ISSN 1330-3651 (Print), ISSN 1848-6339 (Online) https://doi.org/10.17559/TV-20240608001761
Original scientific paper

Ant Colony Optimized Convolution Neural Network-Long Short Term Memory Based Energy Conservation in Variation-Tolerant Near-Threshold Processor

C. THIRUVENGADAM, Mashael MAASHI, Jamal ALSAMRI, Raghu GUNDAALA*

Abstract: When considering the reduction of profit margins and improvement of energy efficiency in processors operating at near-threshold and sub-threshold levels, exploring timing error resilience might be seen as a potentially profitable option. Near-threshold (NT) circuit design is increasingly used for building energy-efficient digital circuits. One drawback was a drop in performance due to a reduction in driving current. To maximize energy economy and minimize performance deterioration, some studies advocate non-traditional (NT) chip multiprocessors. The creation of energy-efficient artificial intelligence technologies has gained popularity recently. Graphics processing units and general-purpose CPUs could not match the goal of efficiency and performance. A semiconductor with many processing units was created to do this. This goal was achieved by engineering the hardware with unique features. Therefore, this study constructs an ACO_CNN+LSTM (Ant Colony Optimised Convolution Neural Network-Long Short Term Memory) neural network to achieve lower power consumption without compromising performance. In this method, Ant Colony Optimization (ACO) is used to explore and select optimal configurations of CNN and LSTM architectures, minimizing energy usage through efficient layer selection, filter size adjustment, and neuron pruning. By incorporating Dynamic Voltage and Frequency Scaling (DVFS) and operating at near-threshold voltages (NTV) - where the supply voltage is just above the transistor threshold - power consumption is further reduced, as the processor can function at lower voltage levels while tolerating variations in performance. This hybrid ACO-driven optimization enhances both CNN feature extraction and LSTM's temporal processing by selecting the most energy-efficient network parameters. As a result, the proposed ACO_CNN+LSTM achieves 12.73% of energy consumption, 9.46 sec of latency, 98.45% of normalized frequency, 22.64% of error rate and 31.67% of processor utilization.

Keywords: chip processor; convolution neural network; energy conservation; long short term memory; near-threshold; optimization

1 INTRODUCTION

The energy and latency costs of current machine learning (ML) algorithms prevent real-time inference on sensor-rich platforms like the IoT, wearables, personal biomedical devices, and drones [1]. These applications comprise devices that collect and analyze data to deliver interpretations and actions for automating or monitoring processes without human intervention. These approaches must be capable of incorporating computationally demanding machine learning techniques while meeting strict constraints on size, power consumption, and response time [3]. Memory access has been shown to decrease both the time delay and the energy expenditure involved with understanding machine learning algorithms. Several energy-efficient machine learning systems using digital architectures and IC have been developed in recent years. These applications use techniques such as efficient data flow, data reuse, and computation optimization to minimize memory accesses [4]. Processors across all market categories are experiencing a growing limitation in terms of power and energy resources. Although present designs may be improved gradually, many applications need much greater economy while maintaining excellent performance. The system and circuit levels must change drastically. The interface is the main energy consumer in system-on-chip (SoC) applications such video-enabled sensor networks, implanted medical devices, and mobile devices. This comprises screens, wireless transceivers, and analog-digital converters. These systems allow for the reduction of input/output energy while still achieving system objectives via the implementation of complex and resource-intensive on-chip processing. Recent brain implant research shows that local on-chip parallel DSP approaches like feature extraction and clustering may increase system power by 4.3 and 26 [7]. Parallel video and image processing may extract features from modern sensors and reduce input/output procedures [8]. Parallel half-toning methods improve low-power bi-stable display efficiency [9]. Hence, this objective is to develop a novel category of processors that prioritize ultra-low power consumption while still fulfilling the substantial computational requirements necessary to transfer the energy consumption load from the input/output to the on-chip computation [10]. Currently, there exists a disparity in the tradeoffs between performance and efficiency. In order to facilitate the energy optimizations discussed earlier, there is a need for a novel category of processors that amalgamates the energy efficiency of near-threshold processors with the high-performance throughput of massively parallel architectures. In this study, we introduce our preliminary efforts to tackle the significant obstacles associated with a programmable, highly parallel, and very efficient near-threshold processor. Considering this, the accomplishments of this effort may be summarised as follows:

- A novel application of Ant Colony Optimization (ACO) to optimize the configuration of hybrid Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks.
- ACO to systematically select optimal CNN layer structures and LSTM parameters, the approach achieves significant reductions in power consumption while maintaining performance. This optimization includes layer selection, filter size adjustments, and neuron pruning, which collectively enhance the energy efficiency of the hybrid network architecture.

In the rest of this paper section 2 provides the existing methods. Section 3 proposed a method, and it is divided into several sections. Then we discuss experimental results for algorithm evaluation in Section 4. Finally, Section 5 is conclusion of the work.

