IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 15 May 2024, accepted 10 June 2024, date of publication 17 June 2024, date of current version 26 June 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3416062

RESEARCH ARTICLE

Enhancement of Convolutional Neural Network Hardware Accelerators Efficiency Using Sparsity Optimization Framework

HEMALATHA KURAPATI AND SAKTHIVEL RAMACHANDRAN (Senior Member, IEEE)
School of Electronics Engineering (SENSE), Vellore Institute of Technology, Vellore, Tamil Nadu 632014, India
Corresponding author: Sakthivel Ramachandran (rsakthivel@vit.ac.in)

ABSTRACT Convolutional neural networks (CNNs) accelerators have been utilized widely for several digital applications to improve processing efficiency. However, the traditional CNN accelerator processor performance is insufficient to run the digital smart application as per the user's needs, resulting in high power consumption, delay, Look Up Table (LUT)- Random Access Memory (RAM) usage, and less accuracy and throughput. Hence, the present research study was intended to design the modified CNN accelerator for prediction and data broadcasting applications. Hence, the newly designed accelerator is named a novel Siberian Tiger-based Convolutional Neural Accelerator architecture (STbCNA). Here, the sparse features and the tiger fitness data reuse strategy have been considered to gain the exact prediction outcome. The predicted outcome is transferred to the user to make the rainfall aware to satisfy this parameter. Consequently, the Throughput and other FPGA parameters were calculated and compared with other models. For comparison, all the traditional approaches were executed in the same proposed platform, and the outcomes were compared with those of the proposed approach. In that, the modified CNN (STbCNA) scored the finest outcome by attaining a high Throughput of 150 bps, reduced Power of 0.43W, high accuracy of 92.8%, less delay of 0.5ns, and LUT of 0.001. The presence of the Siberian tiger provided the continuous optimal conditions for the present FPGA implementation. Hence, the STbCNA is significant for the FPGA application in gaining the optimal outcome.

INDEX TERMS Convolutional neural networks, accelerator performance, rainfall prediction, optimization, processing efficiency.

I. INTRODUCTION

Inspired by the natural anxious system, deep knowledge has recently achieved commanding delicacy improvement. Convolutional neural networks (CNNs), the most generally applied model in deep knowledge, hold lived took on in various disciplines, including image and speech recognition [1]. The CNN Accelerator IP is paired with the Chassis Neural Network Complier Tool [2]. The compiler takes the networks elaborated ordinary machine learning fabrics, analyses them for practical operation, simulates them for interpretation and functionality, and also collects them for the CNN Accelerator IP [3]. The suggestive delicacy improvement of CNNs comes at the disbursement of huge computational complications as it requires a total estimate of all the fields across the point maps [4]. Tackle acceleration refers to the operation of technical tackle, Graphics processing units (GPUs), or Application-specific integrated circuits (ASICs) to perform calculations more briskly than a general-ambition central processing unit (CPU) [5]. Tackle acceleration can significantly facilitate the speed and effectiveness of routine and deep literacy conclusion operations in the plot of deep literacy for AI [6]. Tackle acceleration can significantly reduce the time it takes to train and emplace deep literacy models [7]. For illustration, using a GPU can speed training by a factor of 10- 100 assimilated to using a CPU alone [8].

The associate editor coordinating the review of this manuscript and approving it for publication was Jiang Wu.

This can be particularly important for large-scale deep literacy assignments, similar to training image or language models with millions of parameters [9]. Tackle acceleration can similarly facilitate the effectiveness of deep literacy training processors by reducing the quantum of bounce needed to carry out calculation processes [10]. This is particularly consequential for assignments that bear a large quantum of calculation, cognate as training large neural networks [11]. Tackle acceleration can degrade training costs and plant deep literacy models [12]. For illustration, operating a GPU can significantly reduce the cost of training equated to using a CPU, particularly for considerable- scale tasks [13]. Hardware acceleration is a consequential tool for perfecting the velocity and effectiveness of bottomless literacy tasks [14].

By applying technical tackle, similar to GPUs, deep literacy interpreters can significantly downgrade the occasion and outlay of practice and planting abysmal literacy miniatures [15]. Reducing the total compute and memory requirements for such models without sacrificing their high sensitivity is possible by cutting the depth of neural networks [16]. For example, the researchers have demonstrated that some Deep Networks have substantial duplication (up to 90), that may be trimmed without sacrificing delicacy [17]. Reducing methods can open up the fast conclusion procedure by hypothetically reducing the number of processes in the difficulty algorithm [18]. However, frugal CNN designs are unsuited for topologies using large datasets with FPGAs [19]. The final manufacturers optimize the information flow via circle activities, such as unrolling and crossing circles. Through this, it can effectively prevent design challenges associated with sparsely comparable data, such as imbalanced weight and uneven connections. However, CNNs have certain drawbacks that restrict their applicability and understanding. One of the main disadvantages of CNNs is that they bear a substantial quantum of tagged data to train effectively, which can be ultra-expensive and time-devouring to gain and annotate [20]. The key contributions of this present work are described as follows;

- Initially, the required parameters were defined in the Xilinx environment.
- Moreover, a novel STbCNA is created with the optimal functional modulus for improving the training accelerators.
- Then, the created STbCNA is tested for the performance and efficiency of the accelerator, including sparsity and data reuse techniques.
- Finally, the performance improvement score and reliability were measured regarding power utilization, Flip flop, CU utilization, acceleration rate, and LUT.

The paper is arranged as follows; the second section has detailed the recent works of the CNN accelerator. Third section has exposed the traditional CNN with problems, fourth section describes the proposed CNN accelerator, the outcome of the proposed accelerator is exposed in fifth section, and sixth section concludes the research article.

