Alexandria Engineering Journal (2022) 61, 795–809

HOSTED BY

ELSEVIER

Alexandria University

Alexandria Engineering Journal

www.elsevier.com/locate/aej
www.sciencedirect.com

Area and energy efficient shift and accumulator unit for object detection in IoT applications

Anakhi Hazarika [a,*], **Soumyajit Poddar** [a], **Moustafa M. Nasralla** [b], **Hafizur Rahaman** [c]

[a] *Indian Institute of Information Technology Guwahati, India*
[b] *Prince Sultan University, Saudi Arabia*
[c] *Indian Institute of Engineering Science and Technology, Shibpur, India*

Received 23 March 2021; revised 25 April 2021; accepted 28 April 2021
Available online 05 June 2021

KEYWORDS

Convolution operation;
Object detection;
MAC unit;
Approximate computing;
Embedded platform

Abstract Convolutional Neural Networks (CNNs) exhibit significant performance enhancements in several machine learning tasks such as surveillance, intelligent transportation, smart grids and healthcare systems. With the proliferation of physical things being connected to internet and enabled with sensory capabilities to form an Internet of Thing (IoT) network, it is increasingly important to run CNN inference, a computationally intensive application, on the resource constrained IoT devices. Object detection is a fundamental computer vision problem that provides information for image understanding in several artificial intelligence (AI) applications in smart cities. Among various object detection algorithms, CNN has emerged as a new paradigm to improve the overall performance. The Multiply-accumulate (MAC) operations, which are used repeatedly in the convolution layers of CNN, hold extreme computational complexity. Hence, the overall computational workloads and their respective energy consumption of any CNN applications are on the rise. To overcome these escalating challenges, approximate computing mechanism has played a vital role in reducing power and area of computation intensive CNN applications. In this paper, we have designed an approximate MAC architecture, termed Shift and Accumulator Unit (SAC), for the error-resilient CNN based object detection algorithm targeting embedded platforms. The proposed computing unit deliberately trades accuracy to reduce design complexity and power consumption, thus suiting the resource constrained IoT devices. The pipeline architecture of the SAC unit saves approximately 1.8× clock cycles than the non-pipeline SAC architecture. The performance evaluation shows that the proposed computing unit has better energy efficiency and resource utilization than the accurate multiplier and state-of-the-art approximate multipliers without noticeable deterioration in overall performance.

* Corresponding author.
E-mail address: anakhi.hazarika@iiitg.ac.in (A. Hazarika).
Peer review under responsibility of Faculty of Engineering, Alexandria University.

https://doi.org/10.1016/j.aej.2021.04.099

1. Introduction

In the evolution of smart cities, AI and machine learning (ML) especially CNN [1] plays a vital role. These techniques are efficiently and effectively used in several IoT applications [2,3] like intelligent transportation systems [4], unmanned aerial vehicle, smart grids [5], surveillance, and healthcare systems [6] of a smart city [7]. IoT devices that are responsible for generating big-data are integral constituent part of the aforementioned smart systems. The preceding techniques offer state-of-the-art learning and decision capabilities by analysing the big-data to support intelligent systems. The increase of smarter surveillance camera technologies and their advance processing proficiencies gain a lot of advantage in the field of real-time image and/or video analysis [8,9], i.e., objects detection, object tracking, and action recognition.

Object detection is a fundamental as well as a critical task in computer vision problems. Real-time object detection applications require high accuracy at low power for computing on embedded and/or edge-IoT devices. Traditional object detection approaches that use region-based and sliding window algorithms provide low accuracy and high computation time [10,11]. The advancements in object detection algorithms with convolutional neural network (CNN) has led to its application in various computer vision applications, including image classification, human behavior analysis, face recognition, and autonomous driving [12–16]. However, the rapid advancement in the underlying architectures and algorithms of CNN has led to increasing challenges in computations [13,17–20]. The multitude of parameters (input pixels, weights, intermediate pixels, bias, etc.) required to implement the CNN network layers increases the computational complexity [21]. Further, CNN architectures used in object detection algorithms have different layer parameters such as kernel size, number of channels, etc. This nonidentical layer structure makes it challenging to design generic computing hardware to deploy on embedded systems. More hardware resources are required to implement deeper CNN architectures to achieve better detection accuracy.

A CNN model consists of several convolution (Conv.) and pooling layers followed by a fully-connected (FC) layer, and they contrast significantly in the number of computations. A lot of iterative computations are involved in each layer to process the large size of the parameters. Convolution is a basic operation, and it comprises most of the computations of the CNN model. Table 1 illustrates the comparison of different CNN models in terms of the number of computations involved in convolution and fully-connected layers. It shows that the number of operations associated with the convolution layer is significantly higher than the FC layer. Hence, the convolution layers demand a large number of multiply-accumulate (MAC) operations with high execution time, power consumption, and area overhead. This increased computational complexity creates a barrier in the deployment of CNN models on embedded platforms. High throughput MAC units are used for processing the convolution operations as the total execution time of the entire computation depends on the speed of the MAC unit [22]. These limitations have opened up new avenues for computing architectures to reduce the execution latency of the convolution layer in CNN, power consumption, and hardware resources.

Table 1 Comparison in terms of number of computations involved in convolution and fully-connected layers.

Model	Parameter	Operations	
		Conv.	FC
AlexNet [13]	61 M	666.2 M	58.7 M
VGG-16 [20]	138 M	15.3 M	0.1 M
GoogleNet [17]	6.99 M	1.56 M	6 K
ResNet-50 [19]	25.9 M	3800 M	100 M

1.1. Motivation

Convolution operation in image processing is nothing but multiplication and accumulation of overlapping values of two input images. In general, convolution is the sum of dot products of pixels with the kernels across the width and height of the input image. A kernel is a small matrix of numbers with different size and different patterns of number that slides over the entire input image. The choice of the kernel is based on the desired functions for the image (smooth, blur, etc.). MAC operations are computed in an array of MAC processing units that typically consist of a multiplier, an adder, and an accumulator register that stores the result [23]. MAC operations are computed in an array of MAC processing units, as shown in Fig. 1. The Fig. 1 describes the convolution operation between an image matrix I and a 3×3 kernel K. The first pixel O_{11} of the output image O is calculated as:

$$O_{11} = p_{11} \times 1 + p_{12} \times 0 + p_{13} \times 1 + p_{21} \times 0 + p_{22} \times 1 + p_{23} \times 0 + p_{31} \times 1 + p_{32} \times 0 + p_{33} \times 1. \quad (1)$$

In a convolution layer of CNN, repeated MAC operations result in high latency and power consumption. Computational complexity and energy efficiency are the major challenges while implementing convolution layers in hardware. Despite these computing challenges, resource constraints in embedded devices create bottleneck due to the large size of parameters associated with the convolution layers of a CNN model. Hence, real-time CNN architectures demand a large amount of memory space to store and logic units to process the data in embedded devices. Therefore, the architecture of a computing unit that incurs minimal area overhead is necessary to efficiently implement the convolution layers. A plethora of algorithms and architectures of MAC units are available in the literature; however, designing a high throughput MAC architecture for image processing applications with low execution time and high energy efficiency is still a promising research paradigm. Designing the multiplier of a MAC unit with approximate computing is an emerging trend in this domain. Approximate computing introduces errors in the computational results making it suitable for error-resilient applications such as computer vision, machine learning, etc. Thus, reducing the area overhead and power consumption by maintaining the overall performance of a CNN application is the primary motivation of this work.

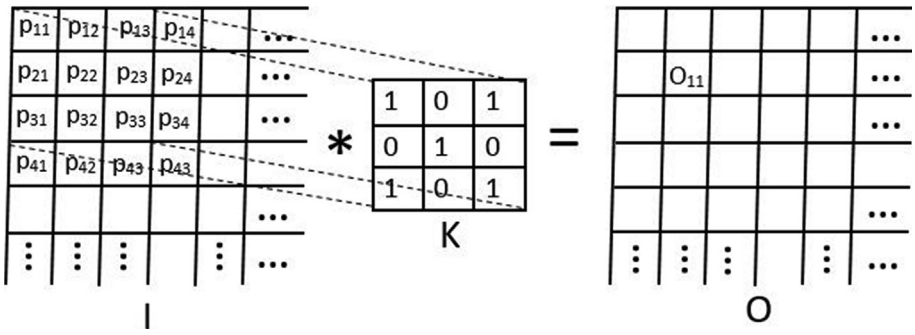


Fig. 1 Illustration of 2D Convolution Operation.