2 RELATED WORKS

Studies have shown that operating in the sub-threshold and near-threshold regimes results in energy efficiency that is ten times greater than conventional processors at the

same technology node. The innovative Error Prediction (EP) approach enhances adaptive prediction capabilities and voltage scaling [11]. The energy consumption is decreased by 18.6-25.1% when compared to the signoff margins. Additionally, it enables the system to function without any energy overhead in comparison to its ideal non-margined critical operating point, resulting in a throughput loss of less than 5%. This design incorporates an 8-kb SRAM macro constructed using 55-nm CMOS technology, as shown in [12]. The mean power consumption for the full chip executing 7 T read and write operations shows a reduction of 19.3% in comparison to the typical 6T SRAM at a voltage of 0.75 V. The approach of metastability inference and avoidance (MIAA) is introduced in order to address the issue of metastability during clock-domain crossover in [13]. By implementing metastability mitigation, we can replace a multi-stage synchronizer with a single flip-flop for synchronisation in the NoC. This results in a 40.2% improvement in latency and a 79% improvement in throughput for the NoC. The SE10T near-threshold SRAM [14] is energy-efficient. A built-in read-assist mechanism and power-gating technique improve read stability and writability, while single-ended read/write operation and transistor stacking in the cell core reduce SE10T power consumption. The breakthrough low-energy single-bitline 11-transistor (SB11T) near-threshold SRAM cell is compared to the standard 6 T at 0.45 V in [15]. The SB11T cell exceeds the usual 6 T cell in hold/read stability and write static noise margin by at least 1.04 × and 1.40 × respectively. A 7 T bit cell with 32 nm features and 300 mV supply voltage is created in [16]. Cell leakage power is 8.4 pW for Q = '0' and 1.2 pW for Q = '1'. The tolerance analysis shows the cell functions and generates dependable outputs during process-voltage-temperature changes for static performance measurements. Lightweight timed error-tolerant flip-flop (ETFF) [17]. Simulations demonstrate that a CNN accelerator employing the suggested timing error-tolerant architecture in the SMIC CMOS 40 nm technology may operate at 1.1–0.3 V with 3.5% area overhead. This design minimizes area overhead by 54.68% and increases energy efficiency by 53.69% at 0.6 V compared to the Razor flip-flop design. In [18], Google's DNN accelerator outperformed its competitors. Low-power Near-Threshold Computing (NTC) TPU PREDITOR. This method improves performance by 3-5× compared to leading error mitigation systems, with little accuracy loss. Consider a unique very stable low-energy 10T (SLE10T) SRAM cell for near-threshold operation [19]. The SLE10T reduces leakage power dissipation by at least 1.10×. Additionally, it enhances read/write energy by at least 1.01×/1.03×. However, the SLE10T bitcell has a 0.02 μm2 area, 1.657×/1.318× greater than the typical 6 T/8 T bitcell. The works reviewed show advancements in low-energy SRAM and processing designs but face notable limitations. Techniques like adaptive prediction and voltage scaling, while reducing energy consumption by up to 25.1%, introduce a slight throughput loss of up to 5%. Specialized SRAM cells (7 T, 10 T, SB11T) improve energy efficiency and stability but increase cell area, with SB11T being up to 1.657× larger than conventional cells. Metastability mitigation methods like MIAA enhance NoC performance but add design complexity. The ETFF design reduces area and energy overhead for CNN accelerators but incurs a 3.5% area overhead. Finally, while PREDITOR offers improved performance in near-threshold computing, it may experience minor accuracy losses. These limitations highlight the need for balanced designs that address trade-offs between energy efficiency, area, and performance. To overcome this limitations Ant Colony Optimized Convolution Neural Network-Long Short Term Memory (ACO_CNN+LSTM) based Energy conservation is adopted in this work.

3 THE PROPESED MODEL

In this study, Ant Colony Optimized Convolutional Neural Network (ACO_CNN) for energy conservation in Variation-Tolerant Near-Threshold Processors (VTNTP) aims to optimize computational efficiency while reducing energy consumption. This approach combines the strengths of Ant Colony Optimization (ACO), a nature-inspired metaheuristic algorithm, with the capabilities of CNNs to perform efficient processing under near-threshold voltage (NTV) conditions, which are known to provide substantial energy savings but introduce variability and potential errors.

3.1 Near-Threshold Voltage (NTV) Operation in Processor

Energy conservation in a Variation-Tolerant Near-Threshold Processor (VTNTP) is achieved by operating the processor at voltages close to its threshold voltage. This approach significantly reduces energy consumption but introduces variability in performance, which the system must handle effectively. In a perfect clock network, cllk_root matches the sequential endpoint clock $\left(clk_{dff_{des}}\right)$. An elevated FP_error signal shows important activity in Prediction Window (PW) at the boundary of $\left(clk_{dff_{des}}\right)$, indicating a timing error. The delay $\left(T_{latency,des}\right)$ between PW and $clk_{dff_{des}}$ is caused by the clock network latency in the chip, which rises with voltage reduction, the High FP_error signal and timing error inequivalence. The error detection circuit detects timing problems when the supply falls after energy-up, increasing PW_status until one processor output is covered. Thus, neighbouring PW widths should be bigger than a processor's output pulse width T_H at distinct corners,

$$\forall i \in [0,6]: PW_{PW_{status}=i+1} - PW_{PW_{status}=i} > T_H \tag{1}$$

A minimum pulse width constraint applies on the $PW_{PW_{status}=1}$. This constraint equals the sum of the worst case propagation delay and setup time for Integrated Clock Gating (ICG) cell:

