# Demystifying Small Language Models for Edge Deployment

**Zhenyan Lu[†1,2], Xiang Li[†1], Dongqi Cai[1,3], Rongjie Yi[1], Fangming Liu[2], Wei Liu[4], Jian Luan[4], Xiwen Zhang[5], Nicholas D. Lane[3,6], Mengwei Xu[1],**

[1]Beijing University of Posts and Telecommunications, [2]Peng Cheng Laboratory, [3]University of Cambridge, [4]MiLM Plus, Xiaomi Inc., [5]Helixon Research, [6]Flower Labs

**Correspondence:** mwx@bupt.edu.cn

## Abstract

Small language models (SLMs) have emerged as a promising solution for deploying resource-constrained devices, such as smartphones and Web of Things. This work presents the first comprehensive study of over 60 SLMs such as Microsoft Phi and Google Gemma that are publicly accessible. Our findings show that state-of-the-art SLMs outperform 7B models in general tasks, proving their practical viability. However, SLMs' in-context learning capabilities remain limited, and their efficiency has significant optimization potential. We identify key SLM optimization opportunities, including dynamic task-specific routing, model-hardware co-design, and vocabulary/KV cache compression. Overall, we expect the work to reveal an all-sided landscape of SLMs, benefiting the research community across algorithm, model, system, and hardware levels.

## 1 Introduction

The evolution of language models is diverging. On one hand, in the pursuit of artificial general intelligence, increasingly large language models (LLM) have been born in datacenters that host hundreds of thousands of GPUs (Kaplan et al., 2020; Xu et al., 2024c). The aim of this path is to demonstrate that machines can solve the most challenging language tasks, with the ultimate goal of advancing human civilization by pushing the boundaries of science and technology. On the other hand, there is a growing focus on **small language models (SLM)**, designed for resource-efficient and ubiquitous deployment on scenarios such as mobile devices and robotics. The vision behind SLMs is to democratize access to machine intelligence, making it both accessible and affordable to a wider range of users. This approach seeks to make intelligence ubiquitous and practical, available to anyone, anywhere, at any time – much like the human brain, which everyone possesses.

Both LLM and SLM are important in reshaping our daily lives, yet the latter receives significantly less attention in academia. There has been very few literature exploring SLM capabilities (Lepagnol et al., 2024; Schick and Schütze, 2020; Zhou et al., 2023) or their runtime cost on devices (Li et al., 2024b; Laskaridis et al.; Xu et al., 2024b), often with limited scale or depth. In industrial, however, SLMs have already been integrated into commercial off-the-shelf (COTS) devices on a massive scale (Yuan et al., 2023; Dubiel et al., 2024). For instance, almost every popular browser has built-in access to local SLM capability, including Google Chrome (chr, 2024), Microsoft Edge (edg, 2024) and Opera (ope, 2024). At system level, the latest Google/Samsung smartphones have built-in LLM services (Gemini Nano), allowing third-party mobile apps to leverage LLM capabilities through prompts and LoRA modules (goo, 2024).

This work presents the first in-depth, systematic investigation of SLMs, thoroughly discussing their capabilities and runtime performance. The scope of this work is limited to those language models with 100M–5B parameters in decoder-only transformer architecture for their wide deployment on edge devices, which covers the range from low-end WoT/wearable gadgets like smartwatches to high-end mobile devices such as smartphones and tablets. In total, we collected 68 popular SLMs released by 24 organizations, spanning from OPT (2022.05) to Llama3.2 (2024.09). The details of those models are shown in Table 1.

We then built up an end-to-end benchmark that comprehensively evaluates the model capabilities (mainly commonsense reasoning and in-context learning) through 10 widely used datasets, as well as their runtime costs (prefill and decode speed, memory footprint, etc.) on two real development edge boards. Through such investigation, we try to

[†]Zhenyan Lu and Xiang Li contributed equally to this work.



*Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14747–14764
July 27 - August 1, 2025 

answer the following crucial questions concerning SLMs: "Can SLMs catch up to LLMs in terms of intelligence?" "What datasets are more likely to produce a highly capable SLM?" "How different SLM architecture (e.g., depth, width, atten type) and the deployment environments (quantization algorithms, hardware type, etc) impact runtime performance?"

Based on the benchmarking results, we obtain valuable insights of SLMs. Here, we summarize a few of them. (1) SLMs capabilities are fast evolving, closing the gap with Llama-7B/8B series (§2.3). (2) Not all SLMs benefit from in-context learning (§2.3). (3) SLMs are typically trained on way more tokens than what Chinchilla recommends ("over trained") (§2.4). (4) Dataset quality is more crucial than dataset size (§2.4). (5) Model architecture has non-trivial impacts on inference speed (§3.1). (6) Quantization gains diminish in long context (§3.2). We will explain how these insights create opportunities for the advancement of SLMs. Overall, our findings paint a promising picture of on-device ubiquitous SLMs, yet also highlight the challenges towards resource efficiency. Specifically, the findings show strong implications for multiple stakeholders of the SLM development pipeline. We expect the work to reveal an all-sided landscape of SLM and benefit the research community, including those working on the algorithm, model, system, and hardware levels.

In summary, we make the following contributions in this work.

- We exhaustively review 68 small language models released in recent years, benchmark their capability as well as on-device cost.
- To this end, we have developed the first mobile SLM evaluation suite, which handles downloading, quantizing, deploying, and measuring the performance of SLMs across heterogeneous edge devices. A leaderboard website is created as well to advance and facilitate the SLM research.
- Through such in-depth experimental investigation combined with comprehensive literature review, we obtain valuable insights from open-sourced SLMs, fostering future light-weight SLM research. We also summarize a few potential research topics in SLM.

## 2 SLM Overview and Benchmarking

### 2.1 Collecting Popular SLMs

| Affiliation | Model name | Size | Date | Attention | Activation | Open training datasets | Max context window |
|---|---|---|---|---|---|---|---|
| Meta | OPT (Facebook, 2022.05) | 125M 350M 1.3B 2.7B | 2022.05 | MHA | ReLU | ✓ | 2k |
| | Galactica (Facebook, 2022.11) | 125M 1.3B | 2022.11 | MHA | GELU | | 2k |
| | Llama 3.2 (Meta, 2024.09) | 1B 3B | 2024.09 | MHA | GELU | | 128k |
| BigScience | Bloom (BigScience, 2022.11a) | 560M 1.1B | 2022.11 | MHA | $GELU_{tanh}$ | ✓ | 2k |
| | Bloomz (BigScience, 2022.11b) | 1.1B 560M | 2022.11 | MHA | | | |
| EleutherAI | Pythia (EleutherAI, 2023.03) | 160M 410M 1B 1.4B 2.8B | 2023.03 | MHA | GELU | ✓ | 2k |
| Cerebras | Cerebras-GPT (Cerebras, 2023.03) | 111M 256M 590M 1.3B 2.7B | 2023.03 | MHA | GELU | ✓ | 2k |
| Microsoft | Phi-1 (Microsoft, 2023.09a) | 1.3B | 2023.09 | MHA | $GELU_{tanh}$ | | 2k |
| | Phi-1.5 (Microsoft, 2023.09b) | 1.3B | 2023.09 | MHA | $GELU_{tanh}$ | | |
| | Phi-2 (Microsoft, 2023.12) | 2.7B | 2023.12 | MHA | $GELU_{tanh}$ | | 2k |
| | Phi-3-mini* (Microsoft, 2024.04) | 3.8B | 2024.04 | MHA | SiLU | | 4k |
| | Phi-3.5-mini* | 2.7B | 2024.09 | MHA | SiLU | | 4k |
| StabilityAI | StableLM-zephyr* (StabilityAI, 2023.11) | 3B | 2023.11 | MHA | SiLU | ✓ | 1k |
| | StableLM-2-zephyr* (StabilityAI, 2024.01) | 1.6B | 2024.01 | MHA | SiLU | ✓ | 4k |
| TinyLlama | TinyLlama (Unknown, 2023.12) | 1.1B | 2023.12 | GQA | SiLU | ✓ | 2k |
| Meituan | MobileLLaMA (Meituan, 2023.12) | 1.4B | 2023.12 | GQA | SiLU | ✓ | 2k |
| Alibaba | Qwen 1 (Alibaba, 2023.11) | 1.8B | 2023.11 | MHA | SiLU | | 8k |
| | Qwen 1.5 (Alibaba, 2024.02a) | 0.5B | 2024.02 | MHA | SiLU | | 32k |
| | Qwen 2 (Alibaba, 2024.02b) | 1.8B 4B | 2024.06 | MHA | SiLU | | 32k 32k |
| | Qwen 2.5 (Alibaba, 2024.09) | 0.5B 1.5B 3B | 2024.09 | GQA | SiLU | | 32k |
| MBZUAI | MobiLlama (MBZUAI, 2024.02) | 0.5B 1B | 2024.02 | GQA | SiLU | ✓ | 2k |
| | LaMini-GPT (MBZUAI, 2023.04) | 774M 1.5B | 2023.04 | MHA | $GELU_{tanh}$ | | 1k |
| Google | Gemma (Google, 2024.02) | 2B | 2024.02 | MQA | GELU | | 8k |
| | recurrentGemma (Google, 2024.04) | 2B | 2024.04 | MQA | $GELU_{tanh}$ | | 8k |
| | Gemma-2 (Google, 2024.07) | 2B | 2024.07 | GQA | $GELU_{tanh}$ | | 8k |
| OpenBMB | MiniCPM (OpenBMB, 2024.04) | 1B 2B | 2024.04 | GQA | SiLU | | 128k 131k |
| | MiniCPM3 (OpenBMB, 2024.09) | 4B | 2024.09 | MLA | SiLU | | |
| Apple | OpenELM (Apple, 2024.04) | 270M 450M 1.1B 3B | 2024.04 | GQA | SiLU | ✓ | 2k |
| H2O | danube3 (H2O.ai, 2024) | 0.5B 4B | 2024.07 | GQA | SiLU | | 8k 8k |
| TensorOpera AI | Fox (TensorOpera, 2024) | 1.6B | 2024.07 | GQA | SiLU | | 8k |
| HuggingFace | SmolLM (HuggingFace, 2024.07) | 135M 360M 1.7B | 2024.07 | GQA GQA MHA | SiLU | ✓ | 2k |
| Toyota | DCLM (Toyota, 2024.08) | 1.4B | 2024.08 | MHA | SiLU | ✓ | 50k |
| DataBricks | Dolly-v2* (DataBricks, 2023.04) | 3B | 2023.04 | MHA | GELU | | 2k |
| AllenAI | OLMo (AllenAI, 2024.04) | 1.18B | 2024.04 | MHA | SiLU | ✓ | 50k |
| Princeton | Sheared-LLaMA (Princeton, 2023.11) | 1.3B 2.7B | 2023.11 2023.11 | MHA MHA | SiLU SiLU | | 4k 4k |
| Xiaohongshu | MiniMA (Xiaohongshu, 2023.11) MiniMA2 (Xiaohongshu, 2024.07) | 3B 1B | 2023.11 2024.07 | UKN | SiLU SiLU | | 4096 4k |
| Nvidia | Minitron (Nvidia, 2024.07) | 4B | 2024.07 | GQA | ReLU2 | | 4k |
| M.A.P. | CT-LLM (M.A.P., 2024.04) | 2B | 2024.04 | MHA | SiLU | | 4k |
| AMD | AMD-Llama (AMD, 2024.08) | 135M | 2024.08 | MHA | SiLU | | 2k |

Table 1: Detailed configurations of SLMs benchmarked. We mainly use the base models in experiments, with exceptions of StableLM, Phi-3/3.5, and Dolly-v2 (marked with *) that only provide the instruct version.

