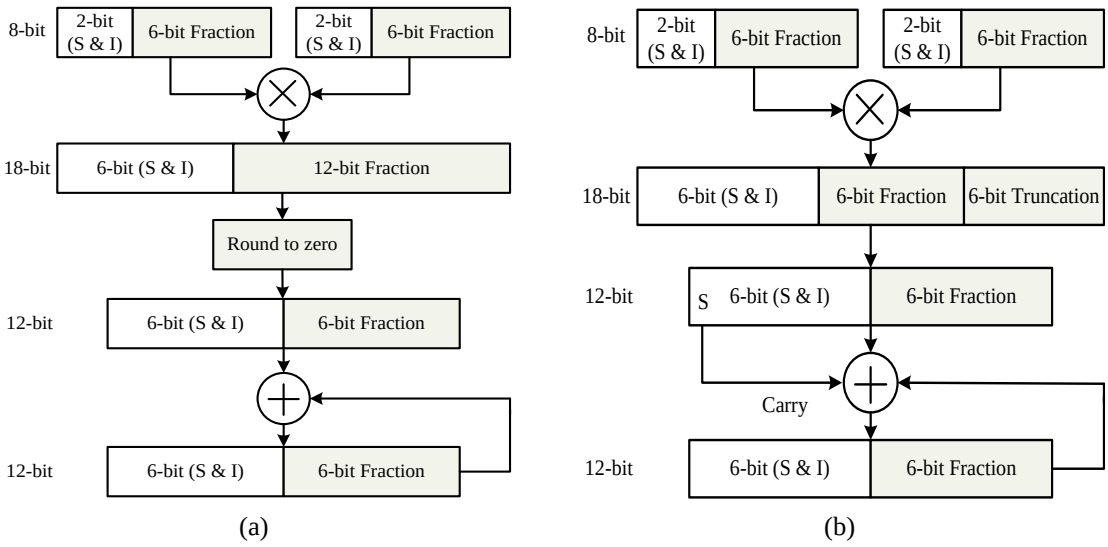


Figure 4. Suggested approximate MAC modules; (a) MAC (rounded) and (b) MAC (carry)

c. Loop parallelism*:* the Vitis HLS tools give HLS pragma commands as a special compiler directive that are used to parallelize loops. This work uses three pragma commands, namely: i) HLS unroll for flattening looping structures, ii) HLS pipeline for smaller segmentation, and iii) HLS dataflow with pipeline to increase the accelerator's speed in the complete system. As illustrated in Figure 5, whenever synthesized and processes within the loop content are executed as multiple processes that run in parallel using the unrolling command, greatly lowering latency at the expense of additional processing power. Substantial optimization is carried out by breaking up activities into smaller segments for simultaneous processing using the HLS pipeline command. The dataflow pipeline command increases the accelerator's speed by enabling activities to run in a pipelined fashion. Every layer needs to run in order since each layer's intake depends on the result of the layers before it. It is not feasible to run more than one layer concurrently for the same system source. To avoid overwriting the inside registers while operating, each layer must also complete evaluating its present input before taking in a new set of data. Consequently, the number of clock cycles (CCs)+1 of the layers that need the most cycles is equal to the minimal number of CCs needed for the accelerator to recognize the subsequent input image.

