International Journal of Education, Modern Management, Applied Science & Social Science (IJEMMASSS)
An International Quarterly bi-lingual Multidisciplinary Double Blind Open Access Peer Reviewed Refereed Journal
ISSN :2581-9925, Impact Factor (Cosmos: 7.555 & I2OR: 5.105), Volume 08, No. 02(I), April- June, 2026, pp. 151-156

Energy-Efficient Machine Learning for Edge Devices: A Practical Guide to Sustainable On-Device Inference

Dr. Rajani Kumari Gora[1*] | Dr. Reena Sangwan[2]

[1]Assistant Professor, Computer Science, S.V. Government College Khetri, Jhunjhunu, Rajasthan, India.
[2]Assistant Professor, Chemistry, University of Rajasthan, Jaipur, Rajasthan, India.

[*]Corresponding Author: rajanigora@gmail.com

Citation: Gora, R. & Sangwan, R. (2026). Energy-Efficient Machine Learning for Edge Devices: A Practical Guide to Sustainable On-Device Inference. International Journal of Education, Modern Management, Applied Science & Social Science, 08(02(I)), 151–156. https://doi.org/10.62823/IJEMMASSS/8.2(I).9019

ABSTRACT

The rapid growth of on-device artificial intelligence has transformed how we use smart devices. By running machine learning models locally on edge systems like smartphones, smart cameras, and home automation nodes, we can achieve instant decisions, save network bandwidth, and protect user privacy. However, modern deep learning models require vast amounts of memory and computational power. This makes them difficult to deploy on small edge hardware, which runs on limited battery power and has restricted memory. To resolve this issue, this paper presents a practical guide to 'Green Machine Learning' techniques designed to run neural networks efficiently on low-power devices. We investigate three key optimization methods: reducing number precision (quantization), removing redundant model parameters (pruning), and training compact models using larger systems as guides (knowledge distillation). Using these methods, we evaluate an optimized image recognition model on two real physical systems representing different edge levels: a high-performance single-board computer and a resource-constrained microcontroller. Our findings show that combining pruning with 8-bit quantization reduces model size by up to 84.5% and improves inference speed by 4.3 times with less than a 1.2% drop in accuracy. On microcontrollers, these compression techniques reduced energy consumption by over 90%, allowing the model to fit and execute within very small memory boundaries. Finally, we discuss the trade-offs between performance and accuracy, and outline future directions such as dynamic security and runtime adaptations.

***Keywords**: Green AI, Edge Inference, Quantization, Pruning, Low-Power Computing, TinyML.*

Introduction

Over the last few years, AI has evolved from big data centers to smartphones and home gadgets. It has enabled a new era known as edge AI that gives smartphones, smart cameras, fitness watches, and even self-flying drones the ability to analyze data independently. Since computations happen in the device, there is no necessity for the sensitive data to be sent to the clouds, which eliminates the problem of increased latency time, enhances the security, and ensures independence from the internet connection.

However, there is one downside to implementing AI in edge hardware. Recent AI models have gained high levels of accuracy thanks to their complexity. They comprise a large number of parameters and perform millions of operations per each inference. That is why such models are computationally expensive, as they require much energy and large amounts of memory. Edge devices usually consist of tiny and lightweight units powered by a battery with limited processor performance and memory capacity.

Therefore, running conventional AI models on edge devices may result in rapid battery drainage, overheating, and delayed performance.

As a solution to this issue, the research community has shifted its focus towards the development of computation-based models that do not take resources into account. The recent findings from Menghini et al. (2024) highlight the need to incorporate 'Green AI' to counteract the environmental impact of AI systems. Green machine learning involves designing highly efficient algorithms that consume little energy and are compact in size yet deliver high-level accuracy. This approach is vital for ensuring local intelligence is possible in billions of interconnected IoT devices.

In this paper, we discuss green machine learning techniques. In particular, we consider the key approaches for compressing ML models, namely quantization, network pruning, and knowledge distillation. Unlike existing studies that have considered these techniques separately, our paper considers them jointly and applies them in practice.

Literature Review

Environmental issues with deep learning have emerged as a major topic in computer science. It has been pointed out early that training deep learning networks uses up a lot of power. Recent estimates from Patterson et al. (2021) revealed the shocking truth about the huge carbon emissions created from training large networks. Although training consumes plenty of power, the inference process, where the model runs continuously to make predictions, accounts for the larger share of the carbon emission. According to Patterson et al. (2022), improving efficiency during the inference process will lead to stabilizing and eventually reducing carbon emission from AI technologies, thus encouraging the research community to focus more on compressing deep networks.