SLMs have gained increasing attention from both the research and industrial communities. Notably, since the end of 2023, the number of SLM models has surged significantly. To understand their capability and cost, we comprehensively collect SLMs based on the following criteria: (1) Decoder-Only Transformer Architecture. (2) Open Weights to evaluate them freely. (3) Parameter Range between 100M and 5B parameters. (4) Base Model Focus: Only base pre-trained models are included, except for cases where only instruct versions exist (e.g., Microsoft Phi, StabilityAI StableLM).

These models, detailed in Table 1, encompass a broad spectrum from both industry and academia, differing in hyperparameters and training datasets. While they share similar architectures,

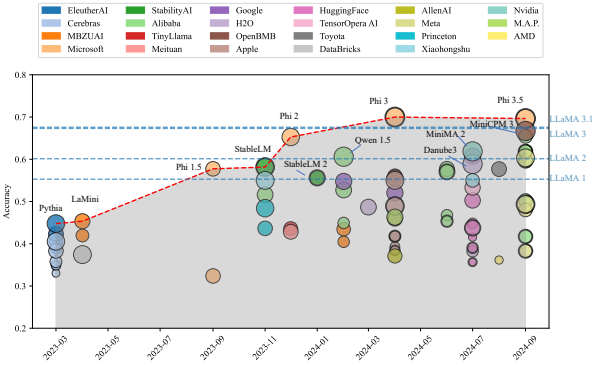



Figure 1: SLM capabilities over time. Performance is the average of all tasks (except for Math). The size of the circle is proportional to the model size. Red dashed lines show the state-of-the-art model at different time, indicating the trend that SLMs are getting better over time. Instruction-tuned models are highlighted with bold borders. LLaMA-7B series models are shown in horizontal blue dashed lines for comparison.

some datasets remain closed-source, leading to performance variations across tasks, as discussed in following sections. The details about our evaluation suite are shown in Appendix A.

## 2.2 Evaluation Datasets and Metrics

| Dataset | Description |
|---|---|
| HellaSwag (Zellers et al., 2019) | Tests narrative completion. |
| TruthfulQA (Lin et al., 2022) | Assesses truthfulness. |
| Winogrande (Sakaguchi et al., 2020) | Evaluates pronoun resolution. |
| CommonsenseQA (Talmor et al., 2019) | Commonsense multiple-choice questions. |
| PIQA (Bisk et al., 2020) | Physical commonsense reasoning. |
| OpenBookQA (Mihaylov et al., 2018) | Open-book science questions. |
| BoolQ (Clark et al., 2019) | Yes/no questions requiring reasoning. |
| ARC Easy (Clark et al., 2018) | Simple science questions. |
| ARC Challenge (Clark et al., 2018) | Complex science questions. |
| MMLU (Hendrycks et al., 2021) | Problem-solving across disciplines. |

Table 2: Datasets used to evaluate SLM capabilities.

We used 10 datasets as described in Table 2 to evaluate the SLM performance. Following (Gao et al., 2024), we use *accuracy* as the primary evaluation metric. Accuracy measures the compute log-likelihood of generating a continuation from a context. The default shown accuracy is instructed by 5 shots, as it is the most common setting in the released model.

## 2.3 SLM Capabilities

Figure 1 illustrates the progress of small language models (SLMs) in commonsense reasoning and problem-solving. From March 2023 to September 2024, SLM performance improved by 12.5% on average, surpassing the 7.5% improvement of LLaMA models over the same period. Notably, SLMs have outpaced LLaMA-7B series (v1–3.1), highlighting their growing potential for on-device tasks.

The Phi family, trained on closed-source datasets, leads in performance, reaching 70% average accuracy, comparable to LLaMA 3.1 (7B parameters). As of September 2024, Phi-3.5-mini (2.7B) achieves the highest accuracy, rivaling LLaMA 3.1 (8B). This advantage likely stems from careful data engineering, instruction tuning, and potential dataset overfitting (Zhang et al., 2024a). These findings suggest that SLMs are rapidly closing the gap with LLMs in general reasoning tasks.

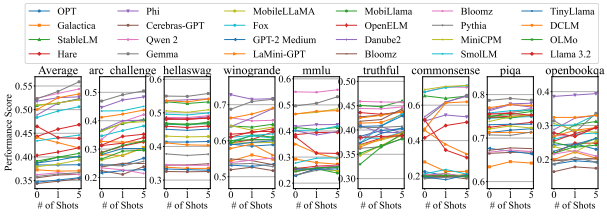
While larger models generally perform better, exceptions exist. Qwen2-1.5B outperforms many 3B-parameter SLMs, demonstrating that smaller models can excel in specific tasks.

The gap between open-source and closed-source SLMs is narrowing, driven by high-quality datasets like DCLM and FineWeb-Edu. Notably, SmolLM (64.2%) and DCLM-1B (63.8%) achieve strong performance in commonsense reasoning, highlighting the impact of high-quality training data.

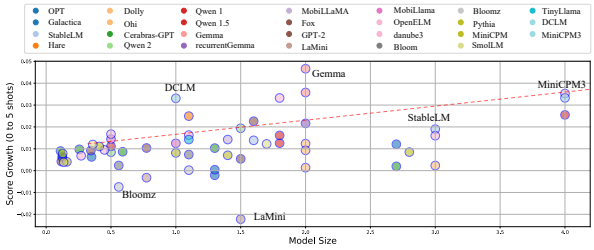
**Insight#1**: We draw following key insights from the evolvement of SLMs: (1) From March 2023 to September 2024, SLMs exhibited significant performance improvements across various language tasks, outpacing the improvements of the LLaMA-7B/8B series. Among them, the `Phi` family consistently achieves state-of-the-art performance across most tasks. (2) Smaller models like Qwen 2-1.5B can excel in specific tasks despite having fewer parameters. (3) SLMs trained on open-source datasets are closing the gap, thanks to high-quality datasets.
**Opportunity#1** State-of-the-art SLMs have surpassed 7B models in general tasks, demonstrating that their capabilities for real-world deployment. Moreover, the exceptional performance of certain SLMs on specific tasks highlights their potential for task-specific model routing, where different models are dynamically assigned based on task requirements to optimize efficiency and accuracy.

**In-context Learning Capabilities.** We evaluate various SLMs and their 2B-parameter variants (or the closest available ones) on 8 tasks, including commonsense reasoning and problem-solving. In-context learning (ICL) generally improves performance, with five-shot ICL increasing zero-shot accuracy by 2.1% on average.

However, HellaSwag and PIQA show minimal




(a) SLM in-context capabilities across tasks.




(b) Average accuracy improvement after in-context learning across different SLM model size.

Figure 2: In-context learning performance with different tasks and models. Red line in (b) highlights the trend of the average score improvement with the increase of model size.

improvement, likely due to their lower complexity compared to datasets like ARC Challenge. LaMini is the only model with a performance drop of over 2%, possibly due to overfitting, where additional context introduces noise. Gemma 2 exhibits the most significant improvement, with accuracy increasing by 4.8%. Notably, ICL effectiveness improves with model size.

**Insights#2**: We draw two key insights from the in-context learning capacity of SLMs: (1) Most SLMs exhibit in-context learning ability, but its effectiveness varies by task. Significant gains are observed in ARC Challenge, while HellaSwag and PIQA show minimal benefits across all models. (2) Larger models generally perform better in in-context learning, while some smaller SLMs experience performance declines.
**Opportunity#2**: Due to the smaller parameter size of SLMs, the effectiveness of ICL is limited. Combining ICL with supervised fine-tuning (SFT) may yield better performance (Zhu et al., 2024).

## 2.4 Training Datasets

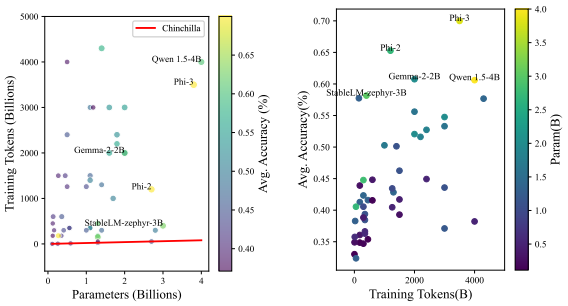
We investigate how the open-sourced pre-training datasets are used in training the SLMs. Overall, we find 12 such datasets being used and show them in Table 3.

| Subcaption | Model | Date | Tokens (B) | Datasets | Acc ↓ |
|---|---|---|---|---|---|
| <1B | SmolLM-360M | 24.07 | 600 | FineWeb-Edu[b],StarCoder,Cosmopedia[a] | 0.448 |
| | OpenELM-450M | 24.04 | 1500 | RefinedWeb, The Pile, RedPajama, Dolma | 0.417 |
| | SmolLM-135M | 24.07 | 600 | FineWeb-Edu[b],StarCoder,Cosmopedia[a] | 0.416 |
| | MobiLlama-0.5B | 24.02 | 1259 | RedPajama, RefinedWeb | 0.405 |
| | OpenELM-270M | 24.04 | 1500 | RefinedWeb, The Pile, RedPajama, Dolma | 0.393 |
| | Pythia-410M | 23.03 | 300 | The Pile | 0.388 |
| | BLOOMZ-560M | 22.11 | 350 | WuDaoCorpora | 0.366 |
| | BLOOM-560M | 22.11 | 350 | WuDaoCorpora | 0.363 |
| | OPT-125M | 22.05 | 180 | RoBERTa, The Pile, PushShift.io Reddit | 0.361 |
| | Cerebras-GPT-590M | 23.03 | 12 | The Pile | 0.358 |
| | OPT-125M | 22.05 | 180 | RoBERTa, The Pile, PushShift.io Reddit | 0.349 |
| | Pythia-160M | 23.03 | 300 | The Pile | 0.347 |
| | Cerebras-GPT-111M | 23.03 | 2 | The Pile | 0.330 |
| 1B–1.4B | DCLM-1B | 24.08 | 4300 | DCLM-baseline[b] | 0.577 |
| | OpenELM-1.1B | 24.04 | 1500 | RefinedWeb, The Pile, RedPajama, Dolma | 0.463 |
| | TinyLlama-1.1B | 23.12 | 3000 | SlimPajama, StarCoder | 0.436 |
| | MobiLlama-1B | 24.02 | 1259 | RedPajama, RefinedWeb | 0.434 |
| | MobileLLaMA-1.4B | 23.12 | 1300 | RedPajama | 0.428 |
| | Pythia-1.4B | 23.03 | 300 | The Pile | 0.423 |
| | OPT-1.3B | 22.05 | 180 | RoBERTa, The Pile, PushShift.io Reddit | 0.413 |
| | Pythia-1B | 23.03 | 300 | The Pile | 0.406 |
| | Bloom-1B1 | 22.11 | 350 | WuDaoCorpora | 0.394 |
| | Bloomz-1B1 | 22.11 | 350 | WuDaoCorpora | 0.384 |
| | Cerebras-GPT-1.3B | 23.03 | 26 | The Pile | 0.383 |
| 1.5B–2B | StableLM-2-zephyr-1.6B | 24.01 | 2000 | RefinedWeb, RedPajama, The Pile, StarCoder, CulturaX | 0.556 |
| | SmolLM-1.7B | 24.07 | 1000 | FineWeb-Edu[b],StarCoder,Cosmopedia[a] | 0.503 |
| 2.5B–3B | StableLM-zephyr-3B | 23.11 | 400 | RefinedWeb, RedPajama, The Pile, StarCoder | 0.582 |
| | Pythia-2.8B | 23.03 | 300 | The Pile | 0.448 |
| | OPT-2.7B | 22.05 | 180 | RoBERTa, The Pile, PushShift.io Reddit | 0.439 |
| | Cerebras-GPT-2.7B | 23.03 | 53 | The Pile | 0.405 |

