PEER-REVIEWED JOURNAL OF COMPUTER SCIENCE (PRJCS)

ISSN Online : Open Access Scholarly Journal

Small Language Models for On-Device and Private Intelligence

Mini T V

Associate Professor, Department of Computer Science, Sacred Heart College (Autonomous), Chalakudy, Kerala, India.

Article information

Received: 15th April 2026
Received in revised form: 28th April 2026
Accepted: 3rd May 2026
Available online: 13th May 2026

Volume: 1
Issue: 5
DOI: https://doi.org/10.5281/zenodo.20151276

Abstract

While the public discussion of language models has been dominated by ever-larger frontier systems, a parallel research line is delivering capable models in the one-to-eight billion parameter regime that run on phones, laptops, and embedded devices. Models such as Microsoft Phi, Google Gemma, Meta Llama 3.2, Apple OpenELM, and Alibaba Qwen-1.5B demonstrate that careful data curation, knowledge distillation, and aggressive post-training quantisation can produce on-device models competitive with much larger ones on common reasoning and instruction-following benchmarks. This paper surveys the small-language-model (SLM) landscape, the algorithmic techniques that make it possible distillation, pruning, quantisation, and architectural innovation and the use cases, deployment patterns, and open challenges of running language models privately on user devices.

Keywords: - Small Language Models, On-Device AI, Knowledge Distillation, Quantisation, GPTQ, AWQ, Edge AI, Private Inference.

I. INTRODUCTION

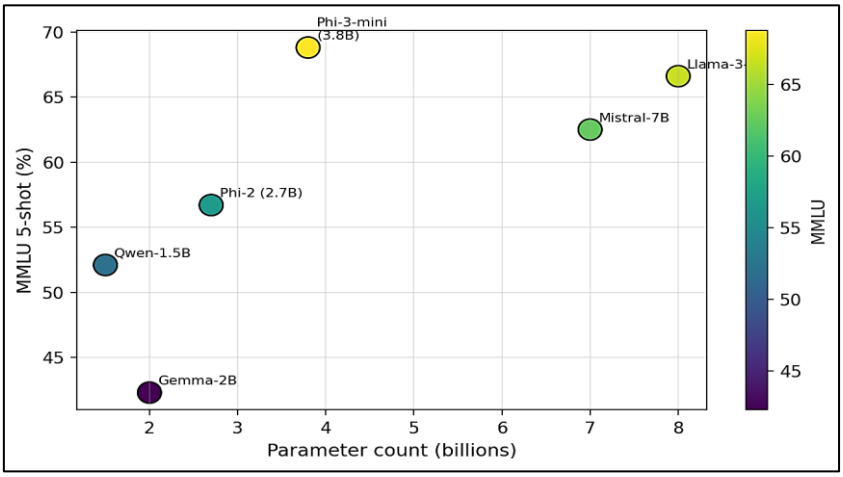
The trajectory of language models from GPT-3 [1] through GPT-4 [2] established a clear scaling narrative: capability rises with parameter count, training tokens, and compute. Hoffmann et al.'s Chinchilla scaling laws [3] refined this view, showing that for a given compute budget, models should be smaller and trained on more data than the prior consensus. A complementary insight emerged from the Phi series of Gunasekar, Li, et al. [4], [5]: models with carefully curated textbook-quality data can substantially outperform larger models trained on web crawl. This established the small-language-model (SLM) paradigm, in which models with one to eight billion parameters are deliberately engineered to run on consumer hardware while delivering capable instruction-following and reasoning.

The motivations for SLMs are operational rather than scientific. They include privacy (data does not leave the device), latency (no network round trip), cost (no per-token cloud charge), availability (offline operation), and personalisation (per-user adaptation without aggregating user data centrally). This paper surveys the SLM landscape, the techniques that make it possible, the resulting deployment patterns, and the open challenges.

II. PROMINENT SMALL LANGUAGE MODELS

By early 2026 the SLM landscape includes several active families. The Phi series from Microsoft [4], [5], [6] uses curated synthetic and textbook-style data to reach strong reasoning at 1.3B-7B parameters, with Phi-3-mini at 3.8B parameters scoring above 68% on MMLU 5-shot [6]. Google's Gemma 2B and 7B are open-weight derivatives of Gemini engineering [7]. Meta's Llama 3.2 1B and 3B [8] prioritise mobile deployment with explicit support for on-device runtimes. Apple's OpenELM [9] released open weights with 270M-3B variants tailored to private on-device inference. Alibaba's Qwen 1.5B and 0.5B [10] have been widely adopted in international deployments. Mistral 7B [11] and its derivatives remain influential, with Mistral demonstrating that grouped-query attention and sliding-window attention deliver efficiency gains. Figure 1 plots representative MMLU performance against parameter count.

