IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Efficient Deep Learning Approaches For Low-Power Approximate Multiplier Architectures

[1]Ajeesh S, [2]Deebu U S, [3]Anoop S

[1]Computer Science and engineering
[1]College of engineering, Kottarakkara, Kerala
[2]Electronics and Communication engineering,
[2]College of engineering, Adoor, Kerala
Electronics and Communication engineering,
[3]College of Engineering, Kottarakkara, Kerala

Abstract: A novel approach for design of low-power approximate multipliers by leveraging Long Short-Term Memory (LSTM) networks within a deep learning (DL)-based framework. Our proposed architecture, referred to as the DL -Based Approximate Multiplier (DLAM), exploits the sequence-to-sequence learning capabilities of LSTMs to enhance the efficiency of approximate multiplication in terms of both accuracy and power consumption. The DLAM model is trained on a diverse dataset, incorporating various input patterns and corresponding approximate multiplication outcomes. Through the integration of LSTM units, the model captures long-range dependencies within the input sequences, enabling more accurate predictions of approximate multiplication results. The trained DLAM exhibits superior performance in terms of both precision and energy efficiency when compared to traditional approximate multiplier designs. Furthermore, we explore optimization techniques to minimize power consumption without compromising the accuracy of multiplication results. Our test findings show that the DLAM accomplishes a significant reduction in power consumption while maintaining competitive levels of accuracy, making it a promising candidate for low-power applications in energy-constrained environments**.**

Index Terms - low-power design, DL, LSTM networks, metaheuristics, Jellyfish Search Optimization algorithm, sequential data.

I. INTRODUCTION

Approximate computing has emerged as a promising paradigm to strike a balance between computational accuracy and energy efficiency in resource-constrained environments. One key aspect of approximate computing is the design of low-power approximate multipliers [1], which play a crucial role in a myriad of applications, including signal processing [2], machine learning (ML), and digital signal processors [3]. This study investigates the incorporation of LSTM [4] networks into a DL-oriented framework to improve the optimal accuracy in the design of approximate multipliers.

LSTM networks are renowned for their ability to capture long-range dependencies in sequential data, making them well-suited for tasks involving input sequences with contextual information. In this study, the LSTM is harnessed to improve the precision of approximate multiplication outcomes by leveraging its sequence-to-sequence learning capabilities. This integration enhances the model's capacity to discern intricate patterns and relationships within the input data, leading to more accurate predictions of approximate multiplication results. Moreover, to further boost the efficiency of the DLAM, a metaheuristics algorithm is incorporated into the training process. Specifically, the Jellyfish Search Optimization algorithm [5], known for its ability to explore solution spaces efficiently, is employed to fine-tune the DLAM parameters. This

synergistic combination aims to enhance precision and resilience of the DLAM model, ensuring optimal execution in terms of precision and energy efficiency.

The utilization of LSTM networks and metaheuristics algorithms in approximate multiplier design represents a novel approach to address the intricate trade-off between accuracy and power consumption. By exploring the synergy between DL and metaheuristics, this research seeks to push the boundaries of efficiency in approximate computing, with potential applications in energy-constrained embedded systems and edge devices. The subsequent sections delve into the methodology, experimental setup, and results, shedding light on the advancements achieved through the integration of LSTM and metaheuristics in the context of approximate multiplier design.

II. LITERATURE REVIEW

Usharani et al.[6] introduces Deep-PowerX1, a novel framework integrating DL and Low Power Design for logic synthesis optimization. Employing a Deep Neural Network (DNN), Deep-PowerX reduces power and area by up to 1.47× and 1.43×, outperforming exact solutions and state-of-the-art tools by 22% and 27%. Limitations include a focus on digital CMOS circuits and a predefined error rate constraint. Nagarajan et al. [7] introduces an energy-efficient 1-bit GDI-based full swing full adder (EAFA) for approximate computing in error-tolerant applications. The proposed 16-bit EAHSETA demonstrates superior performance with 95% classification accuracy, outperforming GDI-CMBAI, GDI-AMBAII, and GDI-HSETAII in area and power consumption by 88.31%, 80.27%, and 77.34%, respectively. Limitations include focus on handwritten digit recognition and potential accuracy trade-offs. The work holds promise for resource-constrained, high-speed, low-power DL applications.