1.2. Contributions and organization

To address the limitations of state-of-the-art MAC architectures [22,24–27], the paper presents an architecture of a convolution processing unit with an approximate computing mechanism that reduces the area overhead and power consumption. The proposed architecture computes pixels of reduced bit length while processing an image for CNN based object detection algorithms. The proposed approximate computing unit, termed as shift and accumulator (SAC), introduces insignificant amounts of errors that slightly degrade the image quality than the image quality processed by the accurate MAC unit. Applications of computer vision, machine learning are inherently resilient to small inaccuracies. Hence, we have successfully simulated the CNN based object detection algorithms with the SAC unit targeting embedded platform. Finally, the performance analysis of SAC architecture along with the object detection algorithms are presented, depicting the impact on logic utilization, power consumption, and object detection.

We summarize the main contributions of this paper as follows.

- A convolution processing architecture (called SAC) is proposed that executes convolution operations with the approximate computing mechanism. In addition, optimized hardware of the SAC unit is presented to further reduce power consumption.
- A pipeline architecture of the SAC unit is developed by using a non-overlapping two-phase clock to accelerate the overall computing speed.
- CNN based object detection algorithms are implemented with the proposed SAC unit. We have adopted tiny YOLO CNN architecture for object detection and MT-CNN architecture for face detection.
- We compare and present the hardware synthesis results of SAC and other state-of-the-art multiplier architectures in terms of logic utilization and power consumption. Also, image quality assessment of the output images is presented. Further, we have simulated the object detection algorithms on Keras deep learning environment and detect objects.

The rest of the paper is organized as follows. Section 2 describes the prior works related to this area. The SAC architecture is described in Section 3. In Section 4, object detection algorithms are explained and discussed. The experimental results are presented in Section 5. Finally, the conclusions are drawn in Section 6.

2. Related research

In this section, architectures of accurate and approximate MAC has been discussed. The state-of-the-art MAC architectures designed for convolution operations are typically comprised of a multiplier, an adder, and an accumulator register [23]. The wide range of available MAC units is compared to each other with the different architectures of the aforementioned operational parts. Cowlishaw et al. [28] present a hardware implementation of the decimal arithmetic unit for commercial and financial applications. This hardware unit addresses the decimal to binary conversion errors. However, the use of decimal arithmetic results in higher execution time compared to binary arithmetic implementation.

In embedded signal processing applications, the multiplication is a computationally expensive operation than other arithmetic operations. To address this issue, Robison [29] proposed a hybrid algorithm by using a fused multiply-add (FMA) operation that reduces the complexity of multiplication operation. The FMA processing chips are commercially available in many processors such as IMB [30], HP [31], and Intel [32]. Samy et al. [33] presents a hardware design of a fully parallel FMA unit for decimal floating-point operations. This unit supports both decimal-64 and decimal-128 IEEE 754-2008 standard format. The proposed unit is specially designed to execute $\pm(A \times B) \pm C$ but can also execute addition, subtraction, and multiplication operation separately. The design of multiplier and adder determines the overall execution speed of the MAC unit while processing a signal. The MAC architectures with reversible logic are presented in [34,35] that reduce the usage of logic gates, garbage outputs, constant inputs, and hence the hardware circuit complexity. Also, power consumption and execution delay are much lower than the existing MAC architectures. However, delay in accumulation operation is higher in these MAC architectures. Sharif and Prasad [22] presents a 64-bit MAC unit that uses Vedic Multiplier and Ripple Carry Adder to reduce the area overhead. In addition, the paper compared the performance of Wallace multiplier and Vedic multiplier and observed the latter to be more efficient in terms of area. In [36], an area efficient and low delay MAC unit is presented for exponent addition in single-precision floating-point operations. In this architecture, the sign, exponent, and mantissa parts of the original number

are extracted separately and then processed parallelly in different operational blocks. Yengade et al. [37] present MAC architectures for 8-bit, 16-bit, and 32-bit operand with Baugh-Wooley Multiplier. The multiplier helps reduce execution delay, and the pipelined architecture increases the power efficiency of the MAC unit. Several MAC architectures are proposed for convolution layers in CNN. Garland et al. [38] compresses the shared weight in the MAC to accelerate the convolution operations. Hardware implementation of this architecture achieves low power consumption and area utilization. Although the above accurate computing units [29,33–35,22,36–38] provide better accuracy in computing results, these units are computationally complex and thus, consume more energy and area on embedded devices.

In recent years, approximate computing has emerged as a low-power design paradigm in hardware that improves performance, power, and area utilization. Several researchers [39,24,40,41] have introduced MAC architectures with approximate computing to leverage their aforesaid benefits. Kim et al. [27] present an in-depth analysis of various approximate MAC architectures used for CNN MAC operations. It identifies the fact that in a MAC operation, the multiplication should be approximated, while the addition should be exact to reduce the overall error that occurs during approximation. Kulkarni et al. [39] proposed a multiplier that calculates the final result by adding outputs of approximate partial product units. The major limitation of this work is that the performance of the computing unit degrades when the input size is increased. Hashemi et al. [24] proposed a multiplier unit (drum) that dynamically selects the most relevant bits of the operands to process the multiplication operations. The rest of the bits are discarded to reduce the computational complexity of the hardware. The approximation of operands in this scheme reduces the size of the multiplier at-least half than the operands bit length to attain the desired accuracy. Further, to minimize the mean errors in computing results, drum introduces higher latency and power consumption. Imani et al. [40] proposed an approximate floating-point multiplier to avoid the expensive multiplication of fractional part by discarding one of the input mantissae. This design provides better area and power trade-off at the cost of maximum output error. Aggressive input approximation increases the error rate by up to 50%. In addition, separate hardware has been made to enable accuracy tuning while recognizing high output errors. Boroumand et al. [41] proposed a multiplier architecture whose accumulation unit has been replaced by an approximate compressor unit. This work also presents a tool that helps the user to explore the design space with minimal errors, area, and power. Neural network applications meet the desirable accuracy at the cost of high energy consumption and resource utilization. Works described in [42,26,25,43], utilizes the approximate multipliers to bound the hardware resource and power consumption within a tolerable range. Lee et al. [42] present a run-time configurable Unified Neural Processing Unit (UNPU) with variable bit-precision. This work configures the MAC unit for three different bit width of weights, and performs $8b \times 8b, 8b \times 4b$ and $8b \times 2b$ operations. Operations of reduced bit width in neural networks introduce a significant amount of errors in accuracy. Maki et al. [26] proposed a MAC architecture that processes weights of variable bit precision. This architecture reduces the execution time and improves errors in accuracy. The filter-wise optimization technique presented in this paper can effectively reduce memory footprints but unable to reduces computational costs. Allocations of bit width for weights depend on the size of convolution filters. TOSAM [25] is a scalable approximate multiplier architecture that truncates and round the input operands. This design achieves the approximate product by shifting, adding, and fixed-width multiplication operations. In this architecture, output errors depend on truncated bit width, and this width cannot be changed dynamically in run-time.

Few more approximate multipliers are found in literature for neural networks inference [44–47]. In [48], authors have conducted an in-depth analysis of different MAC architectures of lower bit precision. The study reveals that undoubtedly at full precision, MAC architecture achieves higher performance. However, processing architectures with reduced bit precision exhibit the best trade-offs between energy consumption and area overheads. Table 2 briefly summarizes the approximation mechanism of floating-point multiplication adopted in closely related works and their advantages and limitations.

To achieve further improvement in area and power efficiency of computation-intensive CNN applications, we present an approximate computing unit that executes convolution

Table 2 Comparison of closely related works.

Work	Mechanism	Advantages	Limitations
[24]	• Dynamically selects the important bits from the operands.	• Scalable MAC architecture; • Reduce computational errors.	• High latency; • Increase hardware overhead in signed multiplications.
[25]	• Truncation and rounding of input operands.	• Reductions in area, energy consumption, and delay compared to the exact multipliers.	• Can not change the width of truncated bits dynamically; • Increase in truncation width increases output error.
[26]	• Filter-wise weight quantization with variable precision.	• Reduces CNN's execution time; • Reduces memory footprints.	• Reduce recognition accuracy; • Unable to reduce computational cost.
[40]	• Discard one of the input mantissa and use the second directly.	• Saves energy in multiplication.	• Aggressive input approximation; • Hardware overhead.
[41]	• Use three approximate compressors for partial product reduction.	• Low design space; • Reduce power consumption.	• Area and latency increases while approximating high-level operations.

operations. The proposed computing architecture with reduced bit precision exhibit the best trade-offs between power consumption and area overheads while implemented on CNN-based object detection algorithms. This approach compromises the image quality without deteriorating the accuracy of object detection. In [49], we presented a preliminary version of the proposed architecture. The differences of this work and [49] are as follows: Firstly, this paper implements a complete and optimized architecture of the proposed computing unit to reduce overall area and power consumption while processing images. Secondly, we have designed a pipeline architecture of the SAC unit to increase the computation speed. Thirdly, the efficacy of the proposed architecture is validated on CNN-based object detection algorithms. Finally, an in-depth analysis of resource and energy consumption by SAC has been made and present a comparative analysis with the state-of-the-art architectures.

