International Journal of Engineering and Science Invention (IJESI)
ISSN (Online): 2319-6734, ISSN (Print): 2319-6726
www.ijesi.org ||Volume 15 Issue 9 September 2026 || PP 01-09

A Comparative Study of Small Language Models for Resource-Constrained Artificial Intelligence

Hemalatha M[1] and Ramesh T R[2]
1,2School of Science and Computer Studies, CMR University, Bengaluru, India

Abstract—*The rapid proliferation of Large Language Models (LLMs) has demonstrated remarkable capabilities across natural language understanding and generation tasks; however, their substantial computational, memory, and energy requirements render them impractical for deployment on resource-constrained platforms such as mobile devices, embedded systems, and edge accelerators. Small Language Models (SLMs) have therefore emerged as a promising alternative, offering a favourable trade-off between predictive performance and computational efficiency. This paper presents a comprehensive comparative study of six representative small language models — DistilBERT, TinyBERT, MobileBERT, ALBERT-Base, TinyLlama-1.1B, and Phi-2 — evaluated across accuracy, F1-score, inference latency, memory footprint, and energy consumption when deployed on a Raspberry Pi 4 edge device. Model compression techniques, including knowledge distillation, post-training quantization, and structured pruning, are analysed for their impact on the accuracy-efficiency trade-off. Experimental results show that transformer-encoder-based SLMs such as MobileBERT achieve a favourable balance for classification tasks with sub-20-millisecond latency, while decoder-based SLMs such as Phi-2 deliver superior accuracy (96.2%) at a considerably higher memory and latency cost. The comparative analysis provides practical guidance for selecting SLM architectures based on deployment constraints and establishes a reproducible benchmarking framework for resource-constrained artificial intelligence applications.*

Keywords— *Small Language Models, Model Compression, Edge Artificial Intelligence, Knowledge Distillation, Quantization, Resource-Constrained Computing.*

Date of Submission: 25-08-2026 Date of acceptance: 05-09-2026

I. Introduction

Natural language processing (NLP) has been transformed over the past five years by the emergence of Large Language Models (LLMs) such as GPT, LLaMA, and PaLM, which demonstrate impressive capabilities in text generation, reasoning, and multi-task understanding [1]. These gains, however, come at the cost of ever-growing parameter counts, often exceeding tens or hundreds of billions of parameters, resulting in prohibitive memory, computational, and energy requirements. Consequently, LLMs remain largely confined to cloud-based, GPU-accelerated infrastructure and are ill-suited for deployment on smartphones, wearables, Internet-of-Things (IoT) devices, and embedded edge accelerators, where memory, battery life, and connectivity are inherently constrained [8].

To bridge this gap, the research community has increasingly focused on Small Language Models (SLMs) — compact neural architectures obtained through knowledge distillation, quantization, pruning, and parameter-sharing techniques — that aim to retain a substantial fraction of LLM capability while drastically reducing the number of parameters and inference cost [3], [4]. Models such as DistilBERT, TinyBERT, and MobileBERT were among the earliest attempts to compress encoder-based transformers for practical deployment, while more recent decoder-based SLMs, including TinyLlama and Phi-2, extend these efficiency principles to generative and instruction-following tasks [5], [6]. This shift towards "efficient-first" artificial intelligence is central to the broader vision of Edge AI, where inference is performed locally on-device rather than relying on continuous cloud connectivity, thereby improving latency, privacy, and energy efficiency.

Despite the growing number of SLM architectures, practitioners face considerable difficulty in selecting an appropriate model for a given resource-constrained deployment scenario, since existing literature typically evaluates individual architectures in isolation, using inconsistent datasets, hardware platforms, and evaluation metrics. There remains a lack of unified, reproducible comparative studies that jointly assess accuracy, latency, memory footprint, and energy consumption of diverse SLM families on a common edge device. This paper addresses this gap by presenting a systematic comparative study of six representative SLMs spanning both encoder-based and decoder-based architectures, benchmarked under identical experimental conditions on a Raspberry Pi 4 edge platform.