$$PW_{PW_{status}=1(ff)} > T_{prob,OR(ss)} + T_{setup} \tag{2}$$

The energy consumption of a processor, especially in CMOS circuits, can be modeled using two primary components: dynamic energy and static (leakage) energy. Dynamic energy is primarily consumed when transistors

switch between states (0 and 1). The dynamic energy consumption is proportional to the square of the supply voltage, and it can be expressed mathematically as:

$$P_{dynamic} = \alpha C V^2 f \tag{3}$$

where, $P_{dynamic}$ is the dynamic energy consumption, α is the activity factor (the fraction of the circuit switching per clock cycle), C is the effective capacitance being charged and discharged in the circuit, V is the supply voltage, f is the clock frequency. Reducing the supply voltage V drastically reduces dynamic energy because $P_{dynamic}$ is proportional to V^2. This is why operating at near-threshold voltage (NTV) offers significant energy savings. However, reducing the voltage also affects the maximum operating frequency f because it takes more time for the transistors to switch between states at lower voltages. This introduces a trade-off between energy savings and performance (speed). Leakage energy is the energy consumed when the transistors are not actively switching but still dissipate energy due to leakage currents. The static energy consumption can be modeled as:

$$P_{static} = V I_{leakage} \tag{4}$$

where, P_{static} is the static energy consumption, V is the supply voltage, $I_{leakage}$ is the leakage current. Leakage current increases exponentially as the threshold voltage decreases. The total energy consumption P_{total} is the sum of dynamic and static energy

$$P_{total} = P_{dynamic} + P_{static} = \propto C V^2 f + V I_{leakage} \tag{5}$$

At near-threshold voltage, **dynamic energy** is significantly reduced, but **leakage energy** may become more prominent. Hence, there is a trade-off between energy savings and the increase in static energy, which needs to be optimized depending on the application and operating conditions.

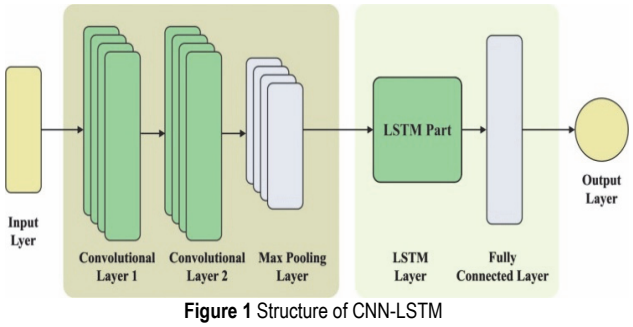

Figure 1 Structure of CNN-LSTM

3.2 AEO Based Hyper Parameter Tuning Process

Ant Colony Optimization (ACO) can be effectively applied to energy conservation in microprocessors by optimizing key design and operational parameters that influence energy consumption, both at the architectural and transistor levels.

a) Initial Solution Construction

The initial solutions have a considerable impact on the end outcomes. In order to enhance the original solution with higher transition density, 10,000 random pairings of patterns $(V1, V2)$ are implemented on the circuits. Following the simulation, the transition density for each pair of patterns may be computed. This figure represents the extent to which this pair of patterns contributes to the overall maximum energy usage. Let's assume that the AND gates A and C have a delay of 2 units, whereas the OR gates B and D have a delay of 3 units.

The delay values are also shown on the gates. During the simulation of the input vector pair (1110, 0100), several transitions take place in the gates. For example, at time 2, gate A changes its state from 1 to 0. Gate B changes state at time 5, but gate D remains unchanged. The transition density (TD) of this pair of vectors is obtained as shown in Fig. 1. The numerator of TD is calculated by summing the product of the transition number and the number of gate outputs for each node. The denominator of TD represents the quantity of capacitive nodes. As the transition density increases, the switching behaviour also increases, resulting in increasing energy consumption.

b) Pheromone Update

The pheromone is adjusted based on the discovered solution's quality. For every main input i, each vector pair would result in four transition conditions: 0→0, 0→1, 1→0, and 1→1. Four parameters, $\tau^i_{0\to0}, \tau^i_{0\to1}, \tau^i_{1\to0}$ and $\tau^i_{1\to1}$, are constructed to reflect the pheromones for each main input i under four different situations. These parameters are used for combining the transition density of every input vector pair. The pheromone values drop in the process of pheromone evaporation at a rate of k. It is denoted by Eq. (6),

$$\tau^i_{0\to0}(t+1) = \Delta\tau^i_{0\to0}(t) + (1-k).\tau^i_{0\to0}(t) \tag{6}$$

where the variable k, which belongs to the range [0, 1], represents the rate at which pheromones degrade. The term $\Delta\tau^i_{0\to0}(t)$ refers to the quantity of pheromones added to the accumulated pheromone $\tau^i_{0\to0}$ at time t. The Eq. (6) expresses the remaining three parameters, with the exception of the transition condition. The basic input consists of four parameters, namely $\tau^a_{0\to0}, \tau^a_{0\to1}, \tau^a_{1\to0}$ and $\tau^a_{1\to1}$. These parameters reflect the pheromones associated with the four transition circumstances of a vector pair. The values of these parameters are set to zero at the beginning. The transition density of the input vector pair $(V1, V2) = (1110, 0100)$ is computed for all inputs. Subsequently, the transition density produced from the transition condition of each main input is used as an indication for updating the pheromone value. The transition condition between $(V1, V2)$ changes from $1 \to 1$ and the parameter $\tau^b_{1\to1}$ rises by 3/8 for the second main input bit b. The corresponding pheromone values for each main input are updated continuously for each input vector pair.