3. Architecture of proposed approximate computing unit

In this section, we present an approximate computing architecture (called SAC), specially designed for convolution operations that intelligently trade off hardware implementation and/or result accuracy for performance or energy gains. Approximate computing deliberately introduces an insignificant amount of error to the output of error-resilient applications to improve the computing hardware unit's overall area and power efficiency [24,27,39–41]. The goal of approximate computing is to obtain gain in computational throughput by eliminating the exactness of traditional computing. The proposed SAC architecture considers the aspect of computation with reduced bit-precision that can be implemented on an embedded platform with fewer hardware resources and power consumption. As discussed above (Section 1), the multiplier unit of a MAC introduces maximum computation delay and computational complexity than the other functional units.

To address this issue, we have designed an approximate multiplier unit that can replace the exact multiplier unit used in MAC operations. The underlying architecture of the SAC unit is shown in Fig. 2. The architecture is primarily comprised of two building blocks: *Bit Extracting Unit* (BEU) and *Bit Replacing Unit* (BRU). The Leading one detector (LOD) present in BEU traverses the input pixels from MSB and detects the occurrence of the first '1'. The remaining bits in the bit stream remain same. A priority encoder takes the bit stream as digital input and gives the position of the first one as binary output. After determining the position of the first 1, we extract the next three consecutive bits, including the first 1. Also, we have designed SAC architectures of different bit-length to deal with the input of larger bit-length to maintain accuracy. Extracted bit-length of the architectures SAC_4 , SAC_5 and SAC_6 is 4, 5 and 6 respectively. The comparator in BRU compares each bit of the extracted bit stream with the 1. If the bit is 1, then it is replaced by the weight of the kernel; otherwise, it remains the same. Finally, we left-shift the bit values with the help of 8-bit barrel shifter and sum up the values together to produce the desired output. The primary advantage of using a barrel shifter is to shift n-bits within a single clock cycle. Fig. 3 illustrates the flow of operations computed in the proposed SAC unit. Re-configurability of this architecture can be claimed with the fact that we can use a variable length of

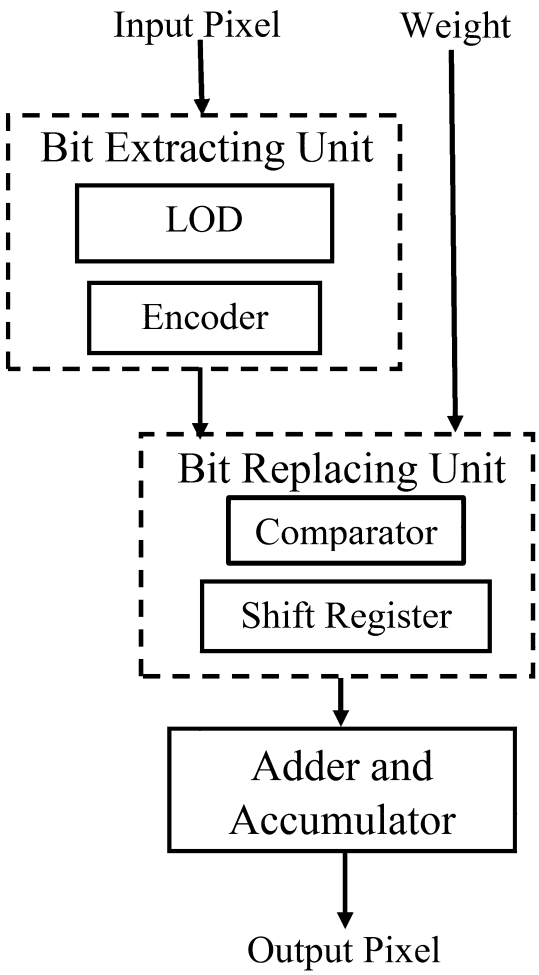


Fig. 2 Proposed SAC architecture.

reduced bit stream depending on the quality of input image and purpose of the convolution operation. To produce the output image, the bit length of the output image pixels must be the same as the input pixels. Hence, a mathematical expression is derived that provides us the number of 0s to be added at the LSB of convolved output. Besides, we truncate the extra bits from MSB to achieve the desired bit-length in the output pixels. The mathematical expression is shown in Eq. (2):

$$Z = B_T - S - B_B, \tag{2}$$

where Z = Number of Zeros to be added; B_T = Total number of bits present in the input pixels; S = Number of bits shifted to achieve first one from MSB; B_B = Number of bits in the extracted bit stream. In this equation, input S is dynamically known as it depends on the position of first one of each input pixels. Hence, S varies in run-time. However, B_T and B_B are user defined static inputs. During hardware realization of Eq. (2), S is fetched from the output of priority encoder via a substractor circuit.

An illustrative example: We consider an illustrative example to understand the working of the proposed computing architecture. A convolution operation with 3×3 kernel consists of nine dot products, as shown in Eq. (1). However, for simplicity in illustration, we consider a single dot product operation computed in the proposed architecture. We assume an

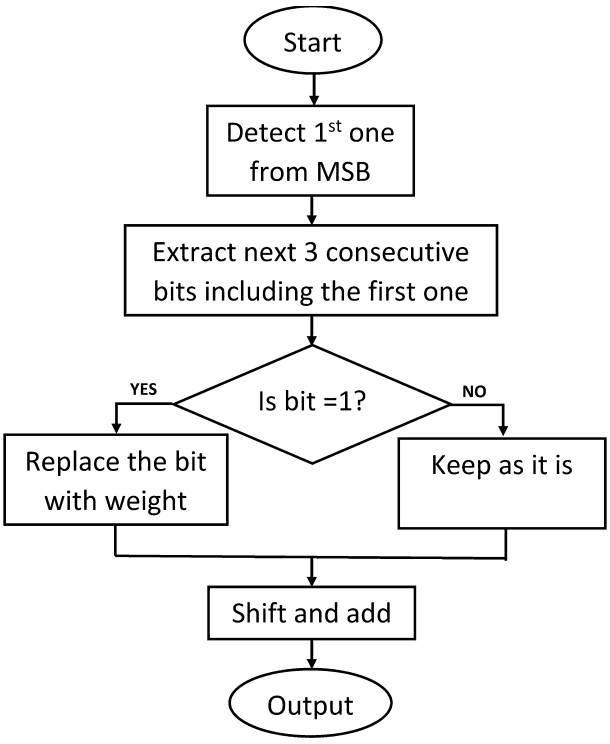
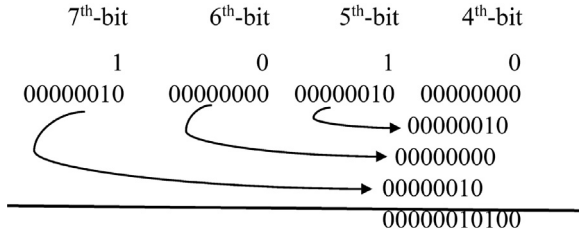


Fig. 3 Flowchart of computational steps in SAC architecture.

input pixel of value 82 whose 8-bit binary equivalent is 01010010 and a weight value 00000010. Firstly, the pixel bit-stream is traversed left to right from MSB, and the first 1 is located at the bit position 7. Therefore, considering the next three consecutive bits, we finally extract 1010 for further processing. Next, 1s are replaced by the weight and then left-shifted and summed up, as shown below:



To maintain the accuracy of the convolution operation, we add 0s at the LSB by following the Eq. (2). In this case, $B_T = 8$, $S = 1$, and $B_B = 4$. Hence, $Z = 8\text{-}1\text{-}4 = 3$. Now, the output pixels become 00000010100000. We have truncated six 0s from MSB to maintain the desired bit length at the output pixel. The decimal equivalent of the final output pixel 10100000 is 160, ideally, which should be 164 ($= 82 \times 2$).

3.1. Design optimization of SAC architecture

Nowadays, approximate computing architectures have emerged as a promising paradigm to reduce the design area and power of real-time, computation-intensive computer vision applications. Considering this paramount aspect, we have optimized the proposed SAC architecture for further reduction of power consumption. In the convolution processing of an entire image, pixels are computed redundantly. As discussed earlier, BEU extracts the bit stream, which is further computed by the proposed SAC architecture. When a kernel slides over an image, few pixels get convolved again with the weights, and hence, BEU processes the same pixels. To avoid this repetition, the SAC architecture is optimized by inserting a multiplexer and a counter, as shown in Fig. 4. In this architecture, the position of the first set of pixels computed by the BEU is fed to the BRU through the signal *pos1*. In the same time, the positions are stored in a register. When the bit extracting unit completes the first set of computations, a *done_signal* generates and enables the counter. The size of the register and state of the counter depends on the size of the kernel. The counter drives the select signal of the multiplexer and fetches the position through the signal *pos2* from the register to the bit replacing unit. During this period, bit extracting unit remains idle and saves power.

