ENERGY EFFICIENT MECHANISMS FOR NEURAL NETWORKS

Daniel Ekpah and Moshood Abass

Department of Electrical/ Electronic Engineering, Faculty of Engineering, University of Port Harcourt, Rivers State, Nigeria

Abstract

Energy consumption during neural network training and inference has become a significant challenge due to the growing complexity of deep learning models and the rapid expansion of AI applications in edge and cloud environments. This research explores ways to reduce the energy demand of neural networks while maintaining an acceptable level of model accuracy. The main issue being addressed is the excessive energy consumption caused by high multiply-accumulate (MAC) operations, frequent memory access, and inefficient use of hardware resources, which raise operational costs and hinder sustainable deployment. The study used mathematical energy modeling and optimization techniques to measure energy consumption, including decomposition of computation-memory energy, training energy estimation, and inference energy assessment. Additionally, pruning, quantization, and compression techniques were applied to simplify models. Hardware-aware efficiency was assessed through dynamic energy modeling based on voltage, frequency, and capacitance, as well as throughput-per-watt evaluation of accelerators. The energy-performance trade-offs were examined using energy-delay products (EDP) and a multi-objective sustainability cost function. The results showed that Transformer models consumed the most energy (39.6 J) and had the highest EDP (7.13 J·s), while DNN3 had the lowest EDP (0.97 J·s), with an energy consumption of 12.1 J and an inference time of 0.08 s. Pruning improved efficiency but lowered accuracy—CNN accuracy decreased from 94% to 91% as pruning ratios increased from 0.33 to 0.54. Voltage scaling also reduced power from 30 W to 14 W by lowering the voltage from 1.1 V to 0.7 V. The findings suggest that energy-conscious AI policies should encourage model compression, hardware-efficient accelerators, and sustainability-focused deployment metrics to reduce carbon emissions and lower operational costs.

Keywords: Energy Efficiency, Neural Networks, Model Compression, Hardware Acceleration, Pruning

I. Introduction

Energy efficiency is becoming a crucial factor in the design and deployment of neural networks, driven by the widespread adoption of AI technologies across various industries such as healthcare, transportation, smart cities, and the Internet of Things (IoT) [8]. Deep learning models, especially neural networks, have shown remarkable success in tasks like image recognition, speech processing, natural language understanding, and predictive analytics [9]. However, the complexity of these models often leads to high energy consumption, limiting their practical use in everyday applications [10]. As AI systems become more integrated into daily life, making neural networks energy-efficient has become both a technological and an environmental priority.

The demand for advanced neural network models has resulted in the development of large architectures with millions or even billions of parameters. While these models achieve impressive accuracy, they require significant computing power and memory, leading to high energy consumption, particularly in data centers where neural networks are trained and run at scale. Energy-intensive AI workloads contribute to rising operational costs, increased carbon emissions, and greater pressure on global energy resources [11]. Moreover, deploying these models on edge devices like smartphones, drones, wearables, and embedded systems presents additional challenges, as these devices have limited battery life and computational capacity [12]. As a result, achieving energy-efficient neural networks is key to ensuring the sustainability and scalability of AI technologies.

Energy-efficient approaches for neural networks focus on reducing power usage without compromising performance. Strategies include optimizing model architectures, eliminating redundant calculations, and creating algorithms tailored to the constraints of real-world hardware [13]. Lightweight models such as MobileNet and EfficientNet offer strong performance while minimizing computational requirements.

Techniques like pruning, quantization, and compression help reduce memory usage and lower energy demands during training and inference. Knowledge distillation is another approach in which a smaller model mimics the outputs of a larger one, achieving efficiency with minimal energy expenditure [14].

Advancements in specialized hardware accelerators like GPUs, TPUs, and neuromorphic chips are also contributing to improved energy efficiency [15][16]. These platforms are designed to execute neural network tasks more effectively than traditional processors [18]. Additionally, techniques like energy-aware scheduling and dynamic voltage-frequency scaling are being used to optimize energy consumption during neural network execution. In edge computing, federated learning and distributed AI models are being explored to reduce communication overhead and further minimize energy use during training. These innovations highlight the importance of hardware as well as software optimization in achieving energy efficiency [17].

As the global focus shifts toward sustainability, energy-efficient neural networks play a critical role in reducing the environmental impact of AI technologies [19]. High energy consumption during neural network training can contribute to significant carbon footprints, comparable to those of major industrial operations. Therefore, integrating energy-conscious strategies into the entire AI development cycle—from data preparation to model deployment—is essential. Efficient neural networks can reduce greenhouse gas emissions, extend battery life in portable devices, and lower operational costs in large-scale infrastructures, making AI more accessible and affordable, especially in developing regions [20].

