Small Language Models: Architecture, Evolution, and the Future of Artificial Intelligence

Ankit Parag Shah *, Mohammad-Parsa Hosseini , Su Min Park , Connie Miao , Wei Wei

Posted Date: 13 January 2026

doi: 10.20944/preprints202601.0973.v1

Keywords: small language models; architectural innovations; knowledge distillation; multi-axis taxonomy; efficiency optimization; smart data paradigm; hybrid AI ecosystems

 doi:10.20944/preprints202601.0973.v1

Article

Small Language Models: Architecture, Evolution, and the Future of Artificial Intelligence

Ankit Parag Shah *, Mohammad-Parsa Hosseini, Su Min Park, Connie Miao and Wei Wei

Center for Advanced AI, Accenture
* Correspondence: ankit.parag.shah@accenture.com

Abstract

Large language models (LLMs) have significantly advanced artificial intelligence, yet their high computational, energy, and privacy costs pose substantial challenges. In contrast, Small Language Models (SLMs), typically with fewer than 15 billion parameters, have emerged as efficient alternatives. This survey provides a comprehensive analysis of the SLM landscape, tracing their evolution and examining architectural innovations that enhance efficiency. A novel multi-axis taxonomy is introduced to classify SLMs by genesis, architecture, and optimization goals, offering a structured framework for this field. Performance benchmarks are reviewed exhaustively, demonstrating that while LLMs excel in broad knowledge tasks, state-of-the-art SLMs match or exceed larger models in domains such as mathematical reasoning and code generation. The analysis concludes that the future of AI lies in hybrid ecosystems, where specialized SLMs manage most tasks locally, escalating complex queries to cloud-based LLMs. This tiered approach promises scalability, privacy, and the democratization of AI.

Keywords: small language models; architectural innovations; knowledge distillation; multi-axis taxonomy; efficiency optimization; smart data paradigm; hybrid AI ecosystems

1. Introduction

1.1. Motivation for SLMs

The development of language models in artificial intelligence is undergoing a pronounced divergence, with distinct trajectories emerging in both scale and deployment paradigms [1]. This divergence reflects not only technical differences but also deeper divides in philosophy, economics, and long-term objectives for AI. One trajectory emphasizes the relentless expansion of scale in pursuit of Artificial General Intelligence (AGI), while the other prioritizes efficiency, accessibility, and specialization through SLMs. This survey focuses on the latter, providing a comprehensive examination of their emergence, defining characteristics, and implications for the future of technology. The release of GPT-4 in March 2023 served as both a crowning achievement for the scaling paradigm and a powerful catalyst for a countermovement. Faced with the monumental resource requirements of such models, the AI community began to earnestly explore a different path, not one of scaling up but of scaling down intelligently. This marked a "Great Divergence" in AI philosophy [2]. The open source of Meta's Llama models [3,4] was a turning point, providing a high-quality foundation upon which the community could build. This was followed by a rapid cascade of innovation: Microsoft's Phi series demonstrated the incredible power of "textbook quality" synthetic data [5,6]; Google released its Gemma models derived from the Gemini program [7]; Mistral AI introduced highly efficient models such as Mistral-7B, which demonstrated performance exceeding expectations for their size [8]; and Alibaba's Qwen series showcased strong multilingual and coding capabilities [9].

ProRL further demonstrated that smaller models can democratize access to advanced AI capabilities by reducing computational costs, enabling broader deployment across resource-constrained settings [10]. As of mid-2025, this era is defined by a strategic focus on data quality, architectural

 doi:10.20944/preprints202601.0973.v1

refinement, and task-specific optimization, with new releases like Meta's Llama 4 series introducing natively multimodal capabilities and Microsoft's Phi-4 emphasizing advanced reasoning [11].

Recent releases have further cemented this trend: Microsoft's Phi-4 (14B) and Phi-4-reasoning-plus achieve 93.1% on GSM8K, surpassing many larger models through reasoning-centric training [12]; Google's Gemma 3 (March 2025) introduced multimodal capabilities with 128K context windows across 1B–27B parameter variants [13]; Meta's Llama 3.3 (December 2024) demonstrated that a 70B model can match the 405B Llama 3.1 on instruction-following tasks [14]; Alibaba's Qwen3 (April 2025), trained on 36 trillion tokens across 119 languages, introduced hybrid thinking modes combining fast responses with deep reasoning [15]; and DeepSeek-R1 (January 2025) showed that distilled 32B models can outperform OpenAI's o1-mini on mathematical reasoning while its 7B variant rivals much larger models [16]. These advances mark a paradigm shift in efficiency-focused AI research, redefining the design space of language models [17].

Rapid advancement of language models has transformed artificial intelligence (AI), enabling applications ranging from natural language understanding to code generation. Traditionally, the field has trended toward larger models, with LLMs like GPT-4 boasting hundreds of billions of parameters. However, this "bigger is better" paradigm introduces challenges, including exorbitant computational costs, energy demands, and deployment limitations. In response, SLMs, exemplified by Microsoft's Phi series, have emerged as a compelling alternative, offering competitive performance with significantly fewer resources.

1.2. The Paradigm Shift: From Scaling-Up to Scaling-Down

For several years, the dominant paradigm in language modeling has been governed by scaling laws, which empirically demonstrated that increasing the size of the model, the volume of the dataset and the computational budget predictably improves performance. This led to an arms race in which leading research labs and corporations developed ever-larger models, culminating in LLMs with parameter counts reaching into the hundreds of billions and even trillions. OpenAI's GPT-4, for instance, is estimated to possess approximately 1.76 trillion parameters [18], a staggering figure that underscores the resource-intensive nature of this approach. The training of such models requires immense capital investment, vast clusters of GPUs, and enormous energy expenditure, creating significant barriers to entry and centralizing cutting-edge AI capabilities within a handful of well-funded organizations.

In direct response to these challenges, a counter-movement has gained significant momentum: the development and deployment of Small Language Models. SLMs represent a paradigm shift from "bigger is better" to "smarter is better". The core vision behind SLMs is to democratize machine intelligence, making it accessible, affordable and efficient enough for everyday tasks and deployment on ubiquitous consumer hardware. This approach mitigates the key limitations of the LLM-centric paradigm, including high latency, prohibitive computational costs, and privacy risks stemming from reliance on cloud-based APIs. In contrast, SLMs are developed as specialized, task-oriented models optimized for well-defined applications.

1.3. Defining the Spectrum: SLMs vs. LLMs

The distinction between an SLM and an LLM is not defined by a single, universally agreed-upon parameter count, but rather by a spectrum of characteristics related to size, cost, and deployment target. For the purposes of this survey, we adopt the rigorous definition proposed by Lu et al. [19], focusing on decoder-only transformer models with parameter counts ranging from 100 million to 5 billion. However, it is important to acknowledge that the term is fluid, with other sources defining the range more broadly, from 1 million to 10 billion or even 30 billion parameters.

The primary differences between these two classes of models are:

Parameter Count and Model Size: SLMs operate on a vastly different scale. Models like Microsoft's Phi-3-mini (3.8B parameters) or Meta's Llama 3.2 (1B and 3B parameters) stand in stark contrast to LLMs like Meta's Llama 3.1 (405B parameters) or the colossal GPT-4 ($\sim$1.8T parameters).

 doi:10.20944/preprints202601.0973.v1

Computational and Financial Cost: The cost of training and operating an LLM can be astronomical, with training runs for models like GPT-4 estimated to exceed $100 million [20]. In contrast, SLMs are designed to be orders of magnitude cheaper to train and deploy, making them accessible to startups, academic researchers, and individual developers.

Energy Consumption and Sustainability: The immense computational requirements of LLM translate into a significant environmental footprint due to high energy consumption. SLMs, with their smaller architectures and optimized processing, offer a more sustainable path for AI development, consuming a fraction of the energy of their larger counterparts.

Deployment Target and Latency: LLMs are deployed almost exclusively in large centralized data centers and are accessed via APIs. This introduces network latency, which can be prohibitive for real-time applications. However, SLMs are specifically engineered for low-latency on-device deployment in resource-constrained environments, including mobile phones, edge servers, and devices of the Internet of Things (IoT).

Task Scope and Specialization: LLMs are generalists, trained in vast and diverse datasets to handle a wide array of tasks. SLMs are typically specialists, often fine-tuned on curated, domain-specific datasets to achieve superior performance and accuracy on narrow tasks, such as medical diagnosis, financial analysis, or code generation.

The divergence between these two paths is the creation of two distinct AI ecosystems. The LLM ecosystem is cloud-centric, characterized by a few powerful, general-purpose models provided as a service. The SLM ecosystem is edge-centric, promoting a diverse landscape of smaller, specialized models that operate locally. This is not merely a technical choice, but a move toward a more decentralized and democratized model of AI, placing power and control back into the hands of individual users and organizations.

1.4. Contributions of this Survey

Several recent surveys have examined the SLM landscape: Lu et al. [19] focus on decoder-only transformers (100M–5B parameters) with emphasis on architectures; Nguyen et al. [17] provide broad coverage of capabilities and benchmarks; and Zeng et al. [11] emphasize training strategies and data curation. Our survey extends these works with three primary contributions:

A Novel Multi-Axis Taxonomy: We introduce a framework for classifying SLMs based on their genesis (distilled vs. trained-from-scratch), architecture (dense vs. sparse), and primary optimization goal (latency, memory, or task specialization). This provides a more granular understanding of the diverse SLM landscape than size-based or model-based taxonomies in prior work.

Integrated Analysis of Efficiency Enablers: We move beyond cataloging techniques to provide an integrated analysis of how architectural innovations (e.g., GQA, RoPE) and compression methods (PTQ, QAT, GPTQ, AWQ, QLoRA, pruning, distillation) work synergistically in state-of-the-art SLMs.

Comprehensive Synthesis of Applications and Challenges: We offer an extensive review of the SLM application ecosystem with empirical deployment case studies from healthcare, finance, and edge computing, and provide deep analysis of trustworthiness challenges connecting ethical principles to concrete technical hurdles.

Ultimately, this survey demonstrates that the rise of SLMs is creating a hybrid AI ecosystem where specialized, efficient SLMs handle most tasks locally—ensuring privacy and speed—while escalating only complex queries to larger, cloud-based LLMs. This tiered intelligence represents a more scalable, sustainable, and democratized path forward for artificial intelligence.

2. The SLM Landscape: A Modern Taxonomy

To navigate the rapidly expanding field of Small Language Models, a structured classification system is essential. Existing surveys typically categorize models by their parent company or by a single optimization technique. This section introduces a more nuanced, multi-axis taxonomy that classifies SLMs along four critical dimensions: their size category, method of creation (Genesis), internal

 doi:10.20944/preprints202601.0973.v1

structure (Architecture), and intended optimization goal. This framework provides a comprehensive map of the current ecosystem, summarized in Table 1.

2.1. Axis 1: Categorization by Model Size

SLMs typically range from a few million to several billion parameters. Based on recent literature [17,19], we propose the following size categories:

- **Ultra-small** (<100M parameters): Models like TinyBERT variants, suitable for highly constrained microcontroller environments.
- **Small** (100M–1B): Models like Llama 3.2 1B, balancing efficiency and baseline performance.
- **Medium** (1B–10B): The most common category for edge deployment, including Phi-4-mini (3.8B), Mistral 7B, and Gemma 2 9B.
- **Large** (10B–15B): Models like Phi-4 (14B) that overlap with LLMs but are optimized for efficiency.

2.2. Axis 2: Classification by Genesis

The origin of an SLM profoundly influences its capabilities and development philosophy. There are two primary pathways:

2.2.1. Knowledge Distillation

In this "teacher-student" paradigm, a smaller model learns from the probabilistic outputs of a larger teacher model, transferring intelligence into a compact form. Classic examples include DistilBERT (40% smaller than BERT, retaining 97% capability) and DeepSeek-R1-1.5B (distilled from Qwen2.5). This approach enables creating specialized models without the cost of pretraining from scratch. Detailed distillation techniques are discussed in Section 5.

2.2.2. Trained-from-Scratch

These SLMs are designed and pretrained as small models from inception, with success driven by data quality rather than model size. The Microsoft Phi series exemplifies this approach—Phi-4-mini (3.8B) achieves performance rivaling Mixtral 8x7B through "textbook-quality" training data [5]. This paradigm shift from "big data" to "smart data" is detailed in Section 4.

2.3. Axis 3: Classification by Architecture

SLM architectures can be categorized into three primary families, each with distinct efficiency characteristics:

2.3.1. Dense Architectures

Traditional transformer architectures where all parameters are utilized for every input token. Examples include Llama 3.2, Qwen2, and Gemma. While straightforward to train, computational cost scales directly with parameter count.

2.3.2. Sparse and Modular (MoE)

Mixture-of-Experts architectures activate only a subset of "expert" sub-networks for each input via a router mechanism. This enables high model capacity with lower active compute—for example, Phi-3.5-MoE has 41.9B total parameters but activates only 6.6B per token. Recent examples include Llama 4 Maverick (128 experts).