For further understanding, let us consider a kernel of size $k \times k$ and stride S that convolved with an image. In the first set of computation, $k \times k$ numbers of pixels are processed through a bit extraction unit. When the kernel slides, the down-counter having $k \times k$ states enables the select signal. The counter keeps the *pos2* signal active from $k \times k$ state to $k \times S$-1 state and *pos1* signal from $k \times S$-1 to 0. The BEU remains idle and saves energy, while the *pos2* signal remains active. The schematic representation of this optimized computing mechanism with S = 1, k = 3 is shown in Fig. 5. When t = 0 (present computation), the bit extracting unit remains idle for repeating pixels, and the results come directly from the registers that have already been computed and stored at time t = -1 (previous computation). The power consumption by each individual design unit of the optimized SAC architecture while computing a 3×3 is summarized in Table 3. The table shows that the amount of power consumed by the bit extracting unit is double the multiplexer and counter unit. Further, an analysis of power comparison between the original SAC and Optimized SAC is presented. Table 4 shows the effectiveness of skipping strategy used in the optimized SAC architecture. The optimized SAC architecture achieves 22% reduction in power consumption than the original SAC architecture. Thus, power can be saved for different hyper-parameter (input size, kernel size, stride, etc.) configurations.

In addition, we design a pipeline architecture of the optimized SAC unit to save clock cycles. The pipeline architecture has four stages, as shown in Fig. 6. Latches B_{12}, B_{23}, and B_{34} are used to isolate the inputs and outputs between two pipeline stages. To ensure correct pipeline operation, a non-overlapping two-phase clock is used for the consecutive stages. When the clock is applied, all latches transfer data to the next stage simultaneously.

4. CNN based object detection

In many computer vision applications, object detection is a challenging task for complete image understanding and to provide critical information [50]. Object detection technique not only locates objects in any image but also classify and label with bounding boxes. Face detection, skeleton detection are the sub-tasks of object detection. Among several object detection algorithms found in literature, deep learning systems are

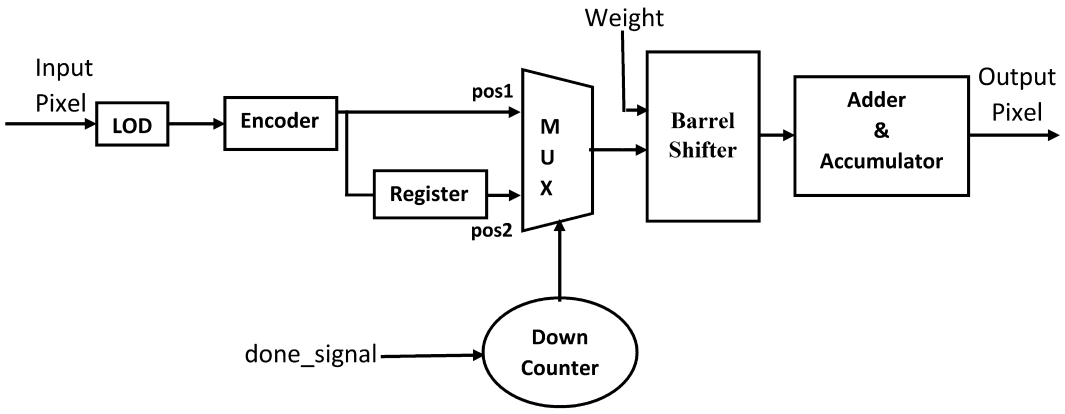


Fig. 4 Optimized SAC architecture.

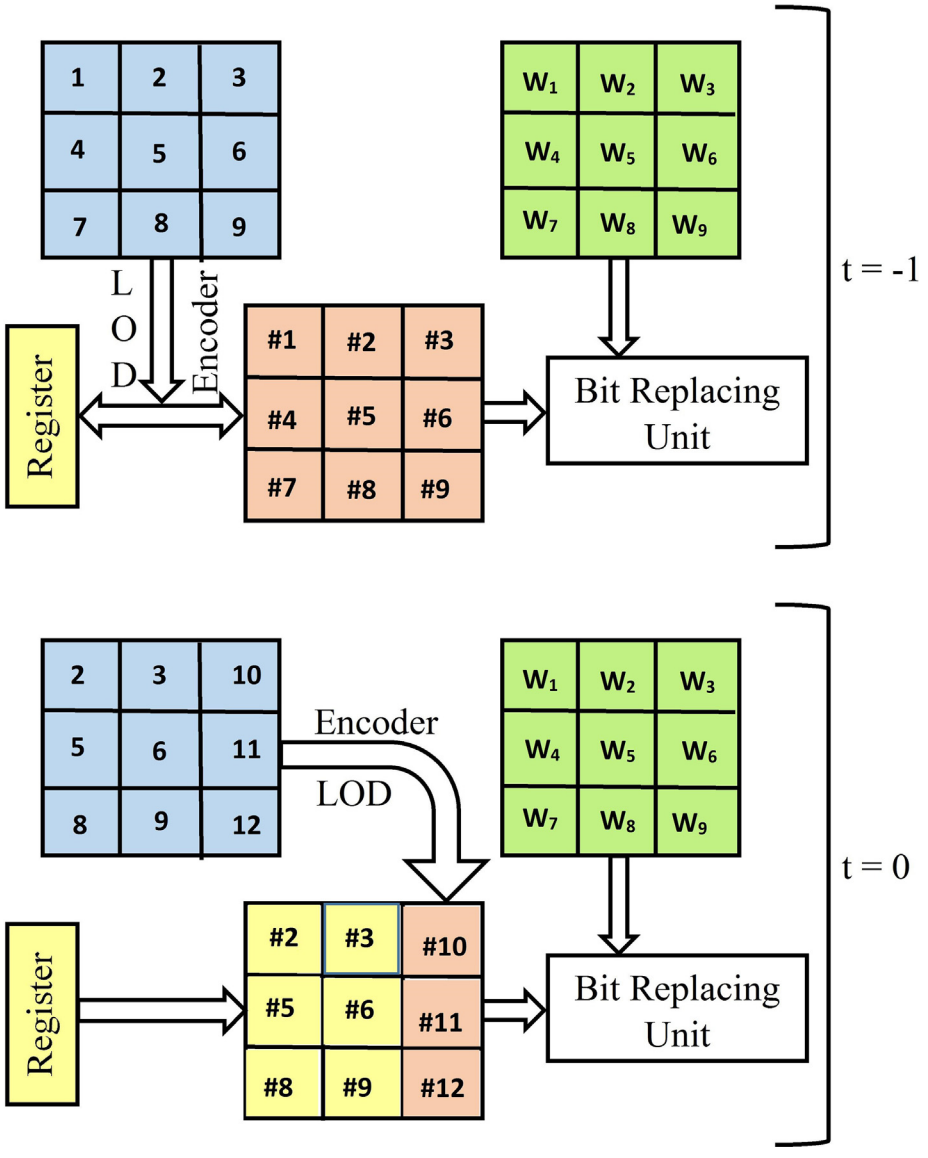


Fig. 5 Schematic representation of optimized computation.

Table 3 Power profile of different hardware units of SAC.

Unit	Power		
	Static (μW)	Dynamic (μW)	Total (mW)
Bit Extraction Unit	519.2	985.8	1.505
Multiplexer + Counter	0.194	0.423	0.617
Bit Replacing Unit + Adder	874.7	1186.6	2.061

Table 4 Power consumption by original and optimized SAC architecture.

Architecture	Total Power
Original SAC	8.328 mW
Optimized SAC	6.492 mW

widely used in this area. Recently, CNN based object detection algorithms such as single-shot-multibox-detection (SSD) [51], faster R-convolutional neural network (R-CNN) [52], multi-task cascaded CNN (MT-CNN) [53], you-only-look-once (YOLO) [54] perform better trade-offs between accuracy and speed of object detection. Deep CNN architectures have the capacity to learn complex features, and different CNN algorithms offer different degrees of object detection accuracy. In SSD architecture VGG-16 CNN model is used as a base network that consumes 80% of total time during training. The integration of any other faster CNN model as a base network could have increased the SSD's performance. R-CNN is the pioneer work in the field of CNN based object detection. It is based on CNN's feature extraction and classification ability; however, R-CNN consumes a high amount of time, energy, and computation while processing the massive data. Although R-CNN models are more accurate in real-time object detection applications, YOLO models are used popularly due to its higher speed and small-sized architecture. MT-CNN approach uses a classical feature-based cascade classifier for face detection. Cascaded CNNs require high computational expense for bounding box calibration in the training stage. Hence, the implementation of these computation-intensive object detection algorithms in hardware leads to considerably high resource utilization and power consumption.