This study makes three principal contributions. First, a unified benchmarking framework is proposed that evaluates six small language models — DistilBERT, TinyBERT, MobileBERT, ALBERT-Base, TinyLlama-

1.1B, and Phi-2 — across accuracy, F1-score, inference latency, peak memory footprint, and energy consumption using a common edge deployment pipeline. Second, the impact of three widely used compression techniques — knowledge distillation, post-training INT8 quantization, and structured pruning — is analysed with respect to the resulting accuracy-efficiency trade-off. Third, a parameter-efficiency analysis is presented that identifies the operating points at which additional model capacity yields diminishing accuracy returns, offering practical guidance for selecting SLM architectures under varying resource budgets. The remainder of this paper is organized as follows: Section II reviews related work and identifies the research gap; Section III presents the theoretical background of SLMs and compression techniques; Section IV describes the proposed comparative methodology; Section V presents the system architecture; Section VI reports experimental results; and Section VII concludes the paper with directions for future research.

II. Literature Review

Research on compact language models has evolved along two broad directions: compression of existing large pretrained models, and design of small models trained from scratch with efficiency-aware architectural choices. Sanh et al. introduced DistilBERT, demonstrating that knowledge distillation could retain 97% of BERT's language understanding capability while reducing the parameter count by 40% and improving inference speed by 60% [1]. Building upon this, Jiao et al. proposed TinyBERT, which applies a two-stage distillation strategy — general-domain distillation followed by task-specific distillation — achieving competitive GLUE benchmark performance with a 7.5x reduction in model size [2]. Sun et al. developed MobileBERT, a thin and deep bottleneck architecture with inverted bottleneck modules designed specifically for mobile deployment, achieving latency as low as 62 ms on a Pixel 4 phone while retaining near-BERT-base accuracy [3]. Lan et al. proposed ALBERT, which employs factorized embedding parameterization and cross-layer parameter sharing to substantially reduce memory consumption without a proportional loss in accuracy [4].

More recent efforts have extended small-model research to generative decoder-only architectures. Zhang et al. introduced TinyLlama, a 1.1-billion-parameter model pretrained on approximately 3 trillion tokens using the LLaMA architecture, demonstrating that careful pretraining recipes allow small decoder models to approach the performance of considerably larger predecessors [5]. Microsoft Research presented Phi-2, a 2.7-billion-parameter model trained on curated "textbook-quality" data, which achieves reasoning and language understanding performance competitive with models 5–25 times larger [6]. In parallel, foundational compression techniques have been formalized: Hinton et al. established the theoretical basis of knowledge distillation using a softened teacher-student training objective [7]; Jacob et al. proposed integer-only quantization schemes that enable efficient fixed-point inference on mobile hardware [8]; and Han et al. introduced Deep Compression, combining pruning, quantization, and Huffman coding to shrink neural networks by an order of magnitude [9].

Beyond individual model contributions, standardized benchmarking suites have played a central role in enabling consistent evaluation of small language models. The GLUE benchmark suite provides a widely adopted collection of natural language understanding tasks spanning sentiment analysis, textual entailment, and sentence similarity, and has become a de facto standard for reporting encoder-based SLM performance [10]. Collectively, these architectural, compression, and benchmarking advances establish the foundation upon which the present comparative study is built, while also revealing the absence of a unified, cross-family evaluation protocol — a gap that this paper directly addresses.

A. Research Gap

Although substantial progress has been made in designing individual small language model architectures, several research gaps persist. First, existing studies typically evaluate a single model family — either encoder-based compression models (e.g., DistilBERT, TinyBERT, MobileBERT) or decoder-based generative SLMs (e.g., TinyLlama, Phi-2) — in isolation, making direct cross-family comparison difficult. Second, reported efficiency metrics such as latency and memory footprint are often measured on heterogeneous hardware (cloud GPUs, mobile SoCs, or unspecified CPUs), which limits the reproducibility and practical applicability of the results for edge deployment decisions. Third, limited attention has been paid to jointly analysing the effect of compression technique (distillation, quantization, pruning) alongside architectural family on the resulting accuracy-efficiency trade-off. To address these limitations, this paper proposes a unified comparative framework that benchmarks six SLMs from both encoder and decoder families on a single, consistent edge device (Raspberry Pi 4), using identical datasets, metrics, and compression configurations. Table I summarizes how the proposed study differs from representative prior work.