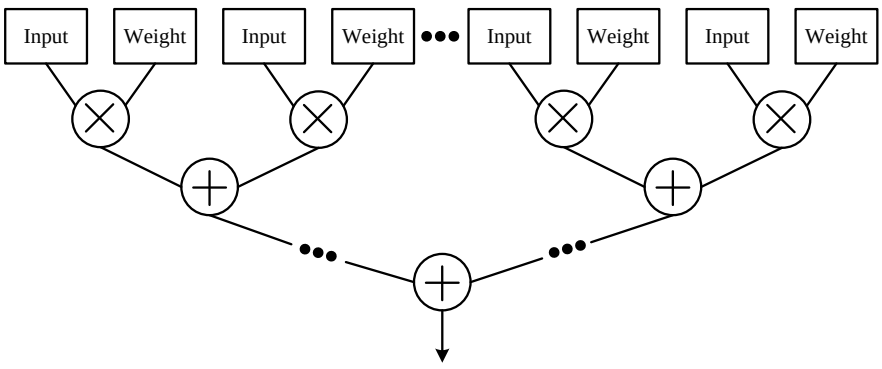


Figure 5. Arithmetic modules with unrolling features

3. RESULTS AND DISCUSSION

This work synthesizes the accelerator architecture aimed at the Xilinx Ultra+ multiprocessor system-on-chip (MPSoC) family of XCZU11EG with the package of FFVB1517 chip using the Xilinx Vitis HLS tool with version 2020.1. The accelerator designs can be used on any different FPGA device because their features can be built on the FPGA chip, despite the need for MPSoC CPU cores. The power calculation analysis from the Vivado tool and after implementation summaries are the sources of the data on the maximum working frequency, resource usage, and power usage. The MNIST handwritten number database, containing training images on CPU, was used to train the CNN utilizing the FLP data type. Measurements about interpretation accuracy were obtained using the HLS tool's testbench simulator. Every model's classification accuracy was determined using a total of 10,000 images from the MNIST test set. After being transmitted to the accelerator designs, the programmed FLP variables undergo an internal hardware function conversion into FP data type.

Tables 2 and 3 provide an overview of the resource usage and timing summary of CNN-based accelerator modules. A format consisting of integer (sign bit inserted at most significant bit (MSB)) bits and fraction bits is used to represent the 32-bit FLP and FP accelerator modules.

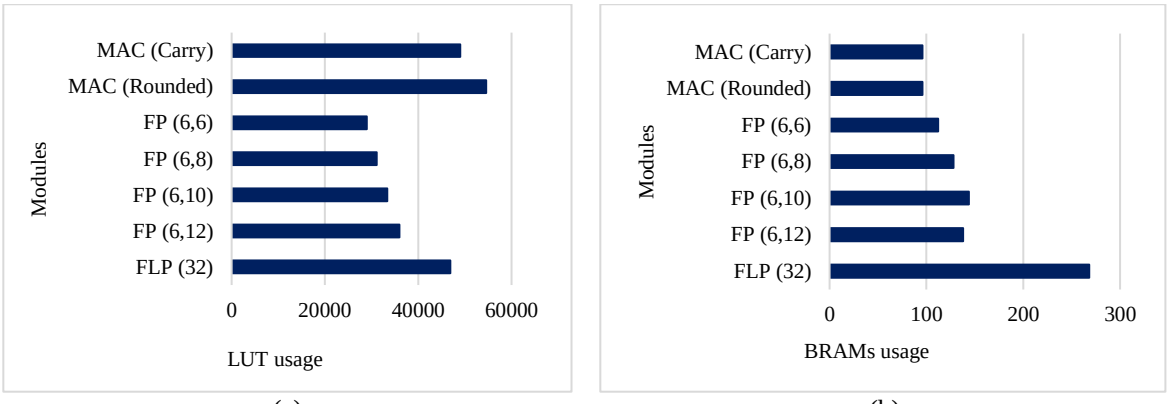
Table 2. Resource usage of CNN-based accelerator modules

Accelerator modules	LUT	FFs	BRAMs	DSP
FLP (32)	46873	38621	268	166
FP (6,12)	35962	23753	138	480
FP (6,10)	33391	20198	144	464
FP (6,8)	31086	18763	128	434
FP (6,6)	28976	16828	112	424
MAC (rounded)	54568	17864	96	106
MAC (carry)	48981	18206	96	106

Table 3. Timing summary of CNN-based accelerator modules

Accelerator modules	Clock period (ns)	Fmax (MHz)	CCs	Latency (ms)
FLP (32)	9.125	109.59	920456	8.399
FP (6,12)	6.866	145.65	572280	3.929
FP (6,10)	7.043	141.98	572260	4.031
FP (6,8)	6.968	143.52	572230	3.987
FP (6,6)	6.637	150.68	572210	3.797
MAC (rounded)	8.286	120.69	432610	3.584
MAC (carry)	7.378	135.54	432610	3.191