To alleviate the issue, this work implements two CNN-based object detection algorithms with the proposed SAC unit. Area and Power analysis of the SAC architecture is presented in Section 5.3. The object detection algorithms MT-CNN and YOLO successfully detect the objects present in the input images. However, an insignificant reduction of output image quality is observed due to the approximate computing mechanism. Further, object detection metric Mean Average Precision (mAP) is studied that helps in finding efficiency of SAC based object detection algorithms. mAP gives the precision of correct predictions for the entire algorithm in numerical values which make it easier to compare.

5. Performance analysis and discussion

In this section, the performance of the proposed SAC architecture is evaluated and compared with the accurate MAC architecture and state-of-the-art approximate computing architectures [24,25]. In SubSection 5.1, the accuracy of the proposed approximate computing has been evaluated with various error metrics and compared with other state-of-the-art approximate computing architectures. Also, the SAC architecture is simulated on MATLAB [55] for output image quality assessment with quality metrics, namely, *PSNR* (Peak Signal-to-Noise Ratio) and *SSIM* (Structural Similarity) index and presented in SubSection 5.2. In addition, hardware analysis of the proposed architecture is evaluated in 5.3 with the help of Synopsys [56] tool. Object detection algorithms are simulated on the Keras [57] environment in the anaconda tool [58].

5.1. Accuracy analysis

In this section, an analytical model of the proposed computing mechanism is implemented on MATLAB for accuracy analysis. Several well-known error metrics used in approximate computing are discussed to evaluate the approximation errors of the proposed architecture [59]. These error metrics are defined as the arithmetic calculation between erroneous and correct binary numbers. Hence, no units are used in error representation. We have considered five error metrics for the proposed SAC architectures whose inputs are of 8-bit each. The mathematical representation of these error metrics are listed below:

- **Error Distance (ED)**: ED is the arithmetic difference between the exact (O) and approximate output (O').

$$\mathrm{ED} = |O' - O| \tag{3}$$

- **Mean Error Distance (MED)**: MED is the average of ED values obtained for a set of n-bit inputs. This metric is effective for multiple bit design.

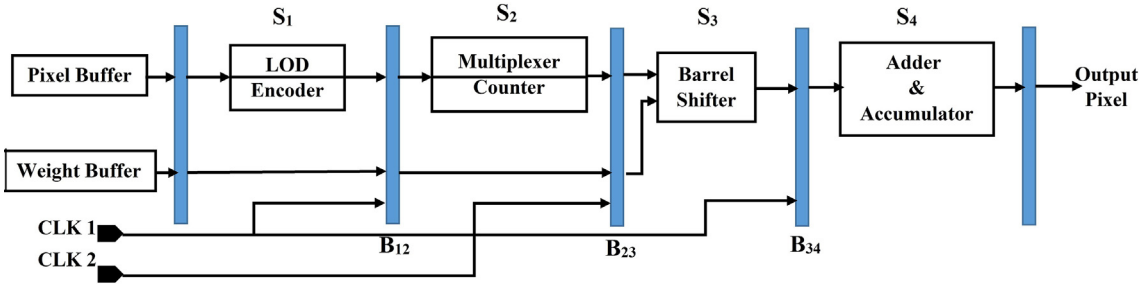


Fig. 6 Pipeline architecture of SAC unit.

$$\mathrm{MED} = \frac{1}{2^{2n}} \sum_{i=1}^{2^{2n}} |\mathrm{ED}_i| \tag{4}$$

- **Normalized Error Distance (NED)**: To compare the architectures of different bit sizes, ED has been normalized by the maximum error (E_{max}).

$$\mathrm{NED} = \frac{\mathrm{MED}}{E_{max}} \tag{5}$$

- **Mean Relative Error Distance (MRED)**: MRED is the average of ratio between ED and O.

$$\mathrm{MRED} = \frac{1}{2^{2n}} \sum_{i=1}^{2^{2n}} \frac{|\mathrm{ED}_i|}{O} \tag{6}$$

- **Mean Square Error (MSE)**: MSE is defined as the average of the squares of the errors (ED).

$$\mathrm{MED} = \frac{1}{2^{2n}} \sum_{i=1}^{2^{2n}} |\mathrm{ED}_i|^2 \tag{7}$$

A set of 1000 random numbers ranges from 0 to 2^8 are used to determine the computing units' approximate errors with the help of these error metrics. Scalability of the proposed SAC unit is studied for three different configurations such as SAC_4, SAC_5, and SAC_6 based on extracted bit length. The length of the processing bits in SAC_4, SAC_5, and SAC_6 are 4, 5 and 6 respectively. As shown in Table 5, SAC_6 provides the best error handling capacity among all the approximate computing architecture. Error metrics of the TOSAM architecture are better than SAC_4 configuration. Fig. 7 demonstrates the histogram distribution for error metrics of different SAC configurations and other approximate computing units. The proposed SAC_6 architecture gives better accuracy than DRUM and TOSAM.

5.2. Image quality assessment

In many computer vision applications such as image classification by CNN, image filtering, etc., traditional convolution processing unit lowers the output image quality as it eliminates or reduces unnecessary details of an image. Computation without noticeable degradation is acceptable in the scenario of compute-intensive applications [60]. Further, approximate computing architectures intentionally introduce errors to reduce computational complexity, area, and power consumption. In this section, we provide a comparative analysis of output image quality that is processed by an accurate computing unit (i.e., MAC) and the proposed approximate computing unit (SAC). PSNR and SSIM index are the parameters used for performance measures. PSNR is the ratio between the maximum power of a signal and power noise that degrades the image quality. Note that, higher the PSNR value better the image quality [61]. The SSIM index measures the similarity between the input image and the output processed image based on luminance, contrast, and structure. Typically, the value of the SSIM index is between −1 and 1, where value 1 indicates perfect, and 0 signifies no structural similarity.

To perform the image quality assessment, we model the accurate MAC and SAC architecture in the MATLAB environment using fixed-point simulation. We evaluate these architectures using two different kernels from the image processing domain. A grayscale image of size 128×128 is convolved by a 3×3 Gaussian-blur kernel and a 3×3 Smoothing kernel separately. Table 6 compares the result of the proposed approximated design with the accurate MAC on the test image by using two different kernels. It depicts that the PSNR and SSIM of the output image approximately degrade by 6% and 9.5% respectively, when processed by the SAC unit. From the analysis, we observe that the compromise in the image quality in the proposed work is insignificant compared to the computationally expensive MAC processing. In real-time image processing applications, minimizing the computational cost is considered the paramount aspect of computational accuracy [60]. Fig. 8 compares visually the input image, the accurate results, and the approximate results of processing kernels.

5.3. Hardware analysis of SAC unit

In this section, we present simulation results of the optimized pipeline architecture of SAC. Fig. 9 shows the timing diagram of computing convolution operation by both the pipeline and non-pipeline SAC architectures. A set of nine pixels is convolved with a 3×3 weight kernel. The waveform depicts the total time required to complete the computation of convolution operations. The non-pipeline architecture completes the computation in 325 ns, whereas pipeline architecture consumes 185 ns. The pipeline architecture computes approximately 1.8× faster than the non-pipeline SAC architecture.

Furthermore, we compute the area and power benefits achieved by the proposed SAC architecture in reference to the accurate MAC and two similar state-of-the-art approximate computing units [24,25]. In this analysis, all the architec-

Table 5 Accuracy analysis of different approximate computing units.

Architecture	Max ED (%)	MED	NED	MRED	MSE
SAC_4	1.08	421	0.058	1.628	2.66×10^4
SAC_5	0.69	279	0.037	0.819	1.37×10^4
SAC_6	0.61	228	0.028	0.679	1.02×10^4
DRUM	0.82	681	0.082	3.064	1.46×10^4
TOSAM	0.97	532	0.051	1.002	2.16×10^4