c) Local search

Ant Colony Optimisation and local search seem to work effectively. Thus, maximum energy consumption

options are searched locally. The top 100 input vector pairs with greater transition density are chosen for local search after 10,000 beginning vector pairs. For each main input, the highest and secondary values of those four parameters may be used to create two input vector pairs for crossover with the top 100 input vectors. The highest and secondary values are chosen to increase vector pair variation, and their difference is sometimes small. Two vector pairs of each main input's highest and secondary parameter values are on the left. The top one is the highest main input parameter value, while the lower one is the secondary parameter value. The four parameters of main input *b* are $\tau^b_{0\to0}, \tau^b_{0\to1}, \tau^b_{1\to0}$ and $\tau^b_{1\to1}$. Select the greatest value $\tau^b_{0\to1}$ to compose the first vector pair. This technique applies to other principal inputs. The secondary parameter values from each main input form the second vector pair. Two vector pairs (1001 → 0110) and (0011 → 1011) are created simultaneously based on the transition condition. Each main input will randomly choose the highest or secondary bit from these two pairings to combine one vector pair. The vector pair (0011 → 1110) includes the secondary vector pair bits *a* and *c*, and the highest vector pair bits *b* and *d*.

d) Optimized CNN structure

The application utilises convolutional 1D layers, pooling 1D layers, and a fully connected layer. In CNN processes time series data by representing it in a one-dimensional format, where the data is organised in a sequential order based on time instants. For the one-dimensional input vector $=\{x1, x2, \ldots.xn\}$, $xn \in R^d$ represents variables, and $ec \in R$ represents energy consumption. The convolution 1D creates a feature map *fm* by applying the convolution operator to input data with a filter $w \in R^{fd}$, where *f* represents the intrinsic features of the input data. The result provides a new set of features for the next block in line. Equation yields a new feature map *fm* from a collection of features *f*.

$$HL_I^{fm} = \tanh\left(w^{fm} x_{1:f}, +b\right) \tag{7}$$

The filter HL in Eq. (7) is applied to each pair of input features $\{x_{1:f}, x_{2:f+1}, \ldots x_{n-f+1}\}$ to create a feature map $HL = \left[HL_1, HL_2, \ldots.HL_{n-f+1}\right]$. Additionally, $b \epsilon R$ is a bias term and $L \epsilon R_{n-f+1}$.

The Multi linear transformation weighted inputs drive convolutional layer output. Normally, linear transformation cannot capture complicated data structures, hence a non-linear activation layer is used after convolutional layers to better train data. We used a ReLU activation function that applies max (0, *x*) to each input in this investigation. The pooling layer down-samples the convolutional layer output. This model applies the max-pooling layer to feature maps $L' = \max\left[HL\right]$. This approach chooses the most important attributes with high values. This is the max-pooling layer output,

$$x_i^{'} = CNN\left(x_i\right) \tag{8}$$

where, x_i is the energy-consuming input vector to the CNN network, and $x_i^{'}$ is the output sent to the next LSTM network. In the suggested architecture, CNN receives the input vector and generates $x_i^{'}$ using Eq. (9). The formulation is

$$i_t = \sigma\left(W_i\left(x_t, y_{t-1}\right)\right) \tag{9}$$

$$f_t = \sigma(W_f\left(x_t, y_{t-1}\right) \tag{10}$$

$$O_t = \sigma\left(W_o\left(x_t, y_{t-1}\right)\right) \tag{11}$$

$$g_t = \tanh\left(W_g\left[x_t, y_{t-1}\right]\right) \tag{12}$$

$$c_t = f.c_{t-1} + i.g \tag{13}$$

$$y_t = o.\tanh\left(c_t\right) \tag{14}$$

where, i, f, o, g and *c* are input, forget, output, and input modulation gates. These are n-dimensional real vectors. Eqs. (9) to (14) use a sigmoid function σ and fully connected neural networks W_i, W_f, W_g, W_o for input, forget, output, and input mod energy consumption, latency, and accuracy. Minimize dynamic energy consumption in the microprocessor by reducing the number of computations (convolutions, activations, memory accesses). Ensure the CNN+LSTM can be processed within a certain time limit or with acceptable throughput for the application. The fitness function could be represented as

$$Fitness = \omega_1 \times P_{total} + \omega_2 \times \frac{1}{T} + \omega_3 \times \frac{1}{A}$$

where, P_{total} is the total energy consumption (dynamic + static), T is the execution time (latency) of the CNN+LSTM, A is the accuracy of the CNN+LSTM model, ω_1, ω_2 and ω_3 are weighting factors to prioritize energy, latency, or accuracy based on design requirements.

