Dawkhar Vaishnavi Ganpat, 2026, 14:5
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology
An Open Access Journal

Small Language Models For Edge Ai: A Review Of Efficient Architectures, Compression Techniques And On-Device Deployment

Dawkhar Vaishnavi Ganpat[1] Prof. M. A. Auti[2] Prof. M. C. Kolse[3] Prof. S. K. Said[4]
Department of Artificial Intelligence and Data Science
Jaihind College of Engineering
Pune, Maharashtra, India

Abstract—Small Language Models (SLMs) bring language understanding and generation directly to resource-constrained devices such as smartphones, wearables, IoT nodes and embedded systems. Large language models demand substantial memory, energy and connectivity, which limits their use in latency-sensitive, privacy-critical and offline settings. This paper reviews SLMs for Edge AI. It examines how compute-optimal scaling and high-quality training data enable compact models to perform well, and surveys contemporary open SLM families. Compression techniques are analysed in depth, including knowledge distillation, post-training quantization to four and eight bits, activation-aware and outlier-smoothing quantization, one-shot pruning, and parameter-efficient fine-tuning with low-rank adapters. Inference optimizations such as memory-efficient attention and speculative decoding are discussed. Trade-offs among accuracy, model size, inference latency and power consumption are compared. A lightweight deployment pipeline is proposed that selects a base model, adapts it to the task, compresses it and benchmarks it on target hardware before deployment. Research gaps, objectives and challenges including reasoning limits, hallucination, evaluation and secure updates are discussed. The review concludes that SLMs make private, low-latency, offline-capable language intelligence practical at the edge.

Index Terms—Small language models, edge AI, quantization, knowledge distillation, pruning, LoRA, on-device inference, efficient transformers

I. INTRODUCTION

Large language models have shown remarkable abilities, but the largest models require clusters of GPUs, send user data to remote servers and stop working without internet access. Many applications, such as a voice assistant on a phone, a translation tool in a remote village or a maintenance assistant on a factory floor, need language intelligence that runs locally, quickly and privately.

Small language models, typically with a few hundred million to a few billion parameters, address this need. Hoffmann *et al.* [1] showed that for a fixed compute budget, smaller models trained on more tokens can outperform larger under-trained ones, and Gunasekar *et al.* [2] demonstrated that carefully curated, high-quality data allows small models to achieve strong results.

This paper reviews SLMs for Edge AI, focusing on compression and deployment, and proposes a practical deployment pipeline.

II. BACKGROUND AND RELATED WORK

1. What Makes A Model Small

A model's size is determined by its number of parameters and the precision in which they are stored. A one-billion-parameter model in 16-bit precision needs about 2 GB of memory for weights; in 4-bit precision, about 0.5 GB. Fig. 1 summarizes the techniques used to make language models fit on edge devices.

© 2026 Dawkhar Vaishnavi Ganpat, This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly credited.

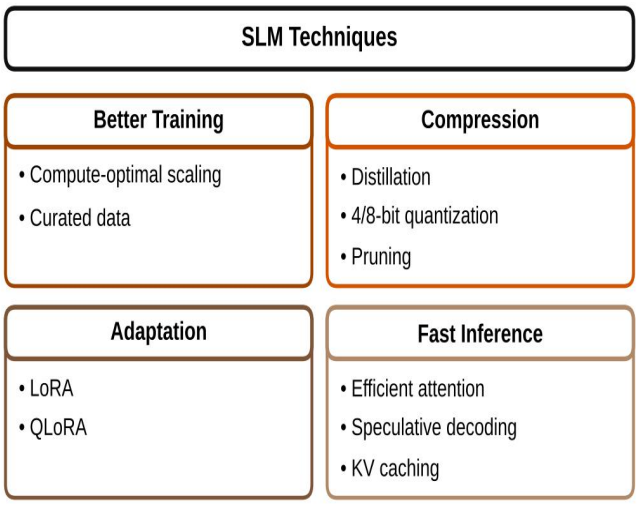


Fig. 1. Techniques for making language models fit on edge devices.

2. Open Slm Families

LLaMA [3] showed that openly released models trained on public data could be competitive, Phi-3 [4] targeted on-phone deployment, and Gemma [5] released lightweight models for developers. These families provide base models for edge applications.

3. Review Of Related Studies

Hinton *et al.* [6] introduced knowledge distillation, and Sanh *et al.* [7] applied it to BERT in DistilBERT, retaining most of the teacher's language understanding with a much smaller and faster model.

For quantization, Frantar *et al.* [8] proposed GPTQ, which quantizes weights to three or four bits using approximate second-order information with little accuracy loss. Lin *et al.* [9] introduced activation-aware weight quantization, protecting the small fraction of weights most important to activations, and Xiao *et al.* [10] proposed SmoothQuant, which migrates activation outliers into weights so that both can be quantized to 8 bits.