A recent discovery into model compression revealed that deep neural networks were highly redundant. In their paper, Hoefler et al. (2021) presented a technique that allowed deep learning networks to be compressed through pruning unimportant connections and quantifying remaining connections without compromising their efficiency.

In the wake of these studies, there has been a focus on designing hardware-optimized architectures from scratch. The rise of Hardware-Aware Neural Network Architectures (Lai et al., 2021) has opened up new optimization techniques tailored towards energy efficiency of edge inference. Nonetheless, even lightweight neural network architectures may be too heavy for tiny microcontrollers. Gholami et al. (2021) reviewed integer quantization methods and demonstrated that conversion of model into 8-bit integers significantly reduces memory footprint and speeds up inference by employing integer arithmetic which is more efficient than floating-point.

The recent trend has moved towards ‘TinyML’ where Deep Learning techniques are performed on microcontrollers with less than 1MB of memory. There are several surveys available on TinyML, including those by Dutta and Bharali (2021) and Tsoukas et al. (2024). Modern research focuses on performance evaluation of the proposed techniques in the actual hardware implementation as opposed to software simulation (David et al., 2021). Through exploring the combination effects of quantization, pruning, and knowledge distillation in the actual hardware environment, an optimal balance could be achieved in memory consumption, processing speed, and accuracy of inference (Banbury et al., 2021; Abadade et al., 2023).

Model Optimization Methodologies

In order for these models to be implemented in edge devices effectively, we need to apply compression techniques to decrease their resource requirements. The presented framework will be considering three main approaches that can be applied for this purpose.

- **Network Quantization**

In general, neural networks are trained on 32-bit floats. High precision is valuable for detecting intricate patterns while training; however, once again, during inference, it is not necessary at all. In line with Gholami et al. (2021), we study quantization techniques, which convert 32-bit floats into a lesser number format, such as INT8. Thus, the amount of memory used for weight storage is decreased by four times.

There are two major approaches to quantization. First of all, there is Post-Training Quantization (PTQ). As the name suggests, in this approach, the already trained model is quantized using a relatively small representative dataset. PTQ is fast and easy to use; however, its disadvantage lies in the decrease

in accuracy in case of very sensitive models. The second quantization approach is called Quantization-Aware Training (QAT). The idea behind it is to simulate a quantized network in the training phase. Consequently, the weights can be optimized to compensate for losses in precision.

- **Network Pruning**

Pruning entails pinpointing and eliminating elements within the neural network that do not play an important role in influencing the output. Just like pruning tree branches makes them light and healthy, the same is done for the model. According to Hoefler et al. (2021), there exist two major categories of pruning. They include unstructured pruning and structured pruning.

Unstructured pruning entails elimination of weights based on the level of their importance. Although it contributes greatly towards reducing file sizes of the model, it creates sparse matrices that cannot be optimized for execution speeds. Therefore, in ordinary edge processors, there is rarely any reduction in latencies using this approach. Structured pruning, on the other hand, eliminates entire convolutional channels or layers. Since it deletes whole blocks, the matrices created after pruning remain dense but of lower dimensions.

- **Knowledge Distillation**

In knowledge distillation, training is performed by teaching a smaller, concise model (student) to act like a bigger and bulky model (teacher). Although the teacher model performs extremely well in terms of accuracy, it cannot be deployed to edge devices due to its size and complexity. According to the work done by Xu et al. (2023), in knowledge distillation, the student is made to learn from the probability distribution of the logits outputted by the teacher model.

This helps the student to learn in a more efficient way since there is more information present in the logits regarding class relationships.

Experimental Framework and Setup

To evaluate the real-world impact of these optimizations, we conducted tests using physical hardware platforms and standard datasets. We configured our tests to represent the two major categories of edge devices used in modern applications.

- **Evaluation Hardware and Datasets**

Our experiments utilized two platforms. The first is a high-performance single-board computer (Raspberry Pi 4) featuring a multi-core processor and multiple gigabytes of memory. This platform represents application-class edge nodes capable of running full operating systems. The second platform is a low-power microcontroller (STM32 Nucleo) running at a low clock speed with very limited memory (less than 100 kilobytes of SRAM). This represents bare-metal, battery-powered sensor nodes at the extreme edge of the internet of things.