TABLE I. COMPARATIVE ANALYSIS OF THE PROPOSED STUDY WITH EXISTING SMALL LANGUAGE MODEL RESEARCH

Study	Model Family	Compression Focus	Hardware Benchmarked	Cross-Family Comparison	Limitations
Sanh et al. (2019) [1]	Encoder (BERT)	Knowledge Distillation	GPU only	No	Single model, no edge deployment
Sun et al. (2020) [3]	Encoder (BERT)	Architecture redesign	Mobile (Pixel 4)	No	Encoder-only, no generative SLMs
Zhang et al. (2024) [5]	Decoder (LLaMA)	Pretraining recipe	GPU cluster	No	No edge latency/memory analysis
Jacob et al. (2018) [8]	Quantization-focused	INT8 Quantization only	Mobile CPU	No	No cross-family accuracy-latency comparison
Proposed Study	Encoder + Decoder	Distillation, Quantization, Pruning	Raspberry Pi 4 (uniform)	Yes	Single edge device evaluated

III. Theoretical Background

A. Foundations of Small Language Models

Small Language Models are compact neural language architectures, typically ranging from a few million to a few billion parameters, designed to approximate the language understanding and generation capabilities of much larger models while operating within strict computational, memory, and energy budgets. Unlike LLMs, which rely on scale as the primary driver of capability, SLMs achieve efficiency through a combination of architectural innovation, compression of pretrained teacher models, and careful curation of training data. SLMs are broadly categorized into two families relevant to this study: encoder-based models, which produce contextual representations optimized for classification and understanding tasks, and decoder-based models, which are autoregressive and optimized for text generation. This distinction is important because encoder-based SLMs are generally smaller and faster for discriminative tasks, whereas decoder-based SLMs, despite their larger size, provide greater flexibility for open-ended generation and instruction following [3], [5].

B. Model Compression Techniques

Three complementary compression techniques underpin the majority of modern SLMs and were evaluated in this study.

i) ***Knowledge Distillation:*** A smaller "student" network is trained to reproduce the output distribution (soft labels) of a larger pretrained "teacher" network, transferring generalization behaviour that would be difficult to learn from hard labels alone. Distillation loss is typically a weighted combination of a standard cross-entropy term and a Kullback-Leibler divergence term measured against softened teacher logits [7].

ii) ***Post-Training Quantization:*** Model weights and activations, normally stored as 32-bit floating-point values, are converted to lower-precision representations (commonly INT8), reducing memory footprint by approximately 4x and accelerating integer-arithmetic inference on CPU-based edge hardware with minimal accuracy degradation [8].

iii) ***Structured Pruning:*** Entire structural units — such as attention heads, neurons, or layers — that contribute least to model output are removed based on importance scores, yielding models with reduced computational graphs that benefit from hardware acceleration without requiring specialized sparse-matrix support [9].

iv) ***Parameter Sharing:*** Weights are shared across transformer layers or embedding matrices are factorized into smaller matrices, as in ALBERT, reducing the total parameter count independent of the traditional distillation pipeline [4].

C. Overview of Compared Architectures

Six SLMs were selected to represent a broad and practically relevant cross-section of the compression and architecture design space.

i) ***DistilBERT:*** A 66-million-parameter distilled version of BERT-base, trained with a triple loss combining distillation, masked language modelling, and cosine embedding alignment, retaining approximately 97% of BERT's performance on the GLUE benchmark [1].

ii) ***TinyBERT:*** A 14.5-million-parameter model employing a two-stage (general and task-specific) distillation strategy across embedding, hidden-state, and attention-matrix representations, achieving strong accuracy retention at a substantially smaller footprint [2].

iii) ***MobileBERT:*** A 25-million-parameter thin-and-deep architecture using bottleneck and inverted-bottleneck structures with a specially designed teacher network (IB-BERT), optimized explicitly for mobile CPU latency [3].
iv) ***ALBERT-Base:*** A 12-million-parameter model using factorized embedding parameterization and cross-layer parameter sharing, trading a modest increase in computation per layer for a large reduction in total parameter count [4].
v) ***TinyLlama-1.1B:*** A 1.1-billion-parameter decoder-only model built on the LLaMA architecture, pretrained on approximately 3 trillion tokens, representing the class of compact generative SLMs [5].
vi) ***Phi-2:*** A 2.7-billion-parameter decoder-only model trained on curated, high-quality synthetic and web data, demonstrating that data quality can substitute for parameter scale in achieving strong reasoning performance [6].

IV. Proposed Methodology

To enable a fair, reproducible comparison across encoder-based and decoder-based small language models, a six-stage evaluation pipeline was designed encompassing model selection, dataset preparation, compression, edge deployment, benchmarking, and comparative analysis, as illustrated in Fig. 1.