Fig. 1. MMLU 5-shot accuracy versus parameter count for representative open small language models.



III. KNOWLEDGE DISTILLATION

Knowledge distillation, formalised by Hinton et al. [12], trains a small student network to match the soft outputs of a larger teacher. In the language-model context, distillation is performed either in pre-training (training on token distributions of the teacher) or in instruction tuning (training on teacher-generated demonstrations). DistilBERT and TinyBERT pioneered this approach for masked language models [13]. Modern SLMs use distillation more aggressively: synthetic instruction data and chain-of-thought traces produced by frontier models are used as training data for SLMs, yielding compact models with markedly stronger reasoning than their parameter count alone would suggest [14].

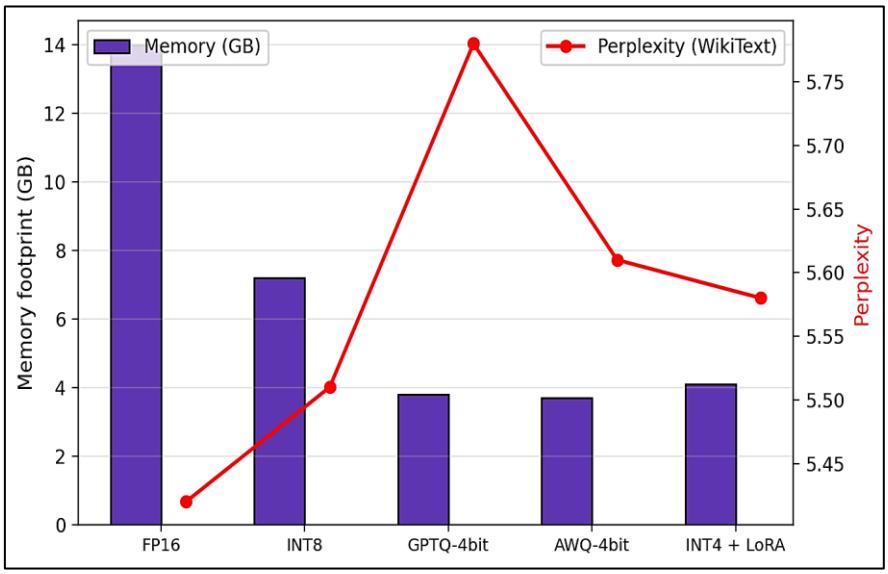
IV. PRUNING AND ARCHITECTURE

Sparsity has long been studied as a route to efficiency. Frankle and Carbin's lottery-ticket hypothesis [15] suggested that dense networks contain sparse subnetworks that train to comparable accuracy. Magnitude-based pruning, structured pruning, and Sun et al.'s Wanda method [16] demonstrate that 50%+ unstructured sparsity can be obtained on LLMs with limited accuracy loss. Architecturally, grouped-query attention [17], sliding-window attention, mixture-of-experts routing, and improved positional encodings (RoPE, ALiBi) reduce inference cost without sacrificing capability. SLMs typically combine several of these innovations.

V. QUANTISATION

Quantisation reduces numerical precision of weights and activations and is the dominant deployment-time compression technique. Post-training quantisation methods including GPTQ [18] and AWQ [19] achieve four-bit weights with negligible perplexity degradation by carefully reconstructing layer outputs over a small calibration set. Activation quantisation is more challenging because outliers drive numerical range; SmoothQuant [20] addresses this with weight-activation rebalancing. QLoRA [21] combines four-bit base quantisation with low-rank adapters and enables fine-tuning of 65B-class models on a single GPU. Figure 2 contrasts memory footprint and perplexity for several post-training schemes applied to a typical seven-billion-parameter model.

Fig. 2. Memory footprint vs. perplexity for representative quantisation schemes (7B model).



VI. INFERENCE RUNTIMES AND DEPLOYMENT

Several inference engines target on-device language models. llama.cpp [22] is a C/C++ engine that supports a wide range of CPUs, including Apple Silicon, ARM, and x86, and is the de-facto baseline for community deployment. MLC-LLM [23] uses Apache TVM to compile language models to GPU, NPU, and WebGPU back-ends. Apple's Core ML and Foundation Models, Google's MediaPipe LLM Inference, and Qualcomm's QNN/Genie support hardware-accelerated SLM inference on respective platforms. ONNX Runtime, llamafile, and Ollama provide portability layers. The combination of four-bit quantisation and modern mobile NPUs makes seven-billion-parameter models practical on flagship phones and three-billion-parameter models comfortable across the mid-range.