II. RELATED WORK

In this document, Li et al. [21] outlined a functional dual complexity approach that isn't embellished and is intended for 3D CNN. In this Algorithm, activating and weighting are either 0 or 1, not - 1 or 1 are processed. Furthermore, a useful complexity cost O2M and classification storage path is proposed, which may greatly reduce the pouring input pixel processing time. A 3D AND- Net accelerator is shown based on the enhancement methodologies. The outcomes demonstrate how excellent our design is regarding power consumption, recommended operation, and Digital Signal Processor (DSP) efficacy when applied to bouncing passes. It ought to be noted that collaborative optimization, as opposed to tone-sustaining optimizing, can yield better results when applied to techniques and threats.

In this exploration paper, Lee and Yoo [22] delineated that DNN training iteratively processes three different ways, causing accidental recollection accesses and operations. Thus, high bounce-operative DNN training tackle is compulsory to ascertain DNN exercise on an edge juggle which has circumscribed computational qualification and command force. For the altitudinous energy-operative DNN training tackle, three challenges must be optimized different dataflow, extrinsic memory turn, and calculation.

In this context, Dhilleswararao et al. [2] reconsidered recent advancements in DNN acceleration on technical tackle infrastructures. Likewise, Embedded AI accelerators for the acerbity terrain retain existed exhaustively bandied. The reappraisal begins with a minute backdrop of DNNs, centering on their pivotal missions and operations. CNNs, which command a thick range of operations, the command has also been numbered in retrospect. To ease the interpretation of the tackle accelerator, we bandied multi-hued computing infrastructures resembling profane and spatial infrastructures.

Mittal et al. [13] carried out a check of accelerator arrangement and optimizations for CNN. They stressed their pivotal inventions and arranged them into several orders. Compactly bandy some exploration cynosures deserving of instantaneous concentration from the experimenters, 3D CNN accelerators have been big-time less analogized to that beseemed into 2D CNN accelerators. For driving added progression in 3D CNNs, it's eventful to distinguish motivating operations and exercise cases, particularly those where 2D CNNs aren't sufficient.

This paper by Xie et al. [23] offers a CNN design that is mindful of sparsity and supports randomized and random pruning methods. To use pruned networks, a preprocessing method called the Weight Pretreatment method is created for condensed weights. A Hybrid Parallelism (HP) on-chip information flow is created according to the suggested scheme to maximize computation effectiveness. Furthermore, the HP information transfer can process the convolutions with different configurations flexibly. The suggested design can significantly lower the quantity of access to outside memories when combined with the full fusing information flow.

III. TRADITIONAL CNN ARCHITECTURE WITH PROBLEM

In many modern programs, processes are executed and stored in chip memory. But, if chip memory is not optimized, it can greatly raise the cost of memory overall, particularly for particular workloads [24]. Numerous accelerator techniques that use neural memory to boost chip performance have been created. Superior results have been obtained while analyzing chip features using neural principles [25]. Traditional deep networks [26] are not sufficient for different applications.

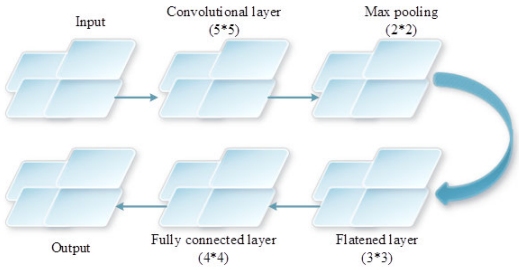

FIGURE 1. CNN architecture with problem.

The kernel size alteration isn't always enough to get the best accelerator performance. Traditional CNN architecture with problem is shown in Fig 1. The main problem that reported by the traditional convolution neural accelerator is high power and memory usage. It is due to the lack of tuning or regulating process in CNN based on different application [27]. If the CNN is not regulated proper then it might take more time and space to execute every execution. To solve these problems, a novel hardware accelerator method has been developed that alters kernel sizes and pooling layers to maximize the accelerator's performance.

IV. PROPOSED METHODOLOGY

The demand for efficient hardware accelerators is increasing due to the rapid growth of data-intensive applications [28]. This research presents a novel Siberian Tiger-based Convolutional Neural Accelerator architecture (STbCNA) shown in fig 2, that uses sparsity and data reuse to improve performance and energy efficiency. In addition, to check the accelerator performance, the rainfall data has been adopted. Finally, the robustness of the designed accelerator was justified by measuring the key parameters and comparing them with the existing approaches such as Bit Parameters CNN (BP-CNN) [29], Deep Convolutional Network Accelerator (DCNA) [30], Embedded features CNN (EF-CNN) [30] and Coati CNN (C-CNN).

A. PROCESS OF PROPOSED METHODOLOGY

This research study mainly intends to build a new CNN architecture with the sparsity features and optimization principle. Hence, the introduced novel approach is called STbCNA. Here, the pooling layer is changed to the optimal layer with the function of the Siberian tiger [31]. At the primary stage, the neuron initialization process was carried out by eq 1.

$$F(s) = \{s_1, s_2, s_3, \ldots, s_n\} \tag{1}$$

Here, the training process is determined as $F(s)$. and the database is described as s. Moreover, the n number of data samples are visualized as $s_1, s_2, s_3, \ldots, s_n$.

Here, p is the number of convolution layers, Q is the number of FC layers, and O is the optimal layer. Moreover, D_1, D_2, D_3 are the data bits, D_r is the dimensionality reduction, DB_1, DB_2, DB_3 are the database, M_1, M_2, M_3 are the mode functional operation, W_1, W_2, W_3 are weights and IFM is the input Feature map. Here, FP is the floating point, FX is fixed integer and OFM is the Optimal Feature map and represents weight of each neuron layers. The internal architecture is illustrated in fig 3 and the bit processing of control unit is shown in fig 4.

