MicroLLM: Hardware-Aware Compression and Runtime Scheduling of Language Models on Microcontrollers

Dr.A.Rehash Rushmi Pavitra[1], Prof. Dr. Vugar Abdullayev[2]

[1]Postdoctoral Researcher, Lincoln University College, 47301, Petaling Jaya, Selangor Darul Ehsan, Malaysia.

[2]Azerbaijan State Oil and Industry University

Email ID: rehashrp@srmist.edu.in, abdulvugar@mail.ru

Abstract: Deploying Large Language Models (LLMs) and Small Language Models (SLMs) on bare-metal microcontroller units (MCUs) enables private, low-latency, and energy-efficient intelligence for edge Internet of Things (IoT) ecosystems. However, standard MCU architectures operate under severe hardware boundaries, typically featuring less than 2 MB of Flash memory, 256 - 512 KB of SRAM, and milliwatt power envelopes. Pushing language model footprints into the sub-megabyte realm requires aggressive co-compression techniques. This survey paper examines the emerging paradigm of MicroLLM deployment, focusing on the synergy between structured sparsity (eg $N:M$ block pruning) and ultra-low-bit quantization (specifically sub-4-bit and 2-bit representations like ternary/quaternary weight formats). We evaluate hardware-software co-design pipelines, compilation toolchains (e.g., CMSIS-NN, TinyEngine), memory-scheduling techniques for runtime activations and accuracy-recovery mechanisms. Finally, we highlight current performance benchmarks, trade-offs in perplexity versus memory footprint, and critical open directions for microcontroller-class language modeling.

Keywords: TinyML, MicroLLM, Microcontrollers (MCUs), 2-Bit Quantization, Structured Sparsity, Model Compression, Edge AI.

Introduction

The transition of natural language processing from centralized data centers to ubiquitous edge devices is driven by stringent requirements for data privacy, low-latency response, and disconnected operability. While cloud-scale Large Language Models (LLMs) depend on gigabytes of High-Bandwidth Memory (HBM) and multi-teraflop GPUs, extreme edge endpoints are governed by microcontroller units (MCUs) based on ARM Cortex-M, RISC-V, or Xtensa architectures.

Microcontroller platforms present three strict operational boundaries:

1. Non-Volatile Storage (Flash/ROM): Typically between 512 KB and 2 MB, which bounds the static footprint of model weights.
2. Volatile Memory (SRAM): Ranging from 64 KB to 512 KB, creating a critical bottleneck for peak dynamic activations and autoregressive Key-Value (KV) caching.

3. Compute and Energy Budgets: Processing units operate at clock frequencies between 40 MHz and 400 MHz, relying on integer-only Arithmetic Logic Units (ALUs) or basic SIMD instruction sets (e.g., ARM Helium/DSP) under milliwatt power budgets [2].

Conventional Post-Training Quantization (PTQ) schemes compressing models to 8-bit (INT8) or 4-bit (INT4) representations (such as SmoothQuant [5] or AWQ [3]) reduce model size by $2\times$ to $4\times$. However, for compact models spanning 25M to 125M parameters, an INT4 checkpoint still occupies 12.5–62.5 MB orders of magnitude beyond standard on-chip MCU Flash storage.

To bridge this resource gap, the MicroLLM paradigm explores the joint optimization of two orthogonal compression vectors:

- Ultra-Low-Bit Quantization (2-Bit / 1.58-Bit): Reducing parameter precision to ternary $\{-1,0,+1\}$ or quaternary values ($\{0,1,2,3\}$), achieving theoretical compression ratios up to $8\times$ over FP16 baselines.
- Structured Sparsity ($N:M$ Patterns): Enforcing deterministic zero-patterns (such as 2:4 vector sparsity) that eliminate redundant multiply-accumulate (MAC) operations while maintaining contiguous memory addressing compatible with bare-metal cache architectures.

Through compiler-level operator fusion and layer by layer SRAM tiling [1], [4], structured sparsity and 2-bit quantization collectively unlock sub-megabyte language modeling on bare-metal hardware.

Summary of Literature Review

Deploying Generative Pre-trained Transformers and Small Language Models directly onto bare-metal Microcontroller Units (MCUs) enables decentralized, zero-latency, and privacy-preserving natural language intelligence across edge Internet of Things ecosystems. However, standard microcontroller architectures operate under strict hardware constraints, where static non-volatile Flash storage is typically bounded between 512 KB and 2 MB, dynamic SRAM is constrained to 64 KB – 512 KB, and computational operations are restricted to integer single-instruction multiple-data (SIMD) units operating under a sub-50 mW power budget. Standard INT8 and INT4 Post-Training Quantization (PTQ) schemes fail to compress multi-million parameter models into this sub-megabyte boundary, creating an urgent need for extreme co-compression frameworks.