Table 1. Representative Open Small Language Models

Model	Year	Parameters (B)	Reported MMLU 5-shot (%)
Phi-2 [5]	2023	2.7	56.7
Gemma-2B [7]	2024	2.0	42.3
Phi-3-mini [6]	2024	3.8	68.8
Llama-3.2-3B [8]	2024	3.0	63.4
Mistral-7B [11]	2023	7.0	62.5
Llama-3-8B [8]	2024	8.0	66.6
Qwen2-1.5B [10]	2024	1.5	52.1

VII. APPLICATIONS

On-device SLMs underpin a growing class of applications. Autocomplete and writing assistance, previously a network-bound feature, can run entirely on device, removing privacy and latency concerns. Mobile photo and document understanding combines vision encoders with SLM language decoders for caption generation, OCR cleanup, and email drafting. Voice assistants increasingly use a small on-device model to handle common queries, falling back to a cloud model only for harder ones. Code completion at the editor cursor and offline translation are further deployed examples. The pattern is generally one of intelligent fallback: an SLM handles fast, private queries, with a frontier model invoked only when necessary [24].

VIII. CHALLENGES AND OPEN PROBLEMS

Several questions are unresolved. Long-context support on device remains constrained by KV-cache memory; sliding-window and state-space variants offer partial relief. Tool use and agentic behaviours have so far been demonstrated mostly with frontier models, although recent work demonstrates competent function calling on Phi-3 and Llama-3.1-8B with appropriate fine-tuning. On-device personalisation that respects user privacy invites federated and differential-privacy techniques [25]. Evaluation is itself complicated, since headline benchmarks like MMLU may be contaminated and do not directly measure utility. Finally, the energy and thermal envelope of sustained on-device inference remains an active engineering concern, particularly for wearables and embedded devices.

IX. ON-DEVICE DEPLOYMENT IN PRACTICE

Shipping a small language model on a phone is now an engineering problem more than a research problem, and the engineering choices are revealing. Modern flagship phones expose three distinct compute substrates: a CPU, a GPU, and a neural processing unit. Apple's A17/A18 Pro Neural Engine, Qualcomm's Hexagon NPU, and Google's Tensor G4 each provide tens of trillions of integer operations per second at modest power. CPU-only inference with llama.cpp [22] is portable and sufficient for short prompts, but sustained throughput requires the NPU. The NPU programming surface is vendor-specific, which has produced a small fragmented ecosystem of runtimes including Core ML, MediaPipe LLM Inference, and Qualcomm Genie. Apple Foundation Models, announced at WWDC 2024, formalises the on-device pattern at the operating-system level.

Memory pressure rather than compute is usually the binding constraint. A four-bit quantised seven-billion-parameter model occupies roughly four gigabytes of RAM at minimum, with a few hundred megabytes of overhead for the inference runtime, the KV cache, and intermediate activations. On phones with eight gigabytes of RAM this is a significant resident-set commitment that competes with the foreground application and the operating system, and aggressive paging strategies tend to harm interactivity. The three-billion-parameter regime exemplified by Phi-3-mini, Llama-3.2-3B, and Gemma 2B sits more comfortably within mid-range hardware envelopes. The trade-off between capability and footprint dominates product decisions in ways that benchmark scores alone do not predict.

Latency, particularly time-to-first-token, is the second binding constraint. Prefill of a long prompt at four-bit precision on a phone NPU takes single-digit seconds for moderate context lengths, while subsequent tokens stream at twenty to fifty per second on flagship devices. The user-perceived effect is acceptable for chat-style interaction but visibly slower than cloud frontiers. Speculative decoding, where a small drafter model proposes tokens that a larger verifier checks in batched fashion, has been adapted for on-device use and yields meaningful end-to-end speedups when both models share a vocabulary. Sliding-window attention and prompt caching of system prompts further reduce per-turn latency, and in some products both techniques are layered.

Personalisation is where on-device language models offer something cloud frontiers cannot easily replicate. Continual fine-tuning on user data, including emails, notes, and chat history, is feasible because the data never leaves the device. LoRA adapters [21] applied to a frozen base model provide a small, swappable personalisation layer of a few megabytes per user. The combination of a strong frozen base, a small personalisation adapter, and a privacy-preserving on-device inference path is a credible architecture for the next generation of personal assistants. Apple's research on Apple Intelligence and Google's AICore for Pixel both move in this direction, and both treat the on-device model and the cloud frontier as complementary rather than competitive parts of the system.