Yang et al. [8] introduces Stochastic Computing as a low-cost, low-power alternative for Convolutional Neural Networks (CNNs), demonstrating MNIST and CIFAR-10 datasets. SC achieves almost 3x learning speed increase for MNIST with 1.37% accuracy degradation and 3.5x acceleration for CIFAR-10 with a 3.39% degradation. The methodology involves two CNN architectures, OpenCL framework include bias optimization on GPUs and FPGAs. Limitations include potential scalability challenges for larger networks. The work provides valuable insights for efficient and accelerated CNN training using stochastic computing. Murillo et al. [9] introduces a Posit Logarithm-Approximate Multiplication (PLAM) scheme to enhance the efficiency of posit multipliers in Deep Neural Network architectures. Experimental results demonstrate remarkable reductions in area, power, and delay, up to 72.86%, 81.79%, and 17.01%, respectively, without compromising accuracy. While promising, limitations may include specific applicability and potential challenges in scaling for more complex networks. The work significantly contributes to advancing the efficiency of Posit arithmetic units in DNNs.

Saravanan et al.[10] explores an innovative approach for an energy-efficient hardware accelerator, utilizing Silicon Nanowire Reconfigurable Field Effect Transistors for ML acceleration. Employing RFET-based Multiply and Accumulate units yields a substantial 70% power reduction compared to traditional CMOS, with minimal impact on accuracy. The suggested RFET-driven accelerator accomplishes 94% on MNIST datasets, demonstrating a remarkable decrease of 93% in size and energy consumption, and 73% reduction in delay. However, limitations may include the generalizability of findings to diverse datasets and application scenarios. The study significantly contributes to advancing energy-efficient ML accelerators through RFET technology. Kim et al. [11] investigates the impact of approximate multiplication on deep convolutional neural networks (CNNs), emphasizing its potential for efficient hardware acceleration. Experiments on recognized network architectures, such as ResNet and Inception-v4, demonstrate near FP32 accuracy with Mitch-w6 multiplication. The findings justify approximating multiplications while ensuring accurate additions in CNNs, offering insights for hardware accelerator design and analytical understanding of approximation techniques. Limitations may include specific network architectures and potential trade-offs in other applications. The work contributes significantly to advancing efficient CNN hardware accelerators.

III. METHODOLOGY

3.1. LSTM model

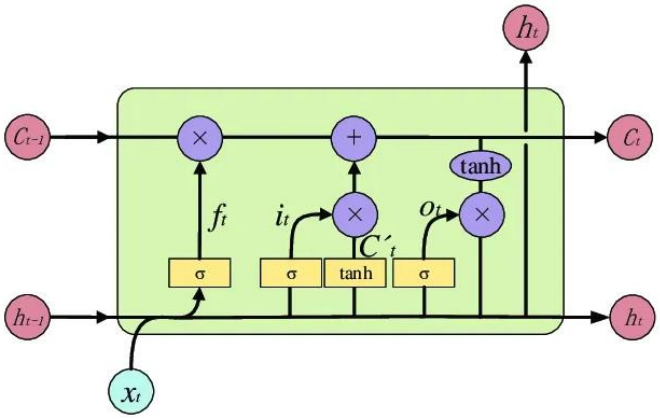

Fig. 1: LSTM model

LSTM is a widely used DL model, especially in sequential modeling. As a variant of recurrent neural networks (RNNs) [12], LSTM excels in capturing long-range dependencies and mitigating the vanishing gradient problem. Its unique architecture incorporates memory cells, input, forget, and output gates, enabling the effective processing of sequential data. LSTMs are particularly favored for applications involving time-series analysis, natural language processing [13], and speech recognition due to their ability to retain and selectively forget information over extended sequences, contributing to their prominence in contemporary DL applications. Figure 1 shows one of the LSTM model used in this study.

The derivation of the forget state, considering the () and (), is expressed through mathematical equations as follows.

$$F_t = \sigma(\mathrm{W_f}[h_t - 1\ ,\ x_t] + b_f \qquad (1)$$

Table 1: Parameters and Its Representation

parameters	symbol
input	X_t
hidden layer activation functions	h_t
Sigmoid activation function	σ
weight matrices	W
threshold bias	b
Input Gate	i_t
new hidden cell state	o_t
Mean Square Errors	MSE
desired output of the approximate multiplier	D_i
predicted output	P_i

$$\sigma(\mathrm{X}) = \frac{1}{1 - e^{-x}} \qquad (2)$$

The activation of the Input Gate and Output Gate leads to the generation of a new hidden cell state, C_t. This process involves the use of sigmoid (σ) and hyperbolic tangent (tanh) activation functions for both gates. Table 1 provides a comprehensive overview of parameters along with their respective representations.