Table 3: Classify according to the model parameter quantity and sort in descending order according to average normalized accuracy. Acc(Avg) is the average of the accuracies of the two types of tasks, Commonsense reasoning/understanding and Problem solving. **a** indicates that this dataset is generated by LLM. **b** indicates that this dataset has been filtered by LLM.

**Comparing the quality of pre-training datasets.** We analyzed the quality of open-source pre-training datasets by evaluating SLM performance across models trained on them. SLMs from the past three years were grouped by parameter size (<0.5B, 1B, 2B, and 3B) and ranked by average accuracy on Commonsense Reasoning/Understanding and Problem Solving tasks (Table 3). The results highlight DCLM and FineWeb-Edu as the top-performing datasets, both employing model-based data filtering. Additionally, many pre-training datasets, including StarCoder, contain coding data, despite SLMs on edge devices not prioritizing coding tasks. This inclusion is likely driven by the belief that coding data enhances reasoning ability (Zhang et al., 2024b).

**The number of training tokens vs. the size of model parameters.** The Chinchilla law (Hoffmann et al., 2022) suggests an optimal parameter-to-token ratio of 1:20 (e.g., a 1B model with 20B tokens). We analyzed SLMs under 4B parameters from 2022 to 2024, as shown in Figure 3(a) and found that recent models use significantly more to-




(a) The relationship between tokens and parameters. (b) The influence of training tokens on accuracy

Figure 3: The relationship between the number of training tokens, the number of model parameters, and the model accuracy. Here, the "accuracy" is averaged across all benchmarks in Table 3. (a) The relationship between the number of training tokens and model parameters size. According to scaling law (Chinchilla), that SLMs are often over-trained for better performance at deployment stage. (b) The influence of the number of training tokens on the model accuracy.

kens (typically over 1.5T) than this guideline (Figure 3(a)). Typically, larger models are trained on more tokens , but some smaller SLMs exceed larger ones (e.g., Qwen 2-0.5B with 12T tokens vs. Qwen 2-1.5B with 7T tokens), indicating over-training. This strategy aims to enhance SLM performance for resource-constrained deployment by increasing training-time FLOPs. However, over-training can lead to performance saturation (Godey et al., 2024).

**The amount of training tokens vs. model accuracy.** Figure 3(b) shows the relationship between the number of training tokens and the accuracy of the model. In general, there is a positive correlation between the two metrics, especially for those with less than 700B tokens. However, the correlation is weak, since the data quality often outweighs the impacts of more training tokens, especially when the training tokens exceed 1T.

**Insights#3**: We make two key observations in SLM training. (1) Data quality plays a crucial role in SLM capability, often outweighing data quantity and model architecture. A key trend in dataset research is model-based filtering, leading to state-of-the-art open-source pre-training datasets: FineWeb-Edu (1.3T/5.4T) (Penedo et al., 2024) and DCLM-baseline (4T) (Li et al., 2024a). (2) Recent SLMs are trained over large amount of tokens (typically >1.5T), disregarding their parameter sizes. In some cases, smaller SLMs

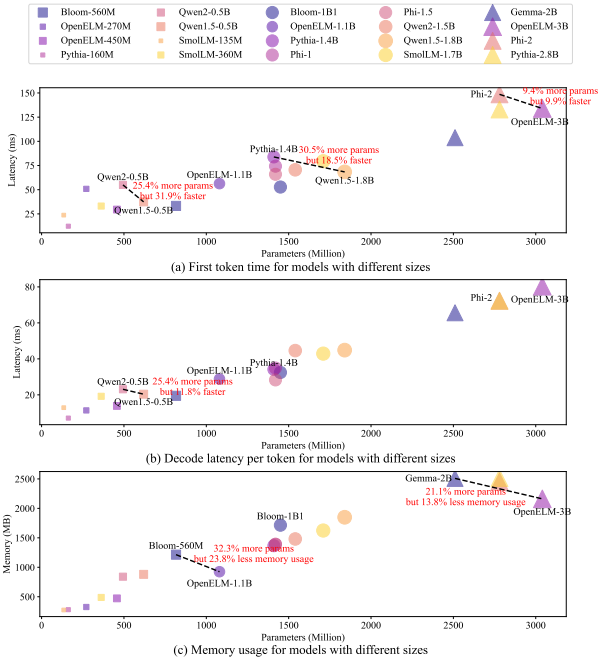



Figure 4: Latency and memory overview.

are trained over even more data (e.g., Qwen2-0.5B at 12T tokens but Qwen2-1.5B at 7T tokens).

**Opportunity#3** SLMs trained on model-based filtering datasets achieve competitive performance with those using closed datasets, significantly improving research reproducibility. Additionally, over-training can cause saturation and performance degradation. Defining an edge-optimized Chinchilla law is essential to ensure that additional tokens contribute to performance gains rather than diminishing returns.

## 3 SLM Runtime Cost

**Setup** In this section, we measure models of different sizes on robotic platform (Jetson Orin) and smartphone. We first analyze the latency and memory usage of models with different parameter sizes. Next, we assess the impact of quantization methods and hardware on model latency. Finally, we break down the latency and memory usage to identify the key factors influencing these metrics across various parts of the model. To eliminate the impact of inference engine implementations, we evaluated 20 models supported by `llama.cpp`, a widely recognized open-source inference engine.

We set a standard prompt length of 50 and a token generation length of 50 unless specified otherwise. To measure larger models, we applied 4-bit

| Device Name | Specifications | Release Time |
|---|---|---|
| Jetson Orin NX | 1024-core, 16G DRAM | Feb. 2023 |
| Pixel 7Pro | GoogleTensor G2,12G RAM | Oct. 2022 |
| Xiaomi 12S | Snapdragon 8Gen1+ ,12G RAM | Jul. 2022 |
| MEIZU 18Pro | Snapdragon 888,8G RAM | Mar. 2021 |

Table 4: Testing devices.

quantization to all models before conducting experiments in all sections except § 3.2.1.

## 3.1 Cost Overview

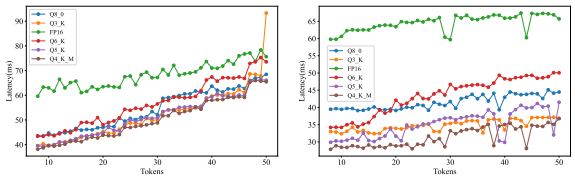
### 3.1.1 Inference Latency

Figure 4 presents inference latency, including first token time and per-token decode latency, for models ranging from 0.1B to 3B parameters. Latency scales with model size across three categories: 0.1-1B, 1-2B, and 2-3B. For models of similar size but different architectures, first token time varies significantly. For example, Qwen2-0.5B's first token time is 1.46× that of Qwen1.5-0.5B and comparable to OpenELM-1.1B, which has 2.18× the parameters. Qwen2's architecture shares the embedding layer and LM head, allocating more parameters to attention and FFN, increasing computational cost. Notably, Pythia-1.4B has higher latency than SmolLM-1.7B, Qwen2-1.5B, and Qwen-1.8B despite being smaller. Phi-2 also exhibits 1.11× the latency of OpenELM-3B, a larger model. The prefill stage dominates on-device LLM inference due to long-context processing for personalization on edge (Xu et al., 2024a).

Decode-stage latency generally follows a linear trend with model size and is primarily memory-bound, unlike the compute-bound prefill stage. Qwen2-0.5B and Qwen1.5-0.5B show similar decode latency, while Pythia-1.4B has lower latency than larger models. Among 2-3B models, Gemma-2B, Phi-2, and OpenELM-3B show a positive correlation between latency and model size. Architectural differences impact compute-bound stages more significantly, with wider, shallower models benefiting from higher parallelism.

### 3.1.2 Memory Footprint

Figure 4 evaluates memory footprint using `llama.cpp` on Jetson for models ranging from 0.1B to 3B parameters, with memory usage between 275MB and 2456MB. Since `llama.cpp` allocates KV cache and compute buffer based on maximum context length, we standardized it to 2048 across all models. Memory usage generally scales linearly with model size, but some models deviate. Gemma-2B, with a vocabulary of 256,000, and Bloom-560M/Bloom-1B1, with 250,880, consume more memory than expected due to their large vocabularies. In contrast, OpenELM models use less memory than similarly sized models, benefiting from a smaller 32,000-token vocabulary (compared to the typical 50,000) and GQA instead of MHA, reducing KV cache requirements. The impact of vocabulary size on memory usage is detailed in § 3.3.2.




(a) First token time (b) Decode latency per token

Figure 5: The relationship between the latency and quantization methods

**Insights#4**: We draw following insights regarding the runtime cost of SLMs on devices. (1) Model architecture has a greater impact on inference latency than model size, especially for smaller models (<1B). The correlation is likely hardware-dependent. (2) The impacts of model architecture on inference speed is more significant at prefill stage than decode stage because the compute bound of prefill stage. (3) Memory footprint scales with model size, but vocabulary size has a disproportionate impact.

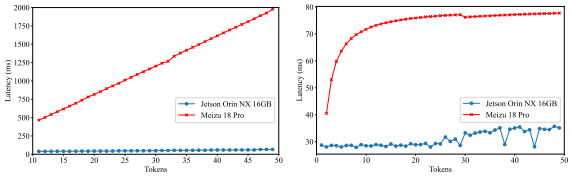
**Opportunity#4**: SLM architectures should align with hardware design, optimizing vocabulary size, FFN width, and layer depth for efficiency. Given different bottlenecks in prefill and decode, cloud systems adopt PD-separated clusters, while edge devices should leverage hardware heterogeneity, using NPUs for prefill and CPUs for decode.

