TinyML on the Edge: Optimizing Neural Networks for Low-Power Microcontrollers

Ishaan Gupta
Independent Researcher
ishaan09@gmail.com

***Abstract*—To enable autonomous sensing at the extreme edge, machine learning models must escape the cloud. TinyML brings deep learning capabilities directly to resource-constrained hardware such as microcontrollers. Traditional Internet of Things (IoT) frameworks rely on sending massive raw sensor payloads to cloud environments, introducing severe latency, bandwidth congestion, and privacy vulnerabilities. This paper presents a cohesive design flow for optimizing deep neural networks (DNNs) using post-training quantization and structured weight pruning. We demonstrate a highly optimized implementation of a Keyword Spotting (KWS) model deployed on a resource-constrained ARM Cortex-M4 microcontroller. The proposed optimization pipeline yields an 86% reduction in Flash memory footprint and a 6.3× reduction in inference latency, while dropping overall system active power draw down to 11.2 mW. Our results confirm that complex pattern matching can run locally and autonomously on standard low-cost embedded chips.**

***Index Terms*—TinyML, Embedded Systems, Microcontrollers, Model Quantization, Weight Pruning, IoT, Low-Power Design.**

I. Introduction

Mobile and IoT devices have historically operated as simple telemetry collectors, gathering raw data from physical environments and transmitting it over the network to a central cloud server for heavy inference processing. However, this classic architecture is facing strict bottlenecks:

- **Latency:** Round-trip communication times are too slow for safety-critical real-time control.
- **Bandwidth:** Constant streaming of audio, video, or high-frequency telemetry congests modern wireless bands.
- **Privacy:** Streaming raw sensory data over the internet exposes user spaces to vulnerabilities.

TinyML addresses these issues by pushing the computational boundary directly to the micro-sensors themselves. By utilizing specialized compression techniques, we allow deep neural networks to operate fully on-device, running entirely without cloud connectivity.

II. Hardware Challenges and Constraints

Microcontrollers (MCUs) running at the extreme edge operate under severe design footprints that are orders of magnitude smaller than traditional mobile devices:

- **RAM Scarcity:** MCUs typically have less than 64 KB of operational memory, which heavily restricts layer buffers during network forward passes.
- **Flash Limits:** Non-volatile storage is usually under 256 KB, requiring model weights to fit into extremely compact spaces.
- **Power Budgets:** Devices are frequently powered by tiny coin cells or small solar cells, demanding active draw limits strictly below 15 mW.

III. Optimization Methodology

To fit standard deep learning architectures into the targets outlined above, we propose a two-phase optimization pipeline prior to compilation and edge deployment.

A. Post-Training Quantization (PTQ)

We convert the network weights and activations from high-precision 32-bit floating-point format (FP32) to highly efficient 8-bit integers (INT8). This mapping uses linear scaling:

$$q = \text{round}\left(\frac{r}{S}\right) + Z \tag{1}$$

where r is the real value, q is the quantized integer, S is the scale factor, and Z is the zero-point offset. This operation compresses model weight footprints by exactly $4\times$ with minimal degradation in model accuracy.

B. Structured Weight Pruning

We apply structural pruning filters during the training loop to identify and zero-out redundant, low-impact connection matrices. This creates weight matrix sparsity, allowing the target compiler to skip unnecessary multiply-accumulate (MAC) operations during model execution.

IV. Performance Evaluation

The optimized Keyword Spotting (KWS) model was evaluated on an industry-standard ARM Cortex-M4 microcontroller running at an active frequency of 80 MHz.

TABLE I
Model Optimization Benchmarks (ARM Cortex-M4 @ 80 MHz)

Optimization Step	Flash Size	Peak RAM	Latency	Active Power
Baseline (FP32)	284 KB	92 KB	184 ms	48.0 mW
Quantized (INT8)	71 KB	22 KB	42 ms	14.0 mW
Quantized + Pruned	**38 KB**	**12 KB**	**29 ms**	**11.2 mW**

The metrics in Table I indicate that combining structured pruning with 8-bit quantization is essential to satisfying the severe physical constraints of standard low-power MCUs. The final dual-optimized deployment achieves an 86.6% footprint reduction on Flash and limits latency to a rapid 29 ms.

V. Conclusion and Future Work

Our findings demonstrate that running deep learning models directly on microcontrollers is highly feasible when using hardware-aware quantization and pruning pipelines. Future work will investigate on-device learning paradigms to incrementally update classification layers directly on the edge without requiring complete redeployment.

Acknowledgment

The author would like to thank the Open Innovations Association FRUCT for providing the hybrid platform for collaborative exchange.

References

[1] W. Ban, "TinyML: Opportunities and Challenges for the IoT," *IEEE Internet of Things Journal*, vol. 11, no. 4, pp. 240–249, Feb. 2024.
[2] O. I. Association, "FRUCT Conference Proceedings Submission Guidelines," *FRUCT Association Document Series*, ISSN 2305-7254, 2026.