$$i_t = \sigma(\mathrm{w_i}[h_{t-1},\ x_t] + b_i \tag{3}$$

$$o_t = \tanh w_o[h_{t-1},\ x_t] + b_o \tag{4}$$

$$\tanh(X) = \frac{e^{x'} - e^{-x'}}{e^{x'} + e^{-x'}},\ \text{between}\ -1\ \text{to}\ 1 \tag{5}$$

JellyFish search optimization is employed for this fine-tuning, yielding optimal results. Utilizing this objective function, the approximate computing multiplier undergoes training with the optimized LSTM, generating a predefined library function.

$$MSE = minimum\left(\frac{\sum_{i=1}^{N}(D_i - P_i)^2}{N}\right) \tag{6}$$

3.2. Proposed Methodology

In Figure 2, the input module retrieves data for computation, directing it to the approximate multiplier. This multiplier encompasses two key processes: a pre-trained library and Jelly-optimized LSTM. Initially, the Jelly-optimized LSTM ensures an optimal solution with heightened accuracy.

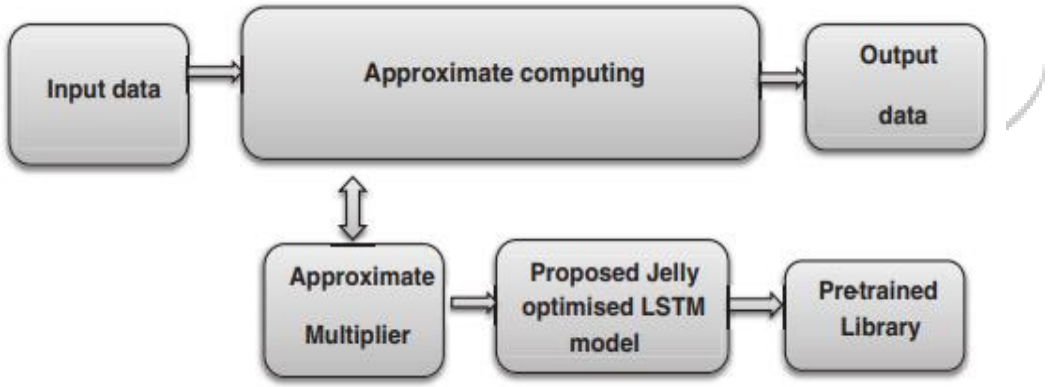

Fig. 2: Novel approximate computing

The fine-tuned values are subsequently saved in a memory unit, forming the refined pre-trained repository. This library serves as a repository for a significant amount of data post the Jelly-optimized LSTM process. The input data affects the rough multiplier, which generates an output value that is sent to the output module. In the end, the approximate computational process yields a high degree of precision, low power consumption, and little mistake.

3.2.1. Innovative Gel-Enhanced (LSTM)

We present the Jelly Optimized LSTM, which refines Jellyfish search optimization for LSTM hyperparameters. The main hyperparameters, including weight and the LSTM's threshold biases. Assessing this objective function produces the best outcome. Following that, the LSTM model is involved in the classification procedure within the approximate computing multiplier. The application of the Optimized LSTM improves accuracy of approximate calculation by attaining optimal input awareness.

3.2.2. Jellyfish optimizer

Jellyfish, globally recognized, inhabit diverse water bodies. Characterized by a gentle, bell-shaped structure and lengthy, diverse appendages furnished with venom for prey capture, jellyfish exhibit unique features, including autonomous movement control. Employing an umbrella structure, they propel themselves forward by expelling water, often drifting with currents and tides. Under favorable conditions, jellyfish aggregate to form a mass known as a Jellyfish bloom. Formation parameters, including seawater movements, levels of oxygen, nutritional content, predatory activities, and temperature impact the formation of swarms. The Jellyfish Search algorithm draws inspiration from these creatures' searching behavior and oceanic movements. The algorithm involves three actions: living in ocean currents or toward the swarm with a time control process, seeking food based on prey quantity, and evaluating an identified prey position. Figure 3 depicts oceanic food-searching behavior of Jellyfish.