## 3.2 Impact of Quantization and Hardware

### 3.2.1 Impact of Quantization

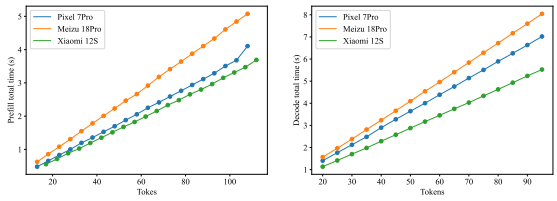
On Jetson and similar mobile devices without low-bit hardware support, quantization improves efficiency by reducing memory access overhead. On server GPUs, it lowers inference latency through higher Tensor Core throughput for int8 operations, lower memory usage leading to higher batch size, and lower memory access overhead.

We evaluated five quantization methods for Phi-1.5 (Figure 5). Qn_K (and Qn_K_M) apply *n*-bit




(a) Prefill (b) Decode

Figure 6: Latency under different hardware.




(a) Prefill (b) Decode

Figure 7: Latency on different smartphones.

quantization using the *k*-quants method, with Qn_0 denoting symmetric quantization. In the prefill stage, quantization reduces latency by at least 25% for short prompts, but the benefit diminishes with longer inputs. At a prompt length of 50, even the best-performing Q4_K_M achieves only a 13% reduction. This is because weights are shared across tokens, diluting the per-token benefit as prompt length increases. In the decode stage, quantization provides more consistent improvements, reducing latency by 17% to 75%, as weights are accessed per token, benefiting memory efficiency. Among methods, Q4_K_M consistently outperforms others, reducing latency by an average of 50%. In contrast, Q6_K and Q3_K become ineffective for long prompts, with latency matching or exceeding FP16. The inferior performance is due to irregular bit-widths, leading to higher overhead from alignment and padding.

**Insights#5**: We draw following insights regarding the quantization technique on SLM deployment. (1) Quantization is more effective in the bandwidth-bound decode stage than in the compute-bound prefill stage, especially when prompt length increasing. (2) Regular quantization precision enhances efficiency by avoiding extra hardware overhead.
**Opportunity#5**: Reducing memory access through quantization is not enough to significantly lower latency in edge deployments. Hardware designed for low-bit computation are essential.

#### 3.2.2 Impact of Hardware

We tested Bloom-1B1 on two edge devices: Jetson Orin NX 16GB (GPU) and Meizu 18 Pro (CPU). The GPU is 40× faster than the CPU in the prefill stage but only 1.84× faster in decode. Prefill benefits from high parallelism, leveraging the GPU's numerous computing units, while decode is sequential, limiting GPU efficiency. In the prefill stage, first token time increases linearly with prompt length, with Jetson's advantage expanding. In decode, latency per token rises as more tokens are generated. On Meizu, latency spikes from 1 to 10 tokens due to thermal throttling, then stabilizes at high latency. Jetson, with better cooling, fluctuates only after 30 tokens. We tested Qwen1.5-1.8B on three smartphones with 60s intervals to reduce power effects. Latency scales linearly with token count. The Xiaomi 12S had the lowest latency, showcasing the efficiency of Snapdragon 8 Gen 1+. The Pixel 7 Pro followed, while the Meizu 18 Pro had the highest latency due to its older Snapdragon 888 and lower memory configuration.

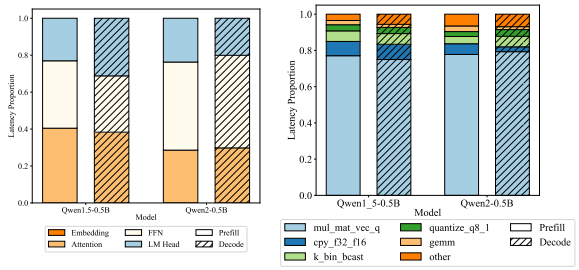
**Insights#6**: We draw following insights of impacts of hardware. (1) GPU has greater advantage in the prefill stage. (2) Jetson maintains consistent latency due to its simpler hardware structure and better heat dissipation, whereas smartphones experience higher thermal fluctuations during long inference tasks. (3) The development of System on a Chip (SoC) generations effectively improves inference efficiency.
**Opportunity#6**: For smartphones, fully utilizing the heterogeneous computing power of the SoC (e.g., GPU, NPU) can significantly improve prefill efficiency. Additionally, power consumption from continuous requests or long-context processing remains a major challenge.

### 3.3 Latency and Memory Breakdown

#### 3.3.1 Latency Breakdown

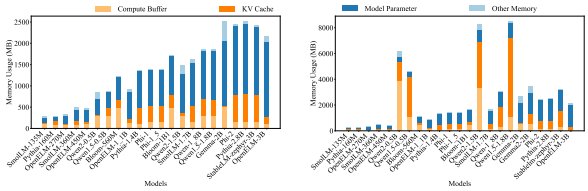
Figure 7 presents a breakdown analysis of Qwen 2-0.5B and Qwen 1.5-0.5B, two models of similar size but different latencies. We measured the time distribution across Embedding, Attention, FFN, and LM Head.

In the prefill stage, both models are dominated by Attention and FFN layers. In Qwen 1.5, Attention accounts for a slightly higher proportion of latency




(a) Layer granularity (b) Op granularity

Figure 8: On-device inference latency Breakdown.




(a) 2048 context window. (b) Max context window.

Figure 9: Memory Breakdown.

than FFN, whereas in Qwen 2, FFN contributes significantly more due to its wider FFN layer. During the decode stage, Attention latency increases in Qwen 1.5, likely due to KV Cache growth, while FFN remains the dominant bottleneck in Qwen 2. At the operator level, `mul_mat_vec_q` (matrix-vector multiplication) accounts for over 80% of total latency in both prefill and decode stages. This proportion is even higher in Qwen 2-0.5B due to its wider FFN layer, further increasing computation time.

### 3.3.2 Memory Breakdown

As shown in Figure 9a, vocabulary size significantly influences memory consumption beyond model size. Larger vocabularies increase compute buffer memory due to the logits, sized as batch_size * sequence_length * vocabulary_size. For instance, Bloom-560M (vocabulary: 250,880) requires 492MB memory, 3.5× more than OpenELM-1.1B (vocabulary: 32,000), while Bloom-1B1 requiring 496MB memory exceeds Qwen2-1.5B (vocabulary: 151,936) by 1.6×. Models using GQA have reduced KV cache size compared to MHA. OpenELM-3B is 3.9× smaller than StableLM-zephyr-3B. At long context lengths, compute buffer and KV cache dominate memory usage. For Qwen2 series (context length: 131,072), they account for 85%, while for Qwen1.5 (context length: 32,768), they make up 87%.

**Insights#7**: We have following insights regarding the breakdown of SLM runtime cost. (1) `Mul_mat_vec` (matrix by vector multiplication) is the most time-consuming operations of SLM, which constitute more than 70% end-to-end inference time. (2) Vocabulary size and Context length is crucial for model runtime memory usage.
**Opportunity#7**: SLMs are increasingly expanding vocabulary size to enhance performance. However, larger vocabulary increases inference latency and memory usage, necessitating compression strategies that preserve model capability. Similarly, as long-context support becomes a key trend, KV cache compression and quantization are crucial for SLMs.

## 4 Related Work

**Benchmarking SLM capability.** Several public leaderboards evaluate the capabilities of LLMs, such as Open LLM Leaderboard (ope) support by Hugging Face clusters, FlagEval (fla). Some datasets have released their own leaderboards, such as SuperCLUE (Xu et al., 2023), C-Eval (cev), and MMLU (mml). These leaderboards include limited SLMs and lack a rich variety of datasets. MobileAIBench (Murthy et al., 2024) and MELTing point (Laskaridis et al., 2024) also evaluate some LLMs on device. Compared to them, we are the first dive into the SLM capability through experiments on a large number of representative SLMs.
**Benchmarking SLM runtime cost.** Currently, some studies have measured the inference throughput and power consumption of LLMs on various hardware devices. MELTing point focuses on the throughput and energy consumption across different hardware. MELODI (Husom et al., 2024) also proposes a framework that focuses on energy consumption of LLMs. Using its dataset, the study explores how prompt attributes, such as length and complexity, correlate with energy expenditure. Additionally, MobileAIBench evaluates the runtime cost of 3 models under 3 billion parameters after 4-bit quantization on an iPhone 14. However, these studies have measured only a limited number of SLMs, and miss crucial observations such as the influence of model architecture on runtime costs.

## 5 Conclusions

In this work, we conduct a comprehensive study on the capabilities and performance of small language models (100M–5B parameters). We evaluate most, if not all, open-source SLMs and analyze their re-

sults, drawing key insights to guide future research. These insights provide a clear understanding of SLM strengths and limitations, identifying areas for architectural improvements and deployment optimizations.

## 6 Limitations

For SLM capability evaluation, we selected 10 commonsense reasoning and problem-solving datasets, excluding math datasets due to the performance gap between SLMs and larger models in mathematical reasoning. For cost analysis, to ensure consistency and eliminate inference engine variations, we evaluated 20 models supported by `llama.cpp`, excluding those not compatible with it.


## Acknowledgments

We thank the anonymous reviewers for their feedback on this work. This work was supported in part by the Major Key Project of PCL under Grant PCL2024A06 and PCL2022A05, in part by the Shenzhen Science and Technology Program under Grant RCJC20231211085918010, and in part by Xiaomi Open-Competition Research Program.


## References


FlagEval - leaderboard.

Leaderboard | C-Eval: A Multi-Level Multi-Discipline Chinese Evaluation Suite for Foundation Models.

MMLU Benchmark (Multi-task Language Understanding) | Papers With Code.

Open LLM Leaderboard 2 - a Hugging Face Space by open-llm-leaderboard.

2024. Copilot in microsoft edge. https://support.microsoft.com/en-us/topic/copilot-in-microsoft-edge-3fe6c1d4-9bd8-4492-a063-2cc6a5d01fef.

2024. Google ai edge sdk for gemini nano. https://developer.android.com/ai/aicore.

2024. Opera becomes the first major browser with built-in access to local ai models. https://press.opera.com/2024/04/03/ai-feature-drops-local-llms/.

2024. Unlocking 7b+ language models in your browser: A deep dive with google ai edge's mediapipe. https://research.google/blog/unlocking-7b-language-models-in-your-browser-a-deep-dive-with-google-ai-edges-mediapipe/.

Alibaba. 2023.11. Qwen 1. https://huggingface.co/alibaba/Qwen-1.

Alibaba. 2024.02a. Qwen 1.5. https://huggingface.co/alibaba/Qwen-1.5.

Alibaba. 2024.02b. Qwen 2. https://huggingface.co/alibaba/Qwen-2.

Alibaba. 2024.09. Qwen 2.5. https://qwenlm.github.io/blog/qwen2.5/.

AllenAI. 2024.04. allenai/olmo-1b-hf. https://huggingface.co/allenai/OLMo-1B-hf.

AMD. 2024.08. Llama. https://huggingface.co/amd/AMD-Llama-135m.

Apple. 2024.04. Openelm. https://huggingface.co/apple/OpenELM.