Algorithm 1: Pseudo code of ACO_CNN+LSTM

Initialization of ACO Parameters $(no_{ants}, eva_{rate}, phre\_rate)$

Set energy and performance constraints

Define Dynamic Voltage Frequency Scaling (DVFS) settings

for iteration in range (max _ *iter*)

for ant in ant colony

build CNN+LSTM architecture

$$tot + pow = pow\left(CNN + LSTM_{config}, voltage, freq\right)$$

$$lat = eval\_lat\left(CNN + LSTM_{config}\right)$$

$$acc = eva + acc\left(CNN + LSTM_{config}\right)$$

$$If\ acc \geq tar\_acc$$

$$fit = pow_{cons} + \left(\frac{1}{lat}\right)$$

else

$$fit = infinity$$

$$update\_phre\left(ant_{path}, fit\right)$$

$$evap\_phre\left(evap_{rate}\right)$$

$$best_{CNN} + LSTMconfig = select_{best} config\left(ants\right)$$

pruned_cnn = prune_cnn_layers(best_cnn_configuration)

deploy_on_processor(quantized_cnn, voltage, frequency)

4 EXPERIMENTAL VALIDATION

Performance analysis

Experimental setup - Integrating both hardware and software components to optimize power efficiency and performance. On the hardware side, the processor should be designed to operate at or near the threshold voltage of transistors, which reduces power consumption but requires mechanisms to handle performance variations. This includes implementing a clock management unit for dynamic adjustment of clock speeds, and a variable voltage regulator to modulate the supply voltage based on workload and performance needs. From a software perspective, the Operating System (OS) has to have the capability to adjust voltage and frequency dynamically (DVFS) and implement power management rules in order to optimise both performance and power consumption. Firmware and drivers should include DVFS support and thermal management to ensure safe operation. Performance monitoring tools and feedback mechanisms are essential for adjusting the processor's settings in response to changes in workload.

Performance metrics - This section presents the experimental validation of the ACO_CNN+LSTM model using four datasets. The validation is conducted by measuring several metrics like Energy Consumption, latency, normalised accuracy, error rate, and Processor Utilisation. The comparison is done between the existing methods such as EP [11], SRAM macro [12] and SE10T [14].

Table 1 Analysis of energy consumption

Normalized frequency	CIFAR-10	IMDB	GTSRB	FEMINIST
1.0×	12.3	11.4	12.45	12.74
1.2×	11.3	11.6	12.53	12.87
1.4×	12.4	11.7	11.56	12.53
1.6×	12.7	12.6	11.65	12.75
1.8×	12.4	11.6	12.75	12.75

From Fig. 2, as the normalized frequency increases from 1.0× to 1.8×, the energy consumption generally rises. This is because energy consumption often scales with the square of the frequency increase. For instance, increasing the frequency from 1.0× to 1.2× could result in an approximate 44% increase in energy consumption, assuming other factors remain constant. The performance data for different datasets shows varying impacts with frequency changes. For CIFAR-10, performance improves up to 1.6× but drops slightly at 1.8×. For IMDB, performance remains relatively stable, while for GTSRB and FEMINIST, performance generally increases with higher frequencies. This indicates that while higher frequencies can enhance performance, they also lead to increased energy consumption.

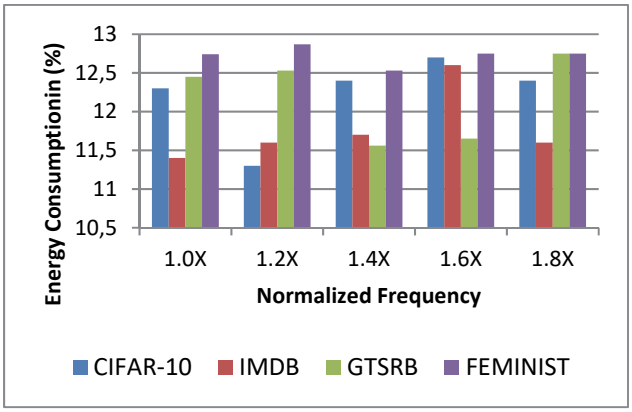

Figure 2 Comparison of energy consumption on various datasets using proposed ACO_CNN+LSTM model

Table 2 Analysis on latency

Normalized frequency	CIFAR-10	IMDB	GTSRB	FEMINIST
1.0×	9.34	8.54	8.54	9.53
1.2×	9.65	8.65	8.75	9.76
1.4×	9.34	8.23	8.23	9.23
1.6×	9.31	8.45	8.68	9.54
1.8×	9.43	8.54	8.53	9.53

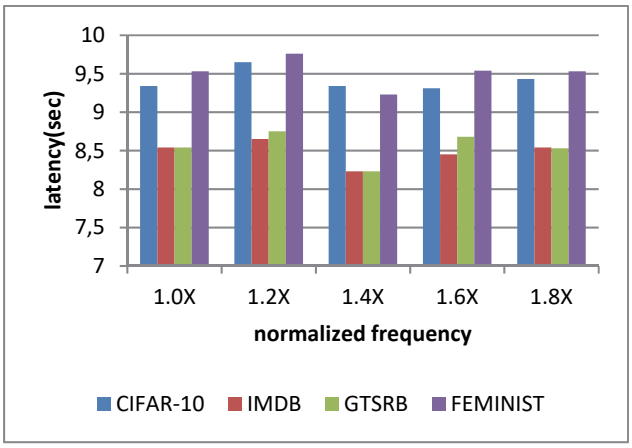

Figure 3 Comparison of latency on various datasets using proposed ACO_CNN+LSTM model