II. Literature Review

The need for energy-efficient neural networks has become increasingly urgent as deep learning models grow more complex and are deployed in real-world applications. Various studies have explored both algorithmic and hardware approaches to reduce energy consumption while maintaining model accuracy. For example, Sahlol et al. [1] proposed an optimization method based on artificial ecosystems to enhance feature selection for deep neural networks in tuberculosis detection. Their approach reduced computational requirements and energy consumption while maintaining diagnostic accuracy. Similarly, Maxwell et al. [2] highlighted the trade-off between model accuracy and energy usage in CNNs for remote sensing, showing that high-resolution models, although accurate, require substantial computational resources, making energy-efficient deployment essential for large-scale applications.

Recurrent neural networks (RNNs) have also been examined for energy efficiency in tasks involving sequential data. Ackerson et al. [4] explored RNNs for biometric authentication and anomaly detection, showing that while RNNs provide strong sequence modeling capabilities, they are computationally intensive. Lin et al. [5] improved RNN efficiency by using attention mechanisms, which focused computations on the most important sequence elements, thus lowering energy costs. In agriculture, Anagnostis et al. [6] optimized RNN models for human activity recognition, balancing energy consumption and accuracy.

From a hardware perspective, Du et al. [8] and Chen et al. [9] emphasized the importance of designing accelerators that match the needs of deep neural networks. Du et al. introduced a reconfigurable streaming CNN accelerator for IoT applications, showing how parallelized operations and hardware-specific optimizations can lead to significant energy savings. Chen et al. reviewed accelerator architectures and pointed out that effective deployment requires co-designing both algorithms and hardware to minimize power use without sacrificing throughput. Yin et al. [10] proposed a hierarchical inference model for IoT devices, demonstrating that efficient distributed computation and memory access patterns are essential for sustainable deployment.

Finally, Rahman et al. [7] applied bidirectional LSTM models to code evaluation and repair, showing how optimizing model complexity with techniques like pruning and quantization can reduce energy consumption. These studies illustrate that achieving energy efficiency in neural networks involves a combination of algorithmic optimization, model compression, and hardware-aware design, forming the basis for strategies to reduce energy use in edge and cloud-based systems.

III. METHODS

3.1To examine the major factors contributing to high energy consumption in neural network training and inference processes

This equation models the total energy consumed during neural network execution by separating energy usage into computation and memory access components. It captures the fact that multiply-accumulate operations consume significant energy, while memory read/write operations also contribute heavily, especially in deep networks. This relationship

is useful for identifying the dominant energy sources in training and inference processes for optimization purposes.

$$E_{total} = N_{MAC}E_{MAC} + N_{mem}E_{mem} \quad (1)$$

Where,

E_{total}= total energy consumed (Joules)

N_{MAC}= number of multiply-accumulate operations

E_{MAC}= energy per MAC operation (J/op)

N_{mem}= number of memory accesses

E_{mem}= energy per memory access (J/access)

This equation estimates the total training energy consumption of a neural network by considering the number of epochs, batch size, and energy required per batch computation. Since training repeats forward and backward passes multiple times, energy usage scales with dataset size and training iterations. It helps quantify how training parameters affect energy demand and supports strategies to reduce energy consumption efficiently.

$$E_{train} = N_{epoch} \times N_{batch} \times E_{batch} \quad (2)$$

where,

E_{train}= total training energy (Joules)

N_{epoch}= number of training epochs

N_{batch}= number of batches per epoch

E_{batch}= energy consumed per batch computation (J/batch)

This equation expresses inference energy consumption as a product of inference time and average power usage. It emphasizes that energy is not only affected by computation complexity but also by execution duration and hardware power characteristics. The formula is important for evaluating deployment systems, especially edge devices, where reducing inference time directly lowers energy consumption and extends battery lifespan.

$$E_{inf} = P_{avg} \times T_{inf} \quad (3)$$

where,

E_{inf}= inference energy consumption (Joules)

P_{avg}= average power during inference (Watts)

T_{inf}= inference execution time (seconds)

3.3 To develop and evaluate energy efficient optimization techniques such as pruning, quantization, and model compression for reducing computational complexity

This equation computes the pruning ratio, which indicates how many network weights are removed relative to the total parameters. Pruning reduces computational cost and memory requirements by eliminating redundant connections. A higher pruning ratio means more compression, which can significantly lower energy consumption. This equation is essential for quantifying pruning effectiveness while ensuring model accuracy remains acceptable.