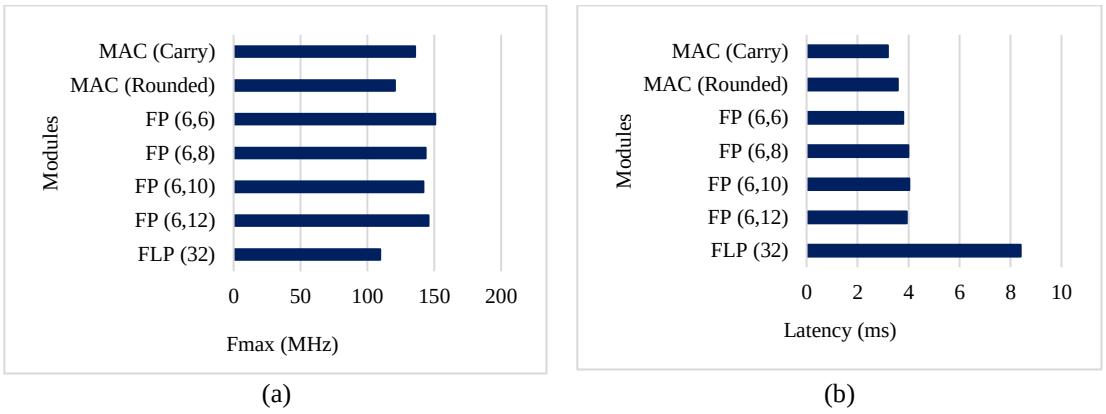
Logic elements are used in the implementation of the suggested approximation MAC modules; LUTs and FFs are used more frequently. Comparing the DSP units to the FLP and FP designs, however, shows a reduction of roughly 37% and 77%, respectively. The block random access memory (BRAM) is decreased by 31% and 64%, respectively, in comparison to the FP and FLP designs. The MAC (rounded) and MAC (carry) designs are compared. The rounded MAC model requires a few fewer FFs yet greater LUTs. The higher number of LUTs utilized to construct the suggested estimated MAC is thought to result in more path delays throughout routing. The representation of resource usage is illustrated in Figure 6. Figures 6(a) and (b) represent the LUT and BRAM usage, respectively, on the Zynq platform.



(a) (b)

Figure 6. Representation of resource usage; (a) LUT and (b) BRAM usage on Zynq platform

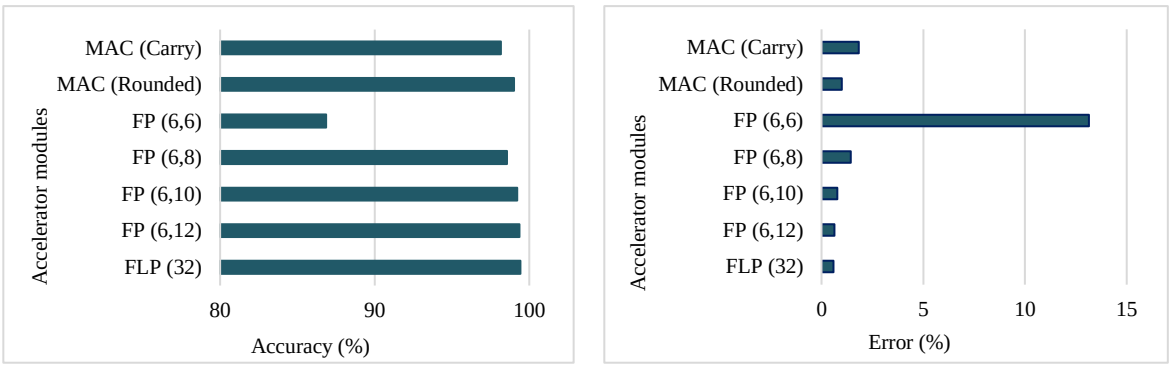
The number of CCs multiplied by the smallest clock period is the latency. Figure 7 shows the timing analysis depiction of CNN-based accelerator modules. Figures 7(a) and (b) display the obtained maximum frequency (Fmax) (MHz) and latency (ms), respectively. Considering the rounding process is more complicated than the carry procedure, the MAC (rounded) design has an 11% longer clock duration over the MAC (carry) design. In comparison to the FLP approach, the MAC (carry) and MAC (rounded) models achieve a 62% and 58% reduction in end latency, respectively. When compared to the FLP and FP designs, the MAC (rounded) and MAC (carry) modules likewise produce fewer CCs. Concerning each kind of network layer, the FC layer shows a significant decrease in cycles, although the CLs and PLs retain the same number of cycles that are used in the FP designs. The FLP design has roughly twice as much delay as the FP design since it has a longer clock period and additional CCs. In contrast to the FP designs, the FLP type requires fewer DSP units despite having a higher latency. Because the FLP design reuses arithmetic blocks across layers, the system seems to consume fewer DSP units. Furthermore, although the FPGA components are assigned to fixed regions, variations in the allocation of resources impact the signal routes, which in turn alter the crucial paths' optimal operating frequency.