In Fig. 3, for CIFAR-10, latency slightly increases from 9.34 seconds at 1.0× to 9.43 seconds at 1.8×, suggesting a minor slowdown as frequency increases. For IMDB, latency remains relatively stable, starting at 8.54 seconds at 1.0× and remaining the same at 1.8×, indicating minimal effect of frequency on processing time. GTSRB shows a decrease in latency from 8.54 seconds at 1.0× to 8.23 seconds at 1.4×, followed by a slight increase to 8.53 seconds at 1.8×, suggesting that latency initially improves with frequency but stabilizes at higher levels. For FEMINIST, latency decreases from 9.53 seconds at 1.0× to 9.23 seconds at 1.4×, and then returns to 9.53 seconds at 1.8×, indicating that while latency improves with moderate frequency increases, it levels off at higher frequencies.

Table 3 Analysis of normalized accuracy

Normalized frequency	CIFAR-10	IMDB	GTSRB	FEMINIST
1.0×	98.45	98.34	97.45	96.24
1.2×	98.56	98.45	97.45	96.53
1.4×	98.4	98.34	97.34	96.34
1.6×	97.34	98.45	97.53	96.52
1.8×	97.78	98.45	97.45	96.51

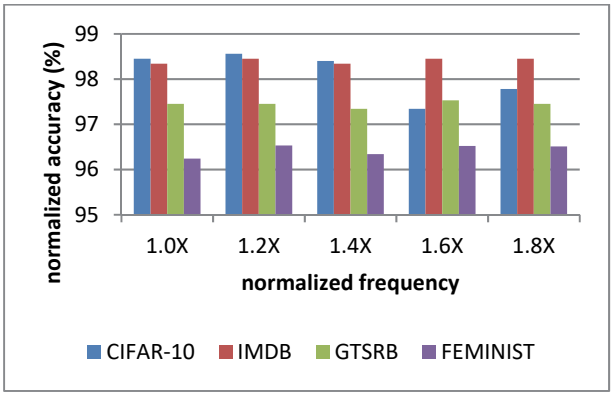

Figure 4 comparison of normalized accuracy on various datasets using proposed ACO_CNN+LSTM model

The normalized accuracy data reveals that as frequency increases, accuracy trends vary slightly across datasets in Fig. 4. For **CIFAR-10**, accuracy increases from 98.45% at 1.0× to 98.56% at 1.2× but then decreases to 97.34% at 1.6×, and slightly recovers to 97.78% at 1.8×, indicating a peak at 1.2X and a slight decline at higher frequencies. For **IMDB**, accuracy remains stable at 98.34% across all frequencies, with only a minor increase to 98.45% at 1.2×, suggesting minimal impact of frequency changes. **GTSRB** shows a small decrease in accuracy from 97.45% at 1.0× to 97.34% at 1.4×, followed by an improvement to 97.53% at 1.6× and a return to 97.45% at 1.8×, reflecting a modest improvement with increased frequency. **FEMINIST** exhibits an increase in accuracy from 96.24% at 1.0X to 96.53% at 1.2×, with subsequent stabilization around 96.51% at higher frequencies, indicating general improvement with frequency increases.

Table 4 Analysis of error rate

Normalized frequency	CIFAR-10	IMDB	GTSRB	FEMINIST
1.0×	23.64	22.54	23.65	22.75
1.2×	22.53	22.15	23.86	22.54
1.4×	23.6	22.75	23.85	22.75
1.6×	23.65	22.64	23.97	22.75
1.8×	23.64	22.75	23.75	22.5

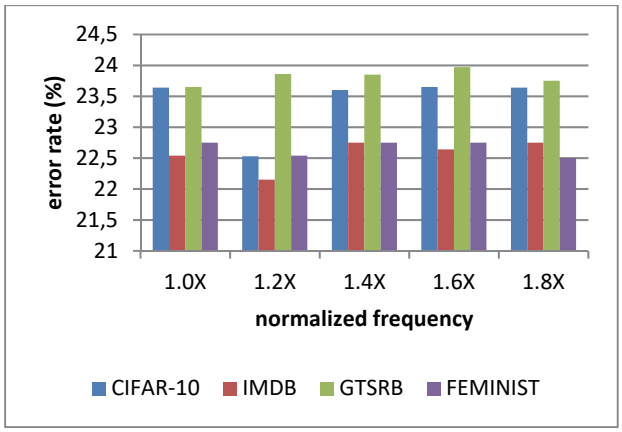

Figure 5 Comparison of error rate on various datasets using proposed ACO_CNN+LSTM model

In Fig. 5 CIFAR-10, the error rate decreases from 23.64% at 1.0× to 22.53% at 1.2×, then increases to 23.60% at 1.4×, and slightly rises to 23.65% at 1.6×, before stabilizing at 23.64% at 1.8X. This indicates an initial improvement in error rate with increased frequency, but then a slight deterioration at higher frequencies. In GTSRB, the error rate shows a slight improvement from 23.65% at 1.0× to 23.86% at 1.2×, then decreases slightly to 23.85% at 1.4×, and further decreases to 23.75% at 1.8×, indicating a general trend of improved error rates with increased frequency, though changes are modest.