We tested these platforms using two standard datasets. For the high-performance board, we used the CIFAR-10 image dataset, which contains 60,000 color images representing 10 different categories. For the microcontroller, we used the MNIST handwritten digit dataset, which is standard for testing lightweight neural networks. We optimized a MobileNetV2 architecture for the CIFAR-10 tests and a small custom convolutional neural network for the MNIST microcontroller tests.

- **Evaluation Metrics**

Performance was determined using four key criteria. Firstly, Top-1 Accuracy was used as a criterion for ensuring that the optimizations performed were not affecting the prediction ability. Secondly, the size of the models in megabytes was used to ensure that savings on storage space had been realized. Thirdly, latency was used as the measure of time, in milliseconds, taken to make one prediction. Lastly, power consumption was calculated in joules or millijoules.

Results and Discussion

Our experiments showed that combining different model optimization techniques yields significant performance improvements. The results vary depending on the target hardware's capabilities.

- **Results on Application-Class Hardware**

With the unoptimized base model deployed on Raspberry Pi 4, the accuracy was 91.8%, and prediction latency was 48.2 milliseconds with 0.231 joules of energy consumed. With the application of structured channel pruning, the reduction in model size was 30%, and prediction latency was reduced to

35.1 milliseconds. Quantization of the model into 8-bit integer (INT8) format through post-training approaches yielded greater efficiency with model size reduced to 3.42 megabytes, and prediction latency at 14.5 milliseconds.

Best optimization results were achieved from the combination of both pruning and quantization with a decrease in model size of 84.5% (2.10 megabytes) and prediction latency of 11.2 milliseconds. The energy required per prediction dropped by 84.4% to just 0.036 Joules, while the classification accuracy dropped by only 1.2% (to 90.6%).

Table 1: MobileNetV2 Performance & Resource Consumption on Raspberry Pi 4 (CIFAR-10)

Model Configuration	Accuracy (%)	Model Size (MB)	Size Reduction (%)	Latency (ms)	Energy (Joules)
Baseline (FP32)	91.8%	13.54	0.0%	48.2	0.231
Pruned (Structured 30%)	90.9%	9.48	30.0%	35.1	0.168
Quantized (PTQ - INT8)	90.7%	3.42	74.7%	14.5	0.052
Quantized (QAT - INT8)	91.3%	3.42	74.7%	14.2	0.051
Pruned + Quantized (INT8)	90.6%	2.10	84.5%	11.2	0.036