(a) (b)

Figure 7. Representation of timing analysis of CNN-based accelerator modules; (a) maximum frequency (Fmax) (MHz) and (b) latency (ms)

Figure 8 and Table 4 represent the error analysis and percentage accuracy of a few CNN-based accelerator modules. The FLP and FP designs were subjected to loop parallelization alone. The three optimization strategies are used in the proposed framework. Six integer bits, including sign bits, and varying numbers of fractional bits, ranging from six to twelve, are used in the FP designs. There is less than 1% accuracy loss in the computations that use FP data with 10 and 12 fractional bits. But 8 bits generated a significant loss of 1.44%, and 6 bits produced a meagre accuracy of 13.14%. The FP (6,6) have poor accuracy. The MAC (carry) model showed a satisfactory accuracy loss of about 1.83%, whereas the one suggested with MAC (rounded) had less than 1% loss. The accuracy and error analysis are illustrated in Figures 8(a) and (b), respectively, of CNN-based accelerator modules.



(a) (b)

Figure 8. Representation of performance analysis including; (a) accuracy and (b) error analysis of CNN-based accelerator modules

Table 4. Percentage accuracy and error analysis of CNN-based accelerator modules

Accelerator modules	Accuracy (%)	Error (%)
FLP (32)	99.42	0.58
FP (6,12)	99.36	0.64
FP (6,10)	99.22	0.78
FP (6,8)	98.56	1.44
FP (6,6)	86.86	13.14
MAC (rounded)	99.01	0.99
MAC (carry)	98.17	1.83

The performance comparison proposed work with existing CNN-based state-of-the-art approaches [5], [6], [11], [16], [18] is tabulated in Table 5. The parameters like the type of design entry, FPGA, precision bits, clock frequency (MHz), area usage (in terms of LUTs, FFs, DSP units, and BRAM usage), power usage, and accuracy are considered for performance comparison. Each work uses a different FPGA family, including Zynq Ultra-Scale+ device [5], Zynq-7 [6], Artix-7 [11], and Virtex-Ultra-Scale+ devices [16], [18]. This comparison uses many accelerators like CNN with optimization of the MAC units [5], an Integer-arithmetic CNN accelerator for facial reconstruction [6], a reconfigurable full precision CNN accelerator [11], a reconfigurable fault-tolerant CNN accelerator [16] and a dataflow-based functional-safe CNN accelerator [18]. The proposed work uses 8-bit or 12-bit FP precision and operates at 100 MHz. The proposed work uses a minimal area (LUTs and FFs) than other accelerators [5], [11], [16]. The proposed work uses minimal BRAMs, DSP units, and offers better accuracy of 99.01% than other state-of-the-art accelerators. However, the results are varied based on the application, device selection, and precision bits.