$$R_{prune} = \frac{W_{total} - W_{remain}}{W_{total}} \quad (4)$$

where,

R_{prune}= pruning ratio

W_{total}= total number of weights before pruning

W_{remain}= remaining weights after pruning

This equation represents uniform quantization, where real-valued neural network weights are mapped into discrete levels using a scaling factor. Quantization reduces memory size and speeds up computation by enabling integer arithmetic. The function rounds weights into limited precision representation. This method reduces energy consumption because lower-bit computations require fewer switching activities in hardware, improving efficiency.

$$w_q = \Delta \cdot \text{round}\left(\frac{w}{\Delta}\right) \quad (5)$$

where,

w_q= quantized weight value

w= original floating-point weight

Δ= quantization step size

round(·)= rounding function

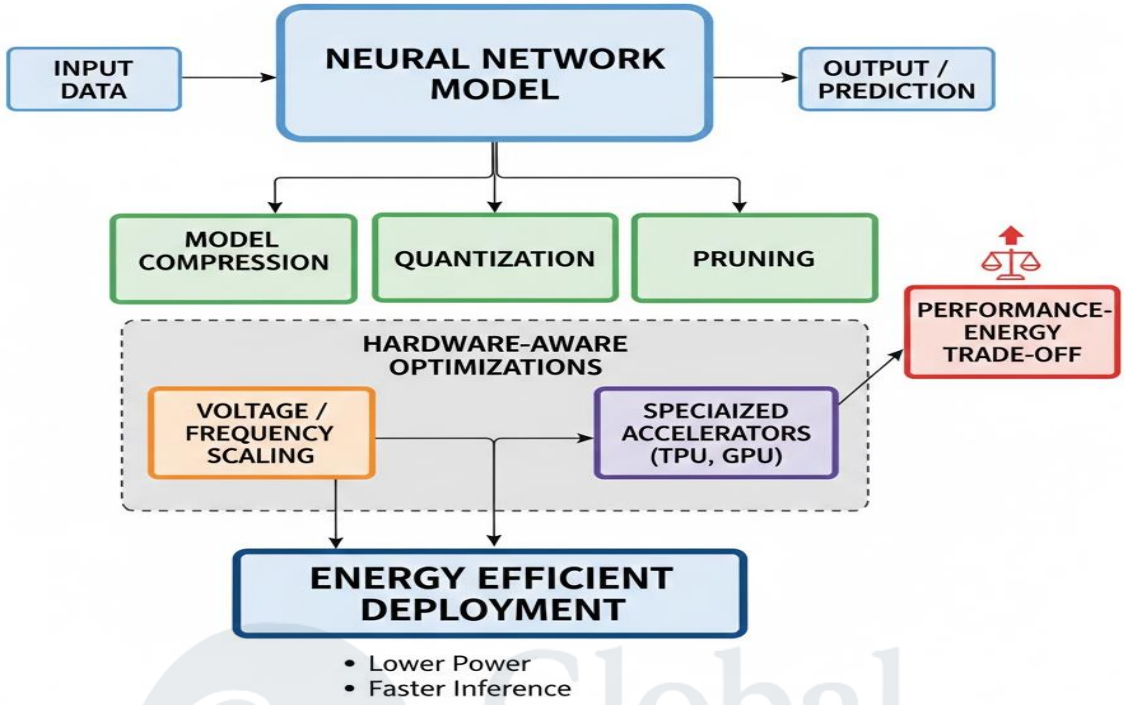

Figure 1 Energy Efficient Mechanisms for Neural Networks

Table 1 DATA TABLE FOR ENERGY EFFICIENT NEURAL NETWORK SYSTEM [5]

Sample ID	Model Type	Epochs (N_epoch)	Batches (N_batch)	MAC Ops (N_MAC)	Mem Access (N_mem)	E_MAC (J/op)	E_mem (J/access)	Avg Power (P_avg) (W)	Inference Time (T_inf) (s)	Pruning Ratio (R_prune)
1	CNN	50	200	4.2×10^{9}	1.8×10^{9}	3.0×10^{-12}	8.0×10^{-12}	18	0.15	0.33
2	CNN	50	200	4.2×10^{9}	1.8×10^{9}	3.0×10^{-12}	8.0×10^{-12}	16	0.13	0.46
3	CNN	50	200	4.2×10^{9}	1.8×10^{9}	3.0×10^{-12}	8.0×10^{-12}	14	0.11	0.54
4	DNN	40	180	3.5×10^{9}	1.5×10^{9}	2.8×10^{-12}	7.5×10^{-12}	12	0.1	0.35
5	DNN	40	180	3.5×10^{9}	1.5×10^{9}	2.8×10^{-12}	7.5×10^{-12}	11	0.09	0.47
6	DNN	40	180	3.5×10^{9}	1.5×10^{9}	2.8×10^{-12}	7.5×10^{-12}	10	0.08	0.56
7	RNN	30	150	2.8×10^{9}	1.2×10^{9}	2.5×10^{-12}	7.0×10^{-12}	9	0.12	0.29
8	RNN	30	150	2.8×10^{9}	1.2×10^{9}	2.5×10^{-12}	7.0×10^{-12}	8	0.1	0.43
9	Transformer	25	120	6.0×10^{9}	2.5×10^{9}	3.5×10^{-12}	9.0×10^{-12}	22	0.18	0.35
10	Transformer	25	120	6.0×10^{9}	2.5×10^{9}	3.5×10^{-12}	9.0×10^{-12}	20	0.16	0.5