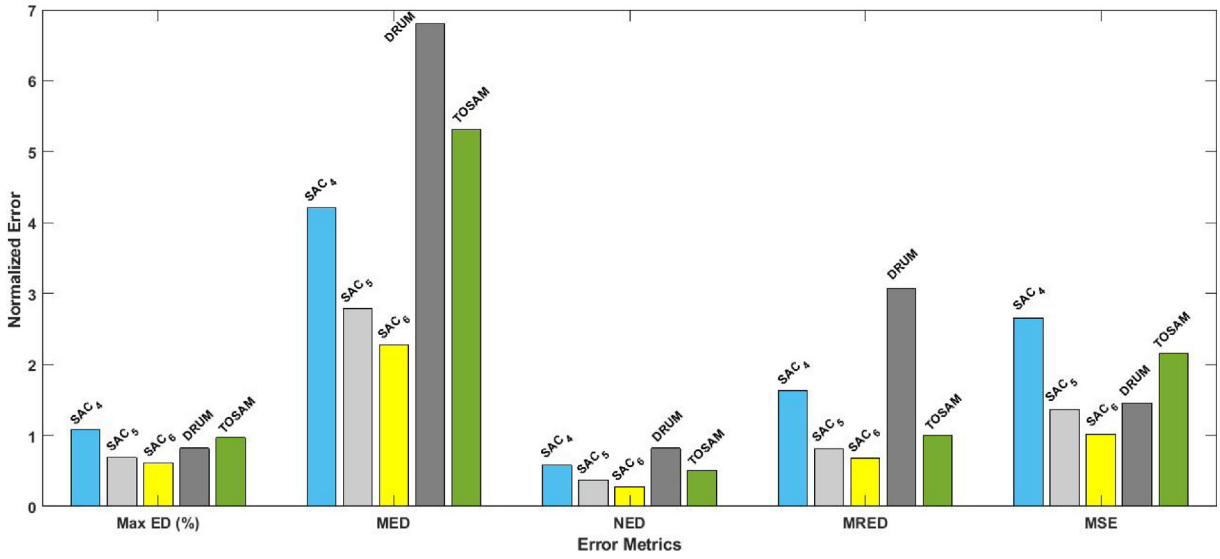


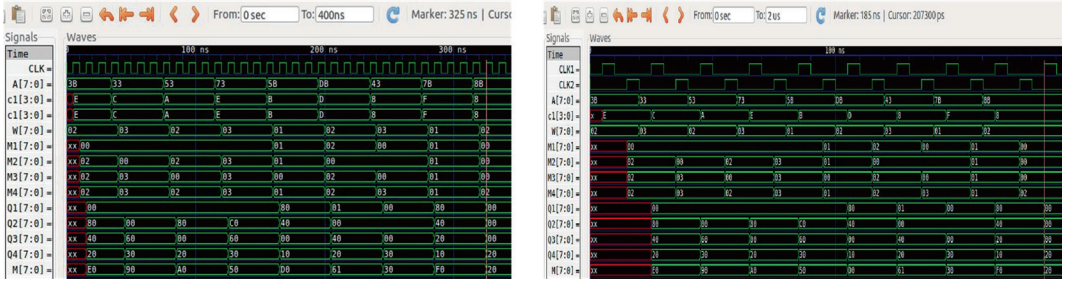
Fig. 7 Histogram distribution of error metrics.

Table 6 Error characteristics of the SAC architecture in reference to an accurate design.

Kernel	MAC		SAC	
	PSNR	SSIM	PSNR	SSIM
Gaussian-blur	14.4287	0.7364	14.3786	0.6766
Smoothing	10.7563	0.6971	9.9383	0.6115

(a) Input Image. (b) Gaussian-blur by MAC. (c) Gaussian-blur by SAC. (d) Smoothing by MAC. (e) Smoothing by SAC.

Fig. 8 Visual illustration of image quality processed by different computing units.



(a) Non-Pipeline Architecture. (b) Pipeline Architecture.

Fig. 9 Timing diagram of the convolution operations computed by SAC architectures.

tures are synthesized from RTL level design with Synopsys Design Compiler using 90 nm technology. Table 7 compares the total cell area and energy consumption of the optimized SAC unit with accurate MAC and the multipliers proposed in [24,25]. We observe that the proposed work results in low resource utilization and power consumption due to the less architectural complexity compared to the other computing architectures. Both the state-of-the-art architectures perform fixed width multiplication to generate the multiplication results. The proposed SAC architecture completely eliminates the multiplication operation by shift and addition operations. The total cell area refers to the table includes the number of ports, combinational and sequential cells, interconnects, buffers, etc. Similarly, dynamic power is the summation of switching and leakage power. Table 8 shows the percentage of area and power saved by the proposed SAC architecture. In both cases (area and power), the SAC architecture performs better than the other accurate and approximate computing architectures. Fig. 10 graphically demonstrates that the proposed SAC architecture outperforms the other state-of-the-art computing architectures.

Table 8 Comparison of % savings by SAC.

Architecture	Area Savings (%)	Power Savings (%)
TOSAM	4.2	14.1
DRUM	17.6	45.6
Accurate MAC	38.1	70.4

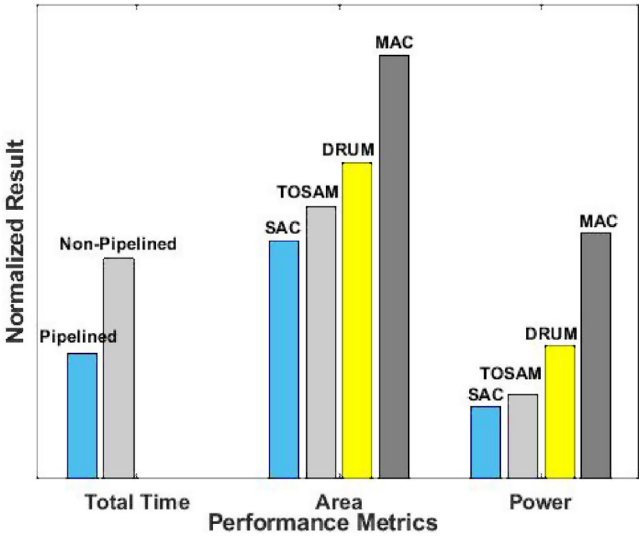


Fig. 10 Graphical representation of computing performance.

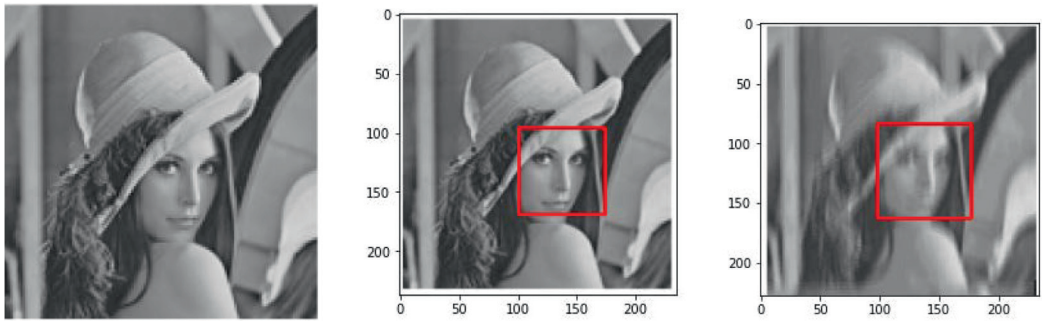
5.4. Evaluation of object detection algorithm

In this section, we validate the working of the proposed SAC architecture in two CNN-based object detection algorithms. To simulate the algorithms, we have used Keras deep learning library running on top of the Tensorflow back end in Anaconda code editors with OpenCV, Numpy, and Scipy support. In both the algorithms, the traditional accurate computing unit is replaced by the proposed SAC unit. Firstly, we have used the cascade classifier from OpenCV library to build MT-CNN network for face detection. Secondly, tiny YOLO architecture is used for object detection with both MAC and SAC unit. In both the cases we have used pre-trained CNN models.

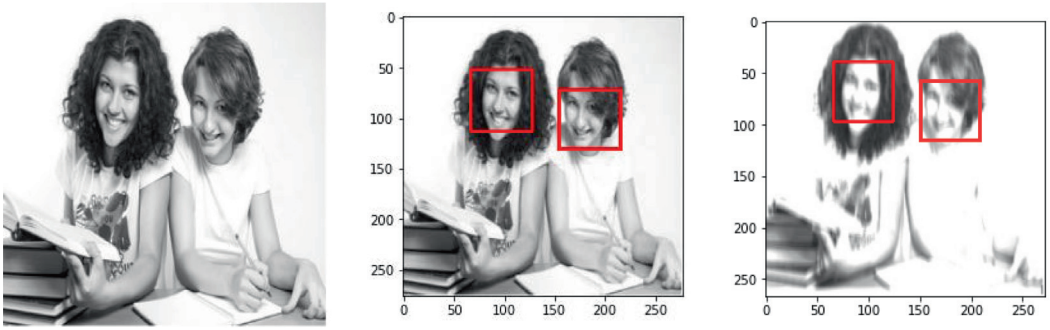
As discussed earlier, it has been noticed that the approximate computing mechanism degrades the quality of the output image in both the detection algorithms when processed by the SAC unit. Fig. 11 and Fig. 12 represents a set of output images of the object detection algorithms computed by MAC and SAC separately for visual analysis. As shown in Fig. 11, face detection is successfully performed for both the images by the proposed SAC architecture. Moreover, detection accuracies shown in the images are found to be similar for both the computing units. Also, We observed in Fig. 12 that the MT-CNN detection algorithms accurately detect the faces and objects present in the input images. Further, we have selected five classes of objects that is person, dog, horse, car, and bicycle to evaluate the object detection metric mAP for the YOLO object detection Scheme. Images from these classes are used for testing the YOLO object detection algorithm. Table 9 shows the mAP values for each class of images. From this table, we observe that the mAP values appear similar for all the classes when computed by the MAC and the proposed SAC unit. Hence, we can conclude that in real-time object detection applications our proposed computing architecture shows satisfactory performance with the better area and power efficiency.