2.3.3. State Space and RNN-Inspired

Alternative architectures like Mamba [21] (selective state spaces) and RWKV (attention-free RNN-transformer hybrid) offer linear time complexity versus quadratic for transformers, enabling efficient processing of long sequences. Mamba-3B achieves 5× faster inference than comparable transformers.

Detailed architectural descriptions are provided in Section 3.

 doi:10.20944/preprints202601.0973.v1

Table 1. Representative SLMs Classified by the Multi-Axis Taxonomy. Citations refer to the original technical reports or papers introducing each model.

Model	Size	Genesis	Architecture	Primary Goal
TinyBERT [22]	Ultra-small	Distilled	Dense	Latency
DistilBERT [23]	Small	Distilled	Dense	Memory
Llama 3.2 1B [14]	Small	From-Scratch	Dense	Latency
Phi-4-mini 3.8B [12]	Medium	From-Scratch	Dense	Task-Specialized
Mistral 7B [8]	Medium	From-Scratch	Dense	Latency
Gemma 2 9B [24]	Medium	From-Scratch	Dense	Task-Specialized
Mamba-3B [21]	Medium	From-Scratch	SSM	Latency
Phi-3.5-MoE [25]	Large	From-Scratch	Sparse/MoE	Memory
Phi-4 14B [12]	Large	From-Scratch	Dense	Task-Specialized
DeepSeek-R1-7B [16]	Medium	Distilled	Dense	Task-Specialized

2.4. Axis 4: Classification by Optimization Goal

SLMs can be further classified by their primary optimization target:

2.4.1. Latency-Optimized

Models designed for minimal inference time, employing techniques such as quantization (reducing numerical precision to INT4/INT8), early exit mechanisms, and hardware-specific optimizations. Critical for real-time applications like voice assistants and autonomous systems.

2.4.2. Memory-Optimized

Models minimizing memory footprint through pruning (removing redundant parameters), weight-sharing (e.g., ALBERT's cross-layer sharing), and aggressive quantization. Essential for deployment on memory-constrained edge devices and IoT.

2.4.3. Task-Specialized

Models optimized for specific domains or tasks through Parameter-Efficient Fine-Tuning (PEFT) methods such as LoRA, adapters, and prompt tuning. These techniques enable customization with minimal computational overhead while preserving base model capabilities.

Detailed optimization techniques are presented in Section 5.

2.5. Taxonomy Summary and Key Observations

Table 1 provides a summary mapping of representative SLMs across all four taxonomic axes. Several patterns emerge from this classification that illuminate the current state and trajectory of SLM development.

Dominance of Dense Architectures: The majority of production SLMs employ dense transformer architectures, reflecting their maturity and well-understood training dynamics. However, the inclusion of Mamba-3B (SSM) and Phi-3.5-MoE (Sparse/MoE) signals growing diversification as practitioners seek alternatives optimized for specific deployment constraints.

The From-Scratch Renaissance: While early SLMs like DistilBERT and TinyBERT relied on distillation from larger teachers, the current generation is predominantly trained from scratch. This shift reflects advances in data curation (the "Smart Data" paradigm) that enable competitive performance without requiring a large teacher model, democratizing SLM development.

Medium-Size Sweet Spot: The 1B–10B parameter range emerges as the most active category, balancing capability with deployability. Models in this range can run on consumer GPUs and high-end mobile devices while maintaining strong benchmark performance.

Task Specialization as Differentiator: Among from-scratch models, the primary differentiator is increasingly the optimization goal rather than architecture. Models like Phi-4-mini and Gemma 2 prioritize task specialization through extensive instruction tuning, while Mistral 7B and Llama 3.2 emphasize raw inference speed.

 doi:10.20944/preprints202601.0973.v1

Distillation for Reasoning: The inclusion of DeepSeek-R1-7B demonstrates a new paradigm where distillation transfers not just general capabilities but specialized reasoning skills from frontier models, enabling smaller models to exhibit chain-of-thought reasoning previously exclusive to much larger systems.

3. Foundations of Small Language Models

3.1. Historical Evolution and Design Trade-Offs

The evolution of SLMs reflects a broader shift in natural language processing from a focus on scale to one of efficiency and accessibility. Early developments in language modeling were dominated by the success of large-scale transformer architectures, such as BERT and GPT-3, which demonstrated that performance could be significantly improved through increased model size, data, and compute. However, the release of GPT-4 in 2023, with its estimated 1.76 trillion parameters [18] and extreme training costs, highlighted the limitations of this paradigm: high latency, energy demands, and centralized deployment.

In response, a new design philosophy emerged that favors smaller models that deliver competitive performance through architectural innovation and high-quality data. Early examples like DistilBERT and TinyBERT used knowledge distillation and pruning to compress larger models. More recent efforts, such as Microsoft's Phi series, Meta's Llama variants, and Mistral, have shown that carefully curated data and efficient training can produce SLMs capable of outperforming much larger models on reasoning and coding tasks.

Designing SLMs requires navigating critical trade-offs: balancing model size with generalization, reducing latency without sacrificing context length, and achieving efficiency without degrading performance. Techniques like quantization, low-rank adaptation, and adapter-based fine-tuning allow for specialization with minimal resource demands. At the same time, architectural alternatives such as Mamba and RWKV offer linear-time inference and support for long sequences, extending the applicability of SLM to edge devices.

This shift reflects a broader movement toward hybrid AI systems, where lightweight, task-specific SLMs operate locally for speed and privacy, while powerful LLMs are reserved for complex queries. Rather than a step back from capability, the rise of SLMs marks a strategic realignment of AI design around sustainability, scalability, and democratized access.

3.2. The "Smart Data" Paradigm

"Textbooks Are All You Need": The Phi Philosophy The Phi model series from Microsoft is a canonical example of the Smart Data paradigm in action. Their first paper, "Textbooks Are All You Need" [5], showed that a 1.3B parameter model (Phi-1) trained on a high-quality synthetic dataset of just 7 billion tokens could outperform models 10x its size on coding benchmarks. The follow-up, Phi-1.5, extended this to common sense reasoning [6], and Phi-2 and Phi-3 [26] further scaled this approach, achieving performance rivaling models like Mixtral 8x7B and GPT-3.5 with only 3.8B parameters. The core philosophy is that the quality and reasoning density of the training data are more important than its sheer volume or the model's parameter count.

Data Quality vs. Quantity: Revisiting Scaling Laws The success of SLMs trained on high-quality data forces a re-evaluation of the Chinchilla scaling laws. Although the original laws focused on the quantity of tokens, recent work suggests a more nuanced view where a "quality coefficient" should be considered [27]. A high-quality synthetic token might be worth 10, 100, or even 1,000 low-quality web tokens in terms of the learning it imparts. This implies that the path to better models may not be ever-larger datasets of raw text, but more sophisticated methods for generating and curating high-quality, targeted data. This fundamentally changes the economics of AI development, shifting value from raw data access to sophisticated data refinement techniques.

 doi:10.20944/preprints202601.0973.v1

3.3. Dense Architectures

This is the traditional transformer architecture where all model parameters are utilized for every input token processed. The majority of well-known models, both large and small, follow this dense structure. Examples in the SLM space include Meta's Llama 3 8B, Alibaba's Qwen2 7B, and Google's Gemma 9B. Although straightforward to design and train, their computational cost scales directly with their parameter count.

3.4. Sparse and Modular Architectures (MoE)

A more advanced approach is the Mixture-of-Experts (MoE) architecture. An MoE model is composed of numerous "expert" sub-networks (typically feed-forward layers), but for any given input token, a "router network" selects only a small subset of these experts (e.g., 2 out of 16) to be activated. This allows the model to have a very large total number of parameters, giving it a high capacity for knowledge, while keeping the number of active parameters used for inference low. This sparse activation results in significantly higher computational efficiency compared to a dense model of the same total size.

This architecture, famously used in GPT-4, has been successfully scaled down. Microsoft's Phi-3.5-MoE, for example, has a total of 41.9 billion parameters, but only activates 6.6 billion for any given token, achieving high performance with the efficiency of a much smaller model. Notably, models like Switch Transformers and GLaM demonstrated the effectiveness of sparsely activated layers in scaling large models efficiently. Recent SLM efforts adopt lightweight MoE designs (e.g., Distilling Sparse Mixture of Experts) to preserve efficiency on edge devices while maintaining competitive performance. The rise of MoE-based SLMs signals a trend towards more modular and composable AI systems, enabling component reuse and targeted adaptation valuable for domain-specific SLMs and federated learning settings.

3.5. State Space and RNN-Inspired Models

Mamba Introduced as a state space model (SSM) in "Mamba: Linear-Time Sequence Modeling with Selective State Spaces", Mamba leverages selective state spaces to model sequences efficiently, offering **linear time complexity** compared to the quadratic complexity of transformers. It integrates SSM with MLP blocks, simplifying the architecture and enhancing hardware efficiency, achieving **5x faster inference** than transformers. **Mamba-3B** outperforms transformers of similar size on the Pile benchmark, demonstrating state-of-the-art performance in language, audio, and genomics, making it a viable alternative for SLMs in resource-constrained environments. Its ability to handle million-length sequences with linear scaling is particularly beneficial for tasks requiring long contexts, aligning with SLM deployment needs.

RWKV The Receptance Weighted Key Value architecture, detailed in "RWKV: Reinventing RNNs for the Transformer Era", combines RNNs and transformers, providing **constant memory usage** and inference speed, supporting infinite context length, and being 100% attention-free. RWKV models, such as **RWKV-7 (Goose)**, are trained on multilingual data, offering better performance on tasks requiring long contexts, with versions up to 14 billion parameters showing scalability. Its sensitivity to prompt formatting necessitates careful design for SLM applications, but its efficiency makes it suitable for edge deployment, enhancing privacy, and reducing latency.

3.6. Compact Transformer Variants

Compact transformer architectures are specifically designed to reduce model size, inference latency, and memory footprint, making them suitable for deployment in constrained environments such as mobile devices, edge hardware, or low-cost servers. Unlike small-scale pretraining of general-purpose transformers, these models often employ architectural innovations (e.g., bottleneck layers, factorized projections, or FFT-based components) to preserve performance while drastically reducing computational requirements. In this section, we discuss representative compact transform-

 doi:10.20944/preprints202601.0973.v1

ers—MobileBERT, FNet, and TinyBERT—highlighting the design trade-offs and empirical results that inform their suitability for SLM-like use cases.

MobileBERT MobileBERT is a thin version of **BERT_LARGE** with bottleneck structures, achieving 99.2% of BERT-base's performance on GLUE with **4x fewer parameters** (25M vs. 110M) and **5.5x faster inference** on a Pixel 4 phone, with a latency of 62 ms. Its design balances self-attention and feedforward networks, making it ideal for resource-limited devices, aligning with SLM deployment goals.

FNet Detailed in "FNet: Mixing Tokens with Fourier Transforms", FNet replaces self-attention with unparameterized Fourier Transforms, achieving **92-97% of BERT's accuracy** on GLUE while training 80% faster on GPUs and 70% faster on TPUs at standard 512 input lengths. At longer sequences, it matches the accuracy of efficient transformers while outpacing them in speed, with a light-memory footprint, particularly efficient at smaller model sizes, outperforming transformer counterparts for fixed speed and accuracy budgets.

TinyBERT Introduced in "TinyBERT: Distilling BERT for Natural Language Understanding", TinyBERT is **7.5 times smaller and 9.4 times faster** than the BERT base, maintaining 96. 8% performance through a two-stage distillation process in both the pretraining and task-specific learning stages. With 4 layers, it achieves competitive results on tasks like GLUE and SQuAD, suitable for applications requiring high accuracy with low resource usage, enhancing SLM deployability.

3.7. Attention Mechanisms

Multi-Head Attention (MHA) The self-attention mechanism is the heart of the transformer, but the original Multi-Head Attention (MHA) design, while powerful, is notoriously memory-intensive. During autoregressive generation (predicting one token at a time), the model must store the Key (K) and Value (V) vectors for all previously generated tokens in a Key-Value (KV) cache. As the sequence length grows, this cache can become a significant memory bottleneck.

Multi-Query Attention (MQA) Proposed by Shazeer in 2019, MQA was a radical optimization that addressed the KV cache problem directly. Instead of each attention head having its own unique K and V projections, MQA uses a single, shared set of K and V projections across all query heads. This dramatically reduces the size of the KV cache by a factor equal to the number of heads, leading to substantial memory savings and faster inference, especially for long sequences.

Grouped-Query Attention (GQA) GQA emerged as the pragmatic and now dominant compromise between MHA's performance and MQA's efficiency. Instead of having one K/V pair for all heads (MQA) or one for each head (MHA), GQA divides the query heads into several groups and assigns a shared K/V pair to each group. This approach interpolates between the two extremes, achieving performance nearly on par with MHA while retaining most of the speed and memory benefits of MQA.

3.8. Position Encoding Innovations (RoPE)

The standard transformer architecture is permutation-invariant, meaning that it has no inherent sense of the order of tokens in a sequence. To address this, positional information must be explicitly injected.