Table 5. Performance comparison of the proposed work with existing CNN-based state-of-the-art approaches

Parameters	Ref [5]	Ref [6]	Ref [11]	Ref [16]	Ref [18]	Proposed
Design entry	HLS	HLS	RTL	RTL	RTL	HLS
FPGA	XCZU9EG	XC7Z045	7A100T	VUC118	VUC 108	XCZU11EG
Precision (bits)	12 and 8	32	32	32	32	12
Clock (MHz)	100	250	200	200	100	100
LUTs (K)	61.72	16	101.4	120.2	33.64	54.568
FFs (K)	27.87	33	NA	33.015	27.76	17.864
DSP	123	205	240	955	176	106
BRAM	102	120	NA	43	1728	96
Power (W)	1.673	4.31	1.821	0.599	3.62	1.42
Accuracy (%)	98.21	86.58	97.29	97.33	95.5	99.01

There are still a number of limitations even though the suggested FPGA-based CNN accelerator significantly improves latency, accuracy, and resource efficiency for LeNet-5 on MNIST. The work does not use interpretability tools to visualize the way approximations impact feature maps or decision boundaries, nor does it use ablation investigations to separate the influence of each optimization (FP quantization, approximate MAC, and loop unrolling). Furthermore, the lack of statistical validation across several synthesis cycles and weight initializations reduces trust in the reliability and consistency of the outcomes that have been published. Additionally, the approach's scalability to larger datasets (such as CIFAR-10 and ImageNet) or deeper CNNs (like ResNet and MobileNet) is still untested, where approximation errors could become more severe.

Timing closure under aggressive pipelining and loop unrolling is still crucial from an electrical engineering standpoint since longer combinational routes may affect the maximum frequency that can be achieved. Similarly, power gating techniques, which are crucial for low-power designs in edge artificial intelligence (AI), were not investigated, and clock domain crossover (CDC) problems may occur when scaling the accelerator for heterogeneous FPGA subsystems. For reliable large-scale deployments, these hardware-level issues must be resolved. In terms of informatics, the design does not provide model compression processes (such as pruning and quantization-aware training), effective scheduling across FPGA resources, or compiler-level support for automated mapping of more advanced CNN models. Adding cloud-to-edge procedures, in which models are dynamically distributed to FPGA nodes after being trained on cloud servers, could increase flexibility even more. Feasibility in real-world deployment goes beyond efficiency to application domains: lightweight accelerators need to be customized for applications such as industrial automation nodes, drones, and smart cameras, where latency, power budgets, and flexibility under changing workloads are crucial.

There is some concern regarding reproducibility under various tool optimizations and hardware mappings because the outcomes are based on a single synthesis and evaluation procedure, and consistency across numerous syntheses runs and diverse input distributions has not been confirmed. It may be necessary to partially retrain in order to recover lost performance on more complicated datasets due to the accumulation

of precision loss from FP quantization and approximation MAC operations across layers. Additionally, deployment in real-world edge AI or FPGA inference engines may encounter limitations like power budgets, memory bandwidth restrictions, and portability overheads, underscoring the necessity of cautious adaptation and retraining techniques.

Future research should fill up these gaps by expanding evaluations to more intricate networks and datasets and integrating interpretability techniques with ablation analysis to more accurately measure and explain approximation effects. Promising approaches for increasing efficiency and guaranteeing wider application in real-world AI workloads include adaptive approximation techniques, layer-specific precision scaling, and hybrid optimization with pruning or compression. However, by using pipelined loop unrolling, FP quantization, and approximate MAC design to demonstrate a resource-optimized LeNet-5 accelerator on MNIST, this study lays a useful and expandable basis for future approximation-aware FPGA accelerators.