Table 7 Area and Power characteristics for computing architectures.

Architecture	Total Cell Area	Power		
		Static (mW)	Dynamic (mW)	Total (mW)
SAC	770778	3.42	7.31	10.73
TOSAM [25]	804681	3.75	8.74	12.49
DRUM [24]	935780	8.89	10.82	19.71
Accurate MAC	1244587	16.41	19.86	36.27

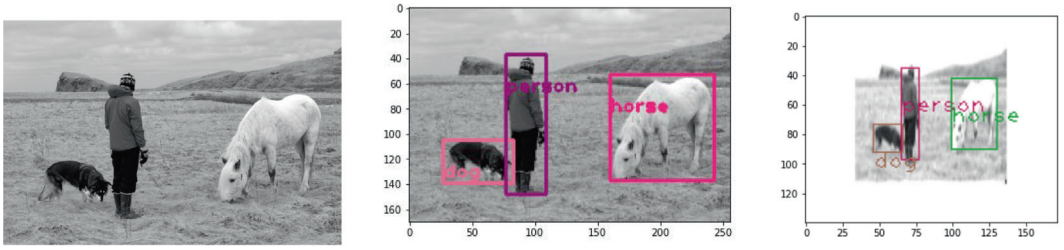


(a) Input Image. (b) computed by MAC (accuracy: 99.94%). (c) computed by SAC (accuracy: 99.37%).

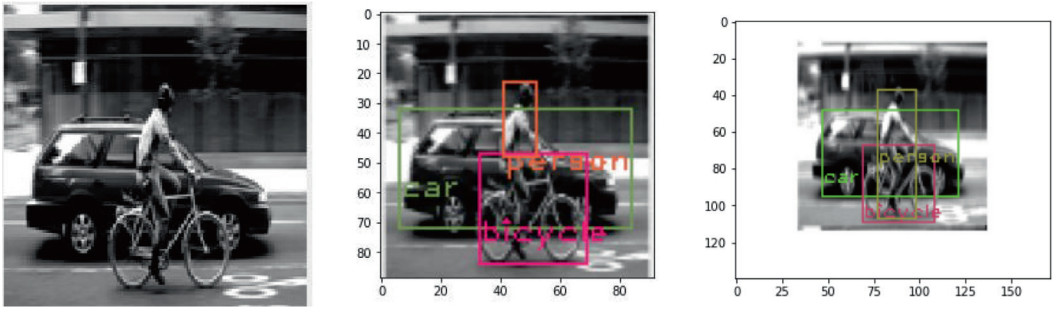


(d) Input Image. (e) computed by MAC (accuracy: 99.87%). (f) computed by SAC (accuracy: 99.12%).

Fig. 11 Face detection by MT-CNN model.



(a) Input Image. (b) computed by MAC. (c) computed by SAC.



(d) Input Image. (e) computed by MAC. (f) computed by SAC.

Fig. 12 Object detection by YOLO architecture.

Table 9 Mean Average Precision (mAP) for YOLO object detection algorithm.

Class	Person	Dog	Horse	Car	Bicycle
mAP for SAC	0.762	0.548	0.652	0.713	0.652
mAP for MAC	0.786	0.507	0.681	0.749	0.637

6. Conclusion

Computer vision plays a major role in smart city applications. Integration of CNN algorithms with IoT devices have opened up a new era for computer vision tasks including object detection, segmentation and recognition. The convolution layers of a CNN architecture demand a large number of MAC operations. Real-time CNN applications on embedded and/or edge-IoT devices increase the challenges of MAC computations with high execution time, power consumption, and area overhead. In this paper, we have addressed the challenges of computational complexity associated with the convolution computing units in terms of time, area, and power. We proposed an architecture of approximate computing unit to execute convolution operations. The computing unit, SAC, is based on the processing of pixels with reduced bit-precision that minimizes area overhead and power consumption on hardware implementation. Moreover, the paper implements two CNN-based object detection algorithms with the proposed SAC unit. Both the algorithms successfully detect faces and objects present in the input images. The hardware implementation of the proposed computing architecture improves approximately 1.2% in total cell area and 1.5% in total power consumption than the other related architectural schemes. This work may limit the performance of applications such as medical imaging that requires output images of high resolution. Implementation of this approximate computing unit in various CNN applications is left as future work.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors would like to acknowledge Prince Sultan University and Smart Systems Engineering lab for their valuable support and provision of research facilities that were essential for completing this work. Also, the authors would like to acknowledge the support of Prince Sultan University for Article Processing Charges (APC) of this publication. This work is also supported by Visvesvaraya PhD scheme for Electronics and IT, Ministry of Electronics and Information Technology, Government of India, MEITY-PHD-3043.

References

[1] A. Kumar, Y. Zhou, C. Gandhi, R. Kumar, J. Xiang, Bearing defect size assessment using wavelet transform based deep convolutional neural network (dcnn), Alexandria Eng. J. 59 (2) (2020) 999–1012.

[2] S.B. Baker, W. Xiang, I. Atkinson, Internet of things for smart healthcare: Technologies, challenges, and opportunities, IEEE Access 5 (2017) 26521–26544.

[3] W. Hu, H. Li, A blockchain-based secure transaction model for distributed energy in industrial internet of things, Alexandria Eng. J. 60 (1) (2021) 491–500.

[4] M.M. Nasralla, N. Khan, M.G. Martini, Content-aware downlink scheduling for lte wireless systems: A survey and performance comparison of key approaches, Comput. Commun. 130 (2018) 78–100.

[5] K. Wu, R. Cheng, W. Cui, W. Li, A lightweight sm2-based security authentication scheme for smart grids, Alexandria Eng. J. 60 (1) (2021) 435–446.

[6] M.M. Nasralla, M. Razaak, I.U. Rehman, M.G. Martini, Content-aware packet scheduling strategy for medical ultrasound videos over LTE wireless networks, Comput. Netw. 140 (2018) 126–137.

[7] C. Zhu, L. Shu, V.C. Leung, S. Guo, Y. Zhang, L.T. Yang, Secure Multimedia Big Data in Trust-Assisted Sensor-Cloud for Smart City, IEEE Commun. Mag. 55 (12) (2017) 24–30.

[8] M.M. Nasralla, M. Razaak, I. Rehman, M.G. Martini, A comparative performance evaluation of the HEVC standard with its predecessor H. 264/AVC for medical videos over 4G and beyond wireless networks, in: International Conference on Computer Science and Information Technology (CSIT), IEEE, 2018, pp. 50–54.

[9] M.M. Nasralla, C.T. Hewage, M.G. Martini, Subjective and objective evaluation and packet loss modeling for 3d video transmission over LTE networks, in: International Conference on Telecommunications and Multimedia (TEMU), IEEE, 2014, pp. 254–259.

[10] S. Gould, T. Gao, D. Koller, Region-based segmentation and object detection, Adv. Neural Inform. Process. Syst. (2009) 655–663.

[11] C. Wojek, G. Dork, A. Schulz, B. Schiele, Sliding-windows for rapid object class localization: A parallel technique, Joint Pattern Recognition Symposium, Springer, 2008, pp. 71–81.

[12] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, T. Darrell, Caffe: Convolutional architecture for fast feature embedding, in: Proceedings of the 22nd international conference on Multimedia, ACM, 2014, pp. 675–678.

[13] A. Krizhevsky, I. Sutskever, H. Geoffrey E., Imagenet classification with deep convolutional neural networks, Adv. Neural Inform. Process. Syst. (2012) 1097–1105.

[14] Z. Cao, T. Simon, S.-E. Wei, Y. Sheikh, Realtime multi-person 2D pose estimation using part affinity fields, in: Proceedings of the conference on computer vision and pattern recognition, IEEE, 2017, pp. 7291–7299.

[15] Z. Yang, R. Nevatia, A multi-scale cascade fully convolutional network face detector, in: 23rd International Conference on Pattern Recognition (ICPR), IEEE, 2016, pp. 633–638.

[16] C. Chen, A. Seff, A. Kornhauser, J. Xiao, Deepdriving: Learning affordance for direct perception in autonomous driving, in: Proceedings of the International Conference on Computer Vision, IEEE, 2015, pp. 2722–2730.