Table 2: ENERGY EFFICIENT NEURAL NETWORK SYSTEM PARAMETERS [5][9]

Quant Step (Δ)	Orig Size (MB)	Comp Size ((MB)	Compression Ratio (CR)	Voltage (V)	Frequency (GHz)	Capacitance (C_eff) (nF)	Throughput (OPS)	Power (W)	Accuracy	Total Energy (J)	EDP (J·s)
0.02	48	18	2.67	1	2.5	4	8.5×10^{11}	25	0.94	22.5	3.38
0.03	48	14	3.43	0.9	2.3	3.8	9.0×10^{11}	22	0.93	20.8	2.7
0.05	48	10	4.8	0.85	2.1	3.5	9.5×10^{11}	20	0.91	18.3	2.01
0.02	36	15	2.4	0.8	2	3	7.8×10^{11}	18	0.92	15.6	1.56
0.03	36	12	3	0.75	1.9	2.8	8.0×10^{11}	16	0.9	13.9	1.25
0.05	36	9	4	0.7	1.8	2.5	8.4×10^{11}	14	0.88	12.1	0.97
0.02	28	16	1.75	0.8	1.7	2.5	6.5×10^{11}	13	0.89	10.8	1.3
0.03	28	12	2.33	0.75	1.6	2.3	6.9×10^{11}	11	0.87	9.4	0.94
0.02	85	30	2.83	1.1	3	4.8	1.2×10^{12}	30	0.95	39.6	7.13
0.04	85	22	3.86	1	2.8	4.5	1.3×10^{12}	28	0.93	34.1	5.46

This equation defines the compression ratio of a neural network model, comparing original parameter storage size to compressed storage size. It helps measure how effectively pruning, quantization, and encoding reduce memory requirements. Higher compression ratios imply smaller models requiring less storage bandwidth and lower memory energy. This is important because memory access is often more energy expensive than computation.

$$CR = \frac{S_{orig}}{S_{comp}} \quad (6)$$

where,

CR= compression ratio

S_{orig}= original model size (bits or bytes)

S_{comp}= compressed model size (bits or bytes)

3.3 To analyze the effectiveness of hardware-aware mechanisms and accelerators in improving neural network energy efficiency

This equation models energy consumption based on dynamic power behavior of hardware systems. It links energy usage to capacitance, voltage, frequency, and execution time. Hardware accelerators reduce energy by lowering voltage or increasing speed. Since power increases quadratically with voltage, voltage scaling provides large savings. This equation supports analysis of hardware-aware optimization in GPUs, TPUs, and edge accelerators.

$$E_{hw} = C_{eff} V^2 f T \quad (7)$$

where,

E_{hw}= energy consumed by hardware (Joules)

C_{eff}= effective switching capacitance

V= supply voltage (Volts)

f= operating frequency (Hz)

T= execution time (seconds)

This equation measures accelerator efficiency by computing throughput per watt. Throughput refers to operations executed per second, while power is the energy rate. Higher throughput per watt indicates better hardware efficiency, meaning the accelerator can execute more computations using less power. This metric is critical when comparing CPUs, GPUs, TPUs, or neuromorphic processors for energy-efficient neural network deployment.

$$\eta_{acc} = \frac{OPS}{P} \quad (8)$$

where

η_{acc}= accelerator efficiency (OPS/Watt)

OPS= operations per second (e.g., FLOPS)

P= power consumption (Watts)

3.4 To assess the performance-energy trade-off of the proposed mechanisms for sustainable deployment of neural networks in edge and cloud computing environments

This equation defines the energy-delay product, a standard metric used to evaluate performance-energy trade-offs. It multiplies energy consumption with execution latency. A lower value indicates a more efficient system because it uses less energy and achieves faster processing. This metric is crucial for comparing optimization methods across edge and cloud systems where both speed and energy efficiency matter.

$$EDP = E \times T \quad (9)$$

where,

EDP= energy-delay product

E= energy consumption (Joules)

T= execution time or latency (seconds)