Frantar and Alistarh [11] showed with SparseGPT that large models can be pruned to substantial sparsity in one shot. For adaptation, Hu *et al.* [12] proposed LoRA, which trains small low-rank matrices instead of all weights, and Dettmers *et al.* [13] combined LoRA with 4-bit quantization in QLoRA, enabling fine-tuning on a single GPU.

For inference speed, FlashAttention [14] reduces memory traffic in attention, and speculative decoding [15] uses a small draft model to propose tokens that a larger model verifies. Table I summarizes these works.

Ref.	Authors	Year	Technique Used	Merits	Demerits
[7]	Sanh et al.	2019	DistilBERT	Smaller, faster BERT	Encoder only
[1]	Hoffmann et al.	2022	Compute-optimal scaling	Smaller models on more data perform well	Large-scale training study
[12]	Hu et al.	2022	LoRA low-rank adaptation	Cheap task fine-tuning	Rank choice matters
[8]	Frantar et al.	2023	GPTQ post-training quantization	3–4 bit weights, little loss	Weights only
[10]	Xiao et al.	2023	SmoothQuant	8-bit weights and activations	Hardware support needed
[11]	Frantar and Alistarh	2023	SparseGPT one-shot pruning	High sparsity without retraining	Sparse speed-ups vary
[13]	Dettmers et al.	2023	QLoRA	Fine-tuning quantized models	Training still needs GPU

Ref.	Authors	Year	Technique Used	Merits	Demerits
[9]	Lin et al.	2024	Activation-aware quantization	Protects salient weights	Calibration data needed
[4]	Abdin et al.	2024	Phi-3 small models	Capable model running on phones	Limited factual knowledge

TABLE I Summary of Literature Survey

III. RESEARCH GAP AND OBJECTIVES

1. Identified Research Gaps

- 1. Reasoning and factual accuracy of SLMs lag behind large models on complex tasks.
- 2. Aggressive quantization can harm specific abilities in ways that standard benchmarks miss.
- 3. Energy and latency on real phones and microcontrollers are rarely reported.
- 4. Support for Indian languages in small models is limited.
- 5. Secure updating and protection of on-device models are under-studied.

2. Objectives Of The Review

- **O1:** To review the principles that enable capable small language models.
- **O2:** To examine distillation, quantization, pruning and parameter-efficient fine-tuning.
- **O3:** To propose a deployment pipeline for SLMs on edge devices.
- **O4:** To compare compression techniques in accuracy, size and speed.
- **O5:** To identify challenges and future research directions.

IV. DESIGN OF THE PROPOSED SYSTEM

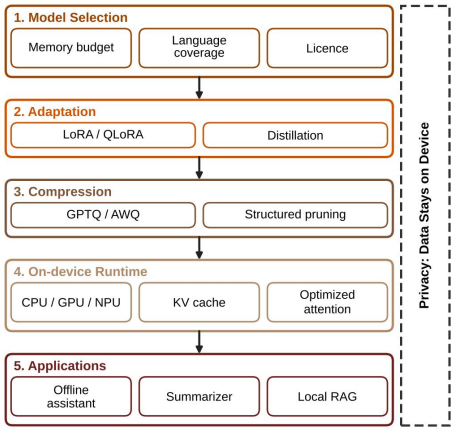


Fig. 2 presents the proposed deployment pipeline.

Fig. 2. Proposed pipeline for deploying small language models at the edge.

1. Model Selection Layer

A base SLM is chosen according to memory budget, language coverage and licence.

2. Adaptation Layer

The model is adapted to the target task with LoRA or QLoRA on domain data, and optionally distilled from a larger teacher.

3. Compression Layer

Weights are quantized with GPTQ or AWQ; optionally, structured pruning removes redundant components.

4. Runtime Layer

An efficient on-device runtime with optimized attention and key–value caching executes the model on CPU, GPU or NPU.

5. Application Layer

Applications such as offline assistants, summarizers and command interpreters call the runtime, with retrieval from local documents where needed.

6. Methodology

Fig. 3 shows the workflow. The memory needed for weights is approximately

V. M ≈ N_PARAMS · B / 8 BYTES (1)

where b is the number of bits per weight. LoRA represents a weight update as the product of two low-rank matrices:

$$W' = W + B \cdot A \quad (2)$$

where B is d×r, A is r×k and the rank r is much smaller than d, so only a small number of parameters are trained.