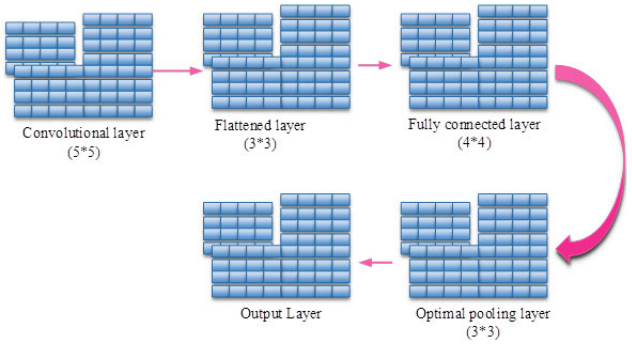

FIGURE 2. STbCNA:architecture.

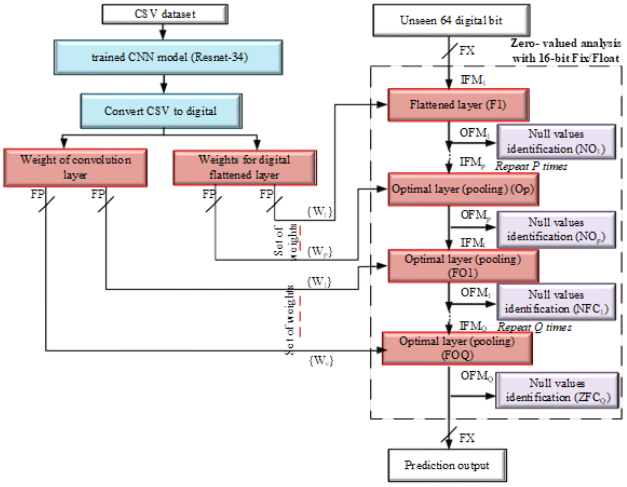

FIGURE 3. STbCNA:internal structure.

Also, the optimal model operation is exposed in algorithm1. This algorithm explained the optimization of pooling layer of the CNN. Generally, the pooling layer of the CNN carried out the dimensionality reduction of the features in the test data. By introducing the Siberian tiger fitness process [31] at the pooling layer of the CNN achieved the optimal features for the specific prediction in the dataset by tuning the pooling layer parameters.

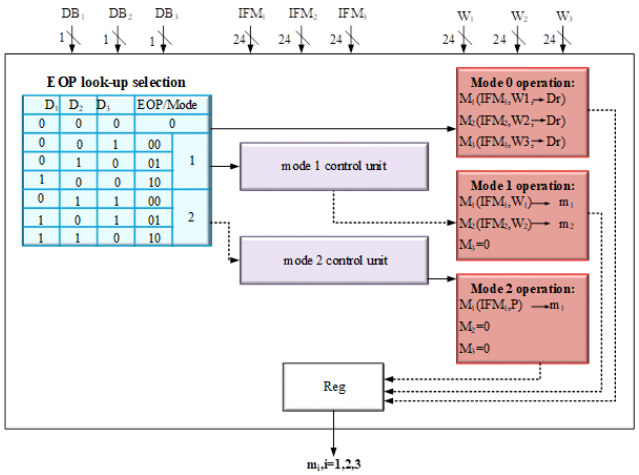

FIGURE 4. STbCNA:bit processing.

The optimized dimensionality reduction process attained maximum accuracy in the prediction. After the neuron initialization phase, the sparse features were updated as the normalization and null value remover. Hence, the outcome of the sparsity is a pre-processed normalized outcome. Henceforth, feature map functional blocks were designed in dual phases before and after the pooling function. So, the feature map operation after the convolutional layer process is mentioned in eq 2.

$$l^* = \frac{l-t+g}{h} + 1 \tag{2}$$

$$m^* = \frac{m-t+g}{h} + 1 \tag{3}$$

Here, the output feature map height is denoted as l^* and l defined feature map input height. The convolutional operation is represented as h, g denoted the padding function of convolution operation and t is the filter size. Besides, m^* is the width of the output feature map and m is the input feature map width. In addition, the feature map size of the STbCNA after the pooling function is measured by eq 3.

$$l^* = \frac{l-p^*}{h} \tag{4}$$

$$m^* = \frac{m-p^*}{h} \tag{5}$$

Here, the size of pooling area is represented as p^* and h is the pooling operation. In addition, the ReLU activation process is formulated in eq 4 and eq 5.

$$f(s)_{\text{ReLU}} = \max(0, s) \tag{6}$$

After designing all those functioning phases, the rainfall prediction database was considered, and the prediction was made in eq 6. Here, the obtained rainfall data is initialized in the neuron initializing function. Then, the noise filtering is executed by processing the sparse function, the null values were neglected using eq 7. The Operational architecture of

Algorithm 1 Optimal Pooling Layer(3*3)

1: **Start()**
2: $if\,(m_2, m_3 = 0, M_1 = p)$;
3: $M_1 = IFM(r_d)$
4: $W = W_1, W_2, \ldots, W_n$
5: $elseif\,(m_2, m_3 = 1)$
6: $m_2, m_3 = D_r$
7: $if\,(M_1(p^*) = max - acc)$
8: process end
9: else continue(tune-optimal pooling 3*3)
10: **Stop()**

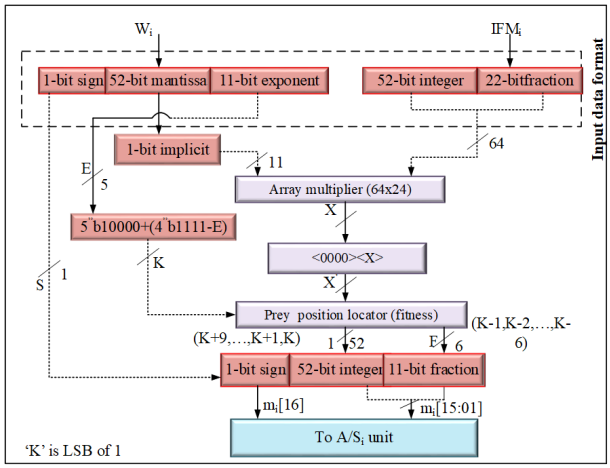

FIGURE 5. Operational architecture.

the novel CNN is exposed in Fig 5.

$$R = R_{i,j} + \frac{P_r - v}{R_s} \tag{7}$$

The rainfall prediction constraint is determined as R, and the rainfall constraints, such as humidity and temperature, are represented as $R_{i,j}$. Also, P_r is the predicted rainfall rate,v is the stored rainfall data and R_s is total rainfall rate. Moreover, the selected features from the feature map are denoted as the total rainfall statistics, which is stored in the STbCNA memory.