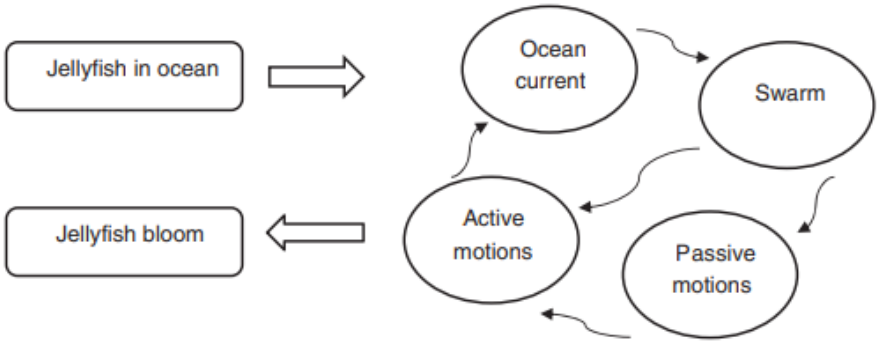

Fig 3: Behaviour of Jellyfish sea

3.2.3. Ocean waves

Jellyfish are drawn to ocean currents abundant with nutrients. The direction or trend is mathematically expressed in Equation (1).

$$\overrightarrow{trend} = X^* - \beta * random(0,\ 1) * \mu, \qquad (7)$$

$X^* \rightarrow$ current best position of the jellyfish

Here, denotes the, calculated as the mean value of each jellyfish location. The parameter β, where β > 0, signifies a coefficient influencing the distribution. Positions is expressed through formulae:

$$X_i(t+1) = X_i(t) + randon(0.1) * \overrightarrow{trend} \qquad (8)$$

3.2.4. Jellyfish swarm(j)

A collective denotes a considerable assemblage of 'j,' preserving its initial location, referred to as passive movement or Type A, or embracing fresh location, recognized movement or Type B. In instances of Type A movement, 'j' revise their positions using the subsequent formula.

$$X_i(t+1) = X_i(t) + \gamma * randon(0.1) * (UB - LB) \qquad (9)$$

In this context, UB signifies the upper limit of the exploration area, LB indicates the lower limit of the exploration area, and γ represents the motion coefficient. The adjustment of the coefficient with the extent of the motion takes place, and a particular value is designated, where γ is established at 0.1.

In Equation (9), we simulate Type B motion, which involves randomly selecting jellyfish ('j') to assess the course of movement. The jellyfish vector was selected out of interest 'i' directed toward the selected ('j'). If the anticipated prey chosen ('j') position exceeds that with 'i,' it resets to the initial position. Conversely, if the available prey for the chosen ('j') is minimal compared to the position of 'i,' it moves directly away from that

location. This iterative process enables each jellyfish to continually determine enhanced prey positions by adjusting their locations.

$$\overrightarrow{Dir} = \begin{cases} X_j(t) - X_i(t) & if\ f(X_i) \geq f(X_j) \\ X_j(t) - X_j(t) & if\ f(X_i) \geq f(X_j) \end{cases} \quad (10)$$

$$X_i(t+1) = -X_i(t) + \overrightarrow{step} \quad (11)$$

To evaluate the types of motion throughout time, a time regulation system is implemented. This system oversees the entire swarm, encompassing both type A and type B motions, and guides the motion of jellyfish toward an sea wave. The details of the mechanism for temporal control are as follows.

3.2.5. Time control mechanism

Jellyfish are consistently drawn to ocean currents abundant with nutritious plants, often influenced by factors such as temperature, wind, and atmospheric changes carried within these currents. As a response, jellyfish swarms migrate to alternate ocean currents, forming new swarms. Within a swarm, jellyfish exhibit movement between Type A and Type B motions. Initially, Type A motion is predominantly selected, gradually transitioning to a preference for Type B motion. Where,

$c(t) \rightarrow$ time control mechanism

$$c(t) = \left| 1 - \frac{1}{Max_{iter}} * (2 * random(0,\ 1) - 1) \right| \quad (12)$$