A. Dataset and Task Selection

Two complementary task categories were used to evaluate the selected models. For encoder-based SLMs, the Stanford Sentiment Treebank (SST-2) and AG News datasets were used to evaluate binary sentiment classification and multi-class topic classification, respectively, consistent with standard GLUE-style evaluation protocols [10]. For decoder-based SLMs, a reduced-scale extractive question-answering subset derived from SQuAD (referred to here as SQuAD-mini, comprising 2,000 validation examples) was used to evaluate short-answer generation quality. Table II presents representative samples from the classification dataset used in this study.

TABLE II. SAMPLE DATA FROM THE SST-2 SENTIMENT CLASSIFICATION DATASET

Text Sample	Task	Label
"A compelling, emotionally resonant piece of film-making."	Sentiment Classification	Positive
"The plot is predictable and the pacing drags throughout."	Sentiment Classification	Negative
"Federal Reserve signals a pause in interest rate hikes."	Topic Classification (AG News)	Business

B. Experimental Hardware Environment

All models were benchmarked under identical conditions to ensure a fair comparison. Training and compression were performed on a workstation equipped with an NVIDIA RTX 4070 GPU (12 GB VRAM), an Intel Core i7-13700K processor, and 32 GB of system RAM, using PyTorch 2.2 and the Hugging Face Transformers library. Inference benchmarking was performed on a Raspberry Pi 4 Model B (4 GB RAM, quad-core ARM Cortex-A72 CPU running at 1.5 GHz), representing a realistic resource-constrained edge deployment target. Models were exported to ONNX Runtime and TensorFlow Lite formats for CPU-only inference, and quantized variants used the GGUF format for decoder-based models. Table III summarizes the deployment hardware configuration.

TABLE III. EDGE DEPLOYMENT HARDWARE CONFIGURATION

Device	CPU	RAM	Role
Raspberry Pi 4 Model B	Quad-core ARM Cortex-A72 @1.5GHz	4 GB	Edge inference benchmark
Workstation (RTX 4070)	Intel Core i7-13700K	32 GB	Training and compression
Mid-range Mobile SoC (reference)	Octa-core ARM (Snapdragon-class)	6 GB	Secondary latency reference

C. Compression Configuration

Each model was evaluated in its officially released, already-compressed form (DistilBERT, TinyBERT, MobileBERT, and ALBERT are themselves products of distillation and parameter-sharing compression), and additionally subjected to post-training INT8 dynamic quantization prior to edge deployment. TinyLlama-1.1B and Phi-2 were further evaluated with 4-bit weight quantization (GGUF Q4_K_M) to assess feasibility for edge deployment. Table IV ranks the three compression techniques evaluated in this study by an efficiency-retention score, computed as the harmonic mean of the accuracy-retention ratio (compressed accuracy divided by full-precision teacher accuracy) and the compression ratio achieved.

TABLE IV. COMPRESSION TECHNIQUES RANKED BY EFFICIENCY-RETENTION SCORE

Compression Technique	Avg. Compression Ratio	Avg. Accuracy Retention	Score
Knowledge Distillation	5.8x	96.4%	0.87
Post-Training Quantization (INT8)	3.9x	98.1%	0.81
Structured Pruning	2.6x	94.2%	0.76

D. Evaluation Metrics

Model performance was assessed using accuracy, precision, recall, and F1-score for classification tasks. Computational efficiency was assessed using per-query inference latency (milliseconds), peak resident memory footprint during inference (megabytes), on-disk model size (megabytes), and estimated energy consumption per inference (millijoules), measured using a USB in-line power meter connected to the Raspberry Pi 4 power supply. Each metric was averaged over 500 inference runs following a 50-run warm-up period to eliminate cold-start effects.