4. CONCLUSION

The CNN-based hardware accelerator on an FPGA framework that optimizes resources is presented in this paper. The LeNet-5 structure is used in the scheme to classify handwritten digits employing the MNIST dataset. The CNN accelerator employs three distinct optimization techniques in this work: loop unrolling characteristics with a pipelining method to minimize the hardware resources, approximation MAC processes, and FP data optimization with shorter bits. These strategies enhance the CNN architecture's overall performance and latency with minimal chip area. The Vivado HLS tool is used in this work and synthesized on the Zynq Ultra-scale+ FPGA platform. Compared to FP designs, the estimated MAC (rounded) unit utilizes 31% and 37% less BRAM and DSP units, respectively. The latency is 58% lower with the MAC (rounded) unit than with the FLP design. The MAC (rounded) and MAC (carry) units achieve low accuracy loss, achieving 99.01% and 98.17%, respectively. The suggested accelerator outperforms the most advanced accelerators in terms of accuracy, power, and area. The work is limited to FPGA implementation, and it can be extended in future to implement on the ASIC or GPU platform to strengthen the validation and real-time analysis.

FUNDING INFORMATION

Authors state no funding involved.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Pooja Alana Puttaswamy	✓	✓		✓	✓	✓			✓	✓			✓	
Kasaragod Poornima Kamath		✓				✓		✓	✓	✓	✓			
Priyadarshini Jainapur	✓		✓	✓			✓			✓	✓		✓	
Karanam Sunil Kumar	✓	✓		✓	✓		✓	✓	✓	✓	✓	✓		
Kumar Puttaswamy Gowda		✓	✓	✓	✓	✓		✓	✓					

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

Data availability is not applicable to this paper as no new data were created or analyzed in this study.

REFERENCES

[1] M. A. Arshad, S. Shahriar, and A. Sagahyroon, "On the Use of FPGAs to Implement CNNs: A Brief Review," in *2020 International Conference on Computing, Electronics & Communications Engineering (iCCECE)*, Southend, UK, 2020, pp. 230-236, doi: 10.1109/iCCECE49321.2020.9231243.

[2] D. Ghimire, D. Kil, and S.-H. Kim, "A survey on efficient convolutional neural networks and hardware acceleration," *Electronics (Basel)*, vol. 11, no. 6, p. 945, 2022, doi: 10.3390/electronics11060945.

[3] M. M. Taye, "Theoretical understanding of convolutional neural network: Concepts, architectures, applications, future directions," *Computation (Basel)*, vol. 11, no. 3, p. 52, 2023, doi: 10.3390/computation11030052.

[4] F. Ge, N. Wu, H. Xiao, Y. Zhang, and F. Zhou, "Compact convolutional neural network accelerator for IoT endpoint SoC," *Electronics (Basel)*, vol. 8, no. 5, p. 497, 2019, doi: 10.3390/electronics8050497.

[5] M. Cho and Y. Kim, "FPGA-based convolutional neural network accelerator with resource-optimized approximate multiply-accumulate unit," *Electronics (Basel)*, vol. 10, no. 22, p. 2859, 2021, doi: 10.3390/electronics10222859.

[6] J. Kim, J. -K. Kang, and Y. Kim, "A Resource Efficient Integer-Arithmetic-Only FPGA-Based CNN Accelerator for Real-Time Facial Emotion Recognition," *IEEE Access*, vol. 9, pp. 104367-104381, 2021, doi: 10.1109/ACCESS.2021.3099075.

[7] J. Kim, J. -K. Kang, and Y. Kim, "A Low-Cost Fully Integer-Based CNN Accelerator on FPGA for Real-Time Traffic Sign Recognition," *IEEE Access*, vol. 10, pp. 84626-84634, 2022, doi: 10.1109/ACCESS.2022.3197906.

[8] A. K. Jameil and H. Al-Raweshidy, "Efficient CNN Architecture on FPGA Using High Level Module for Healthcare Devices," *IEEE Access*, vol. 10, pp. 60486-60495, 2022, doi: 10.1109/ACCESS.2022.3180829.

[9] K. Shi, M. Wang, X. Tan, Q. Li, and T. Lei, "Efficient dynamic reconfigurable CNN accelerator for edge intelligence computing on FPGA," *Information (Basel)*, vol. 14, no. 3, p. 194, 2023, doi: 10.3390/info14030194.