A subtler lesson from the past two years is that small language models are not simply smaller versions of frontier models. The training recipes that work best for Phi-3 or Llama-3.2-3B include curated synthetic data, distillation from larger teacher models, careful filtering of low-quality web text, and training-token counts well above the Chinchilla-optimal point. Each of these techniques borrows from the frontier but adjusts for the constraint of small parameter budgets. The result is that an SLM in 2026 with two or three billion parameters is roughly competitive with a 2022-vintage model an order of magnitude larger. The implication for product builders is that the small-model curve is moving as fast as, or faster than, the frontier curve, and shipping today on a current SLM will not lock a product into yesterday's capability. The implication for researchers is that small-model research is not a poor cousin of frontier research; it has its own distinct questions, including the right ratio of synthetic to natural data, the limits of distillation, the optimal balance between depth and width at small scales, and the degradation profile under aggressive quantisation. Several of these questions are not yet answered, and the open-weights releases from Microsoft, Google, Meta, Apple, and Alibaba have made the relevant experiments tractable for a much broader research community than was previously the case. Anyone with a single high-end GPU can now contribute meaningfully to small-model training research, and that democratisation matters for the pace of the field.

X. CONCLUSION

Small language models have moved from a research curiosity to a deployed product category. The combination of curated training data, distillation, pruning, four-bit quantisation, and modern inference runtimes makes capable language models practical on consumer hardware. The next phase will probably feature stronger on-device tool use, longer effective contexts via hybrid architectures, and tighter integration of SLMs with cloud frontier models in cooperative patterns. As privacy, latency, and operating-cost pressures grow, the role of on-device intelligence is set to expand further across consumer and enterprise software.

REFERENCES

[1] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, *et al*., "Language models are few-shot learners," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2020.

[2] OpenAI, "GPT-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[3] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, *et al*., "Training compute-optimal large language models," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2022.

[4] S. Gunasekar, Y. Zhang, J. Aneja, C. Cesar Teodoro Mendes, A. Del Giorno, S. Gopi, *et al*., "Textbooks are all you need," *arXiv preprint arXiv:2306.11644*, 2023.

[5] Y. Li, S. Bubeck, R. Eldan, A. Del Giorno, S. Gunasekar, and Y. T. Lee, "Textbooks are all you need II: Phi-1.5 technical report," *arXiv preprint arXiv:2309.05463*, 2023.

[6] M. Abdin, S. Aneja, H. Awadalla, A. Awadallah, A. A. Awan, N. Bach, *et al*., "Phi-3 technical report: A highly capable language model locally on your phone," *arXiv preprint arXiv:2404.14219*, 2024.

[7] Gemma Team, Google, "Gemma: Open models based on Gemini research and technology," *arXiv preprint arXiv:2403.08295*, 2024.

[8] Meta AI, "Llama 3.2: Edge-class open foundation models," *Meta Technical Note*, 2024.

[9] S. Mehta, M. H. Sekhavat, Q. Cao, M. Horton, Y. Jin, C. Sun, *et al*., "OpenELM: An efficient language model family with open-source training and inference framework," *arXiv preprint arXiv:2404.14619*, 2024.

[10] A. Yang, B. Yang, B. Hui, B. Zheng, B. Yu, C. Zhou, *et al*., "Qwen2 technical report," *arXiv preprint arXiv:2407.10671*, 2024.

[11] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, *et al*., "Mistral 7B," *arXiv preprint arXiv:2310.06825*, 2023.

[12] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[13] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT," in *Proc. NeurIPS EMC2 Workshop*, 2019.

[14] L. C. Magister, J. Mallinson, J. Adamek, E. Kharitonov, and R. Aharoni, "Teaching small language models to reason," in *Proc. Annu. Meeting Assoc. Comput. Linguistics (ACL)*, 2023.

[15] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2019.

[16] M. Sun, Z. Liu, A. Bair, and J. Z. Kolter, "A simple and effective pruning approach for large language models," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2024.

[17] J. Ainslie, J. Lee-Thorp, M. de Jong, Y. Zemlyanskiy, F. Lebrón, and S. Sanghai, "GQA: Training generalized multi-query transformer models from multi-head checkpoints," in *Proc. Conf. Empirical Methods Nat. Lang. Process. (EMNLP)*, 2023.

[18] 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)*, 2023.

[19] J. Lin, J. Tang, H. Tang, S. Yang, X. Dang, and S. Han, "AWQ: Activation-aware weight quantization for LLM compression and acceleration," in *Proc. Mach. Learn. Syst. (MLSys)*, 2024.

[20] 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. Int. Conf. Mach. Learn. (ICML)*, 2023.
[21] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "QLoRA: Efficient finetuning of quantized LLMs," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2023.
[22] G. Gerganov *et al*., "llama.cpp: Inference of LLaMA model in pure C/C++," *GitHub repository*, 2023–2025.
[23] MLC-AI, "MLC-LLM: Universal LLM deployment engine," *GitHub repository*, 2023–2025.
[24] Y. Ye, B. Xie, S. Zheng, R. Lou, P. Yang, *et al*., "On-device LLMs: A survey," *arXiv preprint arXiv:2409.00088*, 2024.