Table 5 Analysis of processor utilization

Normalized frequency	CIFAR-10	IMDB	GTSRB	FEMINIST
1.0×	32.53	31.64	32.56	31.56
1.2×	32.5	31.54	32.67	31.74
1.4×	32.5	31.56	32.76	31.67
1.6×	32.76	31.67	32.75	31.86
1.8×	32.4	31.56	32.89	31.78

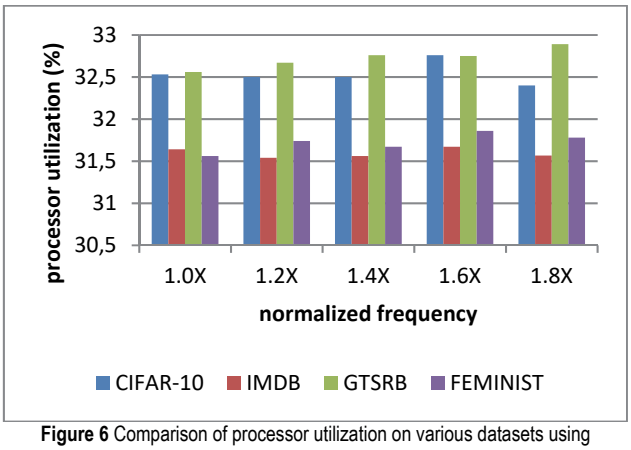

Figure 6 Comparison of processor utilization on various datasets using proposed ACO_CNN+LSTM model

The processor utilization data reveals how processor demand changes with increasing frequency in Fig. 6. For CIFAR-10, utilization starts at 32.53% at 1.0X, decreases slightly to 32.50% at 1.2× and 1.4×, increases to 32.76% at 1.6×, and then decreases to 32.40% at 1.8×, indicating minor fluctuations. For **IMDB**, utilization decreases from 31.64% at 1.0× to 31.54% at 1.2× and remains stable around 31.56% at higher frequencies, showing minimal impact of frequency changes. **GTSRB** shows a steady increase in utilization from 32.56% at 1.0× to 32.67% at 1.2×, 32.76% at 1.4×, and 32.75% at 1.6×, with a slight rise to 32.89% at 1.8×, indicating a general trend of increased demand with higher frequencies. For **FEMINIST**, utilization starts at 31.56% at 1.0×, rises to 31.74% at 1.2×, decreases slightly to 31.67% at 1.4X, and stabilizes at 31.78% at 1.8×, reflecting a moderate increase with frequency changes.

Table 6 Comparative analysis between existing and proposed methods

Parameters	EP	SRAM macro	SE10T	ACO_CNN+LSTM
Energy consumption / %	45.5	37.6	42.5	12.73
Latency / sec	23.6	31.5	21.6	9.46
Normalized accuracy / %	78.4	83.7	75.7	98.45
Error rate / %	35.6	45.6	32.6	22.64
Processor utilization / %	45.5	78.5	56.3	31.67

5 CONCLUSION

The integration of Ant Colony Optimized Convolutional Neural Network (ACO_CNN) with Long Short-Term Memory (LSTM) for energy conservation in a Variation-Tolerant Near-Threshold Processor demonstrates significant advancements in optimizing processor efficiency. The ACO_CNN effectively enhances feature extraction and pattern recognition capabilities, while LSTM contributes to capturing temporal dependencies and managing dynamic workloads. This combination results in improved energy efficiency and performance for near-threshold processing environments, where power consumption and computational efficiency are critical. The approach shows promising results in reducing energy consumption while maintaining or even enhancing processing capabilities compared to traditional methods. Future research could focus on further refining the ACO_CNN+LSTM architecture to enhance its robustness and adaptability to various types of workloads and processor configurations. Exploring advanced optimization techniques and hybrid models may offer additional improvements in performance and energy efficiency.

Acknowledgments

Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2025R729), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia. Research Supporting Project number (RSPD2025R787), King Saud University, Riyadh, Saudi Arabia.

6 REFERENCES

[1] Shan, W., Shang, X., Wan, X., Cai, H., Zhang, C., et al. (2019). A wide-voltage-range half-path timing error-detection system with a 9-transistor transition-detector in 40-nm CMOS. *IEEE Transactions on Circuits and Systems I: Regular Papers*, *66*(6), 2288-2297. https://doi.org/10.1109/TCSI.2019.2900463

[2] Thiruvengadam, C., Palanivelan, M., Senthil Kumar, K., & Jayasankar, T. (2020). Low power approximate adder based repetitive iteration cord (LP-ARICO) algorithm for high-speed applications. *Microprocessors and Microsystems*, *78*, 103260. https://doi.org/10.1016/j.micpro.2020.103260

[3] Wang, S., Chen, C., Xiang, X. Y., & Meng, J. Y. (2017). A Variation-tolerant near-threshold processor with instruction-level error correction. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, *25*(7), 1993-2006. https://doi.org/10.1109/TVLSI.2017.2674139

[4] Pandey, P., Basu, P., Chakraborty, K., & Roy, S. (2019). GreenTPU: Predictive design paradigm for improving timing error resilience of a near-threshold tensor processing unit. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, *28*(7), 1557-1566. https://doi.org/10.1109/TVLSI.2020.2991032