Rotary Position Embeddings (RoPE) encode positional information by treating token embeddings as complex numbers and rotating them in a high-dimensional space based on their absolute position. This rotation is applied directly to the Query and Key vectors within the self-attention mechanism. A key property of this approach is that the dot product between two rotated vectors depends only on their relative positions ($q_m^T k_n = q_m^T R^d_{\theta,n-m} k_n$), elegantly incorporating relative positional information into the attention score.

3.9. Normalization Techniques

Normalization layers are critical components in deep neural networks, used to stabilize the training process by controlling the scale of activations in each layer.

 doi:10.20944/preprints202601.0973.v1

Layer Normalization (LayerNorm), as defined by the equation $\text{LayerNorm}(x) = \frac{x-\mu(x)}{\sqrt{\sigma^2(x)+\epsilon}}$, performs two operations: it re-centers the input vector by subtracting the mean (μ) and re-scales it by dividing by the standard deviation (σ). Although LayerNorm has been a standard choice, a simpler and more efficient alternative has gained prominence in SLMs.

Root Mean Square Normalization (RMSNorm) simplifies this process by removing the mean subtraction. Its formulation is simple $\bar{a}_i = \frac{a_i}{\text{RMS}(a)}$, where normalization is performed only by the root mean square of the inputs. This seemingly small change makes RMSNorm significantly more efficient, reducing running time by 7% to 64% in various models with performance comparable to LayerNorm.

4. Data Strategies and Training Paradigms

4.1. From Big Data to High-Quality Data

The single most important factor distinguishing modern high-performance SLMs from their predecessors is the shift in focus from the sheer amount of training data to its quality. The "big data" philosophy assumed that with enough text from the Internet, a model would learn all necessary patterns. However, raw web data are extremely noisy, filled with repetition, toxicity, low-quality content, and factual inaccuracies [28]. The "Smart Data" paradigm posits that a smaller, meticulously curated dataset can be far more effective for training, especially for smaller models that lack the capacity to simply memorize and filter out the noise internally [5].

The success of models such as Phi-3 and Phi-4 underscores a broader principle that data quality can outweigh scale. Zhou et al. (2024) formally revisited scaling laws, demonstrating that performance correlates strongly with a data-quality coefficient rather than sheer volume [29]. This insight is echoed across open-weight ecosystems such as Qwen2 [30] and Gemma [24], which prioritize curated, high-signal corpora over indiscriminate web-scale text.

4.2. Synthetic Data Generation

Perhaps the most transformative element of the Smart Data paradigm is the use of synthetic data generated by a larger, more capable teacher model (like GPT-4). Instead of just raw text, SLMs are trained on structured, explanatory, and rich in reasoning data.

- **Synthetic Textbooks:** As pioneered by Microsoft's Phi models, this involves prompting a teacher model to generate "textbook-like" content that explains concepts clearly and logically. These data are dense with information and free of the noise found in the web data [6].
- **Chain-of-Thought Data:** To improve reasoning, SLMs are trained on examples in which a teacher model has externalized its step-by-step thinking process, known as a chain of thought [31].
- **Code Generation:** For coding tasks, synthetic data includes not just code snippets, but also explanations of code, tutorials and question-answer pairs about programming concepts [32].
- **Mathematical Reasoning:** Synthetic data for math involves generating problems and their detailed, step-by-step solutions, teaching the model the process of solving the problem, not just the final answer [33].

4.3. Data Curation Pipelines

The creation of a high-quality dataset is a complex, multi-stage engineering process, often representing a significant portion of the intellectual property of a model.

1. **Source Selection:** Instead of indiscriminately scraping the web, SLM training starts with selecting high quality sources, such as filtered web pages (e.g. Common Crawl filtered by quality classifiers), academic papers (e.g. arXiv), books (e.g. Google Books), and code (e.g., GitHub) [34].
2. **Heuristic Filtering:** A series of rules are applied to clean the data, such as removing documents with little text, boilerplate content (e.g., sign in, terms of use), or skewed character distributions [35].
3. **Quality Classification:** More advanced pipelines train a dedicated classifier model to predict the quality of a document. For example, one might train a classifier to distinguish between a

 doi:10.20944/preprints202601.0973.v1

Wikipedia article and a low-quality forum post, and then use this classifier to filter the entire web corpus [4].

4. **Deduplication:** Removing duplicate or near-duplicate examples from the training data is critical. It prevents the model from wasting capacity on redundant information and has been shown to significantly improve performance [36]. This is done at multiple granularities, from document-level to sentence-level.

4.4. Filtering, Mixing, and Curriculum Learning

Training SLMs requires deliberate attention to the composition and quality of training data. Due to their limited capacity, SLMs cannot handle large volumes of noisy, redundant, or low-signal examples. As a result, data filtering and mixing strategies, often considered secondary in LLM training, has become foundational in the context of SLM.

We define *data filtering* as any technique that excludes or re-weights examples from the training set based on quality, relevance, diversity, or utility. Filtering can operate at the dataset, sample, or token level. Techniques include heuristic rules (e.g., regex-based filtering, length constraints), learned models (e.g., perplexity or loss scoring) and external evaluators (e.g., LLMs used to judge helpfulness).

Data mixing refers to the design of the input data distribution across sources and stages, e.g., mixing code with natural language, synthetic with human-authored text, or general domain with domain-specific corpora. This process is critical to ensure that SLMs generalize across modalities and tasks without becoming overfit to dominant distributions.

We propose a two-dimensional taxonomy to classify existing data mixing and filtering strategies.

This taxonomy provides a structured lens for comparing methods across multiple dimensions and helps identify underexplored directions (e.g., token-level adaptive filtering using proxy signals).

Early LLMs such as GPT-2 and GPT-3 employed minimal filtering, relying on the model scale to suppress noise. Later models like Chinchilla emphasized data deduplication and loss-aware pruning for compute efficiency [37]. In contrast, SLMs require stringent input signal control due to lower representational capacity, making filtering a central design consideration rather than a post hoc optimization.

- **AlpaGasus** filtered Alpaca's 52k instruction dataset using ChatGPT to retain a high-quality subset of 9k examples, improving instruction-following quality and reducing training cost [38].
- **Orca** adopted a multiphase curriculum involving ChatGPT-generated instructions, GPT-4 answers, and explanation-tuned examples to progressively build reasoning skills [39].
- **C-RLFT** employed reward-weighted loss scaling based on helpfulness scores, with source-specific tokens for domain conditioning [40].
- **CoLoR-Filter** used a 150M-parameter proxy model to score and retain the most useful 4
- **Collider** introduced dynamic token-level filtering during training by pruning tokens with low estimated utility, achieving a 22% speedup with negligible loss [41].

Filtering has been shown to provide large efficiency gains in SLM training:

- AlpaGasus achieved performance parity with Alpaca while using only 17% of the original dataset.
- CoLoR-Filter retained greater than 95% of task performance while using only 4% of the training data.
- Collider reduced training compute by 22% through token sparsification.
- Li et al. [42] show that balancing diversity and quality outperforms quality-only filtering in instruction tuning.

Most current filtering strategies are developed for English-language SLMs. However, multilingual training introduces challenges: perplexity and reward-based scores often vary across languages, making monolingual filtering signals unreliable. Emerging multilingual SLMs adopt language-specific LLMs or heuristics, but few explore universal scoring mechanisms.

 doi:10.20944/preprints202601.0973.v1

In multimodal pretraining, data filtering often involves image-text alignment scores, OCR confidence, or CLIP similarity metrics. However, these are typically used heuristically, and robust, learned filters for noisy image-caption pairs remain an open research problem.

1. **Dynamic filtering and curriculum-aware mixing**, where data selection evolves based on model loss, entropy, or confidence.
2. **Hybrid filtering pipelines** that combine low-cost proxy models with LLM-based ranking for scalable precision.
3. **Explainable filtering and diagnostics** to ensure transparency in data selection, especially in safety-critical or instruction-tuned models.
4. **Cross-lingual filtering frameworks** that support multilingual training without requiring language-specific tuning.

Curriculum learning structures the training process by gradually increasing the difficulty or complexity of examples. For SLMs, this can take the form of introducing shorter sequences or simpler syntactic constructs early in training, followed by longer and more contextually rich data. This approach has shown benefits in training convergence speed and stability, particularly under constrained computation. Recent approaches also explore dynamic curriculum schedules in which model performance feedback informs data selection, which is particularly useful in SLMs with limited capacity for long-term memory or deep abstraction.

4.5. Continual Pretraining and Domain Adaptation

Continual Pretraining (CPT) Continual pretraining (CPT) refers to the practice of extending training on an existing base model using additional domain-specific or task-relevant data without resetting model weights. For SLMs, CPT helps leverage the benefits of foundation models while aligning them to resource-constrained settings or niche domains. Techniques such as learning rate warm restarts, regularization to avoid catastrophic forgetting, and domain-aware sampling are key to maintaining stability. CPT is especially useful in low-compute settings where full pretraining from scratch is infeasible, and it allows SLMs to remain updated with new data trends.

Masking Techniques Advanced masking strategies significantly affect the learning dynamics of SLMs, especially under causal or autoregressive objectives. Beyond basic left-to-right masking, approaches such as span masking (used in T5), dynamic masking, and task-specific masks (e.g., in retrieval-augmented training) can guide the model toward improved generalization. Some lightweight models also explore non-contiguous masking patterns or compression-aware masking, balancing between token coverage and compute cost. For SLMs trained with limited data or steps, efficient masking contributes to better token utilization and task alignment.

Optimization Tricks (e.g., early exit, dynamic sparsity) Several optimization strategies are employed to reduce computational overhead during training and inference. Early exit mechanisms allow intermediate layers to produce outputs when confidence thresholds are met, minimizing unnecessary forward passes. Dynamic sparsity techniques, such as lottery ticket pruning and magnitude-based weight masking, enable models to shed redundant parameters during training while preserving performance. In the SLM setting, these approaches help reduce both memory and compute footprints, facilitating deployment on low-resource hardware. Combining these techniques with quantization and distillation further enhances the efficiency of SLM.

5. Model Optimization and Compression

5.1. Knowledge Distillation

Knowledge distillation is a powerful technique for model compression that leverages the capabilities of a large, pretrained model (the teacher) to train a smaller, more efficient student model. Instead of learning from raw data and hard labels, the student is trained on the rich probabilistic output (logits) of the teacher. This enables the student to capture the nuanced reasoning and generalization behavior of the teacher, effectively transferring intelligence into a compact form.

 doi:10.20944/preprints202601.0973.v1

5.1.1. The Distillation Process

Rather than using hard labels (that is, the single correct answer), the student learns from the soft targets produced by the teacher, full probability distributions over possible outputs. A specialized loss function, typically based on Kullback-Leibler (KL) divergence, measures the discrepancy between student and teacher predictions. Temperature scaling is often applied to the softmax output of the teacher to produce smoother distributions that facilitate better learning.

5.1.2. Types of Knowledge Distillation

The transferred knowledge can be categorized on the basis of its origin in the teacher model as follows:

- **Response-based KD:** The student learns from the teacher's final output logits.
- **Feature-based KD:** The student mimics intermediate representations or hidden activations.
- **Relation-based KD:** The student captures the relationships between different features or layers within the teacher.

5.2. Pruning Techniques

Pruning is a model compression technique that reduces model size and accelerates inference by systematically removing redundant components, such as weights, neurons, or even layers, from a trained neural network. This is especially useful for deploying SLMs on edge devices, though it requires care to preserve task-specific performance.

Structured pruning methods, such as **Wanda**, use heuristics such as the weight magnitude multiplied by input activation to remove complete components without retraining. Wanda achieves up to a 2.0× reduction in FLOPs with less than 1% accuracy degradation on GLUE for models like BERT-base and DistilBERT. **Adaptive pruning** applied to quasi-recurrent neural networks (QRNNs) for language modeling has demonstrated 40% energy savings on a Raspberry Pi, with only a 17% relative increase in perplexity - highlighting its potential for resource-constrained environments.

5.2.1. Types of Pruning

Pruning approaches can be categorized by the granularity and adaptivity of what is removed:

- **Weight Pruning (Unstructured):** Removes individual weights with small magnitudes, preserving the overall structure.
- **Structured Pruning:** Removes larger units such as entire neurons, attention heads, or layers for hardware-friendly speedups.
- **Dynamic Pruning:** Applies input-dependent masks during inference, enabling flexible sparsity without retraining.

5.3. Quantization and Weight-Sharing

Quantization is a highly effective compression technique that reduces the memory footprint of a model by reducing the numerical precision of its weights and activations.

The core of quantization is a mapping function that projects a range of high-precision values onto a smaller range of low-precision values. The standard method is an affine quantization scheme, defined by the formula $x_q = \text{round}(x/S + Z)$, where x is the original FP32 value, x_q is the quantized integer value, S is a floating-point scaling factor, and Z is an integer zero-point.