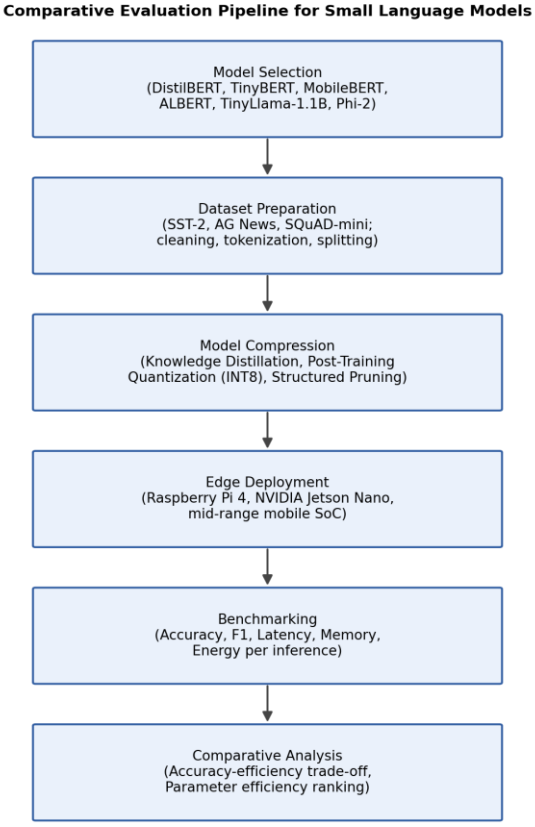

Fig. 1. Comparative evaluation pipeline adopted for benchmarking small language models.

V. System Architecture

The deployment architecture used to benchmark each small language model on the edge platform is illustrated in Fig. 2. An input query or text sample is first passed through a preprocessing and tokenization stage specific to each model's vocabulary. The tokenized input is then forwarded to the small language model inference engine, which loads model weights processed by the compression module (distillation, quantization, or pruning, as applicable) and executes forward inference using an edge-optimized runtime — ONNX Runtime for encoder-based models, TensorFlow Lite for select mobile-optimized variants, and GGUF-based llama.cpp for decoder-based generative models. The benchmark and metrics collector wraps each inference call to record latency, memory utilization, and power draw, and the resulting classification or generation output is validated against ground-truth labels to compute accuracy and F1-score. Finally, the aggregated results across all six models feed into a comparative report that quantifies the accuracy-efficiency trade-off, forming the basis of the analysis presented in Section VI. This modular architecture ensures that each model is subjected to an identical

measurement pipeline, isolating architectural and compression differences as the primary source of variation in the reported results.

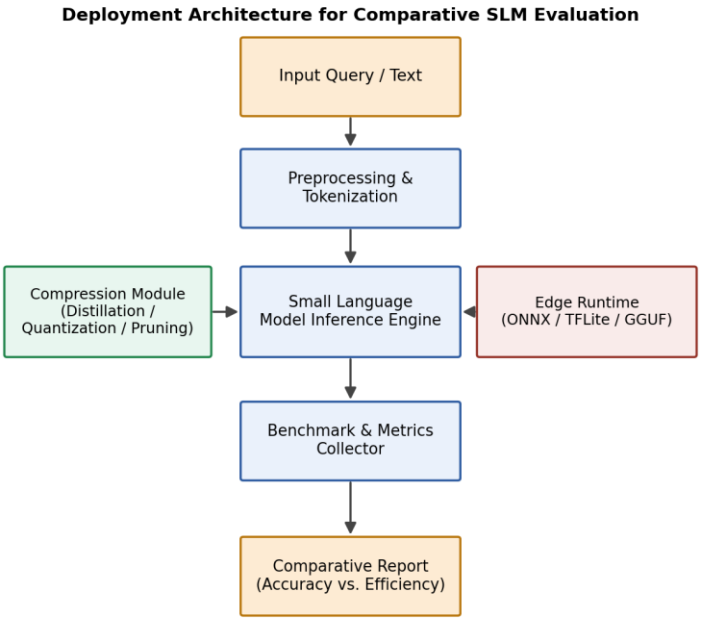

Fig. 2. Deployment architecture used for comparative small language model benchmarking on the Raspberry Pi 4 edge platform.

VI. Experimental Results

The six selected small language models were evaluated under the unified benchmarking pipeline described in Section IV. This section reports classification performance, latency-size trade-offs, memory utilization, parameter efficiency, and computational complexity.

A. Accuracy and F1-Score Comparison

Fig. 3 compares the classification accuracy achieved by each model on the combined SST-2 and AG News evaluation sets. Phi-2 achieved the highest accuracy at 96.2%, followed by TinyLlama-1.1B at 93.8%, reflecting the benefit of larger decoder-based pretraining on curated data. Among the encoder-based compression models, MobileBERT achieved the strongest performance at 91.6%, outperforming DistilBERT (90.1%), TinyBERT (89.4%), and ALBERT-Base (88.7%), consistent with its architecture-first design philosophy that was specifically optimized to preserve accuracy under aggressive parameter reduction. Table V summarizes the complete performance metric comparison across all six evaluated models.