[5] Akyildiz, I., Melodia, T., & Chowdhury, K. (2007). A survey on wireless multimedia sensor networks. *Computer Networks*, *51*(4), 921-960. https://doi.org/10.1016/j.comnet.2006.10.002

[6] Chandler, R., Gibson, S., Karkare, V., Farshchi, S., Markovic, D., & Judy, J. (2009). A system-level view of optimizing high-channel-count wireless biosignal telemetry. *EMBC'09*. https://doi.org/10.1109/IEMBS.2009.5333587

[7] Krashinsky, R., Batten, C., Hampton, M., Gerding, S., Pharris, B., Casper, J., & Asanovic, K. (2004). The Vector-Thread Architecture. *ISCA'04*. https://doi.org/10.1109/ISCA.2004.1310750

[8] Liang, X., Wei, G., & Brooks, D. (2008). Revival: A variation-tolerant architecture using voltage interpolation and variable latency. *ISCA'08*. https://doi.org/10.1109/ISCA.2008.19

[9] Mirasol. (n.d.). *MEMS Drive IMOD Reflective Technology*. See http://www.mirasoldisplays.com/mobile-display-imodtechnology

[10] Woh, M., Seo, S., Mahlke, S., Mudge, T., Chakrabarti, C., & Flautner, K. (2009). AnySP: Anytime Anywhere Anyway Signal Processing. ISCA'09. https://doi.org/10.1109/ISCA.2009.5062542

[11] Yu, R., Li, Z., Deng, X., Wang, Z., Zhang, H., & Liu, Z. (2024). Margin Elimination in a 55 nm Near-Threshold Microcontroller with Adaptive Prediction Capability and Voltage Scaling. *Electronics*, *13*(7), 1211. https://doi.org/10.3390/electronics13071211

[12] Deng, X., Yu, R. Z., Li, Z. H., Zhang, H. M., & Liu, Z. L. (2024). A Low-Power Variation-Tolerant 7T SRAM With Enhanced Read Sensing Margin for Voltage Scaling. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. https://doi.org/10.1109/TCAD.2024.3012876

[13] Shao, L., Lai, M., Xu, S., Lin, C., & He, W. (2023, May). A Metastability Inference and Avoidance Technique for Near-Threshold-Voltage Network-on-Chip. *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*, 1-5. https://doi.org/10.1109/ISCAS.2023.10137846

[14] Abbasian, E. & Sofimowloodi, S. (2023). Energy-efficient single-ended read/write 10T near-threshold SRAM. *IEEE Transactions on Circuits and Systems I: Regular Papers*, *70*(5), 2037-2047. https://doi.org/10.1109/TCSI.2023.3272763

[15] Abbasian, E., Grailoo, B., & Nayeri, M. (2023). Design of a 10-nm FinFET 11 T near-threshold SRAM cell for low-energy internet-of-things applications. *Circuits, Systems, and Signal Processing*, *42*(5), 3138-3151. https://doi.org/10.1007/s00034-022-02249-0

[16] Rawat, B. & Mittal, P. (2023). A low power single bit-line configuration dependent 7T SRAM bit cell with process-variation-tolerant enhanced read performance. *Analog Integrated Circuits and Signal Processing*, *115*(1), 77-92. https://doi.org/10.1007/s10470-022-01976-4

[17] Fan, X., Liu, H., Li, H., Lu, S., & Han, J. (2022). Design of light-weight timing error detection and correction circuits for energy-efficient near-threshold voltage operation. *Electronics*, *11*(18), 2879. https://doi.org/10.3390/electronics11182879

[18] Gundi, N. D., Pandey, P., Roy, S., & Chakraborty, K. (2022). Implementing a Timing Error-Resilient and Energy-Efficient Near-Threshold Hardware Accelerator for Deep Neural Network Inference. *Journal of Low Power Electronics and Applications*, *12*(2), 32. https://doi.org/10.3390/jlpea12020032

[19] Abbasian, E. (2022). A highly stable low-energy 10T SRAM for near-threshold operation. *IEEE Transactions on Circuits and Systems I: Regular Papers*, *69*(12), 5195-5205. https://doi.org/10.1109/TCSI.2022.3210383

Contact information:

C. THIRUVENGADAM
Department of Electronics and Communication Engineering,
Anjalai Ammal Mahalingam Engineering College,
Thiruvarur 614 403, Tamil Nadu, India
E-mail: cthiruvengadam81@gmail.com

Mashael MAASHI
Department of Software Engineering,
College of Computer and Information Sciences,
King Saud University,
Po box 103786, Riyadh 11543, Saudi Arabia
E-mail: mashi@ksu.edu.sa

Jamal ALSAMRI
Department of Biomedical Engineering,
College of Engineering,
Princess Nourah bint Abdulrahman University,
Saudi Arabia
E-mail: jalsamri@pnu.edu.sa

Raghu GUNDAALA, Research Scholar
(Corresponding Author)
Department of ECE,
Saveetha School of Engineering,
Saveetha Institute of Medical and Technical Sciences,
Chennai, India
E-mail: raghugundaalasimts@gmail.com