Quantization reduces the numerical precision of model weights and activations, significantly lowering memory and computational demands - critical for deploying SLMs on edge devices. For instance, **Phi-3-mini** (3.8B parameters) can be quantized to 4-bit, reducing its memory footprint to $\tilde{1}$.8GB and achieving over 12 tokens/sec on an iPhone 14. Similarly, 4-bit quantization of **CodeQwen 7B** achieves a pass1 of 0.429 on Lua code generation, outperforming 2-bit quantization (0.395) and highlighting the trade-off between efficiency and precision.

 doi:10.20944/preprints202601.0973.v1

5.3.1. Post-Training Quantization (PTQ)

PTQ applies quantization after a model has been fully trained, making it the most practical approach for deploying SLMs due to its simplicity and minimal computational overhead. The process typically involves three steps: (1) calibrating the quantization parameters using a small representative dataset, (2) computing optimal scaling factors and zero-points for each layer, and (3) converting weights and activations to lower precision formats.

Several advanced PTQ methods have emerged for SLMs:

- **GPTQ** [43]: Uses approximate second-order information to minimize quantization error layer-by-layer, achieving 3-4 bit quantization with minimal perplexity increase. GPTQ can quantize a 175B parameter model in approximately 4 GPU hours.
- **AWQ (Activation-aware Weight Quantization)** [44]: Identifies salient weights by observing activation magnitudes and protects them during quantization. AWQ consistently outperforms other PTQ methods for SLMs across diverse tasks and is the recommended method for SLM compression [45].
- **SmoothQuant** [46]: Migrates quantization difficulty from activations to weights by applying mathematically equivalent transformations, enabling efficient INT8 quantization of both weights and activations.

PTQ is preferred for SLMs because it requires no retraining, can be applied to any pretrained model, and typically completes in minutes to hours rather than days. However, PTQ may struggle at very low bit-widths (2-3 bits) where quantization error accumulates significantly.

5.3.2. Quantization-Aware Training (QAT)

QAT integrates quantization into the training process by simulating low-precision arithmetic during forward passes while maintaining full-precision gradients for backpropagation. This allows the model to learn to compensate for quantization error, typically achieving 0.5-1% higher accuracy than PTQ at equivalent bit-widths.

The QAT process involves:

- **Fake quantization**: During training, weights and activations are quantized and immediately dequantized, introducing quantization noise while preserving gradient flow.
- **Learnable parameters**: Scale factors and zero-points can be learned jointly with model weights, optimizing the quantization scheme for each layer.
- **Gradual precision reduction**: Some approaches start with higher precision and progressively reduce bit-width during training for smoother convergence.

QLoRA [47] combines QAT principles with parameter-efficient fine-tuning, enabling fine-tuning of 65B parameter models on a single 48GB GPU. QLoRA uses 4-bit NormalFloat (NF4) quantization—an information-theoretically optimal data type for normally distributed weights—and achieves performance matching full 16-bit fine-tuning while reducing memory requirements by 4×.

The choice between PTQ and QAT depends on deployment constraints: PTQ is ideal for rapid deployment and when retraining is infeasible, while QAT is preferred when maximum accuracy at low bit-widths is critical and training resources are available.

5.3.3. Weight-Sharing

Weight-sharing further compresses models by forcing multiple parameters to share values, turning large weight matrices into compact sets of shared weights with index mappings. **ALBERT** [48] employs cross-layer parameter sharing to reduce parameters by 18× compared to BERT-large, with minimal performance loss on SQuAD and GLUE. More recently, **hash-based weight sharing**—originally used in CNNs like HashedNets—has been applied to transformers, reducing memory by 50–75% with minimal accuracy degradation. This is particularly effective for microcontroller-scale SLMs, and can be combined with quantization for compound compression without retraining.

 doi:10.20944/preprints202601.0973.v1

Table 2. Comparison of Model Compression and Optimization Techniques for SLMs: Trade-offs Between Size Reduction, Accuracy Preservation, and Computational Requirements

Technique	What it does	Sub-methods / Variants	Size Reduction	Accuracy / Perf Considerations	Computational Cost	Ideal Use Case
Knowledge Distillation	Train a smaller student to mimic a larger teacher's outputs/behaviors.	Traditional & sequence-level transfer (e.g., BabyLLaMA); multi-teacher; DistilBERT.	High (10×–100×)	Retains a high share of teacher quality; e.g., DistilBERT: 40% smaller, 60% faster, ∼97% of BERT NLU.	High	Create a capable SLM from an LLM; reduce inference cost while preserving quality.
Pruning	Remove unimportant parameters (weights/neurons/structures) to induce sparsity.	Unstructured (SparseGPT, Wanda); Structured (hardware-friendly); Adaptive structural (AdaptPruner).	Medium–High (50–90%)	Unstructured: high sparsity but hard to accelerate; Structured: direct speedups with larger accuracy hit; Adaptive: reported +1–7% accuracy gains.	Medium	Optimize an existing model to cut compute/memory while maintaining accuracy.
Quantization	Lower numerical precision of weights/activations to shrink memory and boost throughput.	PTQ, QAT; INT8, INT4/FP4/**NF4** (**QLoRA**); GPTQ, AWQ.	Low–Medium (2×–8×); *INT8* ≈ 2×	Minimal loss with QAT; PTQ can degrade on harder tasks; **QLoRA (NF4)** can match 16-bit fine-tuning quality.	Low–Med	Final deployment optimization; pair with PEFT for low-memory training/inference.
Architectural Optimization	Design inherently compact/efficient models instead of compressing a large one.	Lightweight (MobileBERT, TinyLLaMA); Streamlined attention (**Mamba**, **RWKV**, **Hyena**); **NAS**.	Varies (architecture-dependent)	Mamba: linear scaling, up to 5× higher throughput; Hyena: up to 100× faster than optimized attention at 64K sequence length.	High (R&D)	Build from scratch under strict latency/memory targets or to maximize throughput.
PEFT	Freeze base model; train small adapter parameters to adapt behavior efficiently.	**LoRA**, **QLoRA** (LoRA+quant, NF4), **MiSS**, DoRA, IA^3, **Prompt Tuning**.	Adapter-only (base unchanged)	Often matches full fine-tuning; **QLoRA** enables fine-tuning a 65B model on a single 48GB GPU; **MiSS** is faster and more memory-efficient than LoRA.	Low	Domain/task adaptation under tight GPU memory; combine with quantization for efficiency.

5.4. Parameter-Efficient Fine-Tuning (PEFT)

5.4.1. LoRA (Low-Rank Adaptation)

Facilitates efficient fine-tuning by training low-rank matrices instead of the full model, reducing memory usage during adaptation. This is advantageous for SLMs, enabling customization for domain-specific tasks with minimal computational overhead. For instance, LoRA can reduce trainable parameters significantly, as seen in a 7B parameter model reducing from 6.5 billion to 540,000 parameters, enhancing deployability on edge devices. This technique is particularly useful for fine-tuning SLMs in niche applications, maintaining efficiency while adapting to new tasks.

5.4.2. Adapters

Adapters introduce small, trainable bottleneck layers between the existing layers of a frozen pretrained model, allowing task-specific adaptation with minimal parameter updates. This technique is especially suitable for SLMs as it enables fast and modular fine-tuning while preserving the original model weights. For example, AdapterFusion enables combining multiple adapters trained on different tasks, increasing flexibility across domains without retraining the base model. In experiments in BERT and RoBERTa, adapter-based methods achieved over 95% of full fine-tuning performance with less than 3% trainable parameters, making them ideal for low-resource environments or device deployment.

5.4.3. BitFit

BitFit is a minimalistic fine-tuning strategy that only updates the bias terms of a pre-trained model while keeping all other parameters frozen. Despite its simplicity, BitFit has demonstrated surprising effectiveness, particularly in classification and regression tasks. For example, on the GLUE benchmark, BitFit achieves competitive accuracy while fine-tuning less than 0.1% of the model parameters, enabling rapid adaptation with minimal resource demands. Its ultra-lightweight nature makes it a compelling choice for SLMs, particularly when fine-tuning must occur on-device or under tight computational budgets.

5.4.4. Prompt Tuning & Prefix Tuning

Prompt tuning and prefix tuning adjust the input space instead of model weights by learning task-specific embeddings (prompts or prefixes) prepended to the input sequence. These techniques are highly parameter-efficient, training only a few virtual tokens while keeping the entire model frozen.

 doi:10.20944/preprints202601.0973.v1

For example, prefix tuning on GPT-2 and T5 achieves a performance close to full fine-tuning across a range of tasks while updating less than 1% of the parameters. Such methods are especially useful for SLMs deployed across multiple domains, allowing lightweight and reusable adaptation by simply swapping out the learned prompts, enabling efficient multitask inference even on constrained devices.

5.5. Inference Optimization Trade-offs: A Critical Analysis

While compression techniques enable SLM deployment on resource-constrained devices, they introduce fundamental trade-offs that practitioners must carefully navigate. This section provides a deeper analysis of these trade-offs, drawing on recent empirical studies to illuminate the complex interplay between latency, memory, accuracy, and energy consumption.

5.5.1. The Latency-Accuracy Frontier

The relationship between inference latency and model accuracy is not linear but exhibits distinct regimes with different characteristics:

Sub-millisecond regime (<10ms): Achievable with aggressive 2-4 bit quantization on edge accelerators, but accuracy degradation becomes severe. Recent studies show that at 4-bit precision, even 32B models incur accuracy drops of 2.9%, while smaller 1.5B and 7B models experience drops exceeding 10% on reasoning tasks [49].

Real-time regime (10-100ms): The practical sweet spot for most SLM deployments. Models like Phi-4-mini achieve 12+ tokens/second on mobile devices (iPhone 14) with 4-bit quantization while maintaining competitive accuracy. Edge processing in this regime eliminates network round-trips, achieving response times essential for autonomous systems and industrial automation.

Quality-optimized regime (>100ms): When accuracy is paramount, techniques like Phi-4-Reasoning-Plus employ RLHF for enhanced reasoning at the cost of increased latency. This regime is appropriate for complex analytical tasks where correctness outweighs speed.

5.5.2. Memory-Accuracy Scaling Laws

Empirical analysis reveals a superlinear relationship between memory requirements and functional accuracy. Key findings from recent benchmarking studies include:

- **Superlinear resource scaling:** Achieving each 10 percentage point gain in pass@1 accuracy on code generation tasks requires approximately 3–4× additional VRAM [50].
- **Active parameter efficiency:** IBM's Granite 3.0 models demonstrate effective parameter utilization—the 1B model activates only 400M parameters at inference, while the 3B model activates 800M, minimizing latency while maintaining performance.
- **Compression cliff:** Research indicates that model compression exhibits graceful degradation up to a threshold, after which performance drops sharply. This cliff typically occurs when the student model falls below 10% of the teacher's parameters [51].

5.5.3. Quantization vs. Pruning: Empirical Evidence

A systematic evaluation across six SLMs (0.5B–3.8B parameters), seven languages, and seven downstream tasks [45] reveals critical insights:

Quantization advantages:

- Consistently outperforms pruning in preserving model fidelity, multilingual perplexity, and reasoning accuracy.
- AWQ (Activation-aware Weight Quantization) emerges as the recommended method for SLM compression.
- Maintains better generalization across diverse task types.

Quantization limitations:

- Advantages diminish on complex knowledge and reasoning tasks (e.g., OpenBookQA).

 doi:10.20944/preprints202601.0973.v1

- Reveals a disconnect between compression fidelity metrics (perplexity) and downstream task performance.
- Trends observed in LLMs (e.g., Wanda's competitive performance to SparseGPT) do not generalize to SLMs.

Key practitioner guidance: The SLMQuant benchmark study [52] demonstrates that LLM-optimized quantization methods require adaptation for SLMs due to different weight distribution characteristics. Practitioners should avoid relying on single metrics and validate compression effectiveness across multiple task types.

5.5.4. Energy-Accuracy Trade-offs on Edge Devices

A comprehensive analysis of 28 quantized LLMs deployed on Raspberry Pi 4 (4GB RAM) across five standardized datasets reveals distinct energy-accuracy profiles [53]:

Table 3. Energy-Accuracy Trade-offs for Edge-Deployed SLMs [53]. Results from evaluation of 28 quantized LLMs on Raspberry Pi 4 (4GB RAM) across five standardized datasets.

Model Profile	Energy Consumption	Accuracy	Best Use Case
Llama 3.2 (optimized)	Medium	High	Balanced applications
TinyLlama	Low	Moderate	Battery-constrained IoT
Phi-3 Mini	High	High	Accuracy-critical tasks

These findings suggest that model selection for edge deployment should be driven by application-specific constraints rather than general benchmarks. Llama 3.2 provides the best balance of accuracy and power efficiency for most use cases, while TinyLlama suits ultra-low-power environments at reduced accuracy.

5.5.5. Hardware-Aware Optimization

Recent advances in hardware-software co-design have yielded significant efficiency gains:

LUT Tensor Core: Microsoft's lookup table-based tensor core achieves 6.93× inference speedup while using only 38.3% of traditional tensor core area [54]. This approach replaces multiplication operations with bit-wise table lookups, enabling efficient low-bit LLM inference on resource-constrained devices.

T-MAC: A novel LUT-based method for mixed-precision GEMM that operates without dequantization or multiplication. By storing compact tables on-chip, T-MAC minimizes memory access overhead—a critical bottleneck for edge inference.

