PhoneLM: an Efficient and Capable Small Language Model Family

Anonymous ACL submission

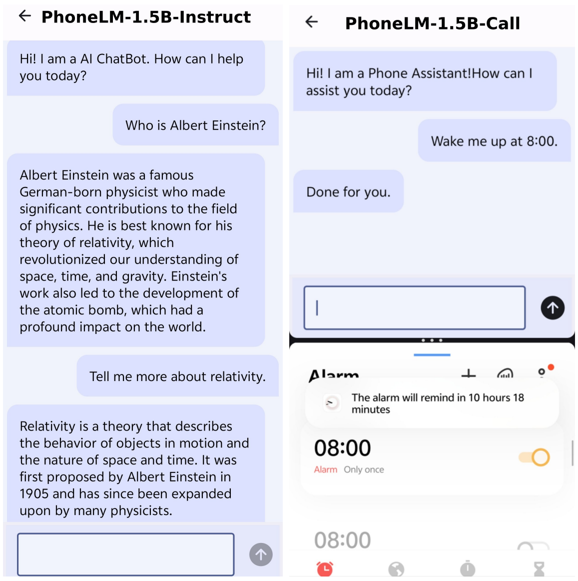
Abstract

The interest in developing small language models (SLM) for on-device deployment is fast growing. However, the existing SLM design hardly considers the device hardware characteristics. Instead, this work presents a simple yet effective principle for SLM design: architecture searching for optimal runtime efficiency before pre-training. Guided by this principle, we develop PhoneLM SLM family (with 0.5B and 1.5B versions), that acheive the state-of-the-art capability-efficiency trade-off among those with similar parameter size. We fully open-source the code, weights, and training datasets of PhoneLM for reproducibility and transparency, including both base and instructed versions. We also release a finetuned version of PhoneLM capable of accurate Android Intent invocation, and an end-to-end Android demo.

1 Introduction

In last few years, the striking progress has been made in large language models, attributed to the scaling-up ability of transformer. One the other hand, we also notice growing interests in small language models (SLMs), which typically encompass sub- or a few billions of parameters and facilitate on-device deployments (Lu et al., 2024; Yuan et al., 2024). In practice, SLMs have been shipped to commercial off-the-shelf devices on a vast scale. For instance, the latest Google/Samsung phones have built-in LLM service (Gemini Nano), through which third-party mobile apps can freely enjoy LLM capability through text prompts or LoRA (Hu et al., 2021). Apple also introduces SLMs to facilitate privacy-preserving on-device intelligence tasks such as refining text and prioritizing notifications in iOS (Inc., 2024a).

On-device SLM deployment is extremely challenging due to the resource scarce of edge devices (Xu et al., 2024). While there has been plenty of open-sourced SLMs, e.g., Microsoft Phi family (Microsoft, 2024.04), that are claimed to be designed for resource-constrained devices, we found rare evidences supporting it except its relatively small parameter size. Motivated by the absence of a high-level principle for SLM design, we ask a question: *beyond using a small parameter size, what else can model developers do to better support on-device deployment with limited resources?*



(a) Chat (b) Android Intent Invocation

Figure 1: An end-to-end Android demo of PhoneLM's capability. (a) shows an example of a user having a conversation with PhoneLM-1.5B-Instruct; (b) shows an example of a user invokes an Android intent through chatting with PhoneLM-1.5B-Call.

In this work, we propose an intuitive yet effective principle for constructing on-device small language models: **searching for an resource-efficient architecture on a given hardware before pre-training**. It fundamentally differs from traditional SLM pipeline in that it moves the consideration of resource efficiency ahead of pre-training, while existing practice typically puts performance optimiza-