Jason Baumgartner, Savvas Zannettou, Brian Keegan, Megan Squire, and Jeremy Blackburn. 2020. The pushshift reddit dataset. In *Proceedings of the international AAAI conference on web and social media*, volume 14, pages 830–839.

Loubna Ben Allal, Anton Lozhkov, Guilherme Penedo, Thomas Wolf, and Leandro von Werra. 2024. Cosmopedia.

BigScience. 2022.11a. bigscience/bloom-560m. https://huggingface.co/bigscience/bloom-560m.

BigScience. 2022.11b. bigscience/bloomz-1b1. https://huggingface.co/bigscience/bloomz-1b1.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Cerebras. 2023.03. cerebras/cerebras-gpt-111m. https://huggingface.co/cerebras/Cerebras-GPT-111M.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. In *arXiv preprint arXiv:1803.05457*.

Together Computer. 2023. Redpajama: an open dataset for training large language models.

DataBricks. 2023.04. databricks/dolly-v2-3b. https://huggingface.co/databricks/dolly-v2-3b.

Mateusz Dubiel, Yasmine Barghouti, Kristina Kudryavtseva, and Luis A Leiva. 2024. On-device query intent prediction with lightweight llms to support ubiquitous conversations. *Scientific Reports*, 14(1):12731.

EleutherAI. 2023.03. Eleutherai/pythia-410m. https://huggingface.co/EleutherAI/pythia-410m.

Facebook. 2022.05. facebook/opt-125m. https://huggingface.co/facebook/opt-125m.

Facebook. 2022.11. facebook/galactica-125m. https://huggingface.co/facebook/galactica-125m.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. 2020. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2024. A framework for few-shot language model evaluation.

Nathan Godey, Éric de la Clergerie, and Benoît Sagot. 2024. Why do small language models underperform? studying language model saturation via the softmax bottleneck. *arXiv preprint arXiv:2404.07647*.

Google. 2024.02. Gemma. https://huggingface.co/google/Gemma.

Google. 2024.04. recurrentgemma. https://huggingface.co/google/recurrentGemma.

Google. 2024.07. Gemma-2. https://huggingface.co/google/Gemma-2.

H2O.ai. 2024. h2o-danube3-4b-base. https://huggingface.co/h2oai/h2o-danube3-4b-base.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

HuggingFace. 2024.07. Smollm. https://huggingface.co/huggingface/SmolLM.

Erik Johannes Husom, Arda Goknil, Lwin Khin Shar, and Sagar Sen. 2024. The Price of Prompting: Profiling Energy Use in Large Language Models Inference. *arXiv preprint*. ArXiv:2407.16893 [cs].

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Stefanos Laskaridis, Kleomenis Katevas, Lorenzo Minto, and Hamed Haddadi. Mobile and edge evaluation of large language models. In *Workshop on Efficient Systems for Foundation Models II@ ICML2024*.

Stefanos Laskaridis, Kleomenis Katevas, Lorenzo Minto, and Hamed Haddadi. 2024. MELTing point: Mobile Evaluation of Language Transformers. *arXiv preprint*. ArXiv:2403.12844 [cs].

Pierre Lepagnol, Thomas Gerald, Sahar Ghannay, Christophe Servan, and Sophie Rosset. 2024. Small language models are good too: An empirical study of zero-shot classification. *arXiv preprint arXiv:2404.11122*.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, Hritik Bansal, Etash Guha, Sedrick Keh, Kushal Arora, et al. 2024a. Datacomp-lm: In search of the next generation of training sets for language models. *arXiv preprint arXiv:2406.11794*.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, Qian Liu, Evgenii Zheltonozhskii, Terry Yue Zhuo, Thomas Wang, Olivier Dehaene, Mishig Davaadorj, Joel Lamy-Poirier, João Monteiro, Oleh Shliazhko, Nicolas Gontier, Nicholas Meade, Armel Zebaze, Ming-Ho Yee, Logesh Kumar Umapathi, Jian Zhu, Benjamin Lipkin, Muhtasham Oblokulov, Zhiruo Wang, Rudra Murthy, Jason Stillerman, Siva Sankalp Patel, Dmitry Abulkhanov, Marco Zocca, Manan Dey, Zhihan Zhang, Nour Fahmy, Urvashi Bhattacharyya, Wenhao Yu, Swayam Singh, Sasha Luccioni, Paulo Villegas, Maxim Kunakov, Fedor Zhdanov, Manuel Romero, Tony Lee, Nadav Timor, Jennifer Ding, Claire Schlesinger, Hailey Schoelkopf, Jan Ebert, Tri Dao, Mayank Mishra, Alex Gu, Jennifer Robinson, Carolyn Jane Anderson, Brendan Dolan-Gavitt, Danish Contractor, Siva Reddy, Daniel Fried, Dzmitry Bahdanau, Yacine Jernite, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro von Werra, and Harm de Vries. 2023. Starcoder: may the source be with you!

Xiang Li, Zhenyan Lu, Dongqi Cai, Xiao Ma, and Mengwei Xu. 2024b. Large language models on mobile devices: Measurements, analysis, and insights. In *Proceedings of the Workshop on Edge and Mobile Foundation Models*, pages 1–6.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Truthfulqa: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958*.

Yinhan Liu. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

M.A.P. 2024.04. Ct-llm. https://huggingface.co/m-a-p/CT-LLM-Base.

MBZUAI. 2023.04. Mbzuai/lamini-gpt-774m. https://huggingface.co/MBZUAI/LaMini-GPT-774M.

MBZUAI. 2024.02. Mobillama. https://huggingface.co/mbzuai/MobiLlama.

Meituan. 2023.12. Mobilellama. https://huggingface.co/meituan/MobileLLaMA.

Meta. 2024.09. meta-llama/llama-3.2-3b. https://huggingface.co/meta-llama/Llama-3.2-3B.

Microsoft. 2023.09a. microsoft/phi-1. https://huggingface.co/microsoft/phi-1.

Microsoft. 2023.09b. microsoft/phi-1_5. https://huggingface.co/microsoft/phi-1_5.

Microsoft. 2023.12. microsoft/phi-2. https://huggingface.co/microsoft/phi-2.

Microsoft. 2024.04. microsoft/phi-3-mini. https://huggingface.co/microsoft/phi-3-mini.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Rithesh Murthy, Liangwei Yang, Juntao Tan, Tulika Manoj Awalgaonkar, Yilun Zhou, Shelby Heinecke, Sachin Desai, Jason Wu, Ran Xu, Sarah Tan, Jianguo Zhang, Zhiwei Liu, Shirley Kokane, Zuxin Liu, Ming Zhu, Huan Wang, Caiming Xiong, and Silvio Savarese. 2024. MobileAIBench: Benchmarking LLMs and LMMs for On-Device Use Cases. *arXiv preprint*. ArXiv:2406.10290 [cs].

Thuat Nguyen, Chien Van Nguyen, Viet Dac Lai, Hieu Man, Nghia Trung Ngo, Franck Dernoncourt, Ryan A Rossi, and Thien Huu Nguyen. 2023. Culturax: A cleaned, enormous, and multilingual dataset for large language models in 167 languages. *arXiv preprint arXiv:2309.09400*.

Nvidia. 2024.07. Minitron. https://huggingface.co/nvidia/Minitron-4B-Base.

OpenBMB. 2024.04. Minicpm. https://huggingface.co/openbmb/MiniCPM.

OpenBMB. 2024.09. Minicpm3. https://huggingface.co/openbmb/MiniCPM3.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, Thomas Wolf, et al. 2024. The fineweb datasets: Decanting the web for the finest text data at scale. *arXiv preprint arXiv:2406.17557*.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. 2023. The refinedweb dataset for falcon llm: outperforming curated corpora with web data, and web data only. *arXiv preprint arXiv:2306.01116*.

Princeton. 2023.11. Sheared-llama. https://huggingface.co/princeton-nlp/Sheared-LLaMA-1.3B.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Timo Schick and Hinrich Schütze. 2020. It's not just size that matters: Small language models are also few-shot learners. *arXiv preprint arXiv:2009.07118*.

Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, Valentin Hofmann, Ananya Harsh Jha, Sachin Kumar, Li Lucy, Xinxi Lyu, Nathan Lambert, Ian Magnusson, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew E. Peters, Abhilasha Ravichander, Kyle Richardson, Zejiang Shen, Emma Strubell, Nishant Subramani, Oyvind Tafjord, Pete Walsh, Luke Zettlemoyer, Noah A. Smith, Hannaneh Hajishirzi, Iz Beltagy, Dirk Groeneveld, Jesse Dodge, and Kyle Lo. 2024. Dolma: an Open Corpus of Three Trillion Tokens for Language Model Pretraining Research. *arXiv preprint*.

StabilityAI. 2023.11. stabilityai/stablelm-zephyr-3b. https://huggingface.co/stabilityai/stablelm-zephyr-3b.

StabilityAI. 2024.01. stabilityai/stablelm-2-zephyr*. https://huggingface.co/stabilityai/stablelm-2-zephyr.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

TensorOpera. 2024. Fox-1-1.6b. https://huggingface.co/tensoropera/Fox-1-1.6B.

Toyota. 2024.08. Dclm. https://huggingface.co/TRI-ML/DCLM-1B.

Unknown. 2023.12. Tinyllama. https://huggingface.co/tinyllama.

Xiaohongshu. 2023.11. Minima. https://huggingface.co/GeneZC/MiniMA-3B.

Xiaohongshu. 2024.07. Minima2. https://huggingface.co/GeneZC/MiniMA-2-1B.

Weikai Xie, Li Zhang, Shihe Wang, Rongjie Yi, and Mengwei Xu. 2024. Droidcall: A dataset for llm-powered android intent invocation. *Preprint*, arXiv:2412.00402.

Daliang Xu, Hao Zhang, Liming Yang, Ruiqi Liu, Gang Huang, Mengwei Xu, and Xuanzhe Liu. 2024a. Empowering 1000 tokens/second on-device llm prefilling with mllm-npu. *arXiv preprint arXiv:2407.05858*.

Jiajun Xu, Zhiyuan Li, Wei Chen, Qun Wang, Xin Gao, Qi Cai, and Ziyuan Ling. 2024b. On-device language models: A comprehensive review. *arXiv preprint arXiv:2409.00088*.

Liang Xu, Anqi Li, Lei Zhu, Hang Xue, Changtai Zhu, Kangkang Zhao, Haonan He, Xuanwei Zhang, Qiyue Kang, and Zhenzhong Lan. 2023. Superclue: A comprehensive chinese large language model benchmark. *Preprint*, arXiv:2307.15020.

Mengwei Xu, Wangsong Yin, Dongqi Cai, Rongjie Yi, Daliang Xu, Qipeng Wang, Bingyang Wu, Yihao Zhao, Chen Yang, Shihe Wang, et al. 2024c. A survey of resource-efficient llm and multimodal foundation models. *arXiv preprint arXiv:2401.08092*.