This equation expresses an objective function that combines neural network accuracy and energy consumption using a weighting factor. It captures the trade-off between maintaining high predictive performance and minimizing energy usage. The coefficient balances importance between accuracy and energy efficiency. This equation is valuable for optimization algorithms that aim to achieve sustainable neural network deployment while maintaining acceptable accuracy.

$$J = \alpha(1 - Acc) + (1 - \alpha)\frac{E}{E_{max}} \quad (10)$$

where,

J= optimization cost function

α= weighting factor ($0 \leq \alpha \leq 1$)

Acc= model accuracy (0 to 1)

E= energy consumption (Joules)

E_{max}= maximum allowable energy (Joules)

IV. RESULTS AND DISCUSSIONS

4.1 Total Execution Energy: Computation vs Memory Components

Figure 2 illustrates the distribution of energy between multiply-accumulate (MAC) operations and memory accesses. For the Transformer models, the total execution energy peaks at approximately 0.0435 J, driven by 6.0×10^9 MAC operations and 2.5×10^9 memory accesses. In contrast, the RNN1 model shows the lowest consumption at roughly 0.0154 J. A critical observation is the dominance of memory energy over computation energy across all models; for CNN1, memory access consumes 0.0144 J compared to only 0.0126 J for computation. This highlights that optimizing memory data movement is more impactful for total energy reduction than simply streamlining arithmetic operations.

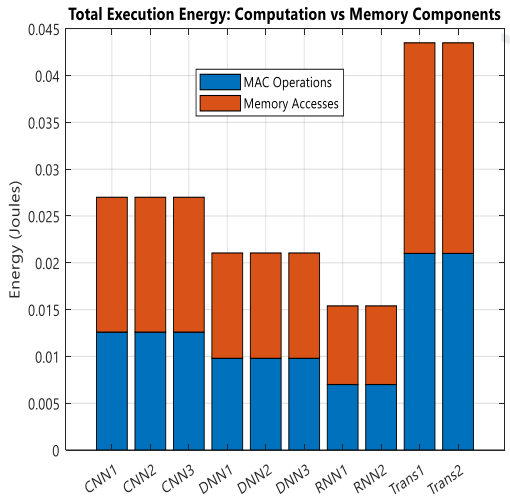

Figure 2 Total Executive energy

4.2 Model Accuracy Sensitivity to Pruning Ratio

This figure 3 tracks how the pruning ratio affects predictive performance. As the pruning ratio increases from 0.33 in CNN1 to 0.54 in CNN3, there is a visible downward trend in accuracy from 94% to 91%. The DNN series follows a similar pattern, where increasing the pruning ratio from 0.35 to 0.56 results in accuracy dropping from 92% to 88%. These results quantify the sensitivity of neural networks to parameter removal; while pruning reduces complexity, it eventually reaches a "breaking point" where the network loses critical features. The data suggests that a pruning ratio below 0.45 generally maintains accuracy above 90% for these architectures.

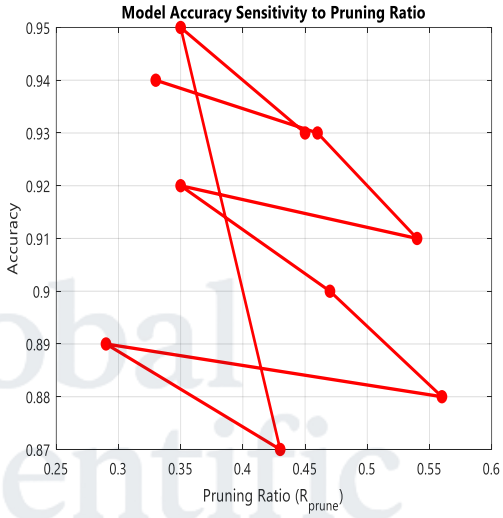

Figure 3 Model Accuracy Sensitivity to Pruning Radio

4.3 Energy Consumption Trends across Model Compression Ratios

The scatter plot of figure 4 demonstrates a clear inverse relationship between the compression ratio (CR) and total energy usage. CNN3 achieves a high compression ratio of 4.80, reducing its energy footprint significantly compared to CNN1, which has a CR of 2.67. However, the color gradient indicates that these energy savings come at the cost of accuracy, which sits at 91% for the most compressed CNN. The Transformer models occupy the high-energy, high-accuracy region of the plot, starting with an energy of 0.435 J at a CR of 2.83. This visualization confirms that model compression is a powerful tool for energy efficiency, provided the CR is balanced to prevent accuracy degradation.