The edge computing landscape is shifting rapidly from simple sensory processing toward generative natural language understanding and localized automation. While Large Language Models have achieved remarkable reasoning capabilities, their deployment has historically depended on cloud infrastructure equipped with high-bandwidth memory and power-hungry GPUs. Even edge accelerators with multi-gigabyte LPDDR memory remain too costly and power-intensive for low-power edge nodes, industrial sensors, and smart wearables [6]. Microcontroller units such as ARM Cortex-M and RISC-V cores represent the most ubiquitous computing substrate in existence, yet running a 100-million parameter model stored in FP16 requires approximately 200 MB of storage, exceeding typical on-chip Flash by two orders of magnitude [7]. Compressing such architectures into the sub-megabyte realm requires an integrated pipeline combining ultra-low-bit (2-bit/ternary) quantization, structured hardware-aligned sparsity, and activation memory virtualization [8].

In the domain of ultra-low-bit quantization, severe degradation typically occurs when pushing weight precision down to 2 bits due to activation outlier distortion, spectral misalignment, and quantization grid collapse. To address this, Chee et al and Tseng et al. [11] introduced incoherence processing via randomized orthogonal Hadamard transformations paired with vector quantization, successfully suppressing outlier energy and achieving the first stable 2-bit model compression with up to an 8-fold storage reduction [9], [10]. Moving beyond post-training quantization, Ma et al. [5] tackled the high silicon area and energy cost of hardware floating-point multipliers by developing BitNet b1.58, a Quantization-Aware Training framework that constrains parameters to ternary states {-1, 0, +1}. This architecture completely replaces costly matrix multiplications with lightweight integer additions and subtractions while matching 16-bit Transformer perplexity. Addressing the representational loss of scalar rounding, Egiazarian et al. [12] introduced Additive Quantization for Language Models (AQLM), formulating compression as a multi-codebook vector optimization problem that attains Pareto-optimal accuracy at an effective 2-bit parameter rate.

Alongside precision reduction, sparsification plays a pivotal role in minimizing computational cycles. While unstructured pruning achieves high theoretical parameter reduction, its non-contiguous zero distributions require substantial pointer metadata and induce execution pipeline stalls on single-issue micro-ALUs. To resolve this, Frantar and Alistarh introduced SparseGPT, a second-order one-shot pruning framework enforcing 2:4 semi-structured sparsity across weight matrices, which halves the active parameter count without linguistic collapse and enables contiguous SIMD register packing. Similarly addressing the storage overhead of Compressed Sparse Row indexing on microcontrollers, Mishra et al. developed SIMD-aware block pruning aligned with embedded vector lanes (such as ARM Helium/MVE), achieving up to a 2.4-fold hardware speedup and a 48% energy reduction without pointer metadata overhead.

At the runtime and system execution level, microcontrollers suffer from a severe memory asymmetry where dynamic SRAM is vastly smaller than Flash, frequently causing inference failures due to activation tensor spikes rather than static parameter footprint. Lin et al addressed this via MCUNet and the TinyEngine compiler by enforcing static activation memory scheduling, in-place operator execution, and layer-by-layer buffer reuse, cutting peak SRAM consumption by 4 to 8 times and establishing the blueprint for embedded activation ping-ponging. To maximize processor efficiency, Lai et al designed the CMSIS-NN framework, implementing handcrafted SIMD assembly kernels tailored for ARM Cortex-M cores that yielded up to a 4.6-fold performance improvement and a 4.9-fold energy efficiency boost over generic interpreted runtimes [13].

Synthesizing these research vectors demonstrates that structured 2:4 pruning cuts active parameters by 50%, 2-bit quantization compresses surviving parameters down to 0.25 bytes per weight, and ahead-of-time memory scheduling restricts peak activations to available SRAM scratchpads, reducing a 100M-parameter model down to an executable 12.5 MB footprint [14]. Ultimately, the MicroLLM framework bridges the multi-order-of-magnitude hardware gap, proving that generative models can operate on bare-metal microcontrollers through cross-layer co-design. Overcoming remaining hurdles, specifically minimizing the instruction cycle overhead of 2-bit SIMD unpacking and adopting constant-memory architectures like State Space Models, will be essential for deploying private and localized generative intelligence across billions of embedded devices worldwide [15].