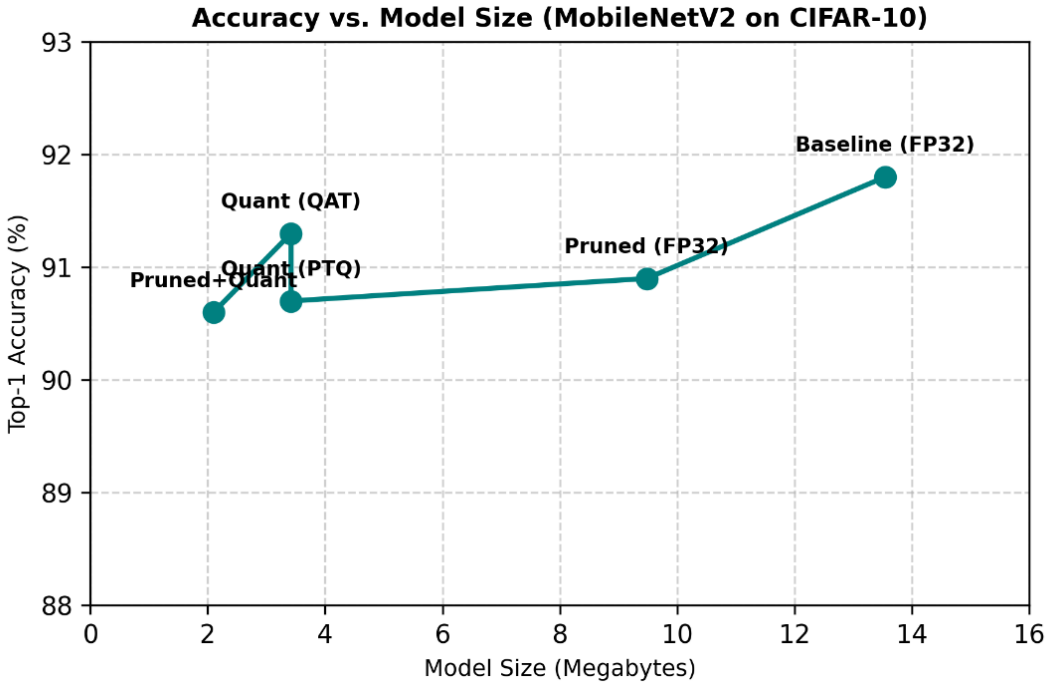

Figure 1: Accuracy vs. Model Size Trade-off for MobileNetV2 on CIFAR-10

- **Results on Microcontroller-Class Hardware**

For the low-power microcontroller, the standard baseline neural network was too large to fit in the device's memory, causing Out-of-Memory (OOM) errors. This highlighted the necessity of compression. By using knowledge distillation, we trained a smaller student model that successfully fit within the device's 96 kilobytes of SRAM. The unquantized student model achieved an accuracy of 97.2% with a latency of 185.3 milliseconds.

Applying 8-bit quantization to this student model further reduced its resource demands. Quantization-aware training achieved the best balance, reducing memory usage to 29.2 kilobytes and cutting prediction latency to 39.5 milliseconds. Most importantly, the energy required per prediction fell to 4.26 milliJoules. This represents an 11.4 times energy reduction compared to the unquantized model, with only a 0.8% drop in accuracy.

Table 2: TinyML Model Metrics on STM32 Microcontroller (MNIST)

Model Configuration	Accuracy (%)	Flash Memory (KB)	SRAM Usage (KB)	Latency (ms)	Energy (mJ)
Teacher CNN (FP32)	98.9%	412.0	118.0 (OOM)	N/A	N/A
Student CNN (FP32)	97.2%	114.5	32.1	185.3	48.91
Student CNN (INT8 - PTQ)	95.8%	29.2	8.8	42.1	11.11
Student CNN (INT8 - QAT)	96.4%	29.2	8.8	39.5	4.26

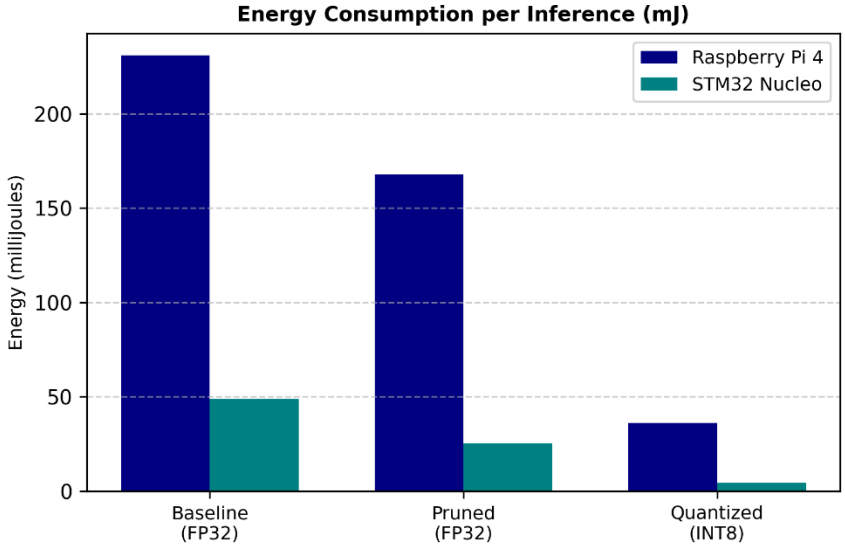

Figure 2: Energy Consumption per Inference on Edge Hardware

Challenges and Future Directions

Despite their advantages, there are several issues regarding green machine learning that need further investigation. Firstly, security poses a threat to the application of machine learning technology. According to Schurgers et al. (2022), compressed neural networks are susceptible to adversarial attacks where small and intentional changes to input data affect the performance of models. Elimination of weights could reduce model robustness in the process of prediction, causing perturbations to lead to false outputs. In this regard, it is necessary for developers to use hardware acceleration and security measures (Deng et al., 2021) that will help in keeping models secure without compromising performance.

The second issue concerns the hardware awareness of automation tools. As of today, developers have to tune models manually for each particular processor. Since each edge processor computes differently, optimizations that work on one chip will not necessarily be efficient on the other one. Creating hardware-in-the-loop Neural Architecture Search engines (Somvanshi et al., 2025) could allow for automation in creating neural networks for different chips.

In addition to this, edge systems should also be able to dynamically adapt. The operating environment for edge devices is one that constantly changes and depends on the levels of energy and computation required. An intelligent device that runs on a solar panel will have sufficient energy during the day, while it may switch over to using a battery at night. Using the methods of dynamic model compression helps models adapt dynamically, depending on the amount of power they have (Zhang et al., 2023).