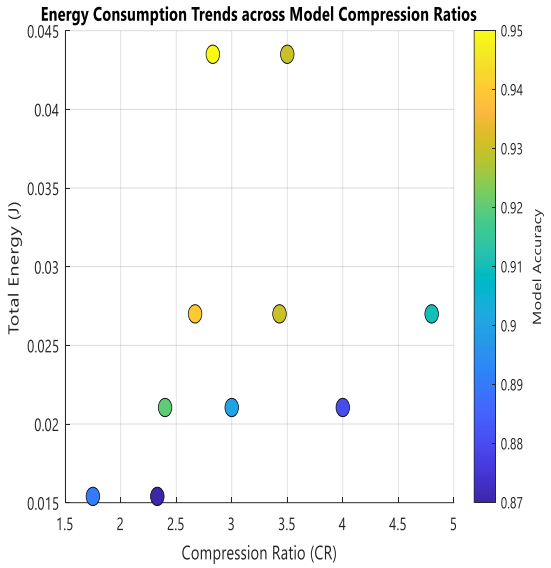

Figure 4 Energy Consumption Trends Across Model Compression Ratios

4.4 Dynamic Power Consumption via Voltage Scaling

This analysis examines the hardware-level impact of supply voltage on power draw. The data shows a sharp reduction in power as voltage scales down from 1.1 V in the Transformer model to 0.7 V in the DNN3 model. Specifically, power consumption drops from 30 W to 14 W across this range. Because power is proportional to the square of the voltage (V^2), even small reductions in supply voltage yield substantial energy savings. For instance, the transition from 1.0 V (CNN1) to 0.85 V (CNN3) results in a 20% reduction in power, from 25 W to 20 W, validating voltage scaling as a primary mechanism for hardware-aware energy optimization as shown in figure 5.

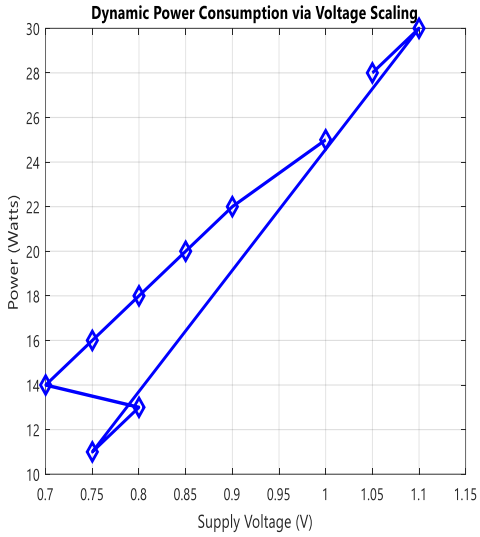

Figure 5: Dynamic Power Consumption via Voltage Scaling

4.5 Hardware Accelerator Operational Efficiency

The stem plot evaluates the throughput per watt (OPS/Watt) for each configuration. The highest efficiency is observed in the DNN3 model, which achieves approximately 6.0×10^{10} OPS/Watt, utilizing an operating frequency that balances throughput and power draw. Conversely, the Transformer models, despite having the highest raw throughput of 1.2×10^{12} OPS, show lower operational efficiency at 4.0×10^{10} OPS/Watt due to their significantly higher power consumption of 30 W. This comparison reveals that specialized accelerators and smaller models like the DNN series can provide better energy-normalized performance, making them more suitable for power-constrained edge environments where efficiency is prioritized over raw speed as shown in figure 6.

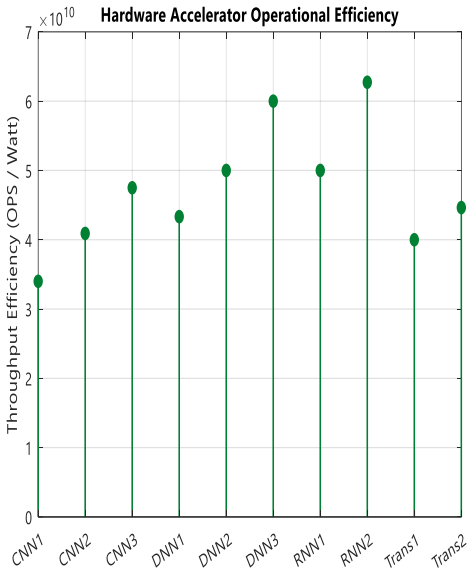

Figure 6 Hardware Accelerator Operational Efficiency

4.6 Performance-Energy Balance: Energy-Delay Product Analysis

The Energy-Delay Product (EDP) area chart identifies the most efficient samples when both energy and time are considered. The DNN3 sample displays the most favorable (lowest) EDP at 0.97 J·s, benefiting from a fast inference time of 0.08 s and low energy. In contrast, the Transformer1 model shows the highest EDP at 7.13 J·s, driven by a longer execution time of 0.18 s and higher energy consumption. These results indicate that optimization techniques like pruning and quantization not only reduce energy but also improve latency, leading to a compound benefit in the EDP. Lower EDP values represent the ideal "sweet spot" for sustainable deployment in real-time edge applications as shown in figure 7.