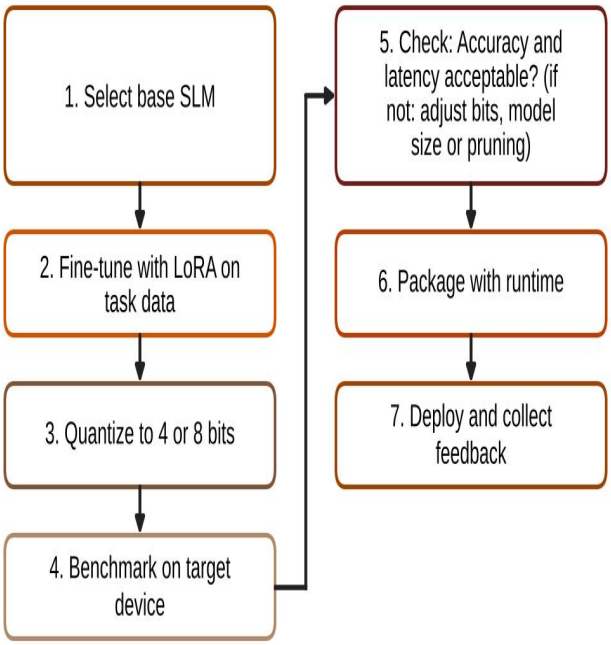


Fig. 3. SLM optimization and deployment workflow.

The compressed model is benchmarked for accuracy on task data, tokens per second, time to first token, peak memory and energy per query.

VII. DISCUSSION AND COMPARISON

Table II compares compression techniques. Quantization gives the largest memory savings with modest accuracy loss, distillation produces smaller architectures, pruning helps when hardware supports sparsity, and LoRA makes adaptation cheap.

Technique	Size Reduction	Accuracy Impact	Effort
8-bit quantization	~2× vs FP16	Very small	Low
4-bit quantization	~4× vs FP16	Small	Low–Medium
Distillation	Architecture-level	Small–Medium	High
Pruning	Variable	Small–Medium	Medium
LoRA (adaptation)	Tiny trainable set	Task gain	Low

TABLE II Comparison of Compression Techniques

VIII. CHALLENGES AND FUTURE DIRECTIONS

Table III lists challenges. Future work includes hybrid systems that answer simple queries on-device and escalate complex ones to the cloud, multilingual SLMs for Indian languages, hardware-aware architecture search and federated fine-tuning on user devices.

Challenge	Mitigation
Weaker reasoning	Task fine-tuning; cloud fallback
Hallucination	Local retrieval grounding
Quantization side effects	Task-specific evaluation
Indian languages	Multilingual training data
Model security	Encrypted weights; signed updates

TABLE III Challenges and Mitigation

IX. CONCLUSIONS

Small language models make it possible to run language intelligence privately, quickly and offline on everyday devices. The reviewed literature shows that compute-optimal training, high-quality data, distillation, low-bit quantization, pruning and parameter-efficient fine-tuning together shrink models dramatically while preserving much of their capability. With careful benchmarking and attention to accuracy limits, SLMs can bring the benefits of language AI to the edge.

1. Acknowledgment

The authors express their sincere gratitude to the Department of Artificial Intelligence and Data Science, Jaihind College of Engineering, Pune, for providing the facilities, guidance and encouragement needed to carry out this work. The authors also thank the faculty members and colleagues whose valuable suggestions helped improve the quality of this review.

REFERENCES

1. J. Hoffmann et al., "Training compute-optimal large language models," in Proc. NeurIPS, vol. 35, 2022, pp. 30016–30030.
2. S. Gunasekar et al., "Textbooks are all you need," arXiv:2306.11644, 2023.
3. H. Touvron et al., "LLaMA: Open and efficient foundation language models," arXiv:2302.13971, 2023.
4. M. Abdin et al., "Phi-3 technical report: A highly capable language model locally on your phone," arXiv:2404.14219, 2024.
5. Gemma Team, "Gemma: Open models based on Gemini research and technology," arXiv:2403.08295, 2024.
6. G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," arXiv:1503.02531, 2015.
7. V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter," arXiv:1910.01108, 2019.
8. E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh, "GPTQ: Accurate post-training quantization for generative pre-trained transformers," in Proc. ICLR, 2023.
9. J. Lin et al., "AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration," in Proc. Mach. Learn. Syst. (MLSys), 2024.
10. G. Xiao, J. Lin, M. Seznec, H. Wu, J. Demouth, and S. Han, "SmoothQuant: Accurate and efficient post-training quantization for large language models," in Proc. ICML, 2023, pp. 38087–38099.
11. E. Frantar and D. Alistarh, "SparseGPT: Massive language models can be accurately pruned in one-shot," in Proc. ICML, 2023, pp. 10323–10337.
12. E. J. Hu et al., "LoRA: Low-rank adaptation of large language models," in Proc. ICLR, 2022.
13. T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "QLoRA: Efficient finetuning of quantized LLMs," in Proc. NeurIPS, vol. 36, 2023.
14. T. Dao, D. Y. Fu, S. Ermon, A. Rudra, and C. Ré, "FlashAttention: Fast and memory-efficient exact attention with IO-awareness," in Proc. NeurIPS, vol. 35, 2022, pp. 16344–16359.
15. Y. Leviathan, M. Kalman, and Y. Matias, "Fast inference from transformers via speculative decoding," in Proc. ICML, 2023, pp. 19274–19286.