To offer the predictive outcome, the stored statistics were matched with the test rainfall features then the precipitation level was determined. Once the upcoming rainfall rate is predicted, that information is shared to the user clouds, making the public aware of the rainfall occurrence. The overall flow model is exposed in Fig 6, and the Algorithm is given in Algorithm 2.

V. RESULTS AND DISCUSSION

The newly designed novel accelerator is tested in the Xilinx platform and running in the Windows 10 platform. At the primary phase, the CNN accelerator is designed with all those steps process then the CNN architecture layer is changed by the optimal features and the sparsity condition. Hence, the

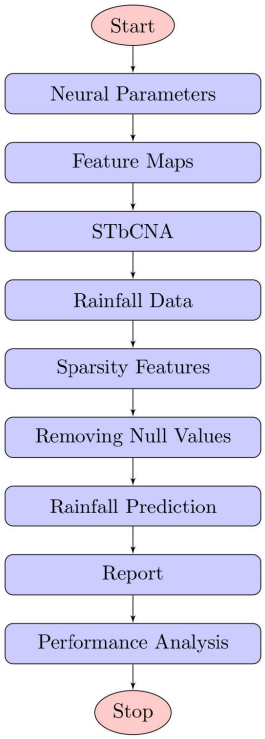

FIGURE 6. Process flow of STbCNA.

Algorithm 2 STbCNA

1: **Start()**
2: int $s = 1, 2, 3, \ldots, n$;
3: **Feature Map Creation()**
4: int l, m, l^*, m^*;
5: feature $= I/O$(height + width)
6: **ReLU Activation()**
7: fn(ReLU) $= \max(0, s)$
8: **Testing()**
9: int R, P_r, R_s, v
10: $P_r = R_s, R_{i,j}$
11: **Data Broadcasting()**
12: Broadcast($P_r \rightarrow$ cloud, user)
13: **Stop()**

TABLE 1. Execution constraints.

Execution Elements	
Optimization	Siberian Tiger
Deep Network	CNN
Software	Win 10
Programming Environment	Xilinx (Vivado 2018.2)
Language	VHDL
Testing Application Environment	MATLAB R2021a
FPGA Family	Kintex Ultrascale
Dataset	Rainfall data, flood data, Land use and Land cover
Data Type	CSV files

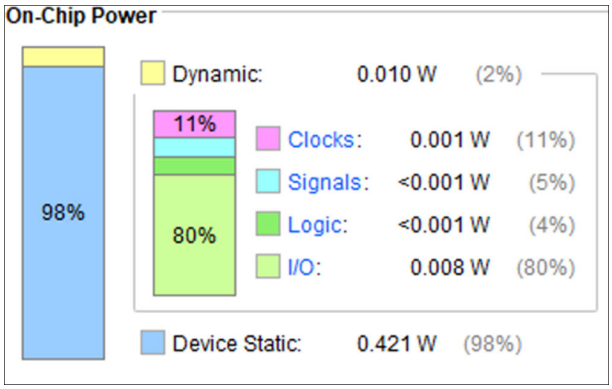

FIGURE 7. STbCNA chip status.

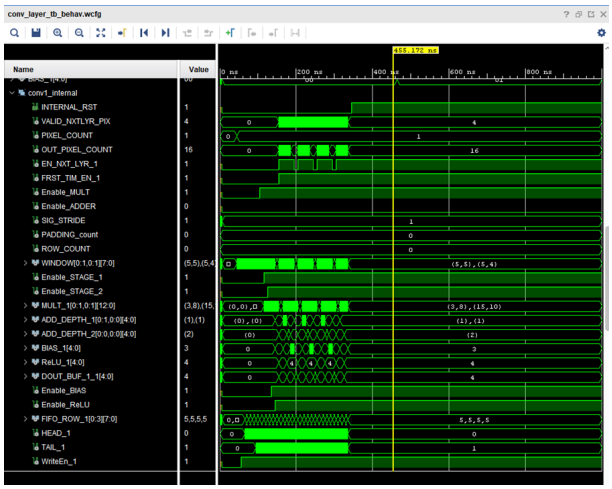

FIGURE 8. Waveform of proposed accelerator.

newly built accelerator is STbCNA; the execution constraints are exposed in Table 1.

After, the design of CNN accelerator, the performance testing process is done in the MATLAB environment. Here, for testing the accelerator, rainfall prediction application is considered.

The accelerator chip status of the newly implemented STbCNA is exposed in fig 7. It contains the power constraints that include static and dynamic Power. Then, the functional modules like clocks, signals, logic units, and I/O were measured.

The waveform outcome of the proposed accelerator is exposed in Fig 8, and the accelerator constraints details are tabulated in Table 2.

Here, the total LUT of the designed accelerator is 216960 in that 0.08% was utilized for rainfall prediction. In addition, the total FF is 433920, and the utilization percentage is 0.06%. Also, the total IO buffer is 304; 8.55% of the buffer was utilized. Moreover, the total buffer port is 256, and 0.39% was utilized. The sampling frequency of each channel signal is 218 Hz. The hyper-parameters of the model includes the number of layers, learning rate, number of kernels, kernel size, length of strides and pooling size which directly influence the training of the designed accelerator. The sparsity condition and the fitness process of the Siberian Tiger optimize all the hyperparameters of the

TABLE 2. Accelerator performance.