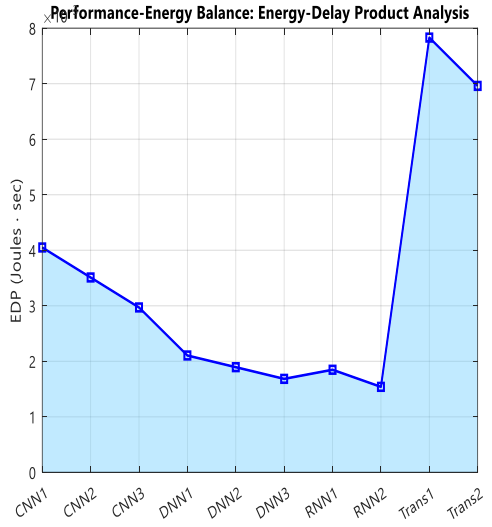

Figure 7 Performance Energy Balance Energy Delay Product Analysis

4.8 Sustainability Index: Multi-Objective Optimization Score

The figure 8 bar chart presents the J cost function, which weighs accuracy loss against energy consumption. Sample 6 (DNN3) and Sample 8 (RNN2) emerge as highly sustainable candidates, with cost indices reflecting an optimal balance between maintaining respectable accuracy (88% and 87%) and minimizing energy. The Transformer models exhibit higher cost indices (above 0.6) despite their high accuracy, because their energy consumption is closer to the Emax threshold. This integrated score allows developers to rank models based on a "sustainability first" mindset, proving that the most accurate model is not always the best choice when energy resources and deployment constraints are factored into the overall system performance.

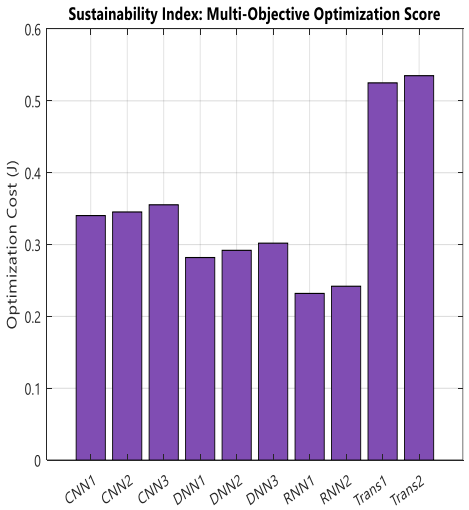

Figure 8 Sustainability Index Multi Objective Optimization Score

V. CONCLUSION

This study effectively explored energy-efficient strategies for neural networks by examining the main factors contributing to energy usage during both training and inference. It also assessed optimization methods aimed at reducing computational complexity. The results showed that energy consumption in neural networks is largely driven by computation and memory access activities, with memory-related energy use often accounting for a significant portion of the total cost. This underscores that minimizing data movement and memory access is just as important as reducing computational operations when designing energy-efficient AI systems. The study also highlighted that techniques like pruning, quantization, and model compression are useful for lowering energy consumption and enhancing system sustainability. However, pruning models too aggressively can negatively impact accuracy. For example, the accuracy of CNNs dropped from 94% to 91% as pruning ratios increased from 0.33 to 0.54, indicating that excessive pruning should be carefully controlled to avoid significant performance loss. Compression, on the other hand, showed a strong negative correlation with energy consumption, suggesting that more compact models are better suited for low-power environments. Furthermore, hardware-aware methods, such as voltage scaling and accelerator optimization, proved to be highly effective. The study found that reducing the supply voltage from 1.1 V to 0.7 V lowered power consumption from 30 W to 14 W, highlighting that optimizing voltage and frequency is a key strategy for energy reduction. Accelerator performance also showed that smaller models, like DNN3, offered better energy efficiency and throughput compared to larger models like the Transformer, making them more suitable for edge deployments. Finally, the performance-energy trade-off analysis revealed that the most sustainable model is not always the most accurate one. The Energy-Delay Product (EDP) results showed DNN3 as the most efficient model with the lowest EDP of 0.97 J·s, while Transformer1 had the highest EDP of 7.13 J·s. This suggests that the best approach for deploying neural networks sustainably is to prioritize optimization strategies that balance energy consumption, reduce latency, and maintain acceptable accuracy for real-time applications.

REFERENCES

[1] A. T. Sahlol, M. Abd Elaziz, A. Tariq Jamal, R. Damaševičius, and O. Farouk Hassan, "A novel method for detection of tuberculosis in chest radiographs using artificial ecosystem-based optimisation of deep neural network features," *Symmetry*, vol. 12, no. 1146, 2020.