t →the number of iterations within a specified time

$Max_{iter} \rightarrow$ maximum allowable number of iterations

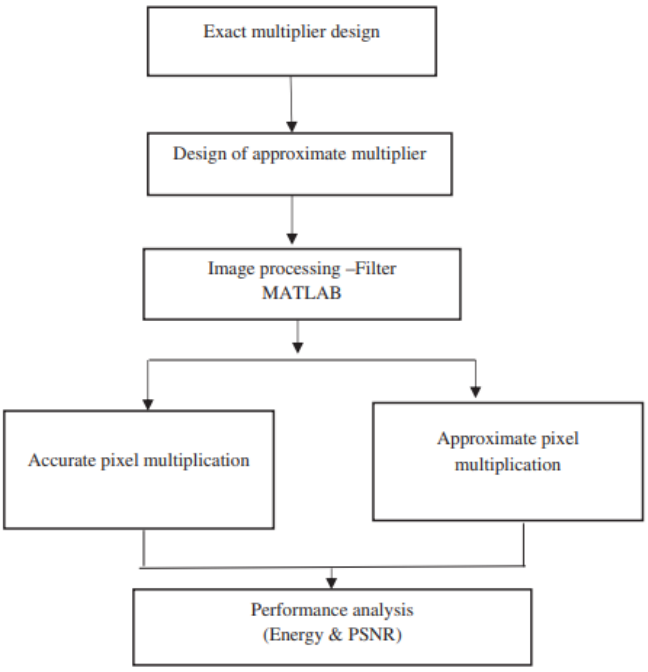

Fig. 4: Suggested assessment of image processing efficiency.

IV. RESULT AND DISCUSSION

Synopsys compiler is used to simulate the proposed multiplier, which is written in Verilog. The proposed multiplier is evaluated taking into account variables like area, delay, power, mean relative error (MRE), and normalized error distance (NRE), in contrast to conventional provide multipliers like Under Designed Multiplier (UDM), Partial Product Perforation (PPP), Static Segment Multiplier (SSM), Approximate Compressor-Based Multiplier (ACM), and ML-based Approximate Multiplier (MLAM). In the context of image processing applications, a Geometric Mean Filter (GMF) is utilized to enhance the visual effects of images by executing geometric mean operations on pixels, effectively eliminating undesired noise. The resulting output image post the GMF application is expressed as follows.

$$GMF(x,\ y) = \left[\prod I(i;\ j) \right]^{1/mn} \tag{13}$$

Here, I present the original picture. The numbers of pixels within the GMF increased to the power of 1/mn are multiplied for each pixel in the GMF treated image at coordinates (x, y). In order to assess the effectiveness of the proposed DL-based multiplier, two grayscale images with Gaussian noise and 8 bits per pixel are used. A 3*3 mean filter with surrounding pixels focused around them is applied to the noisy images. Figure 4 shows the procedure involving the suggested Peak Signal-to-Noise Ratio (PSNR) and energy required. Table 2 presents the performance metrics of the proposed multiplier. The ratio of an image's maximal potential power to the amount of corrupting noise that degrades its quality is known as PSNR.

In the suggested DL multiplier framework, a approximated library is created for every conceivable input combination. The model undergoes training using different levels of output truncation, accompanied by associated area, energy, and error measurements. This training assists the model in determining the most suitable truncation bits and error compensation values. In the case of an 8-bit multiplier, 216 input combinations and error metrics are taken into account for numerous truncation and error compensation values, guiding the LSTM model in choosing the optimal multiplication while considering input variations.

Table 2: Power, Delay Analysis And Area Of Proposed Multiplier

TYPE	AREA	POWER
Multiplier without approximation	2412.6	872.19
DLAM	853.19	151.51
MLAM	913.66	167.69
AM	1079.14	251.3
ACM	1365.56	219.22
SSM	1977.51	630.49
PPP	2270.37	786.43
UDM	1970.3	662.44

Table 3: NRE and MRE analysis

TYPE	MRE	NRE
DLAM	0.187	0.003
MLAM	0.296	0.03
AM	0.305	0.005
ACM	0.376	0.006
SSM	0.319	0.006
PPP	0.452	0.007
UDM	0.332	0.007

Relative to alternative approximate multipliers, the suggested DL multiplier exhibits enhanced efficiency concerning area, power, and delay. Table 3 presents a contrast of error metrics for the proposed multiplier. The noted diminished values of Mean Relative Error (MRE) and Normalized Error Distance (NRE) signify the improved adaptability of the proposed multiplier for applications demanding substantial power conservation with heightened tolerance for errors.