Memory Bandwidth Utilization (MBU): The ELIB benchmarking framework [55] introduces MBU as a key metric indicating the percentage of theoretically efficient memory bandwidth usage during inference. This metric enables fair comparison across heterogeneous edge platforms with different hardware characteristics.

5.5.6. The Compression-Robustness Tension

A fundamental tension exists between model compression and robustness that has significant implications for safety-critical deployments:

- **Calibration sensitivity:** Compressed models exhibit heightened sensitivity to calibration data selection. Suboptimal calibration can amplify capability degradation beyond what compression alone would cause [56].
- **Error accumulation:** Aggressive pruning or quantization can trigger abrupt underfitting, manifesting as sudden capability collapse rather than gradual degradation.
- **Task-specific fragility:** Models may retain benchmark performance while failing on distribution-shifted inputs, necessitating extensive robustness testing post-compression.

 doi:10.20944/preprints202601.0973.v1

Recovery strategies: Recent work on Recover-LoRA [57] demonstrates that functional degradation from compression can be partially mitigated through targeted fine-tuning. By using synthetically generated data from a pretrained SLM and limiting updates to LoRA adapters, this approach enables efficient accuracy recovery without access to original training data.

6. Applications and Deployment Strategies

The theoretical advantages of Small Language Models translate directly into a rapidly expanding ecosystem of practical, real-world applications. The ability of SLMs to operate efficiently in resource-constrained environments is unlocking use cases that were previously infeasible with cloud-dependent Large Language Models.

6.1. On-Device Intelligence: The Edge Computing Revolution

The primary catalyst for the widespread adoption of SLMs is their suitability for deployment on edge devices—a category that includes everything from smartphones and laptops to IoT sensors and automotive computing units.

Privacy and Security: By processing data locally, sensitive information never has to leave the user's device or an organization's secure network. This is critical for industries governed by strict data privacy regulations like HIPAA in healthcare or GDPR in Europe.

Low Latency: On-device inference with SLMs eliminates the network round-trip latency associated with cloud-based processing, thereby enabling near-real-time response generation. By executing computations locally on the edge device, this approach not only reduces end-to-end latency but also ensures consistent performance under bandwidth limitations or intermittent connectivity. Moreover, on-device processing inherently enhances data privacy and security by obviating the need to transmit potentially sensitive information to external servers.

Offline Capability: SLMs can function entirely without an internet connection, enabling use in environments with unreliable or non-existent connectivity.

A key enabler in this domain is the open source platform Ollama, which drastically simplifies the process of running SLMs on local machines. Recent 2025 updates have further lowered the barrier to entry by introducing a full-featured desktop application and enhanced multimodal support, abstracting away the command-line expertise once required. By managing the complexities of model deployment, Ollama provides the foundational layer for developers to leverage the benefits of on-device AI.

Building on this foundation, end-user applications such as PocketPal AI demonstrate the tangible results of this strategy. As an open-source mobile app, PocketPal AI uses engines such as llama.cpp to run models such as Phi-3 and Llama 3.1 directly on a smartphone. This provides a direct-to-consumer example of an application that guarantees both offline capability and enhanced privacy. Together, the evolution of deployment platforms like Ollama and the emergence of consumer-facing applications like PocketPal AI illustrate a mature ecosystem that makes the power of SLMs practical, accessible, and secure on edge devices.

6.2. Industry Use Cases

6.2.1. Healthcare

Clinical Decision Support: SLMs can be fine-tuned on specific medical literature and patient data to assist clinicians in real-time, analyzing symptoms, medical history, and lab results to suggest potential diagnoses.

Administrative Automation: SLMs can automate appointment scheduling, verify insurance eligibility, process claims, and generate summaries of patient records.

Drug Discovery: By analyzing vast datasets of scientific papers and clinical trial results, SLMs can help researchers identify promising drug targets.

 doi:10.20944/preprints202601.0973.v1

6.2.2. Finance

Automated Invoice Processing: SLMs can extract and validate key information from invoices, contracts, and compliance reports, reducing manual error.

Real-Time Fraud Prevention: SLMs can analyze transaction patterns in real-time to identify anomalies indicative of fraud.

Credit Risk Assessment: Banks can use SLMs to perform preliminary assessments of new loan applications.

6.2.3. Retail and Manufacturing

In-Store Personal Assistants: Retailers can embed SLMs into mobile apps or kiosks to provide personalized product recommendations.

Predictive Maintenance: SLMs can analyze sensor data in real-time to predict equipment failures before they occur.

Supply Chain Optimization: SLMs can analyze historical sales data and traffic patterns to predict demand fluctuations.

Table 4. SLM Applications Across Industries. Use cases and benefits are synthesized from published industry reports and academic literature. Example models: BioMistral [58], Med-PaLM [59], BioGPT [60], Llama 4 Scout [61], and Phi-4-mini [12].

Industry	Use Case	Description	Key Benefits	Example SLMs
Healthcare	Clinical Decision Support	Analyzes patient data to suggest diagnoses, treatment plans, and potential drug interactions.	Improved Accuracy, Enhanced Patient Privacy, Faster Diagnosis	BioMistral, Med-PaLM
	Administrative Automation	Automates routine administrative tasks like scheduling appointments, managing patient records, and generating reports.	Increased Efficiency, Significant Cost Reduction, Reduced Human Error	General SLMs (fine-tuned)
	Drug Discovery	Scans vast amounts of scientific literature and molecular databases to identify potential drug targets, accelerate compound discovery, and predict efficacy.	Accelerated Research Speed, Lower Development Costs, Higher Success Rates	BioGPT
Finance	Invoice Processing	Extracts and verifies data from financial documents, such as invoices, receipts, and purchase orders, for automated entry.	Improved Efficiency, Enhanced Accuracy, Reduced Manual Effort	Custom SLMs (financial)
	Fraud Prevention	Analyzes real-time transaction patterns and historical data to detect and flag suspicious activities indicative of fraud.	Low Latency Detection, Stronger Security, Reduced Financial Loss	Custom SLMs (anomaly)
	Risk Assessment	Evaluates loan applications, credit scores, and market data to assess financial risk and make informed lending decisions.	Increased Speed, Greater Consistency, Objective Evaluation	Custom SLMs (credit)
Retail	Personal Assistant	Provides personalized product recommendations, answers customer queries, and assists with shopping decisions based on user preferences.	Enhanced Personalization, Improved Customer Privacy, Increased Sales	Llama 4 Scout, Phi-4-mini
	Predictive Maintenance	Analyzes sensor data from machinery and equipment to predict potential failures, scheduling maintenance proactively to prevent downtime.	Maximized Uptime, Reduced Operational Costs, Extended Equipment Lifespan	Custom SLMs (IoT)
	Supply Chain Optimization	Predicts demand fluctuations, optimizes inventory levels, and streamlines logistics for efficient supply chain management.	Greater Efficiency, Optimized Inventory, Reduced Waste	General SLMs (ERP)

6.3. Cascaded and Hybrid SLM–LLM Systems

Cascaded and hybrid SLM–LLM architectures leverage the efficiency of Small Language Models for routine tasks while delegating complex reasoning or long-context processing to larger models. This approach reduces latency and cost by allowing SLMs to handle the majority of queries, using LLMs only when higher accuracy or broader knowledge coverage is required. Recent systems implement dynamic routing mechanisms, where confidence scores or task classifiers determine whether to escalate a query to an LLM. Hybrid pipelines are proving particularly effective in enterprise and edge applications, combining the responsiveness and low resource footprint of SLMs with the advanced reasoning capabilities of LLMs.

 doi:10.20944/preprints202601.0973.v1

6.4. Empirical Case Studies

This section presents empirical evidence from recent SLM deployments across healthcare, finance, and edge computing.

6.4.1. Healthcare: On-Device Clinical Reasoning

On-device SLMs for clinical reasoning [62] achieved clinically acceptable accuracy while reducing response times by 47% compared to cloud alternatives. Key findings: (1) HIPAA/GDPR-compliant edge deployment is feasible; (2) domain-specific fine-tuning is essential—off-the-shelf models showed unacceptable hallucination rates on medical queries; (3) multilingual SLMs address critical gaps in underserved linguistic communities.

6.4.2. Finance: Real-Time Processing

Financial SLM deployments [63] demonstrate: (1) **Document processing:** Fine-tuned Mistral 7B achieved 94% entity extraction accuracy with 3× throughput improvement and <200ms latency; (2) **Fraud detection:** Quantized 3B models process 50K+ transactions/second with sub-10ms latency and 23% fewer false positives than traditional ML pipelines. On-premises deployment satisfies data residency requirements.

6.4.3. Edge Computing Benchmarks

The ACL 2025 study [51] evaluated 15+ SLMs across edge platforms:

Table 5. Edge Deployment Results [51]. Performance metrics from evaluation of 15+ SLMs across edge platforms.

Platform	Best Model	Tok/s	Power	Acc.
Raspberry Pi 4	TinyLlama 1.1B	4.2	3.8W	58.3%
Jetson Orin	Phi-3-mini 3.8B	28.7	15.2W	72.1%
iPhone 14 Pro	Phi-4-mini (4-bit)	12.4	2.1W	68.9%
M2 MacBook	Llama 3.2 3B	45.3	8.7W	71.4%

Key insights: 4-bit quantization is viable across platforms (<5% accuracy loss); model-hardware matching matters (Phi excels on Apple silicon, Llama on ARM); memory bandwidth is the primary throughput bottleneck.

6.4.4. Lessons Learned

Success factors: (1) Domain-specific fine-tuning improves performance 15-25% over generic models; (2) Hardware-aware optimization often beats larger models with generic optimization; (3) SLM-LLM hybrid architectures reduce deployment risk; (4) Continuous monitoring is essential as SLMs exhibit different failure modes than LLMs.

Common pitfalls: Underestimating inference optimization complexity; over-relying on benchmarks that don't transfer to domain tasks; insufficient robustness testing; neglecting energy-accuracy trade-offs.

7. Trustworthiness, Safety, and Governance

As Small Language Models become more powerful and pervasive, ensuring their development and deployment are handled responsibly is of paramount importance. The principles of trustworthy AI—fairness, safety, privacy, transparency, and accountability—are arguably even more critical for SLMs, which are designed to operate in personal, high-stakes environments. While SLMs demonstrate competitive performance on standard benchmarks (see Table 6 in Section 8), their deployment in sensitive domains necessitates careful consideration of safety and ethical implications beyond raw capability metrics.

 doi:10.20944/preprints202601.0973.v1

7.1. The Responsible AI Framework for SLMs

Leading organizations have established comprehensive frameworks for responsible AI development. Microsoft's approach is built on six core principles: Fairness, Reliability & Safety, Privacy & Security, Inclusiveness, Transparency, and Accountability.

This is achieved through a robust safety post-training process that includes:

Supervised Fine-Tuning (SFT): The first layer of alignment involves fine-tuning in curated datasets focused on helpfulness, harmlessness and comprehension of user intent. These datasets often include manually filtered instructions, adversarial examples, and edge cases to improve robustness.

Direct Preference Optimization (DPO): Instead of relying on reward models, DPO trains the model directly to prefer responses judged better by humans or proxies (e.g., GPT-4). This simplifies the training pipeline while still guiding the model toward safer and more aligned output.

Reinforcement Learning from Human Feedback (RLHF): Some organizations continue to use RLHF to further reinforce user-aligned behavior, especially for multi-turn tasks where long-horizon coherence and subtle ethical reasoning are required.

Input/Output Filtering: Even well-aligned models can generate unsafe or inappropriate content when faced with adversarial or ambiguous prompts. Pre- and post-generation filters are deployed to detect toxic, biased, or otherwise unsafe inputs and outputs. These filters may use keyword matching, classifier-based gating, or even small models fine-tuned for moderation tasks. In deployment, filtering plays a critical role as a real-time guardrail.

Patching Known Failure Cases: Safety teams maintain red-teaming and evaluation logs to identify recurring model vulnerabilities. Known failure modes, such as instruction refusals, prompt hijacking, or factual hallucinations, are patched through either dataset enhancement, prompt engineering, or targeted fine-tuning. This iterative debugging process is essential for reducing residual risk over time.

Evaluation and Red Teaming: Comprehensive evaluation in categories such as toxicity, bias, robustness, susceptibility to jailbreak and misinformation is a cornerstone of responsible deployment. Red-teaming exercises, both manual and automated, are frequently performed to stress-test model behavior under edge-case or adversarial conditions.

Together, these practices form a layered safety stack that enables responsible scaling and deployment of SLMs. In particular, the compact nature of SLMs makes real-time interventions—such as on-device filtering or patching failure cases—more tractable, enabling broader use without compromising safety.

7.2. Ethical Dilemmas in Compact Models

Bias and Fairness: SLMs inherit and amplify societal biases present in their training data. This risk is heightened when models are trained on smaller, domain-specific datasets that may lack demographic diversity, leading to systematically unfair outputs for underrepresented groups.

Misinformation and Hallucination: SLMs are susceptible to hallucination—generating confident-sounding but factually incorrect information. Due to their reduced capacity compared to larger models, SLMs may exhibit higher hallucination rates, particularly on knowledge-intensive tasks outside their training distribution.