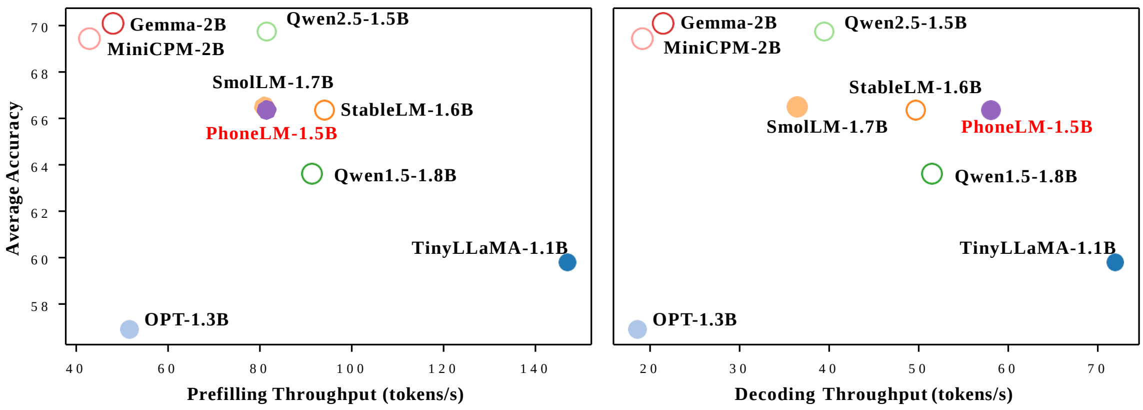


Figure 2: The comparison of the average accuracy and runtime performance between PhoneLM-1.5B and SLMs with similar parameter quantities (1B to 2B). The length of input prompt is 64 tokens. The average accuracy select seven NLP tasks to reflect the ability of the models (same as table reftab:performance), and the prefill/decode throughput is tested using the CPU on the Xiaomi 14 mobile phone. The closer the model is to the upper right corner, the better it is. Solid dots represent that the training data of the model is open source, and hollow dots represent that the training data of the model is closed source.

tions after pre-training (e.g., PTQ) but searches for an architecture with best capability (e.g., through observations on loss curve) (Hu et al., 2024). The principle is reasoned with two observations. (1) According to the scaling law (Kaplan et al., 2020), the final model accuracy is not sensitive to the model configurations in a wide range; yet our experiments in §2 demonstrate the opposite finding for inference speed, where the same-sized SLMs (1.5B) can run with up to 3.13× speed gap (compared with OPT-1.3B) on the same smartphone. (2) The cost of pre-training SLMs for different devices will be amortized by deploying SLM as a system-level service that delivers language ability to third-party apps, e.g., Google AICore (Inc., 2024b). In such circumstance, the pre-training cost of SLMs for each device is one shot, regardless of how many applications it serves (Yin et al., 2024).

Guided by this principle, we develop and release PhoneLM for smartphone hardware (e.g., Qualcomm Snapdragon SoC), a family of pre-trained and instructed SLMs. It now includes 5 model variants: PhoneLM-0.5B, PhoneLM-0.5B-Instruct, PhoneLM-1.5B, PhoneLM-1.5B-Instruct, and PhoneLM-1.5B-Call. The first two are base models, while the other three are finetuned for instruction following and system-level function call in Android. We also provide a few quantized versions to facilitate fast deployments.

There are three notable features of PhoneLM:

First, PhoneLM is extremely efficient through exhaustive ahead-of-pretraining architecture search on smartphone hardware. For instance, PhoneLM-1.5B runs at 58 tokens/second on Xiaomi 14 (Snapdragon 8Gen3 SoC) CPU, which is 1.2× faster than StableLM 2 1.6B and 1.6× faster than SmolLM 1.7B with similar parameter size. The prefilling speed of PhoneLM-1.5B even achieves 654 tokens/second on Xiaomi 14 NPU. The underlying architecture of PhoneLM is against recent SLM designs that converge to using SiLU (PhoneLM adopts ReLU) (Elfwing et al., 2018) and a width-height ratio between 54.6–88.6 (PhoneLM uses 134.7). Such architecture not only offers speed advantage on CPU, but also facilitates the NPU-friendly quantization (Xu et al., 2025) and sparse activation (Liu et al., 2023).