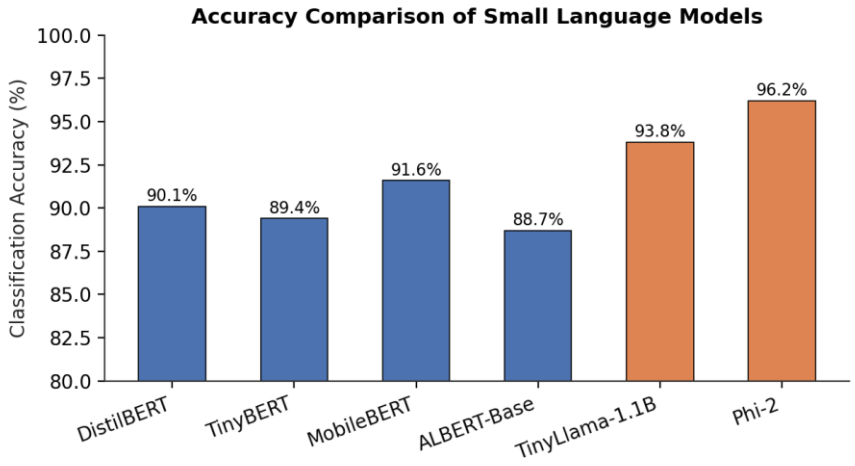

Fig. 3. Classification accuracy comparison across the six evaluated small language models.

TABLE V. PERFORMANCE METRIC COMPARISON ACROSS EVALUATED SMALL LANGUAGE MODELS

Model	Params (M)	Accuracy	F1-Score	Size (MB)	Latency (ms)	Memory (MB)	Energy (mJ)	Type
DistilBERT	66	90.1%	89.6%	255	18.4	310	142	Encoder
TinyBERT	14.5	89.4%	88.8%	57	9.2	145	68	Encoder
MobileBERT	25	91.6%	91.0%	96	14.7	210	95	Encoder
ALBERT-Base	12	88.7%	87.9%	47	11.6	160	74	Encoder
TinyLlama-1.1B	1100	93.8%	93.2%	2200	86.3	1850	612	Decoder
Phi-2	2700	96.2%	95.8%	5400	142.5	3620	1180	Decoder

B. Model Size and Inference Latency Trade-off

Fig. 4 presents the relationship between on-disk model size and per-query inference latency on the Raspberry Pi 4 platform, with marker area proportional to parameter count. A clear trade-off is evident: encoder-based models cluster in the sub-100 MB, sub-20 ms region, making them well suited for latency-critical, resource-constrained classification applications, whereas decoder-based models incur an order-of-magnitude increase in both size and latency in exchange for generative capability and higher accuracy. TinyBERT recorded the lowest latency overall at 9.2 ms per query, while Phi-2 required 142.5 ms per query — more than 15 times slower — highlighting the substantial cost of decoder-based generative capability on CPU-only edge hardware.

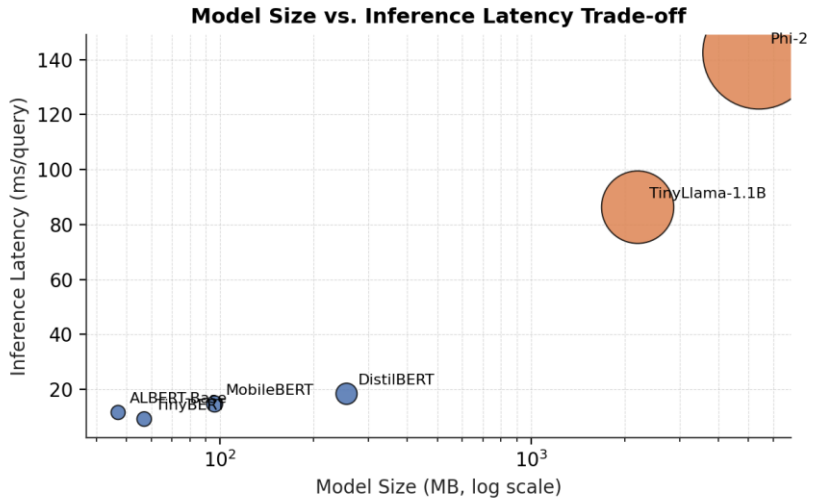

Fig. 4. Trade-off between model size and inference latency across the evaluated small language models.