Accelerator Parameters			
Parameters	**Available**	**Utilization**	**Utilization (%)**
LUT	216960	166	0.08
FF	433920	266	0.06
IO	304	26	8.55
Buffer Port	256	1	0.39

TABLE 3. Performance of traditional CNN and proposed CNN.

Performance Parameters		
Parameters	**CNN**	**Proposed**
Throughput (*bps*)	120	150
Energy (*nJ*)	10	12
Aera (μm^2)	3000	2500
GOPS	50	70
DSP	10	8
DSP Efficiency	0.9	0.95

model for the better learning. The optimal parameters are chosen through the validation strategy. From the total dataset, 70% is taken for training, 15% utilized for validation and 15% for testing process of the designed accelerator. The validation performance of the model was accessed by the k-fold cross validation strategy. The key factors influencing the designed CNN accelerators are the computational resources such as energy, memory, bandwidth and utilization of processing element array. The computational resources measure the upper limit of the FPGA accelerator efficiency. The performance of the designed accelerator is measured through some of the evaluation metrics such as throughput, delay, area, DSP usage and energy. The model reduced the memory size by 57.6×. The small memory size of the optimized network comes out to 7.4Mb. The software running on the ARM handles the load and pre-processes the weights, bias and test data of the designed network. The pre-processing phase includes the rearranging the data in a titled format for the created accelerator.

A. PERFORMANCE ANALYSIS

To value the working function of the newly implemented model, a few key parameters were considered, and the validation was made with other traditional models. Some of the past studies like Bit Parameters CNN (BP-CNN) [29], Deep Convolutional Network Accelerator (DCNA) [30], Embedded features CNN (EF-CNN) [30] and Coati CNN (C-CNN). The performance parameters for the traditional CNN and the proposed CNN are depicted in Table 3.

The traditional CNN scored 120bps throughput, 10nJ energy utilization, 3000 area, 5-GOPS, 10% DSP utilization, and 0.9 DSP efficiency. Moreover, the proposed accelerator gained 150 bps throughput, 12 nJ energy consumption, 2500 area, 70GOPS, 8% DSP utilization, and 0.95 DSP efficiency.

Here, the FF for the existing models BP-CNN, EF-CNN, DCNA, and C-CNN attains 11.66%, 10.8%, 22.18%, and

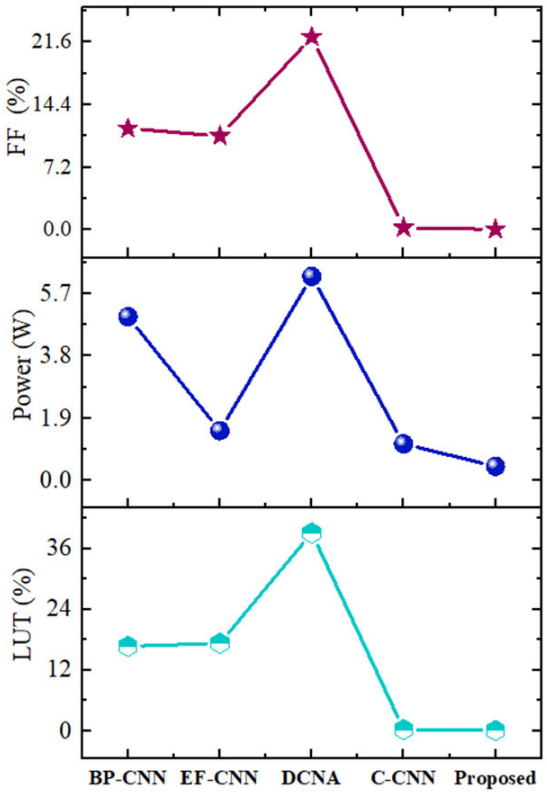

FIGURE 9. Comparison of FF (%), Power (W), LUT (%).

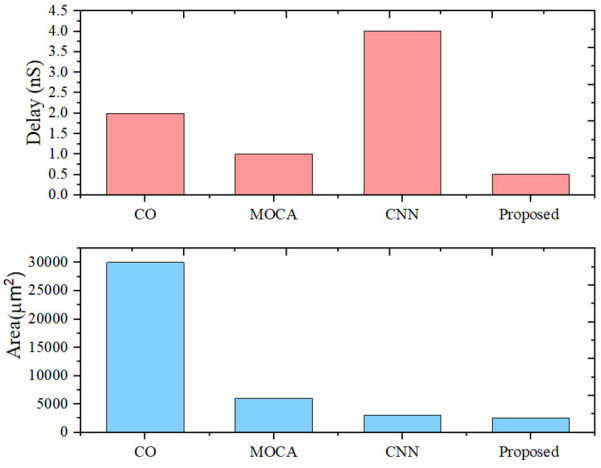

FIGURE 10. Comparison of area and delay.

0.27%, respectively, and the proposed model gains 0.06%. The Power achieved by BP-CNN is 5 W, EF-CNN is 1.52 W, DCNA is 6.23 W, C-CNN is 1.116 W, and the proposed model is 0.431 W. The LUT earned is BP-CNN is 16.71%, EF-CNN 17.3%, DCNA is 39.1%, C-CNN is 0.26% and the proposed model is 0.08%. Hence the comparison shows the performance of the proposed model is better as it gives very low FF, Power, and LUT. The comparison of these metrics is depicted in Fig 9. To validate the performance, traditional models like the Convolutional Operator (CO) and Multiplication Operation Convolutional accelerator (MOCA) [32] were adopted.

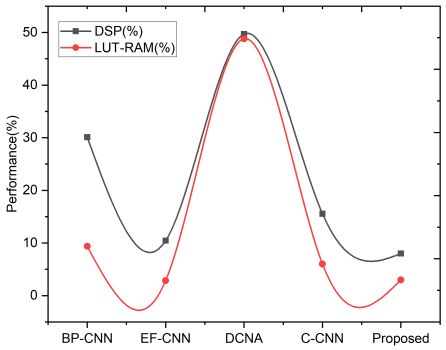

FIGURE 11. DSP performance.