Table 4: Average Energy And PSNR Required For Filtering

Benchmark	MLAM (PSNR)	DLAM (PSNR)	DLAM (Energy-µJ)	DLAM (Energy-µJ)
Lena	67.2	73.5	1.90	1.34
Airplane	74	76.81	2.06	1.89
Baboon	69.5	72.5	1.10	0.94
Peppers	53	56.4	0.98	0.54
Cameraman	71.5	74.9	1.89	1.71
Moon surface	63	72.4	1.13	0.82

Table 4 displays the Peak Signal-to-Noise Ratio (PSNR) and saved energy for images affected by noise, which have undergone processing to generate Geometric Mean Filtered images. The energy requirements for ML and DL based approximate multipliers are 1.92 µJ and 1.35 µJ, respectively. The PSNR outputs for multipliers are recorded as 71.5 and 74.9, respectively.

V. CONCLUSION

The proposed DL multiplier, incorporating a fine-tuned LSTM model and Jellyfish Search Optimization, demonstrates notable advancements in approximate computing. The DL model efficiently handles the generation of an approximated library for diverse input combinations, optimizing truncation levels and error compensation values. Comparative evaluations against conventional approximate multipliers underscore the superiority of the proposed DL multiplier in terms of power, delay, and area. Additionally, experimental results on noisy image processing reveal significant energy savings and improved PSNR with the DL-based approach. The study highlights the efficacy of DL techniques, paving the way for high-performance approximate multipliers in applications demanding energy efficiency and error tolerance.

REFERENCES

[1] Strollo, A. G. M., Napoli, E., De Caro, D., Petra, N., & Di Meo, G. (2020). Comparison and extension of approximate 4-2 compressors for low-power approximate multipliers. IEEE Transactions on Circuits and Systems I: Regular Papers, 67(9), 3021-3034.

[2] Zhang, J. A., Liu, F., Masouros, C., Heath, R. W., Feng, Z., Zheng, L., & Petropulu, A. (2021). An overview of signal processing techniques for joint communication and radar sensing. IEEE Journal of Selected Topics in Signal Processing, 15(6), 1295-1315.

[3] Engel, J., Hantrakul, L., Gu, C., & Roberts, A. (2020). DDSP: Differentiable digital signal processing. arXiv preprint arXiv:2001.04643.

[4] Staudemeyer, R. C., & Morris, E. R. (2019). Understanding LSTM--a tutorial into long short-term memory recurrent neural networks. arXiv preprint arXiv:1909.09586.

[5] Chou, J. S., & Molla, A. (2022). Recent advances in use of bio-inspired jellyfish search algorithm for solving optimization problems. Scientific Reports, 12(1), 19157.

[6] Usharani, M., Sakthivel, B., Priya, S. G., Nagalakshmi, T., & Shirisha, J. (2023). An optimized deep-learning-based low power approximate multiplier design. Comput. Syst. Sci. Eng, 44(2), 1647-1657.

[7] Nagarajan, M., Muthaiah, R., Teekaraman, Y., Kuppusamy, R., & Radhakrishnan, A. (2022). Power and Area Efficient Cascaded Effectless GDI Approximate Adder for Accelerating Multimedia Applications Using Deep Learning Model. Computational Intelligence and Neuroscience, 2022.

[8] Yang, T., Ukezono, T., & Sato, T. (2019, May). Design of a Low-power and Small-area Approximate Multiplier using First the Approximate and then the Accurate Compression Method. In Proceedings of the 2019 on great lakes symposium on VLSI (pp. 39-44).

[9] Murillo, R., Del Barrio, A. A., Botella, G., Kim, M. S., Kim, H., & Bagherzadeh, N. (2021). PLAM: A posit logarithm-approximate multiplier. IEEE Transactions on Emerging Topics in Computing, 10(4), 2079-2085.

[10] Saravanan, R., Bavikadi, S., Rai, S., Kumar, A., & Dinakarrao, S. M. P. (2023, May). Reconfigurable FET Approximate Computing-based Accelerator for Deep Learning Applications. In 2023 IEEE International Symposium on Circuits and Systems (ISCAS) (pp. 1-5). IEEE.

[11] Kim, M. S., Del Barrio, A. A., Kim, H., & Bagherzadeh, N. (2021). The effects of approximate multiplication on convolutional neural networks. IEEE Transactions on Emerging Topics in Computing, 10(2), 904-916.

[12] Staudemeyer, R. C., & Morris, E. R. (2019). Understanding LSTM--a tutorial into long short-term memory recurrent neural networks. arXiv preprint arXiv:1909.09586.

[13] Chowdhary, K., & Chowdhary, K. R. (2020). Natural language processing. Fundamentals of artificial intelligence, 603-649.