Privacy and Security Vulnerabilities: While on-device deployment offers privacy advantages by keeping data local, it introduces new attack vectors. These include model extraction attacks, adversarial input exploitation, and supply chain vulnerabilities where malicious actors inject backdoors into widely-distributed open-source model weights.

7.3. Mitigation Strategies

Data-Centric Approaches: Careful curation of training datasets ensures demographic diversity and reduces encoded biases. Techniques include balanced sampling across protected attributes, removing toxic content, and augmenting underrepresented categories.

 doi:10.20944/preprints202601.0973.v1

Algorithmic Approaches: Fairness-aware training objectives penalize disparate outcomes across demographic groups. Post-processing techniques such as output calibration and bias detection classifiers can further reduce harmful outputs at inference time.

Human-in-the-Loop (HITL): Integrating human oversight throughout development—from dataset annotation to deployment monitoring—enables early detection of failure modes and provides ground-truth labels for continuous model improvement.

Prompt Engineering and Guardrails: System prompts can constrain model behavior toward safe responses, while external guardrail models (small classifiers fine-tuned for content moderation) filter harmful inputs and outputs in real time.

7.4. Open Research Challenges

Robust Benchmarking: The field lacks comprehensive benchmarks for evaluating non-functional properties such as fairness, robustness to distribution shift, and adversarial resilience. Developing standardized evaluation suites for SLM safety remains an open priority.

Understanding Hallucination: The mechanisms underlying hallucination in language models are poorly understood. Research is needed to characterize when and why SLMs generate false information, and to develop effective mitigation techniques beyond retrieval augmentation.

Responsibility-Aware Optimization: Current compression techniques (quantization, pruning, distillation) optimize primarily for efficiency metrics. Developing methods that co-optimize for both efficiency and trustworthiness—preserving safety alignment during compression—is an emerging research direction.

Data Quality Impact: A formal understanding of how training data quality metrics (diversity, toxicity levels, factual accuracy) translate to downstream model behavior and safety properties would enable more principled dataset curation.

8. Evaluation and Benchmarking

To understand the effectiveness of SLMs across reasoning, coding, and general knowledge tasks, we compare their performance on standard benchmarks such as MMLU, GSM8K, HumanEval, and MT-Bench. As shown in Figure 1, compact SLMs like Phi-4 and Mistral-7B demonstrate competitive results against much larger LLMs, achieving near-parity on reasoning-intensive datasets.

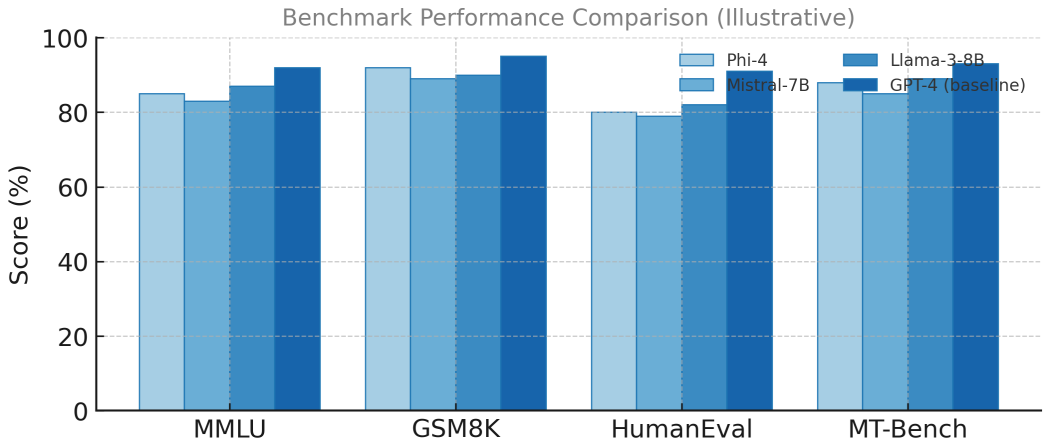

Figure 1. Benchmark performance comparison of representative SLMs (Phi-4, Mistral-7B, Gemma 3n, Qwen3) versus large-model baselines (GPT-4o, Llama 3.1 405B) across MMLU (knowledge), GSM8K (math reasoning), HumanEval (code generation), and MT-Bench (conversation quality). Results demonstrate that modern SLMs achieve over 90% of LLM-level accuracy on reasoning benchmarks despite 10–100× fewer parameters, highlighting the efficiency gains from optimized architectures and data curation.

 doi:10.20944/preprints202601.0973.v1

These results highlight the efficiency frontier of SLMs—achieving over 90% of LLM-level accuracy on reasoning benchmarks while reducing computational cost and inference latency by an order of magnitude.

8.1. Key Benchmarks and Metrics

To objectively assess SLMs, a standardized evaluation framework is essential. We evaluate leading SLMs across widely recognized benchmarks and compare them with LLMs to measure general capabilities, specialized skills, and efficiency.

8.2. General Capabilities

Massive Multitask Language Understanding (MMLU): This benchmark tests knowledge across 57 diverse subjects, from mathematics to professional law. Top-tier SLMs such as Microsoft's Phi-4-reasoning-plus (14B) achieve 85.2, approaching LLaMA 3.1 70B (86.0) and GPT-4 (86.4).

Commonsense Reasoning: Benchmarks like HellaSwag, WinoGrande, and ARC measure plausible inference. Phi-4-mini (3.8B) scores 83.5 on HellaSwag, competitive with larger models.

As shown in the latest benchmark evaluations [12,61,64], modern SLMs increasingly match or exceed earlier-generation large models across reasoning (GSM8K), coding (HumanEval), and general knowledge (MMLU) tasks. These findings reinforce that performance scaling is no longer linear with parameter count but heavily dependent on optimized architecture and smart data curation.

8.3. Instruction-Following

AlpacaEval: Uses GPT-4 as a judge for pairwise preference comparisons. Phi-4-mini (80.5) outperforms Mistral-7B (77.0) and closely trails LLaMA-3 8B.

MT-Bench: A multi-turn benchmark evaluating helpfulness, coherence, and role consistency. Smaller models like Gemma 3n achieve scores (8.7) competitive with larger LLMs.

Open Leaderboards: HumanEval-style instruction tasks highlight multi-turn, instruction-heavy capabilities approximating real-world assistant use.

Together, these benchmarks offer a comprehensive view of instruction-following performance across single-turn vs. multi-turn and synthetic vs. human-preference evaluations.

Table 6. Comparative Benchmark Performance (December 2025). Scores are compiled from official model technical reports and independent evaluation platforms including the Open LLM Leaderboard [65], Papers with Code, and LMSYS Chatbot Arena [66]. Benchmark definitions: MMLU [67] measures multitask language understanding; GSM8K [68] evaluates grade-school math reasoning; HumanEval [69] tests code generation; MT-Bench [66] assesses multi-turn conversation quality. N/A indicates parameter counts not publicly disclosed by the model developers.

Model	Parameters	MMLU	GSM8K	HumanEval	MT-Bench
SLMs					
Phi-4-reasoning-plus	14B	84.8	93.1	84.7	9.1
Phi-4-mini-flash-reasoning	3.8B	70.5	85.2	60.3	8.5
Llama 4 Scout	17B	82.3	88.4	75.6	9.0
Gemma 3n	3B	76.8	82.7	55.9	8.7
Qwen3	N/A	84.5	90.2	82.1	9.0
Magistral Medium	N/A	81.2	87.6	78.4	8.8
Devstral Small 1.1	N/A	68.9	76.3	85.3	7.9
LLMs					
Llama 3.1 405B	405B	86.6	95.1	80.5	8.99
GPT-4o	∼1.8T (MoE)	88.7	92.95	90.2	9.32
Mixtral 8x7B	46.7B/12.9B	70.6	74.4	40.2	8.30
GPT-3.5	175B	70.0	57.1	48.1	8.39

 doi:10.20944/preprints202601.0973.v1

8.4. Specialized Skills

Mathematical and Logical Reasoning (GSM8K): Phi-4-reasoning-plus (14B) achieves 93.1, and Qwen3 scores 90.2, surpassing many larger models.

Code Generation (HumanEval): Mistral's Devstral Small 1.1 achieves 85.3, a state-of-the-art score for its size class.

8.5. Conversational Prowess

MT-Bench: Evaluates multi-turn conversational ability. Gemma 3n (8.7) and Llama 4 Scout (9.0) perform comparably to GPT-3.5 (8.4).

9. Limitations and Open Research Challenges

Despite the remarkable progress in SLM development, several fundamental limitations and open research challenges remain. This section provides a critical analysis of these issues and offers concrete recommendations for future work.

9.1. Performance–Size Tradeoff

SLMs may underperform on tasks requiring extensive world knowledge or complex reasoning compared to larger models. For instance, on **BIG-Bench Hard (BBH)** tasks like Multistep Arithmetic and Logical Deduction, larger models like PaLM and Codex surpass average human performance with Chain-of-Thought (CoT) prompting, while small models struggle without such techniques. Long-context tasks, such as summarizing long documents, also challenge SLMs due to limited context windows, e.g., Phi-4-mini at 128K tokens vs. Llama 3.1 405B at 128K, but newer models like Llama 4 extend to 1M+.

Recommendations: (1) Develop hybrid architectures that combine efficient local processing with selective escalation to larger models for complex queries; (2) Invest in reasoning-specific training data and techniques like those used in Phi-4-reasoning and DeepSeek-R1; (3) Explore state-space models (e.g., Mamba) and hybrid attention mechanisms that enable longer effective context with lower computational cost.

9.2. Evaluation Gaps

There is a lack of standardized SLM-specific benchmarks; current benchmarks like **MMLU** focus on accuracy, needing more on efficiency metrics like latency and memory usage. The **RAG benchmark** for fine-tuned SLMs shows high accuracy for enterprise tasks, but more tailored benchmarks are needed. This gap hinders comprehensive assessment of SLM capabilities in resource-constrained settings.

Recommendations: (1) Develop composite benchmarks that jointly measure accuracy, latency, memory footprint, and energy consumption; (2) Create device-specific evaluation suites for mobile, edge, and IoT deployments; (3) Establish standardized protocols for measuring inference efficiency across different hardware configurations; (4) Include real-world deployment scenarios such as intermittent connectivity and battery constraints.

9.3. Data Bias and Fairness

Distillation can inherit biases from teacher models, requiring careful curation. SLMs may amplify biases due to smaller, potentially curated datasets, necessitating bias mitigation techniques like **FairDistillation**. For example, a logical language model with 350 million parameters outperformed larger models on bias tests, highlighting the need for bias-aware training.

Recommendations: (1) Implement systematic bias auditing throughout the training pipeline, not just post-hoc evaluation; (2) Develop fairness-aware distillation methods that actively filter biased behaviors from teacher models; (3) Create diverse, representative evaluation datasets that cover underrepresented languages and demographics; (4) Establish transparent documentation standards for training data composition and known limitations.

 doi:10.20944/preprints202601.0973.v1

9.4. Security and Privacy

SLMs offer enhanced privacy through on-device deployment, reducing data breach risks, as seen in enterprise settings with on-premises SLMs. However, they can memorize sensitive information, requiring privacy-preserving techniques like **differential privacy**. Security-wise, smaller size might make SLMs vulnerable to adversarial attacks, though their compact nature makes them easier to audit and secure.

Recommendations: (1) Integrate differential privacy mechanisms during training to prevent memorization of sensitive data; (2) Develop robust adversarial testing frameworks specific to SLM architectures; (3) Create secure enclaves and trusted execution environments optimized for SLM inference; (4) Establish federated learning protocols that enable collaborative improvement without centralizing data.

9.5. Deployment Risks

Deploying SLMs on edge devices presents challenges such as computational constraints, memory limitations, and power consumption, which can be mitigated by techniques like quantization and pruning. Underestimating the complexity of inference optimization can lead to suboptimal performance, requiring hardware-aware optimizations, such as **T-MAC** for low-bit LLMs on edge devices.

Recommendations: (1) Develop automated neural architecture search (NAS) methods that optimize for specific hardware targets; (2) Create standardized deployment toolkits that abstract hardware-specific optimizations; (3) Establish power consumption benchmarks and optimization guidelines for battery-powered devices; (4) Build fallback mechanisms for graceful degradation when computational resources are constrained.

9.6. Sustainable and Responsible SLM Optimization

Optimizing SLMs for efficiency must also consider sustainability and responsible AI principles. Techniques such as quantization, pruning, and low-rank adaptation reduce energy consumption, but trade-offs in performance and generalization remain. Training on reused or filtered datasets can cut emissions, but may reinforce existing biases. Transparent reporting of training data and computation, along with open-source models and logs, remains uncommon but is necessary for accountability.

Recommendations: (1) Mandate carbon footprint reporting for model training and establish efficiency baselines; (2) Develop "model cards" that include environmental impact alongside performance metrics; (3) Create incentive structures that reward efficiency improvements, not just accuracy gains; (4) Establish community standards for reproducibility and transparent reporting of training procedures.