Jinliang Yuan, Chen Yang, Dongqi Cai, Shihe Wang, Xin Yuan, Zeling Zhang, Xiang Li, Dingge Zhang, Hanzi Mei, Xianqing Jia, et al. 2023. Mobile foundation model as firmware. *arXiv preprint arXiv:2308.14363*.

Sha Yuan, Hanyu Zhao, Zhengxiao Du, Ming Ding, Xiao Liu, Yukuo Cen, Xu Zou, Zhilin Yang, and Jie Tang. 2021. Wudaocorpora: A super large-scale chinese corpora for pre-training language models. *AI Open*, 2:65–68.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Hugh Zhang, Jeff Da, Dean Lee, Vaughn Robinson, Catherine Wu, Will Song, Tiffany Zhao, Pranav Raja, Dylan Slack, Qin Lyu, et al. 2024a. A careful examination of large language model performance on grade school arithmetic. *arXiv preprint arXiv:2405.00332*.

Xinlu Zhang, Zhiyu Zoey Chen, Xi Ye, Xianjun Yang, Lichang Chen, William Yang Wang, and Linda Ruth Petzold. 2024b. Unveiling the impact of coding data instruction fine-tuning on large language models reasoning. *arXiv preprint arXiv:2405.20535*.

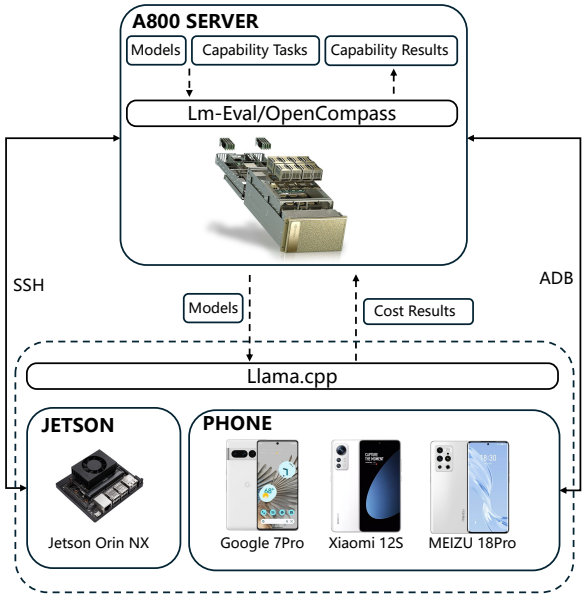



Figure 10: Framework.

Zhengping Zhou, Lezhi Li, Xinxi Chen, and Andy Li. 2023. Mini-giants:" small" language models and open source win-win. *arXiv preprint arXiv:2307.08189*.

Junyi Zhu, Shuochen Liu, Yu Yu, Bo Tang, Yibo Yan, Zhiyu Li, Feiyu Xiong, Tong Xu, and Matthew B. Blaschko. 2024. Fastmem: Fast memorization of prompt improves context awareness of large language models. *Preprint*, arXiv:2406.16069.

## A Evaluation Suite

The framework of our evaluation suites shown in Figure 10 evaluates SLMs across diverse devices, including smartphones (Google Pixel 7 Pro, Xiaomi 12S, MEIZU 18 Pro) and Jetson Orin NX, focusing on capability evaluation and cost analysis. Capability evaluation is conducted on the A800 server using Lm-Eval/OpenCompass, benchmarking models on various tasks.

Cost analysis measures inference efficiency and resource consumption, with both Jetson and smartphones executing cost tasks via `llama.cpp`. In the cost analysis, the A800 server serves as the central hub, managing model deployment, execution, and result aggregation. It downloads and transfers models to target devices, where models are quantized, and prompts are generated to prepare for cost evaluation. After testing, the A800 server analyzes cost results, generating a detailed report on latency, memory usage, and performance breakdowns. Communication between A800 and Jetson occurs via SSH, while smartphones connect via ADB, ensuring seamless task distribution and result collection. By integrating hardware diversity,

benchmarking tools, and evaluation metrics, the framework enables a comprehensive analysis of SLM efficiency across different deployment scenarios.

## B Model Architecture

While we focus on only decoder-only transformer SLMs, their specific configurations still diversify, as shown in Figure 11(a). The core of Transformer is the multi-head self-attention(MHA) mechanism and the Feed-Forward Neural Network(FFN).

**Model architecture analysis.** We conduct statistical analysis on the following several components of the model architecture: 1) The type of self-attention; 2) The type of feed-forward neural network; 3) The intermediate ratio of the feed-forward network; 4) The activation function of the feed-forward neural network; 5) The type of layer normalization; 6) The vocabulary size. Figure 11(a) shows the architecture of SLM and the pie chart shows the distribution of six components. Figure 11(b) shows how these distributions change over time.

1) *The type of self-attention.* The self-attention mechanism is the core of the Transformer model. In general, SLMs mainly use three types of attention mechanism: Multi-Head Attention (MHA), Multi-Query Attention (MQA), Group-Query Attention (GQA) and Multi-Head Latent Attention(MLA). Multi-Head Attention is a mechanism that allows the model to focus on different parts of the input data simultaneously by employing multiple attention heads, which is the most widely used self-attention mechanism in the Transformer models. Multi-Query Attention simplifies multi-head attention by using a single shared query across all heads but allowing different key and value projections. This reduces the complexity in both space and time. Group-Query Attention is a variant of multi-head attention that reduces computational complexity by sharing query representations across multiple heads, while allowing separate key and value representations. The idea is to use fewer query groups but still preserve a level of diversity in the attention mechanism. Multi-Head Latent Attention achieves better results than MHA through low-rank key-value joint compression, and requires much less Key-Value(KV) Cache.

Figure 11(b)① shows the changing situation of choosing three self-attention mechanisms during these time periods from 2022 to 2024. We can see that MHA is gradually being phased out and replaced by GQA.

2) *The type of feed-forward neural network.* Feed-forward network can be summarized into two types: the Standard FFN and the Gated FFN. The Standard FFN is a two-layer neural network with a activation function. The Gated FFN adds an additional gate layer.

The Figure 11(b)② shows the changing situation of type of FFN during these time periods from 2022 to 2024. It shows that Standard FFN is gradually being phased out and replaced by Gated FFN.

3) *The intermediate ratio of the feed-forward neural network.* The intermediate ratio of FFN is the ratio of the intermediate dimension to the hidden dimension. Figure 11(b)③ shows that the intermediate ratio of the Standard FFN is commonly set to be 4, while the intermediate ratio of the Gated FFN is rather diversified ranging from 2 to 8.

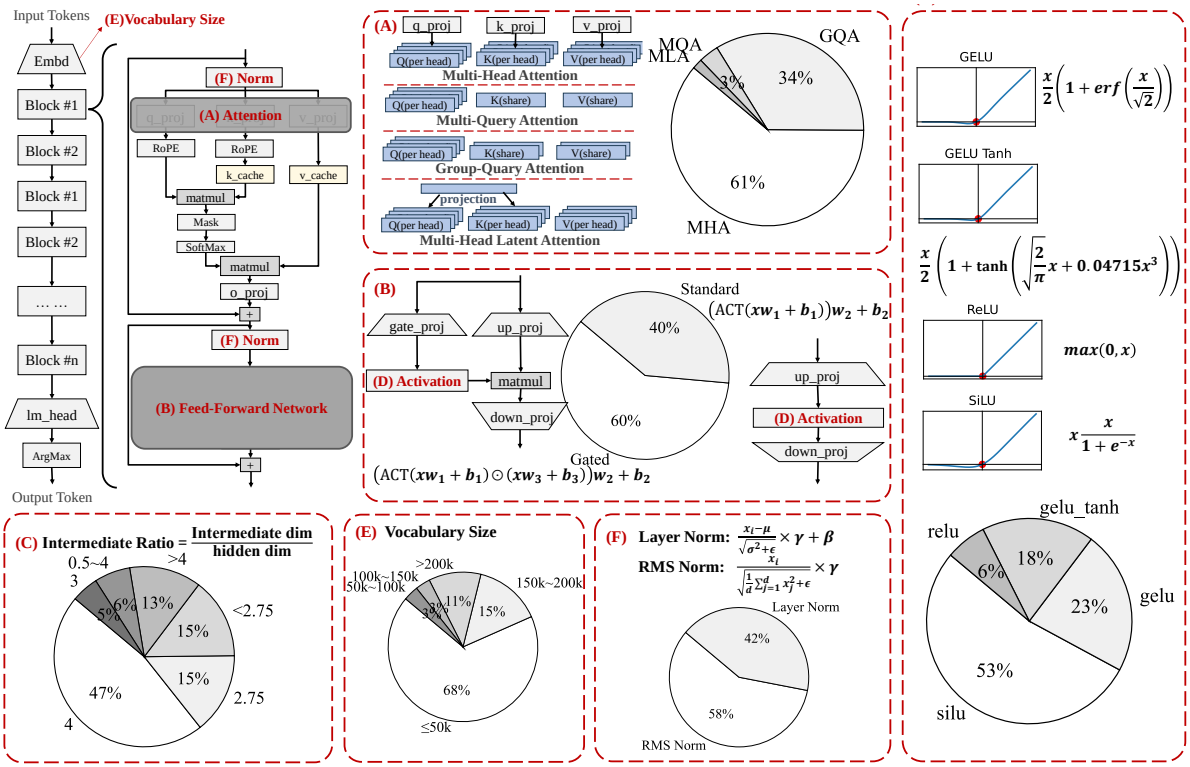
4) *The activation function of the feed-forward neural network.* There are 4 main kinds of activation functions used in FFN: ReLU (Rectified Linear Unit), GELU (Gaussian Error Linear Unit), $\text{GELU}_{\text{tanh}}$, SiLU (Sigmoid Linear Unit). Observed from Figure 11(b)④ , the activation function of FFN was mostly ReLU in 2022, and then changed to GELU and its variants in 2023. For those released in 2024, SiLU becomes the dominant type.

5) *The type of layer normalization.* There are two main types of layer normalization: LayerNorm and RMSNorm. The Figure 11(b)⑤ shows the changing situation of type of the type of layer normalization during these time periods from 2022 to 2024. layer normalization is gradually being phased out and replaced by RMS normalization.