[17] C. Szegedy, W. Liu, J. Yangqing, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going

deeper with convolutions, in: Proceedings of the conference on Computer Vision and Pattern Recognition, IEEE, 2015, pp. 1–9.
[18] F. Iandola, S. Han, M. Moskewicz, K. Ashraf, W. Dally, K. Keutzer, Squeezenet:AlexNet-level accuracy with 50x fewer parameters and < 0.5 MB model size, arXiv preprint arXiv:1602.07360.
[19] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the conference on Computer Vision and Pattern Recognition, IEEE, 2016, pp. 770–778.
[20] I. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556v6.
[21] I.U. Rehman, M.M. Nasralla, N.Y. Philip, Multilayer perceptron neural network-based QoS-aware, content-aware and device-aware QoE prediction model: A proposed prediction model for medical ultrasound streaming over small cell networks, Electronics 8 (2) (2019) 194.
[22] S.M. Sharif, D. Prasad, Design of optimized 64 bit MAC unit for DSP applications, J. Adv. Trends Comput. Sci. Eng. 3 (5) (2014) 456–460.
[23] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.
[24] S. Hashemi, R.I. Bahar, S. Reda, DRUM: A dynamic range unbiased multiplier for approximate applications, in: International Conference on Computer-Aided Design (ICCAD), IEEE, 2015, pp. 418–425.
[25] S. Vahdat, M. Kamal, A. Afzali-Kusha, M. Pedram, TOSAM: An energy-efficient truncation-and rounding-based scalable approximate multiplier, IEEE Trans. Very Large Scale Integr. VLSI Syst. 27 (5) (2019) 1161–1173.
[26] A. Maki, D. Miyashita, K. Nakata, F. Tachibana, T. Suzuki, J. Deguchi, FPGA-based CNN processor with filter-wise-optimized bit precision, in: Asian Solid-State Circuits Conference (A-SSCC), 2018, pp. 47–50.
[27] M.S. Kim, A.A. Del Barrio, H. Kim, N. Bagherzadeh, Effects of approximate multiplication on convolutional neural networks, arXiv preprint arXiv:2007.10500.
[28] M.F. Cowlishaw, Decimal floating-point: Algorism for computers, in: Proceeding of Symposium on Computer Arithmetic, IEEE, 2003, pp. 104–111.
[29] A.D. Robison, N-bit unsigned division via n-bit multiply-add, in: Symposium on Computer Arithmetic (ARITH'05), IEEE, 2005, pp. 131–139.
[30] F.P. O'Connell, S.W. White, POWER3: The next generation of powerPC processors, IBM J. Res. Dev. 44 (3) (2000) 844–873.
[31] A. Kumar, The HP PA-8000 RISC CPU, IEEE Micro Mag. 17 (2) (1997) 27–32.
[32] B. Greer, J. Harrison, G. Henry, W. Li, P. Tang, Scientific computing on the itanium processo, in: Proceedings of the ACM/IEEE SC2001 Conference, 2001, pp. 1–8.
[33] R. Samy, H.A. Fahmy, R. Raafat, A. Mohamed, T. ElDeeb, Y. Farouk, A decimal floating-point fused-multiply-add unit, in: International Midwest Symposium on Circuits and Systems (MWSCS), IEEE, 2010, pp. 529–532.
[34] S. Nasar, K. Subbarao, Design and implementation of MAC unit using reversible logic, J. Eng. Res. Appl. 2 (5) (2012) 1848–1855.
[35] A. Prakash, K. Sharma, Design and analysis of multiplier accumulation unit by using hybrid adder, J. Comput. Trends Technol. 37 (2) (2016) 96–102.
[36] S. Raghav, R. Mittal, Implementation of fast and efficient mac unit on FPGA, J. Math. Sci. Comput. 2 (4) (2016) 24–33.
[37] K.S.N. Yengade, P. Indurkar, Review on design of low power multiply and accumulate unit using baugh-wooley based multiplier, Int. Res. J. Eng. Technol. (IRJET) 4 (2) (2017) 638–642.
[38] J. Garland, D. Gregg, Low complexity multiply-accumulate units for convolutional neural networks with weight-sharing, Trans. Architect. Code Optimiz. (TACO) 15 (3) (2018) 31–54.
[39] P. Kulkarni, P. Gupta, M. Ercegovac, Trading accuracy for power with an underdesigned multiplier architecture, in: 24th Internatioal Conference on VLSI Design, IEEE, 2011, pp. 346–351.
[40] M. Imani, D. Peroni, T. Rosing, CFPU: Configurable floating point multiplier for energy-efficient computing, in: 54th Design Automation Conference (DAC), IEEE, 2017, pp. 1–6.
[41] S. Boroumand, H.P. Afshar, P. Brisk, S. Mohammadi, Exploration of approximate multipliers design space using carry propagation free compressors, in: 23rd Asia and South Pacific Design Automation Conference (ASP-DAC), IEEE, 2018, pp. 611–616.
[42] J. Lee, C. Kim, S. Kang, D. Shin, S. Kim, H.J. Yoo, UNPU: A 50.6tops/w unified deep neural network accelerator with 1b-to-16b fully-variable weight bit-precision, in: International Solid-State Circuits Conference (ISSCC), 2018, pp. 218–220.
[43] G. Gillani, M.A. Hanif, B. Verstoep, S.H. Gerez, M. Shafique, A.B. Kokkeler, MACISH: Designing approximate mac accelerators with internal-self-healing, IEEE Access 7 (2019) 77142–77160.
[44] M.S. Kim, A.A. Del Barrio, L.T. Oliveira, R. Hermida, N. Bagherzadeh, Efficient mitchell's approximate log multipliers for convolutional neural networks, IEEE Trans. Comput. 68 (5) (2018) 660–675.
[45] R. Pilipović, P. Bulić, On the design of logarithmic multiplier using radix-4 booth encoding, IEEE Access 8 (2020) 64578–64590.
[46] M.S. Ansari, V. Mrazek, B.F. Cockburn, L. Sekanina, Z. Vasicek, J. Han, Improving the accuracy and hardware efficiency of neural networks using approximate multipliers, IEEE Trans. Very Large Scale Integr. VLSI Syst. 28 (2) (2019) 317–328.
[47] H. Kim, M.S. Kim, A.A. Del Barrio, N. Bagherzadeh, A cost-efficient iterative truncated logarithmic multiplication for convolutional neural networks, in: 2019 IEEE 26th Symposium on Computer Arithmetic (ARITH), IEEE, 2019, pp. 108–111.
[48] V. Camus, L. Mei, C. Enz, M. Verhelst, Review and benchmarking of precision-scalable multiply-accumulate unit architectures for embedded neural-network processing, J. Emerg. Sel. Top. Circ. Syst. 9 (4) (2019) 697–711.
[49] A. Hazarika, A. Jain, S. Poddar, H. Rahaman, Shift and accumulate convolution processing unit, (TENCON), IEEE (2019) 914–919.
[50] Z.-Q. Zhao, P. Zheng, S.-T. Xu, X. Wu, Object detection with deep learning: A review, IEEE Trans. Neural Netw. Learn. Syst. 30 (11) (2019) 3212–3232.
[51] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A.C. Berg, Ssd: Single shot multibox detector, in: European conference on computer vision, Springer, 2016, pp. 21–37.
[52] S. Ren, K. He, R. Girshick, J. Sun, Faster r-cnn: Towards real-time object detection with region proposal networks, Adv. Neural Inform. Process. Syst. (2015) 91–99.
[53] K. Zhang, Z. Zhang, Z. Li, Y. Qiao, Joint face detection and alignment using multitask cascaded convolutional networks, IEEE Signal Process. Lett. 23 (10) (2016) 1499–1503.
[54] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, in: Proceedings of the conference on computer vision and pattern recognition, IEEE, 2016, pp. 779–788.
[55] T. MathWorks, MATLAB R2018b, MathWorks, 2020 (Online accessed: 25-Sept-2020).
[56] SYNOPSYS, Rtl design and synthesis, 2020 (Online accessed: 28-Dec-2020). https://www.synopsys.com/implementation-and-signoff/rtl-synthesis-test.html.

[57] F. Chollet, Introduction to Keras for Researchers, 2020 (Online accessed: 28-Dec-2020) https://keras.io/getting_started/intro_to_keras_for_researchers/.
[58] ANACONDA, 2020 (Online accessed: 25-Sept-2020) URL https://www.anaconda.com/.
[59] J. Liang, J. Han, F. Lombardi, New metrics for the reliability of approximate and probabilistic adders, IEEE Trans. Comput. 62 (9) (2012) 1760–1771.
[60] Y. Shen, M. Ferdman, P. Milder, Maximizing CNN accelerator efficiency through resource partitionin, in: International Symposium on Computer Architecture (ISCA), 2017, pp. 535–547.
[61] L. Kabbai, A. Sghaiery, A. Douik, M. Machhouty, FPGA implementation of filtered image using 2D gaussian filter, Int. J. Adv. Comput. Sci. Appl. 7 (7) (2016) 514–520.