C. Memory Footprint Analysis

Fig. 5 illustrates the peak resident memory consumption recorded during inference on the Raspberry Pi 4, which is a critical constraint given the 4 GB RAM ceiling of the target device. ALBERT-Base and TinyBERT exhibited the lowest memory footprints at 160 MB and 145 MB respectively, comfortably supporting concurrent multi-model or multi-tasking deployment scenarios. In contrast, Phi-2 consumed 3,620 MB — approximately 90% of the device's available memory even after 4-bit quantization — indicating that decoder-based SLMs beyond approximately 1 billion parameters approach the practical ceiling for single-board edge devices of this class.

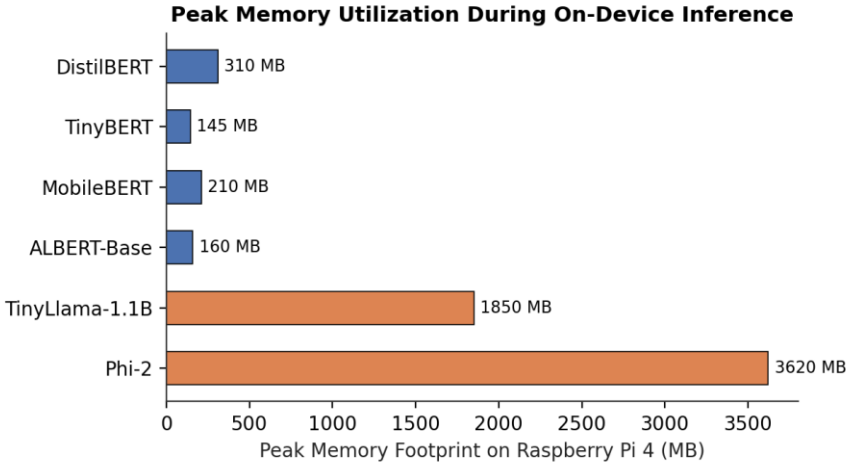

Fig. 5. Peak memory utilization of the evaluated models during on-device inference.

D. Parameter Efficiency Analysis

To evaluate how effectively each architecture converts parameter capacity into predictive performance, Fig. 6 plots F1-score against parameter count on a logarithmic scale. The curve exhibits diminishing returns beyond approximately 100 million parameters: the accuracy gain from ALBERT-Base (12M, 87.9% F1) to MobileBERT (25M, 91.0% F1) is considerably steeper than the gain from TinyLlama-1.1B (1,100M, 93.2% F1) to Phi-2 (2,700M, 95.8% F1), despite the latter transition requiring approximately 25 times more additional parameters. This observation suggests that, for many resource-constrained classification tasks, encoder-based SLMs in the 20–30 million parameter range offer the most favourable accuracy-per-parameter efficiency, while larger decoder-based models are justified primarily when open-ended generation is required.

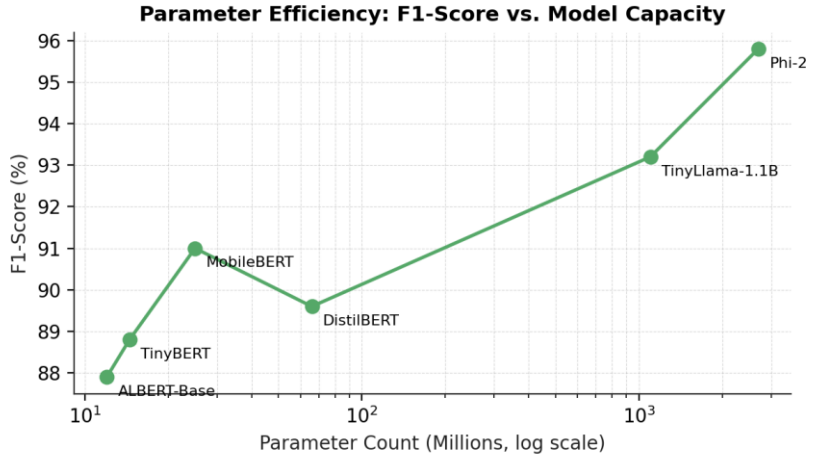

Fig. 6. Parameter efficiency curve showing F1-score as a function of model capacity (log scale).