TABLE 4. Entire comparison.

Comparative Analysis					
Parameters	**BP-CNN**	**EF-CNN**	**DCNA**	**C-CNN**	**Proposed**
LUT	16.71	17.3	39.1	0.26	0.08
Power (W)	5	1.52	6.23	1.116	0.431
DSP (BOPs)	30.11	10.45	49.7	15.56	8
LUT-RAM	9.37	2.86	48.8	0.04	0.001
FF	11.66	10.8	22.18	0.27	0.06

Here, the Area earned by the existing models is CO $30000\mu m^2$, MOCA $6000\mu m^2$, CNN $3000\mu m^2$ and the proposed model earns $2500\mu m^2$. The delay for the existing models CO is $2ns$, MOCA is $1ns$, CNN is $4ns$, and the proposed model is $0.5ns$. The value obtained by both Area and delay is less than the existing models, showing better performance. The comparison of Area and delay is displayed in Fig 10.

DSPs modify and evaluate a signal to enhance its performance or efficiency. It is designed to analog several mathematical algorithms such as addition, subtraction, multiplication, and division very fast and digital signals to generate their quality than the initial signal. The performance of the DSP is displayed in Fig 11.

The entire performance of the proposed model is shown in Table 4. This indicates that the proposed model performs better at a very low rate. The traditional model BP-CNN scored 30.11 DSP and 9.37 LUT-RAM, EF-CNN scored 10.45 DSP and 2.86 LUT-RAM, DCNA has gained 49.7 DSP and 48.8 LUT-RAM. Moreover, the C-CNN attained 15.56 DSP and 0.04 LUT-RAM usage. Considering all these, the newly designed accelerator gained the reduced DSP and LUT-RAM utilization as DSP 8% and LUT-RAM 0.001.

To justify the strength of the executed model, the rainfall data was considered for 2000 to 2023; for the testing, ten years of data were taken. The mean square error is exposed in Fig 12, training and testing presentation is given in Fig 13. Moreover, the outcome of the ten years of data is exposed in Fig 14.

The data collected is in CSV, Andhra Pradesh state, with locations such as Visakhapatnam, Vizianagaram, and Krishna district. In addition, the recorded accuracy for this rainfall prediction is 92.8%. Hence, it justified that the designed accelerator is appropriate for rainfall prediction.

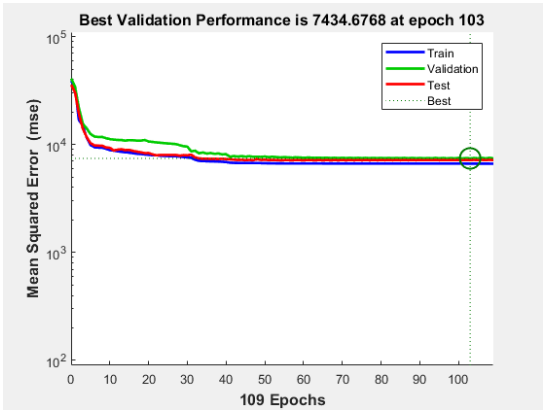

FIGURE 12. MSE validation.

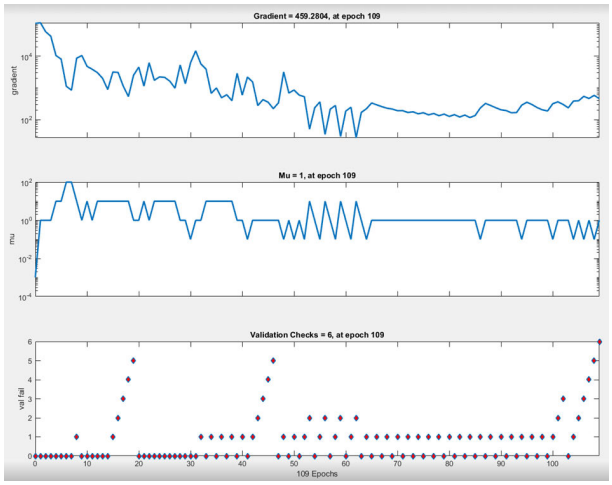

FIGURE 13. Training and testing.

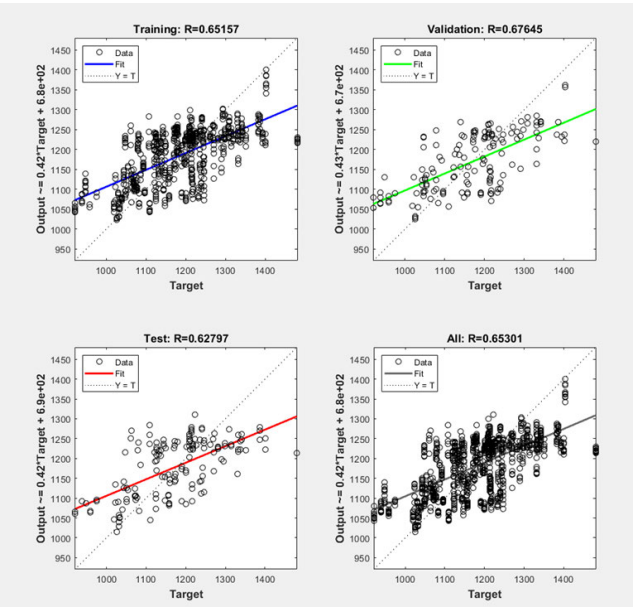

FIGURE 14. Rainfall prediction.

To validate that the present accelerator is better than the past accelerator, some of the existing CNN models for rain prediction were considered, and results were validated with

TABLE 5. Accuracy prediction comparison.

Models	Accuracy (%)
CNN [33]	54.9
U-Net with CNN [33]	72.4
Deep U-Net with CNN [33]	76.63
Proposed	**92.8**

the proposed approach. The proposed accelerator is more accurate than the compared models in Table 5.