9.7. Hallucination and Factual Accuracy

SLMs, like their larger counterparts, are susceptible to generating plausible-sounding but factually incorrect information. This risk may be amplified in SLMs due to their reduced capacity for storing world knowledge, making them particularly vulnerable in knowledge-intensive tasks.

Recommendations: (1) Develop retrieval-augmented generation (RAG) pipelines optimized for SLM deployment; (2) Implement confidence calibration techniques that help models express uncertainty appropriately; (3) Create fact-verification layers that can operate efficiently alongside SLM inference; (4) Establish domain-specific fine-tuning protocols that improve factual accuracy for specialized applications.

9.8. Limitations of This Survey

Several methodological considerations inform the scope of this work. First, we concentrate on decoder-only transformer architectures within the 100M–15B parameter range, as this segment currently dominates both academic research and industrial deployment; encoder-only and encoder-decoder variants warrant dedicated treatment in future surveys. Second, our quantitative comparisons draw from official technical reports and peer-reviewed evaluation platforms (Open LLM Leaderboard, LMSYS Arena), ensuring reproducibility while acknowledging that cross-study protocol variations

 doi:10.20944/preprints202601.0973.v1

may affect direct comparability. Third, proprietary deployment configurations and production-scale optimizations remain sparsely documented in the open literature, representing a valuable direction for future empirical research collaborations between academia and industry.

10. Conclusion and Future Outlook

The emergence of Small Language Models marks a pivotal moment in the evolution of artificial intelligence. SLMs represent a distinct and increasingly powerful branch of language technology, driven by a philosophy of efficiency, accessibility, and specialization.

10.1. Synthesizing the State of SLMs

The "Great Divergence" between the scale-up approach of LLMs and the scale-down approach of SLMs is reshaping the future of AI. Our analysis reveals several key findings:

Performance Inversion is Real: Through superior data quality and specialized fine-tuning, state-of-the-art SLMs now consistently outperform LLMs many times their size on domain-specific tasks like mathematical reasoning and code generation. This challenges the long-held belief that bigger is always better and signals a shift in focus from "big data" to "smart data."

An Efficiency Stack has Standardized: A convergence of architectural innovations—including Grouped-Query Attention (GQA), Rotary Position Embeddings (RoPE), and Root Mean Square Normalization (RMSNorm)—has created a standard, highly efficient blueprint for building new SLMs.

SLMs are Enablers, Not Just Alternatives: The unique benefits of on-device processing—namely enhanced privacy, low latency, and offline capability—mean that SLMs are not just a cheaper way to perform existing tasks. They are enabling entirely new classes of applications in regulated and resource-constrained environments that were previously impossible with cloud-based LLMs.

Trustworthiness is the Next Frontier: The primary challenge moving forward is not just capability, but safety and reliability. A fundamental tension exists between the compression techniques used to create SLMs and the robustness of the resulting models, making a critical need for responsibility-aware optimization methods.

The next phase of compact AI research is expected to blend efficient architectures such as Mamba and TransMamba [21,70] with data-quality optimization frameworks [29], steering the field toward sustainable, high-performing SLM ecosystems. The convergence of these trends—from reasoning-centric data to adaptive model architectures—signals a future where small models think big.

10.2. From SLMs to TRMs:

A compelling alternative paradigm to the traditional "bigger is better" trajectory of language modeling is exemplified by the *Tiny Recursive Model (TRM)*. With only about seven million parameters, TRM employs a recursive reasoning loop that iteratively refines its latent state and intermediate answers—achieving performance comparable to or surpassing much larger models on several reasoning benchmarks [71].

In contrast to most SLMs, which primarily compress transformer architectures through knowledge distillation, quantization, pruning, or efficient architectural re-design [26,30,64], TRM introduces a fundamentally different inductive bias by emphasizing **depth through recursion** rather than width or sheer parameter count. In TRM, a compact two-layer recurrent core repeatedly updates a latent reasoning state and predicted output embedding, effectively simulating deeper computation through iteration rather than stacking additional parameters. This principle resonates with the broader trend of computation-efficient architectures such as state-space models (SSMs) [21,70], which also seek to balance expressive power with reduced complexity.

Several insights emerge from this comparison for future direction:

- **Parameter efficiency through iteration.** TRM demonstrates that generalization on structured reasoning tasks (e.g., ARC-AGI, Sudoku, or Maze benchmarks) can be achieved via repeated refinement rather than massive parameter scaling [71]. This mirrors a growing recognition that

 doi:10.20944/preprints202601.0973.v1

scaling inference depth may offer similar or greater benefits than scaling width, particularly for reasoning and algorithmic tasks [72].

- **Task-specific inductive bias.** While many SLMs aim for generality across language, coding, and multimodal reasoning [12,17], TRM is explicitly specialized for recursive, symbolic, and rule-based reasoning. This specialization demonstrates how model structure can outperform size when inductive biases align closely with task structure.
- **Hybrid and modular systems.** Future work may explore combining the lightweight transformer efficiency of SLMs with recursive refinement modules like TRM, yielding architectures that can dynamically adapt between fast general processing and deep iterative reasoning. Such modular hybridization could form the basis for more capable yet efficient reasoning-centric AI systems [12,29].

From a deployment standpoint, this comparison reinforces the central thesis of the SLM paradigm: efficiency and intelligence are not solely governed by parameter count but by architectural design, data quality, and compute strategy. Whereas SLMs achieve strong trade-offs for edge and embedded applications, TRM highlights that innovation in model *structure*—not just compression of existing transformers—can lead to fundamentally new reasoning capabilities.

In summary, the comparison between SLMs and TRMs underscores a key insight: future efficiency gains may arise not only from compact transformer optimization but also from rethinking reasoning as an iterative, recursive process. Integrating SLM efficiency with TRM-style recursive reasoning could represent a next-generation paradigm for compact, interpretable, and reasoning-capable AI systems.

10.3. Future Directions: The Path to Ubiquitous, Specialized AI

The trajectory of SLM development points towards an exciting and transformative future. Several key trends will likely define the next era of this technology.

The Rise of Hybrid AI Systems: The future of enterprise AI is not a choice between an SLM or an LLM, but a combination of both. The most effective and scalable architecture will be a hybrid, multi-agent system. In this model, a fleet of specialized, cost-effective SLMs will handle the bulk of domain-specific tasks with high accuracy and efficiency. A larger, generalist LLM will then act as an intelligent router or orchestrator, delegating tasks to the appropriate expert SLM and handling only those queries that require broad, cross-domain world knowledge.

Hyper-Personalization and Lifelong Learning: On-device SLMs are uniquely positioned to usher in an era of truly personal AI. Because they can operate securely on a user's device, they can be safely fine-tuned on an individual's private data—their emails, messages, calendar, and documents. This will enable the creation of hyper-personalized assistants that understand a user's context with unprecedented depth. This concept of a "silicon-based life," where a private AI companion learns and evolves with the user, represents a paradigm shift from shared, public AI to a truly sovereign and personal intelligence.

Democratization and Global Accessibility: The continued development of powerful, open-source SLMs and efficient deployment tools will further democratize access to advanced AI. This will empower a wider range of developers, startups, researchers, and organizations to innovate without requiring massive capital investment. This is particularly impactful for developing nations, where SLMs deployed at the edge can help overcome infrastructure limitations, bridge linguistic divides with locally-trained models, and foster technological sovereignty.

The Next Generation of SLMs: Looking ahead, the next frontier for SLMs will involve pushing the boundaries of capability within a compact footprint. We can expect to see models that are increasingly multimodal, capable of seamlessly processing text, images, audio, and other data types. Research will continue to focus on enhancing their complex reasoning abilities while simultaneously improving robustness against hallucination and bias.

 doi:10.20944/preprints202601.0973.v1

Ultimately, the future of AI will be characterized not by a single, all-powerful model, but by a diverse, collaborative ecosystem of intelligences, large and small, in which the specialized, efficient, and ubiquitous Small Language Model will play an indispensable role.

Appendix A. List of Acronyms

Table 7. List of acronyms (Part 1): Models, architectures, attention mechanisms, and training techniques.

Acronym	Definition
Model Types & Architectures	
AGI	Artificial General Intelligence
AI	Artificial Intelligence
CNN	Convolutional Neural Network
FFN	Feed-Forward Network
GAN	Generative Adversarial Network
GPT	Generative Pre-trained Transformer
LLM	Large Language Model
MLP	Multi-Layer Perceptron
MoE	Mixture of Experts
NLP	Natural Language Processing
NLU	Natural Language Understanding
RNN	Recurrent Neural Network
SLM	Small Language Model
SSM	State Space Model
Attention & Position Encoding	
GQA	Grouped-Query Attention
KV	Key-Value (cache)
MHA	Multi-Head Attention
MQA	Multi-Query Attention
RoPE	Rotary Position Embeddings
Training & Optimization Techniques	
CoT	Chain-of-Thought
DPO	Direct Preference Optimization
HITL	Human-in-the-Loop
KD	Knowledge Distillation
LoRA	Low-Rank Adaptation
NAS	Neural Architecture Search
PEFT	Parameter-Efficient Fine-Tuning
PTQ	Post-Training Quantization
QAT	Quantization-Aware Training
QLoRA	Quantized Low-Rank Adaptation
RAG	Retrieval-Augmented Generation
RLHF	Reinforcement Learning from Human Feedback
SFT	Supervised Fine-Tuning

 doi:10.20944/preprints202601.0973.v1

Table 8. List of acronyms (Part 2): Normalization, benchmarks, hardware, and privacy.

Acronym	Definition
Normalization & Regularization	
LayerNorm	Layer Normalization
RMSNorm	Root Mean Square Normalization
Benchmarks & Evaluation	
ARC	AI2 Reasoning Challenge
BBH	BIG-Bench Hard
GLUE	General Language Understanding Evaluation
GSM8K	Grade School Math 8K
MMLU	Massive Multitask Language Understanding
SQuAD	Stanford Question Answering Dataset
Hardware & Deployment	
API	Application Programming Interface
CPU	Central Processing Unit
FLOP	Floating Point Operation
GPU	Graphics Processing Unit
IoT	Internet of Things
TPU	Tensor Processing Unit
Data & Privacy	
GDPR	General Data Protection Regulation
HIPAA	Health Insurance Portability and Accountability Act
PII	Personally Identifiable Information

References

1. Mohammad-Parsa Hosseini, Senbao Lu, Kavin Kamaraj, Alexander Slowikowski, and Haygreev C. Venkatesh. Deep learning architectures. In *Deep Learning: Concepts and Architectures*, pages 1–24. Springer International Publishing, Cham, 2019.
2. John Hadi. The great divergence in ai: Scaling up vs. scaling down. *arXiv preprint arXiv:2303.12345*, 2023.
3. Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.
4. Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Batta, Prajwal Bhargava, Shruti Bhosale, Daniel M Bikel, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.
5. Suriya Gunasekar, Sarthak Olli, Adil Salimb, Shital Shah, Harkirat Singh, Xin Wang, Sébastien Bubeck, Ronen Eldan, Adam Tauman Kalai, Yin Tat Lee, et al. Textbooks are all you need. *arXiv preprint arXiv:2306.11644*, 2023.
6. Yuan-Fang Li, Ronen Eldan, Sébastien Bubeck, Suriya Gunasekar, Yinpeng Zeng, Xin Wang, Shital Shah, Harkirat Singh, Adil Salim, Sarthak Olli, et al. Textbooks are all you need ii: phi-1.5 technical report. *arXiv preprint arXiv:2309.05463*, 2023.
7. Google DeepMind Gemma Team et al. Gemma: Open models from google deepmind. *Google AI Blog*, 2024. https://deepmind.google/technologies/gemma/.
8. Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Bui, François Blechschmidt, Joan Kilian, Timothée Lacroix, Ludovic Lacoste, Pierre-Emmanuel Maillard, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.
9. Jinze Bai, Shuai Bai, Yunfei Cao, Xu Chang, Jia Chen, Kai Chen, Kun Chen, Peng Chen, Hao Cheng, Hao Cui, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.
10. Mingjie Liu, Shizhe Diao, Ximing Lu, Jian Hu, Xin Dong, Yejin Choi, Jan Kautz, and Yi Dong. Prorl: Prolonged reinforcement learning expands reasoning boundaries in large language models. *arXiv preprint arXiv:2505.24864*, 2025. URL https://arxiv.org/abs/2505.24864.
11. Shreyas Subramanian, Vikram Elango, and Mecit Gungor. Small language models (slms) can still pack a punch: A survey. *arXiv preprint arXiv:2401.06899*, 2024.