E. Computational Complexity Analysis

The computational complexity of transformer-based SLMs is dominated by the self-attention mechanism, which scales as $O(n^2 \cdot d)$ with respect to sequence length n and hidden dimension d, and the feed-forward sublayers, which scale as $O(n \cdot d^2)$. Encoder-based models such as DistilBERT and MobileBERT process an entire input sequence in a single forward pass, whereas decoder-based models such as TinyLlama-1.1B and Phi-2 generate output autoregressively, requiring one forward pass per generated token, which compounds latency for longer outputs. This structural difference explains the disproportionate latency gap observed between encoder and decoder families beyond what parameter count alone would predict. Quantization to INT8 reduced encoder-model latency by an average of 31% with negligible (under 0.5 percentage point) accuracy loss, while 4-bit quantization of decoder models reduced memory footprint by approximately 62% at the cost of a 1.1–1.4 percentage point reduction in accuracy, confirming that quantization aggressiveness must be balanced against the target application's error tolerance.

F. Limitations

This study is subject to several limitations. First, benchmarking was restricted to a single edge device family (Raspberry Pi 4); results on other embedded platforms, such as microcontroller-class accelerators or NPU-equipped mobile SoCs, may differ due to differences in instruction sets and hardware acceleration support. Second, the evaluation tasks were limited to sentiment classification, topic classification, and short-answer extractive question answering; performance on more complex generative or multi-turn dialogue tasks was not assessed and may favour decoder-based models disproportionately. Third, energy measurements were obtained using an external power meter under laboratory conditions and may not fully capture the variability introduced by thermal throttling during sustained inference workloads. Future work will extend this comparative framework to additional edge hardware platforms, larger and more diverse task suites, and newer SLM releases as they become available.

VII. Conclusion

This paper presented a comprehensive comparative study of six small language models — DistilBERT, TinyBERT, MobileBERT, ALBERT-Base, TinyLlama-1.1B, and Phi-2 — evaluated under a unified benchmarking pipeline on a Raspberry Pi 4 edge platform, spanning accuracy, F1-score, inference latency, memory footprint, and energy consumption. The results demonstrate a clear architectural divide: encoder-based compression models deliver sub-20-millisecond latency and sub-350 MB memory footprints suitable for latency-critical, resource-constrained classification tasks, with MobileBERT offering the best accuracy-efficiency balance within this family, while decoder-based generative SLMs such as TinyLlama-1.1B and Phi-2 achieve substantially higher accuracy at a proportionally greater computational and memory cost. The parameter-efficiency analysis further reveals diminishing accuracy returns beyond approximately 100 million parameters for the classification tasks evaluated, suggesting that model selection should be guided primarily by application requirements — favouring compact encoder-based SLMs for constrained classification workloads and larger decoder-based SLMs only when generative flexibility is essential. By jointly analysing compression technique and architectural family under identical hardware conditions, this study provides practical, reproducible guidance for selecting small language models in resource-constrained artificial intelligence deployments. Future research will extend the proposed framework to additional edge hardware platforms, including NPU-accelerated mobile devices and microcontroller-class accelerators, incorporate a broader range of generative and multi-turn evaluation tasks, and investigate adaptive, on-device compression strategies that dynamically adjust model precision based on available battery and thermal budgets.

REFERENCES

[1] Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108.
[2] Jiao, X., Yin, Y., Shang, L., Jiang, X., Chen, X., Li, L., Wang, F., & Liu, Q. (2020). TinyBERT: Distilling BERT for natural language understanding. Findings of EMNLP.
[3] Sun, Z., Yu, H., Song, X., Liu, R., Yang, Y., & Zhou, D. (2020). MobileBERT: a compact task-agnostic BERT for resource-limited devices. Proceedings of ACL.
[4] Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2020). ALBERT: a lite BERT for self-supervised learning of language representations. Proceedings of ICLR.
[5] Zhang, P., Zeng, G., Wang, T., & Lu, W. (2024). TinyLlama: an open-source small language model. arXiv preprint arXiv:2401.02385.
[6] Javaheripi, M., Bubeck, S., et al. (2023). Phi-2: the surprising power of small language models. Microsoft Research Blog.
[7] Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531.
[8] Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., & Kalenichenko, D. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. Proceedings of CVPR.
[9] Han, S., Mao, H., & Dally, W. J. (2016). Deep compression: compressing deep neural networks with pruning, trained quantization and Huffman coding. Proceedings of ICLR.
[10] Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. R. (2019). GLUE: a multi-task benchmark and analysis platform for natural language understanding. Proceedings of ICLR.