VI. CONCLUSION

The present research work has built a novel CNN architecture named STbCNA. The designed accelerator has included sparsity and optimal layers. Moreover, the designed accelerator performance is checked in dual cases for prediction and data broadcasting. Here, for prediction, the different data is considered, such as rainfall data, flood data, land use and land cover data is considered to measure the Throughput of the built novel accelerator, and the predicted data is transferred to the cloud user as an alert. Finally, the designed accelerator is tested in the FPGA platform with the Xilinx tool. The designed model attained the optimized LUT of 0.08% utilization, total Power of 0.431 W, DSP 8, LUT-RAM of 0.001, and FF is 0.06%. Hence, the proposed accelerator has improved its performance by 3% compared to the traditional accelerator. It has been verified that the proposed accelerator is most appropriate for digital applications. However, the dictionary features are not incorporated in the present accelerator. In the future, incorporating the dictionary features will improve the technique's flexibility for different applications.

REFERENCES

[1] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," *Artif. Intell. Rev.*, vol. 53, no. 8, pp. 5455–5516, Dec. 2020.

[2] P. Dhilleswararao, S. Boppu, M. S. Manikandan, and L. R. Cenkeramaddi, "Efficient hardware architectures for accelerating deep neural networks: Survey," *IEEE Access*, vol. 10, pp. 131788–131828, 2022.

[3] G. Buchgeher, D. Gabauer, J. Martinez-Gil, and L. Ehrlinger, "Knowledge graphs in manufacturing and production: A systematic literature review," *IEEE Access*, vol. 9, pp. 55537–55554, 2021.

[4] P. Huang, L. Zeng, X. Chen, K. Luo, Z. Zhou, and S. Yu, "Edge robotics: Edge-computing-accelerated multirobot simultaneous localization and mapping," *IEEE Internet Things J.*, vol. 9, no. 15, pp. 14087–14102, Aug. 2022.

[5] A. Mohandu and M. Kubendiran, "Survey on big data techniques in intelligent transportation system (ITS)," *Mater. Today, Proc.*, vol. 47, pp. 8–17, Jan. 2021.

[6] A. A. Theodosiou and R. C. Read, "Artificial intelligence, machine learning and deep learning: Potential resources for the infection clinician," *J. Infection*, vol. 87, no. 4, pp. 287–294, Oct. 2023.

[7] S. Klein, S. Watted, and M. Zion, "Contribution of an intergenerational sustainability leadership project to the development of students' environmental literacy," *Environ. Educ. Res.*, vol. 27, no. 12, pp. 1723–1758, Dec. 2021.

[8] M. Tang, Y. Liu, and L. J. Durlofsky, "Deep-learning-based surrogate flow modeling and geological parameterization for data assimilation in 3D subsurface flow," *Comput. Methods Appl. Mech. Eng.*, vol. 376, Apr. 2021, Art. no. 113636.

[9] X. Han et al., "Pre-trained models: Past, present and future," *AI Open*, vol. 2, pp. 225–250, Jan. 2021.

[10] A. Kumar, S. Bhatia, K. Kaushik, S. M. Gandhi, S. G. Devi, D. A. De J. Pacheco, and A. Mashat, "Survey of promising technologies for quantum drones and networks," *IEEE Access*, vol. 9, pp. 125868–125911, 2021.

[11] P. R. Wills, "Origins of genetic coding: Self-guided molecular self-organisation," *Entropy*, vol. 25, no. 9, p. 1281, Aug. 2023.

[12] H. Wang, S. Guo, Z. Qu, R. Li, and Z. Liu, "Error-compensated sparsification for communication-efficient decentralized training in edge environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 1, pp. 14–25, Jan. 2022.

[13] S. Mittal, P. Rajput, and S. Subramoney, "A survey of deep learning on CPUs: Opportunities and co-optimizations," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 10, pp. 5095–5115, Oct. 2022.

[14] Y. Wang, G. Cao, and L. Pan, "Multiple-GPU accelerated high-order gas-kinetic scheme for direct numerical simulation of compressible turbulence," *J. Comput. Phys.*, vol. 476, Mar. 2023, Art. no. 111899.

[15] L.-H. Gong, J.-J. Pei, T.-F. Zhang, and N.-R. Zhou, "Quantum convolutional neural network based on variational quantum circuits," *Opt. Commun.*, vol. 550, Jan. 2024, Art. no. 129993.

[16] M. Marei, S. E. Zaatari, and W. Li, "Transfer learning enabled convolutional neural networks for estimating health state of cutting tools," *Robot. Comput.-Integr. Manuf.*, vol. 71, Oct. 2021, Art. no. 102145.

[17] Y. Wang, J. Wang, W. Zhang, Y. Zhan, S. Guo, Q. Zheng, and X. Wang, "A survey on deploying mobile deep learning applications: A systemic and technical perspective," *Digit. Commun. Netw.*, vol. 8, no. 1, pp. 1–17, Feb. 2022.

[18] T. Ahmad, R. Madonski, D. Zhang, C. Huang, and A. Mujeeb, "Data-driven probabilistic machine learning in sustainable smart energy/smart energy systems: Key developments, challenges, and future research opportunities in the context of smart grid paradigm," *Renew. Sustain. Energy Rev.*, vol. 160, May 2022, Art. no. 112128.

[19] B. Ramasubramanian, V. S. Reddy, V. Chellappan, and S. Ramakrishna, "Emerging materials, wearables, and diagnostic advancements in therapeutic treatment of brain diseases," *Biosensors*, vol. 12, no. 12, p. 1176, Dec. 2022.

[20] S. U. Rehman, S. U. Rehman, S. Tu, Z. Shah, J. Ahmad, M. Waqas, O. U. Rehman, A. Kouba, and Q. H. Abbasi, "Deep learning models for intelligent healthcare: Implementation and challenges," in *Proc. 7th Int. Conf. Artif. Intell. Secur. (ICAIS)*, Dublin, Ireland. Berlin, Germany: Springer, Jul. 2021, pp. 214–225.