Conclusion

The MicroLLM framework bridges the multi-order-of-magnitude gap between multi-million parameter language models and bare-metal microcontrollers. By uniting 2-bit quantization with $2:4$ structured sparsity, static Flash footprints drop to sub-megabyte thresholds while computational operations convert from floating-point matrix multiplications into hardware-aligned integer additions and lookups. Overcoming remaining runtime challenges particularly the cycle overhead of 2-bit SIMD unpacking and the $\mathrm{O}(\mathrm{N})$ growth of attention activation memory will pave the way for ubiquitous, private, and localized generative intelligence on embedded microcontrollers worldwide.

References

[1] C. R. Banbury et al., "Benchmarking TinyML Systems: Challenges and Direction," IEEE Micro, vol. 41, no. 6, pp. 40–49, Nov.-Dec. 2021.

[2] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh, "GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers," in Proc. Int. Conf. Learn. Represent. (ICLR), Kigali, Rwanda, May 2023.

[3] J. Lin, W.-M. Chen, Y. Lin, C. Gan, and S. Han, "MCUNet: Tiny Deep Learning on IoT Devices," Adv. Neural Inf. Process. Syst. (NeurIPS), vol. 33, pp. 11711–11722, Dec. 2020.

[4] J. Chee, Y. Cai, V. Kuleshov, and C. De Sa, "QuIP: 2-Bit Quantization of Large Language Models with Guarantees," Adv. Neural Inf. Process. Syst. (NeurIPS), vol. 36, pp. 1245–1262, Dec. 2023.

[5] S. Ma, H. Wang, L. Ma, L. Wang, W. Wang, S. Huang, L. Dong, R. Wang, J. Xue, and F. Wei, "The Era of 1-bit LLMs: All Large Language Models are in 1.58 Bits," arXiv:2402.17764, 2024.

[6] E. Frantar and D. Alistarh, "SparseGPT: Massive Language Models Can Be Accurately Pruned in One-Shot," in Proc. Int. Conf. Mach. Learn. (ICML), Honolulu, HI, USA, PMLR, vol. 202, pp. 10323–10337, 2023.

[7] M. Sun, Z. Liu, A. Yao, and Z. Dong, "A Simple and Effective Pruning Approach for Large Language Models," in Proc. Int. Conf. Learn. Represent. (ICLR), Vienna, Austria, May 2024.

[8] L. Lai, N. Suda, and V. Chandra, "CMSIS-NN: Efficient Neural Network Kernels for Arm Cortex-M CPUs," arXiv:1801.06601, 2018.

[9] S. Mishra, R. Sharma, P. Banerjee, and K. Roy, "Accelerating Sparse Deep Neural Networks on Microcontrollers via SIMD-Aware Pruning," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 42, no. 11, pp. 3890–3903, Nov. 2023.

[10] S. Kim et al., "SqueezeLLM: Dense-and-Sparse Quantization," in Proc. Int. Conf. Mach. Learn. (ICML), Vienna, Austria, PMLR, 2024.

[11] A. Tseng, J. Chee, Q. Sun, V. Kuleshov, and C. De Sa, "QuIP#: Even Better LLM Quantization with Hadamard Incoherence and Vector Quantization," in Proc. Int. Conf. Mach. Learn. (ICML), Vienna, Austria, PMLR, vol. 235, 2024.

[12] V. Egiazarian, A. Panferov, D. Kuzmin, P. Haber, and B. Babenko, "Extreme Compression of Large Language Models via Additive Quantization," in Proc. Int. Conf. Mach. Learn. (ICML), Vienna, Austria, PMLR, vol. 235, pp. 11820–11842, 2024.

[13] M. Zhu, T. Chen, and Z. Wang, "FlashSparsity: High-Throughput Structured Sparse Matrix Multiplication on Edge Processors," in Proc. ACM/IEEE Design Autom. Conf. (DAC), San Francisco, CA, USA, 2023, pp. 1–6.
[14] J. Lin, W.-M. Chen, C. Gan, and S. Han, "MCUNetV2: Memory-Efficient Patch-based Inference for Tiny Deep Learning," Adv. Neural Inf. Process. Syst. (NeurIPS), vol. 34, pp. 8785–8798, Dec. 2021.
[15] J. Lin et al., "AWQ: Activation-Aware Weight Quantization for On-Device LLM Compression and Acceleration," in Proc. Mach. Learn. Syst. (MLSys), Santa Clara, CA, USA, vol. 6, 2024, pp. 87–100.