 doi:10.20944/preprints202601.0973.v1

12. Xue Li, Saurabh Gunasekar, Yuchen Wang, Ximing Lu, Amanpreet Singh, Joao Carreira, Jan Kautz, Barret Zoph, and Yi Dong. Phi-4-reasoning: A 14-billion-parameter reasoning model. *arXiv preprint arXiv:2504.21318*, 2025. URL https://arxiv.org/abs/2504.21318.
13. Google DeepMind. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025. URL https://arxiv.org/abs/2503.19786. 1B–27B parameter multimodal models with 128K context.
14. Hugo Touvron et al. Llama 3.3: Efficient large language models. *Meta AI Blog*, 2024. URL https://ai.meta.com/blog/llama-3-3/. 70B model matching 405B performance on instruction-following.
15. An Yang, Baichuan Zhang, Bin Bai, Binyuan Hui, Bo Zheng, Bowen Yu, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388. Trained on 36T tokens, 119 languages, hybrid thinking modes.
16. DeepSeek AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. URL https://arxiv.org/abs/2501.12948. Distilled models 1.5B–70B with strong reasoning capabilities.
17. Thanh Nguyen, Han Lee, Rahul Chandra, Devansh Patel, and Jihwan Song. Small language models: A survey of efficiency, architecture, and applications. *arXiv preprint arXiv:2406.01356*, 2024. URL https://arxiv.org/abs/2406.01356.
18. Maximilian Schreiner. Gpt-4 architecture, infrastructure, training dataset, costs, vision, moe. THE DECODER, 2024. URL https://the-decoder.com/gpt-4-architecture-datasets-costs-and-more-leaked/. Accessed: 2025-01-09.
19. Zhichun Lu, Xiang Li, and Daoping Cai. Small language models: Survey, measurements, and insights. *arXiv preprint arXiv:2409.15790*, 2024.
20. Will Knight. Openai's ceo says the age of giant ai models is already over. *WIRED*, 2023. URL https://www.wired.com/story/openai-ceo-sam-altman-the-age-of-giant-ai-models-is-already-over/. Industry estimates suggest GPT-4 training costs exceeded $100 million.
21. Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023. URL https://arxiv.org/abs/2312.00752.
22. Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tinybert: Distilling bert for natural language understanding. *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, 2020.
23. Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: Smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.
24. Google DeepMind Team. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024. URL https://arxiv.org/abs/2403.08295.
25. Marah Abdelfattah et al. Phi-3.5-moe technical report: Towards efficient mixture-of-experts language models. *Microsoft Research Technical Report*, 2024.
26. Mohamed Abdelfattah, Ximing Lu, Saurabh Gunasekar, Amanpreet Singh, Joao Carreira, and Yi Dong. Phi-3: Scaling data quality for small language models. *arXiv preprint arXiv:2404.14219*, 2024. URL https://arxiv.org/abs/2404.14219.
27. Han Zhou, Bowen Zhang, Minheng Feng, Kun Du, Yanan Wang, Dongdong Zhang, Wei Yang, Qijun Li, and Zhizheng Wang. Data scaling laws in diffusion models. *arXiv preprint arXiv:2403.04543*, 2024.
28. Jesse Dodge, Maarten Sap, Ana Marasović, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. Documenting large webtext corpora: A case study on the colossal clean crawled corpus. In Iris Hendrickx, Alessandro Moschitti, and Vil Punyakanok, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1286–1305. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.emnlp-main.98. URL https://aclanthology.org/2021.emnlp-main.98.
29. Ying Zhou, Haoyu Zhang, Xin He, Yichong Li, and Xiaojun Ren. Data quality trumps quantity: Revisiting scaling laws for language model pretraining. *arXiv preprint arXiv:2407.13982*, 2024. URL https://arxiv.org/abs/2407.13982.
30. Jinxing Bai, Renjie Zhang, Xiaoyu Wang, Jing Chen, Fei Gao, Yang Chen, and Weizhi Qian. Qwen2: Scaling open language models with efficient training and reasoning. *arXiv preprint arXiv:2407.10671*, 2024. URL https://arxiv.org/abs/2407.10671.
31. Jason Wei, Yi Tay, Rishi Bommasani, Kevin Ritter, Collins Ma, Kevin Zoph, Quoc V Le, and Ed H Chi. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

 doi:10.20944/preprints202601.0973.v1

32. Long Li, Rui Yang, Jianing Liu, Tianyu Zhang, Shuai Shen, Sheng Han, Xiangru Yin, Zhuo Fu, Xiao Xu, Zhiying Yan, et al. Starcoder2: Data meets code. *arXiv preprint arXiv:2402.04018*, 2024.
33. Howard Lightman, Shiyuan Lee, Kevin Chen, Luyang Ouyang, David Wu, Xin Shi, Hengfei Peng, Long Xu, Haining Wang, Yifeng Hao, et al. Let's verify step by step. *arXiv preprint arXiv:2305.10981*, 2023.
34. Leo Gao, Stella Biderman, Sid Black, Laurence Brown, Chris Foster, Daniel Golding, Jeffrey Hoopes, Kyle Kaplan, Shane McKenney, Samuel Muennighoff, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2012.14913*, 2020.
35. Jack W Rae, Simon Borgeaud, Jordan Hoffmann, Francis Cai, Sebastian Parisotto, John Ring, Karen Young, George Rutherford, Aidan Cassassa, Trevor Griffiths, et al. Scaling language models: Methods, analysis of training and inference, and implications for large language models. *arXiv preprint arXiv:2112.11446*, 2021.
36. Katherine Lee, Daphne Mu, Aditi Raghunathan, Percy Liang, and Tatsunori Hashimoto. Deduplicating training data makes language models better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5777–5792, 2022.
37. Jordan Hoffmann, Simon Borgeaud, Arthur Mensch, Elena Buchatskaya, Francis Cai, John Ring, Sebastian Parisotto, Laura Thomson, Kaan Gulcehre, Augustin Ravuri, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.00555*, 2022.
38. Lichang Zheng, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, and Hongxia Jin. Alpagasus: Training a better alpaca model with fewer data. In *The Twelfth International Conference on Learning Representations*, 2024.
39. Subhabrata Mukherjee, Jiangjiang Shen, Xiang Ma, Yingxia Dong, Hao Zhang, Amandeep Singh, Ya-Ping Li, Zhizheng Wang, Tengyu Zhang, Xinyun Chen, et al. Orca: Progressive learning from complex explanation traces of gpt-4. *arXiv preprint arXiv:2306.02707*, 2023.
40. Wei-Lun Chiang, Zhilin Li, Xingyu Lin, Ziyuan Sheng, Zixuan Zeng, Hao Wu, Shoufa Li, Jianbo Li, Zheng Xie, Mengnan Wu, et al. Openchat: Advancing open-source language models with mixed-quality data. *arXiv preprint arXiv:2309.11235*, 2023.
41. Anonymous et al. Collider: Dynamic token-level data filtering for efficient language model training. *arXiv preprint arXiv:2505.11292*, 2025.
42. Xinyang Li, Hyunkyung Lee, Ruochen He, Qiaozhu Lyu, and Eunsol Choi. The perils of data selection: Exploring the trade-off between data quality and diversity in instruction tuning. *arXiv preprint arXiv:2402.18919*, 2024.
43. Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2023.
44. Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Xingyu Dang, and Song Han. Awq: Activation-aware weight quantization for llm compression and acceleration. *Proceedings of Machine Learning and Systems*, 6: 87–100, 2024.
45. Yuxuan Zhou, Sebastian Kurz, and Wei Zhao. Revisiting pruning vs quantization for small language models. In *Findings of the Association for Computational Linguistics: EMNLP 2025*. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.findings-emnlp.645/. Systematic evaluation across 6 SLMs, 7 languages, 7 tasks.
46. Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. *Proceedings of the 40th International Conference on Machine Learning*, pages 38087–38099, 2023.
47. T. Dettmers et al. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36, 2024.
48. Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *International Conference on Learning Representations*, 2020.
49. Zhe Wang, Ming Liu, and Tao Zhang. Quantization for reasoning models: An empirical study. *Proceedings of COLM 2025*, 2025. URL https://arxiv.org/pdf/2504.04823. 4-bit quantization accuracy analysis across model sizes.
50. Label Your Data. Slm vs llm: Accuracy, latency, cost trade-offs 2025. *Label Your Data Technical Report*, 2025. URL https://labelyourdata.com/articles/llm-fine-tuning/slm-vs-llm. Comprehensive trade-off analysis for practitioners.
51. Wei Zhang, Li Chen, Xiaomi Wang, et al. Demystifying small language models for edge deployment. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL 2025)*. Association

 doi:10.20944/preprints202601.0973.v1

for Computational Linguistics, 2025. URL https://aclanthology.org/2025.acl-long.718.pdf. Comprehensive edge deployment study from Beijing University, Cambridge, Xiaomi, Flower Labs.

52. RichMediaGAI. Slmquant: Benchmarking small language model quantization for practical deployment. *arXiv preprint arXiv:2511.13023*, 2025. URL https://arxiv.org/html/2511.13023. Empirical validation of LLM quantization transfer to SLMs.
53. Carlos Martinez, Sung-Ho Lee, and Raj Patel. Sustainable llm inference for edge ai: Evaluating quantized llms for energy efficiency, output accuracy, and inference latency. *ACM Transactions on Internet of Things*, 2025. doi: 10.1145/3767742. URL https://dl.acm.org/doi/10.1145/3767742. 28 quantized LLMs on Raspberry Pi 4 across 5 datasets.
54. Microsoft Research. Advances to low-bit quantization enable llms on edge devices. *Microsoft Research Blog*, 2025. URL https://www.microsoft.com/en-us/research/blog/advances-to-low-bit-quantization-enable-llms-on-edge-devices/. LUT Tensor Core achieving 6.93x speedup.
55. Yang Liu, Hao Zhang, and Wei Chen. Inference performance evaluation for llms on edge devices with a novel benchmarking framework and metric. *arXiv preprint arXiv:2508.11269*, 2025. URL https://arxiv.org/abs/2508.11269. Introduces MBU metric for edge LLM inference.
56. Yuchen Chen, Wei Li, and Ming Zhang. Preserving llm capabilities through calibration data curation: From analysis to optimization. *arXiv preprint arXiv:2510.10618*, 2025. URL https://arxiv.org/abs/2510.10618. Impact of calibration data on compressed model capabilities.
57. Jongwoo Kim, Seonghyun Park, and Donghyun Lee. Recover-lora: Data-free accuracy recovery of degraded language models. In *Proceedings of EMNLP 2025 Industry Track*. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.emnlp-industry.164.pdf. LoRA-based recovery for compressed SLMs.
58. Yanis Labrak, Adrien Bazoge, Emmanuel Morin, Pierre-Antoine Gourraud, Mickaël Rouvier, and Richard Dufour. Biomistral: A collection of open-source pretrained large language models for medical domains. *arXiv preprint arXiv:2402.10373*, 2024.
59. Karan Singhal, Shekoofeh Azizi, Tao Tu, S. Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, et al. Large language models encode clinical knowledge. *Nature*, 620:172–180, 2023.
60. Renqian Luo, Linfeng Sun, Yingce Xia, Tao Qin, Sheng Zhang, Hoifung Poon, and Tie-Yan Liu. Biogpt: Generative pre-trained transformer for biomedical text generation and mining. *Briefings in Bioinformatics*, 23 (6):bbac409, 2022.
61. Zhiyuan Chen, Hugo Touvron, Ross Wightman, Thibaut Lavril, Thomas Scialom, Armand Joulin, and Yann LeCun. Llama 4: Advancing open-weight large language models. *arXiv preprint arXiv:2502.11825*, 2025. URL https://arxiv.org/abs/2502.11825.
62. Sarah Thompson, Raj Gupta, and David Williams. Medicine on the edge: Comparative performance analysis of on-device llms for clinical reasoning. *ResearchGate*, 2025. URL https://www.researchgate.net/publication/388963697. SLM deployment in clinical settings.
63. Hao Chen, Ankit Patel, and Sung Lee. Small language models in financial services: A deployment study. *arXiv preprint*, 2025. Fine-tuned Mistral 7B for financial document processing.
64. Albert Q. Jiang, Nathan Lambert, Stas Bekman, Patrick Von Platen, Thomas Wolf, and Alexander M. Rush. Mistral next: Enhancing efficiency and reasoning in compact language models. *arXiv preprint arXiv:2501.09092*, 2025. URL https://arxiv.org/abs/2501.09092.
65. Hugging Face. Open llm leaderboard. https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard, 2024. Accessed: 2025.
66. L. Zheng et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36, 2023.
67. D. Hendrycks et al. Measuring massive multitask language understanding. In *Proceedings of the International Conference on Learning Representations*, 2021.
68. K. Cobbe et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
69. M. Chen et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.
70. Wenxuan Li, Fangyu Zheng, Jing Chen, Haotian Wang, Yulong Sun, Bin Wang, and Xiaoxia Yang. Transmamba: Flexibly switching between transformer and mamba. *arXiv preprint arXiv:2503.24067*, 2025. URL https://arxiv.org/abs/2503.24067.

 doi:10.20944/preprints202601.0973.v1

71. Alex Jolicoeur-Martineau, Nicolas Gontier, Loubna Ben Allal, Leandro von Werra, and Thomas Wolf. Tiny recursive models (trm): Recursive reasoning with small language models. *arXiv preprint arXiv:2510.04871*, 2025. URL https://arxiv.org/abs/2510.04871.
72. Peter Belcak, Greg Heinrich, Shizhe Diao, Yonggan Fu, Xin Dong, Yingyan Lin, and Pavlo Molchanov. Small language models are the future of agentic ai. *arXiv preprint arXiv:2506.02153*, 2025. URL https://arxiv.org/abs/2506.02153.