[21] G. Li, M. Zhang, Q. Zhang, and Z. Lin, "Efficient binary 3D convolutional neural network and hardware accelerator," *J. Real-Time Image Process.*, vol. 19, no. 1, pp. 61–71, Feb. 2022.

[22] J. Lee and H.-J. Yoo, "An overview of energy-efficient hardware accelerators for on-device deep-neural-network training," *IEEE Open J. Solid-State Circuits Soc.*, vol. 1, pp. 115–128, 2021.

[23] X. Xie, J. Lin, Z. Wang, and J. Wei, "An efficient and flexible accelerator design for sparse convolutional neural networks," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 68, no. 7, pp. 2936–2949, Jul. 2021.

[24] K. Asifuzzaman, N. R. Miniskar, A. R. Young, F. Liu, and J. S. Vetter, "A survey on processing-in-memory techniques: Advances and challenges," *Memories-Mater., Devices, Circuits Syst.*, vol. 4, Jul. 2023, Art. no. 100022.

[25] T. Mohaidat and K. Khalil, "A survey on neural network hardware accelerators," *IEEE Trans. Artif. Intell.*, 2024.

[26] B. Peccerillo, M. Mannino, A. Mondelli, and S. Bartolini, "A survey on hardware accelerators: Taxonomy, trends, challenges, and perspectives," *J. Syst. Archit.*, vol. 129, Aug. 2022, Art. no. 102561.

[27] J. Hou, Z. Liu, Z. Yang, and C. Yang, "Hardware trojan attacks on the reconfigurable interconnections of field-programmable gate array-based convolutional neural network accelerators and a physically unclonable function-based countermeasure detection technique," *Micromachines*, vol. 15, no. 1, p. 149, Jan. 2024.

[28] I. Park, Q. Zheng, D. Manno, S. Yang, J. Lee, D. Bonnie, B. Settlemyer, Y. Kim, W. Chung, and G. Grider, "KV-CSD: A hardware-accelerated key-value store for data-intensive applications," in *Proc. IEEE Int. Conf. Cluster Comput. (CLUSTER)*, Oct. 2023, pp. 132–144.

[29] A. Kyriakos, E.-A. Papatheofanous, C. Bezaitis, and D. Reisis, "Resources and power efficient FPGA accelerators for real-time image classification," *J. Imag.*, vol. 8, no. 4, p. 114, Apr. 2022.

[30] X. Li, H. Huang, T. Chen, H. Gao, X. Hu, and X. Xiong, "A hardware-efficient computing engine for FPGA-based deep convolutional neural network accelerator," *Microelectron. J.*, vol. 128, Oct. 2022, Art. no. 105547.

[31] P. Trojovský, M. Dehghani, and P. Hanuš, "Siberian tiger optimization: A new bio-inspired metaheuristic algorithm for solving engineering optimization problems," *IEEE Access*, vol. 10, pp. 132396–132431, 2022.
[32] S. Deepika and V. Arunachalam, "Analysis & design of convolution operator for high speed and high accuracy convolutional neural network-based inference engines," *IEEE Trans. Comput.*, vol. 71, no. 2, pp. 390–396, Feb. 2022.
[33] G. Xu, Y. Wang, L. Wang, L. P. Soares, and C. H. Grohmann, "Feature-based constraint deep CNN method for mapping rainfall-induced landslides in remote regions with mountainous terrain: An application to Brazil," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 15, pp. 2644–2659, 2022.

HEMALATHA KURAPATI received the B.Tech. degree in electronics and communications engineering from Chirala Engineering College, Chirala, Andhra Pradesh, India, in 2008, and the M.Tech. degree from the Sri Indu College of Engineering and Technology and Jawaharlal Nehru and Technological University, Hyderabad, India, in 2012. She is currently pursuing the Ph.D. degree with the School of Electronics Engineering, VIT University, Vellore, India. Her research interests include VLSI designs, digital designs, deep learning, AI/ML, and hardware accelerators for machine learning. Her current research interests include performance improvement in hardware accelerators for convolutional neural networks with CNN models and architectures.

SAKTHIVEL RAMACHANDRAN (Senior Member, IEEE) received the bachelor's degree in electrical engineering from the University of Madras, in 2000, the master's degree in applied electronics from Anna University, in 2004, and the Ph.D. degree in low-power high-speed architecture development for signal processing and cryptography.

He was the Head of the VLSI Division, Vellore Institute of Technology (VIT), Vellore, India, from 2009 to 2012. During his tenure, he was instrumental in setting up the SoC Design Laboratory, FPGA/ASIC Laboratory, and Analog IC Design Laboratory in association with Cadence, Altera, and Texas Instruments, India, respectively. He was a Consultant/Corporate Trainer with Texas Instruments, Bengaluru; BOSCH, Bengaluru and Coimbatore; Pantech Solution, Chennai; and Wipro, Bengaluru. Prior to joining academia, he was the Co-Founder of Silicon to Chip Technologies, Salem, India. He has mentored 30 undergraduate, 60 post graduate students, nine Ph.D. students (three finished Ph.D.'s), and one M.Tech. (research). He is currently a Professor with the School of Electronics Engineering, VIT. He is the coauthor of the book "*Basic Electrical Engineering* published by Sona University in 2001 and the author of the "*VLSI Design*" published by S. Chand in 2007. He has published more than 60 peer-reviewed papers in international conferences/journals. He has delivered around 50 guest lectures/invited talk and hands on workshops in the areas of FPGA-based system design, full custom IC design, RTL to GDSII, and ASIC design. His research interests include low-power VLSI design and developing high-speed architectures for cryptography and image processing. His current research interests include neuromorphic computing and making efficient hardware for AI and ML.