[10] J. Wang, W. Tong, and X. Zhi, "Model parallelism optimization for CNN FPGA accelerator," *Algorithms*, vol. 16, no. 2, p. 110, 2023, doi: 10.3390/a16020110.

[11] Y. Xu, J. Luo, and W. Sun, "Flare: An FPGA-based full precision low power CNN accelerator with reconfigurable structure," *Sensors (Basel)*, vol. 24, no. 7, 2024, doi: 10.3390/s24072239.

[12] J. Jang and H. Yang, "A Runtime Switchable Multi-Phase Convolutional Neural Network for Resource-Constrained Systems," *IEEE Access*, vol. 11, pp. 62449-62461, 2023, doi: 10.1109/ACCESS.2023.3287998.

[13] M. Z. Nisar, M. S. Ibrahim, S. Gorgin, M. Usman, and J. -A. Lee, "DSLR-CNN: Efficient CNN Acceleration Using Digit-Serial Left-to-Right Arithmetic," *IEEE Access*, vol. 12, pp. 174608-174622, 2024, doi: 10.1109/ACCESS.2024.3502918.

[14] H. Kurapati and S. Ramachandran, "Enhancement of Convolutional Neural Network Hardware Accelerators Efficiency Using Sparsity Optimization Framework," *IEEE Access*, vol. 12, pp. 86034-86042, 2024, doi: 10.1109/ACCESS.2024.3416062.

[15] M. A. Shafique and J. -A. Lee, "ON-CNN: Low Latency and High Throughput Online Arithmetic-Based Convolutional Neural Network Accelerator," *IEEE Access*, vol. 12, pp. 175698-175714, 2024, doi: 10.1109/ACCESS.2024.3502665.

[16] R. T. Syed, Y. Zhao, J. Chen, M. Andjelkovic, M. Ulbricht, and M. Krstic, "FPGA Implementation of a Fault-Tolerant Fused and Branched CNN Accelerator with Reconfigurable Capabilities," *IEEE Access*, vol. 12, pp. 57847-57862, 2024, doi: 10.1109/ACCESS.2024.3392240.

[17] H. Fuketa, T. Katashita, Y. Hori, and M. Hioki, "Multiplication-Free Lookup-Based CNN Accelerator Using Residual Vector Quantization and Its FPGA Implementation," *IEEE Access*, vol. 12, pp. 102470-102480, 2024, doi: 10.1109/ACCESS.2024.3432979.

[18] D. Filippas *et al*., "A High-Level Synthesis Library for Synthesizing Efficient and Functional-Safe CNN Dataflow Accelerators," *IEEE Access*, vol. 12, pp. 57194-57208, 2024, doi: 10.1109/ACCESS.2024.3390422.

[19] A. Ghani, A. Aina, and C. H. See, "An optimised CNN hardware accelerator applicable to IoT end nodes for disruptive healthcare," *IoT*, vol. 5, no. 4, pp. 901–921, 2024, doi: 10.3390/iot5040041.

[20] D.-Y. Lee *et al.*, "High-speed CNN accelerator SoC design based on a flexible diagonal cyclic array," *Electronics (Basel)*, vol. 13, no. 8, p. 1564, 2024, doi: 10.3390/electronics13081564.

[21] J. He, M. Zhang, J. Xu, L. Yu, and W. Li, "Optimizing CNN hardware acceleration with configurable vector units and feature layout strategies," *Electronics (Basel)*, vol. 13, no. 6, p. 1050, 2024, doi: 10.3390/electronics13061050.

[22] C. S. Park and S. Park, "CNN Accelerator Performance Dependence on Loop Tiling and the Optimum Resource-Constrained Loop Tiling," *IEEE Access*, vol. 13, pp. 16800-16810, 2025, doi: 10.1109/ACCESS.2025.3532790.