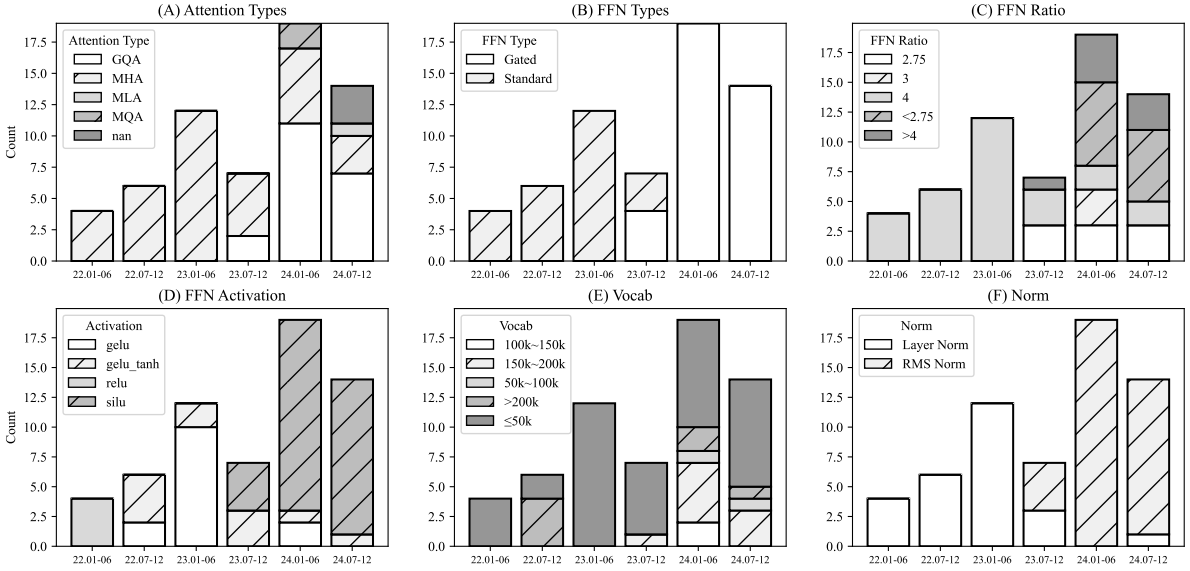
6) *The vocabulary size.* The vocabulary size is the total number of unique tokens that an SLM can recognize. The Figure 11(b)⑥ shows the changing situation of the vocabulary size during these time periods from 2022 to 2024. We can see that the vocabulary size of the model is gradually increasing. The vocabulary of the latest models is often larger than 50k

**Model architecture innovations.** While the vanilla transformer architecture has been well recognized for its scaling ability, there still exist a few architecture-level innovations in the tested SLMs, namely parameter sharing and layer-wise parameter scaling.

1) Parameter Sharing. Parameter Sharing is a technique used in large language models to reuse the same set of weights across different layers or




(a) The architecture.




(b) Architecture distribution.

Figure 11: The architecture analysis of the SLM, highlighting 6 configurations: attention type, FFN type, FFN ratio, FFN activation, vocabulary size, and normalization type. (a) presents the overall structure of the SLM, and the categorizations with usage frequency of the 6 configurations; (b) analyzes the concrete selections of six configurations over time.

components of the network. This approach allows the model to significantly reduce the number of parameters, leading to more efficient training and inference, while maintaining performance.

*Embedding-lm_head sharing.* Sharing the weights of the embedding with the final lm_head layer is the most common weight sharing technique. It is the sharing of the word embedding layer and has nothing to do with the rotary position encoding. Models such as Gemma, and Qwen all used this sharing technique.

*layer-wise attention/FFN sharing.* In this approach, the same set of weights is reused across multiple layers of the model. This is commonly seen in SLM/LLM, where all the transformer layers share the same parameters. For example, MobiLLaMA shares the weights of the FFN of all the transformer blocks; MobileLLM shares the weights of the Attention and FFN of two adjacent transformer blocks.

2) Layer-wise parameter scaling. This technique was proposed and used by OpenELM. Traditional SLMs use the same configuration for each transformer layer in the model, resulting in a uniform allocation of parameters across layers. Unlike these models, each transformer layer in OpenELM has a different configuration (e.g., number of heads and feed forward network dimension), resulting in variable number of parameters in each layer of the model. This lets OpenELM to better utilize the available parameter budget for achieving higher accuracies.

3) Nonlinearity compensation. PanGu-$\pi$ analyzes the state-of-the-art language model architectures and observes the feature collapse problem. PanGu-$\pi$ adopts two techniques for nonlinearity compensation of language model to solve the feature collapse problem. The series activation function is adapted to FFN, and the augmented shortcuts are integrated into MHA, which effectively introduces more nonlinearity into the Transformer architecture.

**Insights**: We make two key observations in SLM architectures. (1) As of August 2024, a typical SLM architecture tends to use group-query attention, gated FFN with SiLU activation, an intermediate ratio of FFN between 2 and 8, RMS normalization, and a vocabulary size larger than 50K. However, the choice of such settings is mostly empirical, without strict and public validation on the superiority of such model's capacity. Instead, the architecture innovations have relative larger impacts on the runtime performance on devices, as shown in §3. (2) The innovations to the transformer architecture is limited in nowaday SLMs. For the few that did contribute architectural innovation (except embedding-lm head sharing), we do not observe strong evidence showing them being significantly superior to the vanilla transformer, and neither are them being generally adopted or studied across different research groups or companies. The significance of those innovations remain to be explored and validated.

## C Training Datasets and Costs

We find 12 open-source datasets being used:

- The Pile (Gao et al., 2020) (825B tokens): a combination of smaller corpora in various domains.
- FineWeb-Edu (Penedo et al., 2024) (1.3T tokens): a collection of educational text filtered from FineWeb.
- StarCoder (Li et al., 2023) (35B tokens): Python tokens.
- Cosmopedia (Ben Allal et al., 2024) (25B tokens): a dataset of synthetic textbooks, blogposts, stories, posts and WikiHow articles generated by Mixtral-8x7B-Instruct-v0.1.
- RefinedWeb (Penedo et al., 2023) (5T tokens): despite extensive filtering, high-quality data extracted from the web remains plentiful, obtained from CommonCrawl.
- RedPajama (Computer, 2023) (1.2T tokens): includes over 100B text documents coming from 84 CommonCrawl snapshots and processed using the CCNet pipeline.
- Dolma (Soldaini et al., 2024): a English corpora, which is deduplicated inner corpus and across corpus using MinHash algorithms.
- WuDaoCorpora (Yuan et al., 2021) (4T tokens): a super large-scale Chinese corpora, containing about 3T training data and 1.08T Chinese characters.

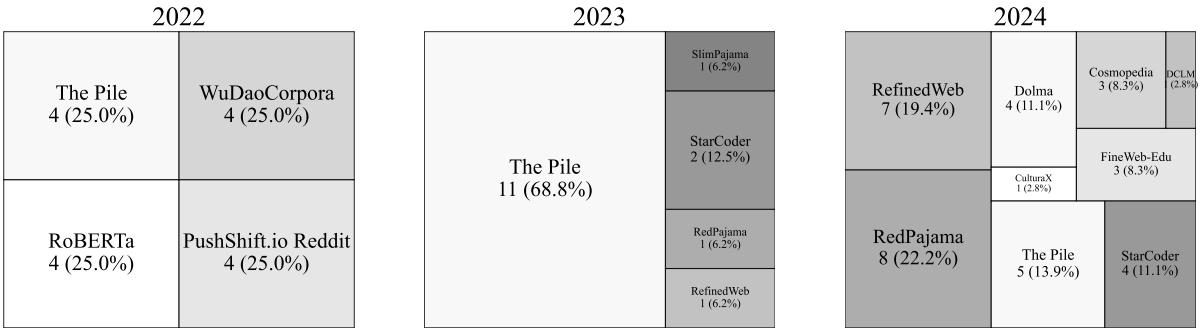
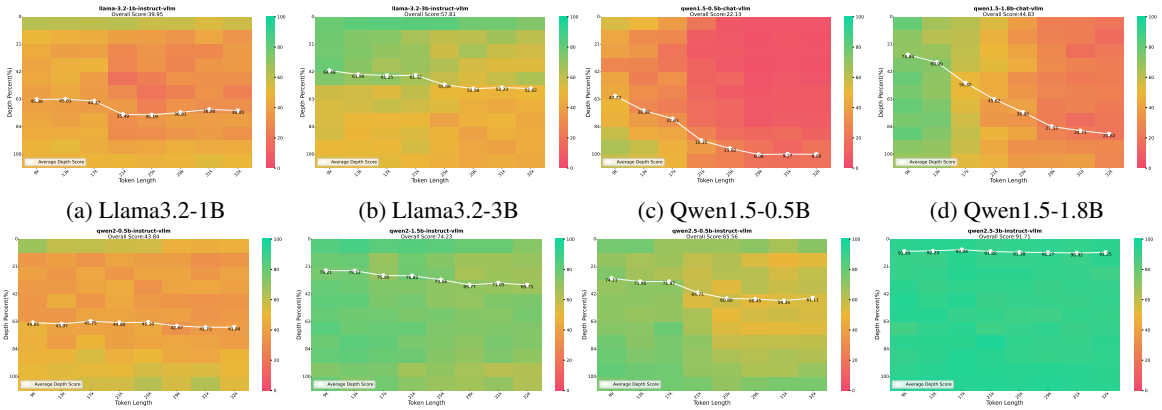



Figure 12: The usage frequency of each open-source pre-training dataset from 2022 to 2024.



(a) Llama3.2-1B (b) Llama3.2-3B (c) Qwen1.5-0.5B (d) Qwen1.5-1.8B

(e) Qwen2-0.5B (f) Qwen2-1.5B (g) Qwen2.5-0.5B (h) Qwen2.5-3B

Figure 13: Long context capabilities of SLMs.

- RoBERTa (Liu, 2019) CCNewsV2: containing an updated version of the English portion of the CommonCrawl News dataset.

- PushShift ().io Reddit (Baumgartner et al., 2020): a social media data collection, analysis, and archiving platform that since 2015 has collected Reddit data and made it available to researchers.

- DCLM-baseline (Li et al., 2024a) (1.35T tokens): a standardized corpus extracted from Common Crawl, effective pretraining recipes based on the OpenLM framework, and a broad suite of 53 downstream evaluations.

- CulturaX (Nguyen et al., 2023) (6.3T tokens): a substantial multilingual dataset in 167 languages.

**The usage preference of pre-training datasets.** We then conducted statistics on the usage frequency of the datasets for training SLM from 2022 to 2024. The results are illustrated in Figure 12. It shows that The Pile is the most widely used pre-training dataset especially in 2022 and 2023; yet more recently, more such datasets are proposed and the choice becomes diversified. In fact, The Pile has been abandoned in pre-training SLMs recently, and datasets such as "RefinedWeb" and "RedPajama" have gradually been widely used. It shows the active research and engineering efforts in constructing pre-training datasts with better quality.

Since pretraining typically uses only one epoch, the Training TFLOPs is calculated as: Training FLOPs $=$ $6 \times$ Model Size $\times$ Training Tokens. Here, the factor 6 accounts for the number of multiplication and addition operations required per token during the model's forward and backward propagation.

When estimating the actual computation time using Training TFLOPs, we need to consider Model FLOPs Utilization (MFU) (Chowdhery et al., 2023). Using the data from DCLM (Li et al., 2024a), we obtain a typical MFU of approximately 30%.

The estimated training time on H100 (BF16

TFLOPS is 1979) is given by:

$$\text{Train Hours} = \frac{\text{Training FLOPs}}{\text{MFU} \times \text{H100 TFLOPs}} \quad (1)$$

As shown in Table 5, we can observe that more tokens are being used to train SLMs over time. The overall training cost has escalated accordingly. And we select 8 models released at 2024 and show that the recent SLMs do not obey the scaling law of training cost and model size. For example, as shown in Table 6, Qwen2 0.5B takes more training tokens and hours than Gemma-2 2B.