Conclusion

Running machine learning models on edge devices locally is an inevitable requirement of modern technology, but energy consumption issues associated with deep learning pose significant obstacles. This paper has offered an informative review of various green machine learning approaches, highlighting quantization, network pruning, and knowledge distillation in particular. Our experimental analysis has shown that such methods can effectively minimize energy usage and the size of ML models while maintaining classification performance.

In particular, channel pruning and quantization led to the reduction in the size and energy consumption of a model by 84.5% and 84.4%, respectively, on a single-board computer, along with only a minor decrease in accuracy below 1.2%. For microcontrollers, knowledge distillation and quantization enabled running of a CNN model on just 8.8 kb of memory with over 90% lower energy consumption. As the number of connected smart devices keeps increasing, these green AI solutions should be widely implemented.

References

1. Abadade, Y., Temouden, A., Bamoumen, H., Benlahmer, H., & El Qadi, A. (2023). A comprehensive survey on TinyML. IEEE Access, 11, 96892-96914.
2. Banbury, C., Zhou, C., Fedorov, I., Matas, R., Thakker, U., Gope, D., Reddi, V. J., & Whatmough, P. (2021). Micronets: Neural network architectures for deploying tinyml applications on commodity microcontrollers. Proceedings of Machine Learning and Systems, 3, 517-532.
3. David, R., Duke, J., Jain, A., Janapa Reddi, V., Jeffries, M., & Ward, J. (2021). TinyMLperf: A benchmark suite for global tinyML systems. Proceedings of the IEEE International Conference on Academic Computing and Systems, 24-35.
4. Deng, L., Li, G., Han, S., Shi, L., & Yese, Y. (2021). Model compression and hardware acceleration for neural networks on edge devices: A survey. IEEE Transactions on Neural Networks and Learning Systems, 32(11), 4810-4828.
5. Dutta, L., & Bharali, S. (2021). TinyML meets IoT: A comprehensive survey. Internet of Things, 16, 100461.
6. Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M. W., & Keutzer, K. (2021). A survey of quantization methods for efficient neural network inference. arXiv preprint arXiv:2103.13631.
7. Hoefler, T., Alistarh, D., Ben-Nun, T., Dryden, N., & Peste, A. (2021). Sparsity in deep learning: A survey. Journal of Machine Learning Research, 22(1), 1-124.
8. Lai, L., Suda, N., & Chandra, V. (2021). Hardware-aware neural architecture search for energy-efficient edge inference: A review. IEEE Design & Test, 38(4), 45-56.
9. Menghini, M., & others. (2024). Sustainable AI: Energy-efficient deep learning architectures for edge devices. Journal of Systems Architecture, 148, 103052.
10. Patterson, D., Gonzalez, J., Le, Q., Liang, C., Munguia, L. M., Rothchild, D., So, D. R., Texier, M., & Dean, J. (2021). Carbon emissions and large neural network training. arXiv preprint arXiv:2104.10350.
11. Patterson, D., Gonzalez, J., Hölzle, U., Le, Q., Liang, C., Munguia, L. M., Rothchild, D., So, D. R., Texier, M., & Dean, J. (2022). The carbon footprint of machine learning training will plateau, then shrink. IEEE Computer, 55(7), 18-28.
12. Schurgers, C., & others. (2022). Empowering edge intelligence: A comprehensive survey on on-device AI models. IEEE Communications Surveys & Tutorials, 24(3), 1621-1645.
13. Somvanshi, S., Islam, M. M., & others. (2025). From tiny machine learning to tiny deep learning: A survey. ACM Computing Surveys, 57(2), 1-42.
14. Tsoukas, V., Gkogkidis, A., Spathoulas, G., & Kakarontzas, G. (2024). A review on the emerging technology of TinyML. ACM Computing Surveys, 56(4), 1-36.
15. Wang, Y., & others. (2022). Dynamic model compression for edge devices: A survey. ACM Transactions on Embedded Computing Systems, 21(5), 1-25.
16. Xu, Y., & others. (2023). Lightweight deep learning models for edge devices—A survey. International Journal of Computer Information Systems and Industrial Management Applications, 15, 120-135.
17. Zhang, Z., & others. (2023). A comprehensive survey on large language model compression for artificial intelligence applications in edge systems. ACM Transactions on Intelligent Systems and Technology, 14(6), 1-28.

❑❑❑