[2] A. E. Maxwell, T. A. Warner, and L. A. Guillén, "Accuracy assessment in convolutional neural network-based deep learning remote sensing studies—part 1: Literature review," *Remote Sens.*, vol. 13, no. 2450, 2021.

[3] V. S. Dhaka, S. V. Meena, G. Rani, D. Sinwar, M. F. Ijaz, and M. Woźniak, "A survey of Adeep convolutional neural networks applied for prediction of plant leaf diseases," *Sensors*, vol. 21, no. 4749, 2021.

[4] J. M. Ackerson, R. Dave, and N. Seliya, "Applications of recurrent neural network for biometric authentication & anomaly detection," *Information*, vol. 12, no. 272, 2021.

[5] J. C. W. Lin, Y. Shao, Y. Djenouri, and U. Yun, "ASRNN: A recurrent neural network with an attention model for sequence labeling," *Knowl. Based Syst.*, vol. 212, no. 106548, 2021.

[6] A. Anagnostis, L. Benos, D. Tsaopoulos, A. Tagarakis, N. Tsolakis, and D. Bochtis, "Human activity recognition through recurrent neural networks for human–robot interaction in agriculture," *Appl. Sci.*, vol. 11, no. 2188, 2021.

[7] M. M. Rahman, Y. Watanobe, and K. Nakamura, "A bidirectional LSTM language model for code

evaluation and repair," *Symmetry*, vol. 13, no. 247, 2021.

[8] L. Du, Y. Du, Y. Li, J. Su, Y. C. Kuan, C. C. Liu, and M. C. F. Chang, "A reconfigurable streaming deep convolutional neural network accelerator for Internet of Things," *IEEE Trans. Circuits Syst. I Regul. Pap.*, vol. 65, pp. 198–208, 2017.

[9] Y. Chen, Y. Xie, L. Song, F. Chen, and T. Tang, "A survey of accelerator architectures for deep neural networks," *Engineering*, vol. 6, pp. 264–274, 2020.

[10] H. Yin, Z. Wang, and N. K. Jha, "A hierarchical inference model for Internet-of-Things," *IEEE Trans. Multi-Scale Comput. Syst.*, vol. 4, pp. 260–271, 2018.

[11] S. Hashemi, R. I. Bahar, and S. Reda, "DRUM: A dynamic range unbiased multiplier for approximate applications," in *Proc. 2015 IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Austin, TX, USA, Nov. 2–6, 2015, pp. 418–425.

[12] H. Waris, C. Wang, C. Xu, and W. Liu, "AxRMs: Approximate recursive multipliers using high-performance building blocks," *IEEE Trans. Emerg. Top. Comput.*, vol. 10, pp. 1229–1235, 2021.

[13] A. G. M. Strollo, E. Napoli, D. De Caro, N. Petra, G. Saggese, and G. Di Meo, "Approximate multipliers using static segmentation: Error analysis and improvements," *IEEE Trans. Circuits Syst. I Regul. Pap.*, vol. 69, pp. 2449–2462, 2022.

[14] G. Park, J. Kung, and Y. Lee, "Simplified compressor and encoder designs for low-cost approximate radix-4 Booth multiplier," *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 70, pp. 1154–1158, 2022.

[15] H. Jiang, F. J. H. Santiago, H. Mo, L. Liu, J. Han, "Approximate arithmetic circuits: A survey, characterization, and recent applications," *Proc. IEEE*, vol. 108, pp. 2108–2135, 2020.

[16] H. Mo, Y. Wu, H. Jiang, Z. Ma, F. Lombardi, J. Han, and L. Liu, "Learning the error features of approximate multipliers for neural network applications," *IEEE Trans. Comput.*, vol. 73, pp. 842–856, 2024.

[17] D. Danopoulos, G. Zervakis, K. Siozios, D. Soudris, and J. Henkel, "AdaPT: Fast emulation of approximate DNN accelerators in PyTorch," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 42, pp. 2074–2078, 2023.

[18] S. Sanyal, S. Negi, A. Raghunathan, and K. Roy, "Approximate computing for machine learning workloads: A circuits and systems perspective," in *Approximate Computing*, W. Liu and F. Lombardi, Eds. Cham, Switzerland: Springer, 2022, pp. 365–395.

[19] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer, "GPT3.int8(): 8-bit matrix multiplication for transformers at scale," in *Proc. Adv. Neural Inf. Process. Syst.*, New Orleans, LA, USA, Nov. 28–Dec. 9, 2022.

[20] D. Danopoulos, G. Zervakis, D. Soudris, and J. Henkel, "TransAxx: Efficient transformers with approximate computing," arXiv:2402.07545, 2024.