Table 5: Token count for SLM pretraining is rising over time.

| Model | Size (B) | Tokens (T) | TFLOPs | Date | H100 h |
|---|---|---|---|---|---|
| OPT | 1.3 | 0.18 | 1.4 | 22.05 | 657 |
| Bloom | 1.1 | 0.35 | 2.3 | 22.11 | 1081 |
| Pythia | 1.4 | 0.30 | 2.5 | 23.03 | 1179 |
| Phi-1.5 | 1.5 | 0.15 | 1.4 | 23.09 | 632 |
| TinyLlama | 1.1 | 3.0 | 19.8 | 23.12 | 9264 |
| OpenELM | 1.1 | 1.5 | 9.9 | 24.04 | 4632 |
| Qwen2 | 1.5 | 7.0 | 63.0 | 24.06 | 29476 |

Table 6: The training cost of recent SLMs.

| Model | Size (B) | Tokens (T) | TFLOPs | Date | H100 h |
|---|---|---|---|---|---|
| SmolLM | 0.135 | 0.6 | 0.49 | 24.07 | 227 |
| SmolLM | 0.36 | 0.6 | 1.30 | 24.07 | 606 |
| Qwen 2 | 0.5 | 12 | 36.0 | 24.06 | 16844 |
| Qwen 2 | 1.5 | 7 | 63.0 | 24.06 | 29476 |
| SmolLM | 1.7 | 1 | 10.2 | 24.07 | 4772 |
| Gemma-2 | 2.0 | 2 | 24.0 | 24.07 | 11229 |
| Phi-3 | 3.0 | 3.3 | 59.4 | 24.04 | 27792 |
| Qwen 1.5 | 4.0 | 4 | 96.0 | 24.02 | 44916 |

## D Long Context Capabilities

We used Needle-In-A-Haystack provided by OpenCompass to explore long-context capabilities of SLMs. The tasks included Single-Needle Retrieval, Multi-Needle Retrieval, and Multi-Needle Reasoning. The scores in Figure 13 are the average of these three tasks. Different models showed large variations in performance. Small models, such as Qwen1.5-0.5B and Qwen2-0.5B, performed less effectively. Qwen1.5-0.5B achieved an average accuracy of 22.13%. Qwen2-0.5B performed slightly better, reaching 43.84%. Qwen1.5-0.5B handled shorter contexts (9k-17k) relatively well. However, its accuracy dropped sharply with longer contexts. This was especially true for middle inserted text (Depth Percent from 20% to 70%). Larger models performed much better. Llama 3.2-3B had an average accuracy of 57.81%. It worked well with shorter contexts but struggled with deeper insertions when contexts exceeded 25k tokens. Qwen2.5-3B achieved an average accuracy of 91.71%. It maintained nearly perfect accuracy across all context lengths and insertion positions. This result highlights its strong ability to handle long contexts and complex scenarios.

**Insights**: We draw two key insights from the long context capacity of SLMs: (1) Larger parameters are crucial for long-context capabilities. Small models, such as Qwen1.5-0.5B and Qwen2-0.5B, perform adequately on short-context tasks but experience a significant drop in recognition accuracy as the context length increases. In contrast, larger models, such as Qwen2.5-3B, excel with outstanding performance, maintaining near-perfect accuracy across all context lengths and insertion positions. (2) "Lost in the Middle" also occurs in small models. Compared to deep or front insertions, the accuracy of middle-position text (Depth Percent 20%-70%) is significantly lower.

## E Tool-use Capabilities

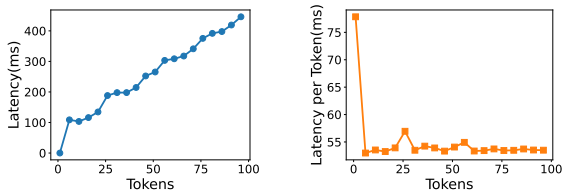
We conducted evaluations on tool-use capabilities with DroidCall(Xie et al., 2024), a dataset for LLM-powered Android intent invocation. Without fine-tuning, SLMs can not catch with LLMs on the tool-use capabilities as shown in Table 7. Qwen2.5-3B, Gemma2, Phi-3.5, and Llama3.2-3B show better performance than GPT-4o after fine-tuning.

Table 7: The tool-use capabilities of SLMs.

| Model | Size | Zero-Shot Acc | Few-Shot Acc | FT Acc |
|---|---|---|---|---|
| Qwen2.5-Instruct | 1.5B | 61.0 | 64.5 | 76.0 |
| Qwen2.5-Instruct | 3B | 62.0 | 71.0 | 83.0 |
| Gemma2-it | 2B | 59.0 | 67.5 | 85.0 |
| Phi-3.5-mini-instruct | 3.8B | 62.0 | 67.5 | 83.5 |
| MiniCPM3-4B | 4B | 67.0 | 75.0 | 74.5 |
| Llama3.2-Instruct | 1B | 31.5 | 60.5 | 75.5 |
| Llama3.2-Instruct | 3B | 66.5 | 72.0 | 82.0 |
| GPT-4o | - | 77.0 | 80.5 | - |
| GPT-4o-mini | - | 71.5 | 76.0 | - |

## F Evaluations on RNN-based Models

To explore the runtime cost of RNN-based architecture models, we evaluated Mamba-1.4B on Jetson Orin NX. Compared to Phi-1.5 with a similar size, Mamba does not have an advantage in prefill time. As shown in Figure 14, it takes more time, likely due to the lack of `llama.cpp` operator optimization for RNN-based models. Decode la-




(a) Prefill (b) Decode

Figure 14: Latency of Mamba on Jetson Orin NX.

tency per token typically increases as the number of tokens increases in Transformer-based models because of the $\mathcal{O}(n^2)$ time complexity of the attention mechanism ($n$ = token count). However, Mamba maintains a very stable decode latency per token, highlighting the advantages of its $\mathcal{O}(n)$ time complexity.

In terms of accuracy performance, we evaluate the Mamba variant: Zamba2. We evaluate Zamba2-1.2B and Zamba2-2.7B on 9 downstream tasks, as shown in Table 8. Zamba2-2.7B demonstrates better performance than Llama2-7B with fewer parameters, while Zamba2-1.2B performs worse than Llama2-7B. Compared to the SLMs released in the same period (May 2024) shown in Figure 1, Zamba2-2.7B performed very well, ranking just behind Phi-3.

Table 8: Capabilities of Zamba2-1.2B and Zamba2-2.7B on 9 downstream tasks.

| Model | ARC-C | ARC-E | BoolQ | CSQA | HellaSwag | MMLU | OBQA | PIQA | TruthfulQA | Avg |
|---|---|---|---|---|---|---|---|---|---|---|
| Zamba2-1.2B | 0.44 | 0.77 | 0.74 | 0.52 | 0.53 | 0.44 | 0.32 | 0.77 | 0.38 | 0.55 |
| Zamba2-2.7B | 0.48 | 0.80 | 0.80 | 0.75 | 0.57 | 0.56 | 0.32 | 0.79 | 0.46 | 0.61 |
| Llama2-7B | 0.45 | 0.74 | 0.80 | 0.76 | 0.76 | 0.41 | 0.34 | 0.79 | 0.38 | 0.60 |

## G Ablation Studies

### G.1 Impact of Model Architecture on Performance

To systematically examine the impact of model architecture, we trained 18 different models of similar parameter sizes, each on the same dataset containing 20B tokens. Table 9 shows that model architecture has a negligible effect on final training loss (ranging from 3.58 to 3.81), suggesting similar next-token prediction capability across variants. However, inference speeds—both in the prefill and decoding stages—vary significantly among architectures, with up to $5.22\times$ difference in decoding speed.

### G.2 Impact of Data Quality on Capabilities

To investigate the effect of data quality, two models with 500M parameters and identical hyperparameters were trained on different datasets: one on RefinedWeb + StarCoder (200B tokens) and one on DCLM (100B tokens). Both models share the same architecture. Evaluation on seven downstream tasks (ARC-Easy, ARC-Challenge, HellaSwag, TruthfulQA, Winogrande, PIQA, MMLU) yields average accuracies of 0.47 and 0.52, respectively. Despite using fewer tokens, the DCLM-trained model outperforms the model trained on larger but less curated data, indicating the importance of data quality over sheer quantity.

These experiments support our important conclusions:

- Model architecture significantly affects inference latency (§3.1).
- Dataset quality has a crucial impact on model performance (§2.4).

Table 9: Ablation on model architectures. All models trained on 20B tokens with comparable size.

| ID | Size (M) | Hidden | FFN | Layers | Act. | QH | KVH | Loss | Prefill (tok/s) | Decode (tok/s) |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 106.73 | 1280 | 2096 | 3 | relu | 16 | 16 | 3.76 | 916.70 | 455.32 |
| 2 | 106.73 | 1280 | 2096 | 3 | silu | 16 | 16 | 3.81 | 877.19 | 424.08 |
| 3 | 101.42 | 768 | 2046 | 9 | relu | 16 | 16 | 3.70 | 742.85 | 258.56 |
| 4 | 101.42 | 768 | 2046 | 9 | relu | 4 | 4 | 3.67 | 784.94 | 266.68 |
| 5 | 101.42 | 768 | 2046 | 9 | relu | 16 | 4 | 3.66 | 871.94 | 260.37 |
| 6 | 101.42 | 768 | 2046 | 9 | silu | 16 | 16 | 3.69 | 788.95 | 260.03 |
| 7 | 101.42 | 768 | 2046 | 9 | silu | 4 | 4 | 3.66 | 773.27 | 255.42 |
| 8 | 101.42 | 768 | 2046 | 9 | silu | 16 | 4 | 3.65 | 853.46 | 252.71 |
| 9 | 99.54 | 704 | 1856 | 11 | relu | 16 | 16 | 3.65 | 720.98 | 228.11 |
| 10 | 99.54 | 704 | 1856 | 11 | silu | 16 | 16 | 3.64 | 753.61 | 228.03 |
| 11 | 100.00 | 576 | 1536 | 18 | relu | 16 | 16 | 3.68 | 601.56 | 154.59 |
| 12 | 100.00 | 576 | 1536 | 18 | relu | 4 | 4 | 3.59 | 652.11 | 164.05 |
| 13 | 100.00 | 576 | 1536 | 18 | relu | 16 | 4 | 3.66 | 705.54 | 153.85 |
| 14 | 100.00 | 576 | 1536 | 18 | silu | 16 | 16 | 3.67 | 614.41 | 151.98 |
| 15 | 100.00 | 576 | 1536 | 18 | silu | 4 | 4 | 3.58 | 640.13 | 160.48 |
| 16 | 100.00 | 576 | 1536 | 18 | silu | 16 | 4 | 3.65 | 691.67 | 150.15 |
| 17 | 101.06 | 448 | 1184 | 33 | relu | 16 | 16 | 3.68 | 469.89 | 89.48 |
| 18 | 101.06 | 448 | 1184 | 33 | silu | 16 | 16 | 3.67 | 481.58 | 87.70 |