[23] P. Thejaswini, G. Suresh, V. Chiraag, and S. Nandi, "Approximate CNN Hardware Accelerators for Resource Constrained Devices," *IEEE Access*, vol. 13, pp. 12542-12553, 2025, doi: 10.1109/ACCESS.2025.3529668.

[24] S. Neelam and A. A. Prince, "VCONV: A Convolutional Neural Network accelerator for FPGAs," *Electronics (Basel)*, vol. 14, no. 4, p. 657, 2025, doi: 10.3390/electronics14040657.

[25] J. Li, Y. Liang, Z. Yang, and X. Li, "An efficient convolutional neural network accelerator design on FPGA using the layer-to-layer unified input Winograd architecture," *Electronics (Basel)*, vol. 14, no. 6, p. 1182, 2025, doi: 10.3390/electronics14061182.

[26] J. Zheng *et al.*, "FPGA-based low-power high-performance CNN accelerator integrating DIST for rice leaf disease classification," *Electronics (Basel)*, vol. 14, no. 9, p. 1704, 2025, doi: 10.3390/electronics14091704.

[27] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.

[28] The MNIST Database of Handwritten Digits. [Online]. Available: http://yann.lecun.com/exdb/mnist/. (Accessed: Jul. 28, 2025).

BIOGRAPHIES OF AUTHORS

Pooja Alana Puttaswamy is currently working as Assistant Professor in the Department of Electronics and Communication Engineering, BMSCE from the past five years. She has overall teaching experience of 10 years. She has handled various course for UG program. She has completed M.Tech. from Malnad College of Engineering in Information and Communication Systems. She has published research papers on reputed journals in image processing domain. Her area of interest includes wireless communication, networking, signal processing, and machine learning. She can be contacted at email: poojaap.ece@bmsce.ac.in.

Kasaragod Poornima Kamath is currently serving as an Assistant Professor at BMS College of Engineering. She completed her Bachelor of Engineering (B.E.), Master of Technology (M.Tech.), and currently pursuing Ph.D. from Visvesvaraya Technological University (VTU), Belagavi. She has a teaching experience of 9 years with research interests are in VLSI design, embedded system, IoT, machine learning, and deep learning. She can be contacted at email: poornimambaliga.ece@bmsce.ac.in.

Priyadarshini Jainapur is currently serving as an Assistant Professor in the Department of Electronics and Communication Engineering at B.M.S College of Engineering. She completed her a Bachelor's degree (B.E.) from BMS Institute of Technology, Bangalore a Master's degree (M.Tech.) from Nitte Meenakshi Institute of Technology, Bangalore. She is currently pursuing her Ph.D. at B.M.S College of Engineering, affiliated with Visvesvaraya Technological University (VTU), Belgaum. With over 10 years of teaching experience, her research interests include VLSI design, embedded systems, Verilog HDL, machine learning, deep learning, and the internet of things (IoT). She has published research papers on reputed journals. She can be contacted at email: priyadarshinijainapur.ece@bmsce.ac.in.

Dr. Karanam Sunil Kumar is currently serving as an Assistant Professor at RV College of Engineering. He completed his Bachelor of Engineering (B.E.), Master of Technology (M.Tech.), and Ph.D. from Visvesvaraya Technological University (VTU), Belagavi. With a career spanning 16 years, he has a wealth of experience in his field. He has contributed to the academic community by publishing six journals, which have been indexed in top-tier categories ranging from Q1 to Q4. His research interests are computer vision, machine learning, big data, and deep learning. He can be contacted at email: ksunilkumar@rvce.edu.in.

Dr. Kumar Puttaswamy Gowda is currently working in Department of Electronics and Communication Engineering, Dayananda Sagar College of Engineering, Bengaluru from past 10 years. He has total teaching experience of 14 years. His area of interest includes signal processing, communication, and digital VLSI. He obtained his Ph.D. degree from Visvesvaraya Technological University, Belagavi India. He has published several research articles in reputed conferences and Journals. He is also a reviewer in IEEE transactions. He can be contacted at email: kumarhuluvadi@gmail.com.