Second, PhoneLM achieves impressive language capability with a small parameter size, as shown in Figure 2. Across 7 typical benchmarks (listed in Table 5), PhoneLM-1.5B scores 67.3% accuracy on average, which is on par with the state-of-the-art SLMs with similar size trained on open datasets (i.e., SmolLM (HuggingFace, 2024.07) 1.7B and DCLM (Toyota, 2024.08) 1.4B). It even achieves better capability than many SLMs trained on proprietary datasets such as Qwen 1.5 1.8B and StableLM 2 1.6B. After finetuned, PhoneLM-1.5B is also capable of having smooth conversations with humans, and controlling smartphones using Android intent through function calls.

Third, PhoneLM is fully open-sourced, reproducible, and demonstrable. PhoneLM is trained on only open datasets without any manipulation. We release the complete codebase to develop PhoneLM, including the data preparation, training, fine-tuning, and evaluation procedures. To showcase the capability of PhoneLM in an end-to-end manner, we also release a demonstrable Android app powered by PhoneLM and mllm (Yi et al., 2023) engine. With the app, users can chat with PhoneLM on devices or invoke OS function calls with human language, as shown in Figure 1.

In a nutshell, PhoneLM achieves the state-of-the-art speed-capability tradeoff for smartphones among the SLMs trained on open datasets. We anticipate PhoneLM, as well as the underlying principle of its development, to bring the community to the attentions on the importance of algorithm-hardware co-design and co-optimizations in SLMs. PhoneLM has risks like being maliciously used to generate false content, so we recommend strict access control and monitoring mechanisms.

2 A Principle for SLM Development

SLM shall adapt to the target device hardware.

A key argument of this work is that, unlike on clouds, the SLM architecture and development shall adapt to the specific hardware for runtime efficiency as the first-class concern. Throughout this work, the "SLM architecture" mainly refers to the hyperparameters of transformer-decoder models, including the types of attention (MHA, GQA, etc.), activation function of feed forward network (FFN), depth and width of the model, etc.

Motivating experiments. To support the principle proposed, we test a bunch of SLMs with 100M and 200M parameters using various configurations on 2B tokens (dataset is the same as used to train PhoneLM). We then compare their loss on the same validation dataset. At the same time, we tested the inference speeds of these models using the inference engine *mllm* (Yi et al., 2023) on a smartphone equipped with the Snapdragon 8Gen3 SoC. The results of average metric (introduced in Section 4.2) and inference speed (throughput) are shown in figure 3. More details of these model architectures are shown in appendix A. We fit a quadratic curve to the loss of the 100M and 200M models when training on the same 2B tokens of data. Overall, fewer transformer layers, a larger model hidden size, and more attention heads tend to have faster inference speeds.

A key observation is that *runtime speed is more sensitive to the SLM architecture than the loss.* For a given model size, the range of its runtime speed is much wider than that of the loss. Comparing the SLMs with different sizes (100M and 200M), there is significant overlap of inference speed, but hardly any overlap of loss. In other words, a model with 200M parameters is consistently more capable than the one with 100M parameters, but does not always run slower on devices. The speed gap could be as large as 5× under the same model size. With more training tokens, the loss gap would even close up according to our experiments.

A principle of SLM development. Based on the insights, we present an intuitive yet effective principle for SLM development: search for the most efficient architecture on given hardware, then pre-train it on datasets with best quality and most quantity as possible. This principle differs from existing approaches that uses model capability as the target metric in SLM architecture search (Hu et al., 2024), leaving runtime optimizations in post-training stages.

3 PhoneLM: Smartphone-native SLM Family

Following the proposed principle, we developed and trained PhoneLM, a smartphone-native SLM family, with the following notable features: (1) Good runtime performance and capability. (2) Convenient for smartphone deployment and more suitable for model inference using NPU.

In this section, we present the architecture and training details of PhoneLM.

3.1 Architecture

Model Size	0.5B	1.5B
Hidden size	1,024	2,560
Intermediate Hidden Size	4,864	6,816
Heads	16	16
Layers	24	19
Vocab size	49,152	49,152
Context Len	2,048	2,048
Training Tokens	1.1T	1.5T

Table 1: PhoneLM hyperparameters and training settings. Notably, only PhoneLM-1.5B is developed with ahead-of-pretraining architecture search.

PhoneLM adopts a transformer decoder architecture with two variants (0.5B and 1.5B parame-

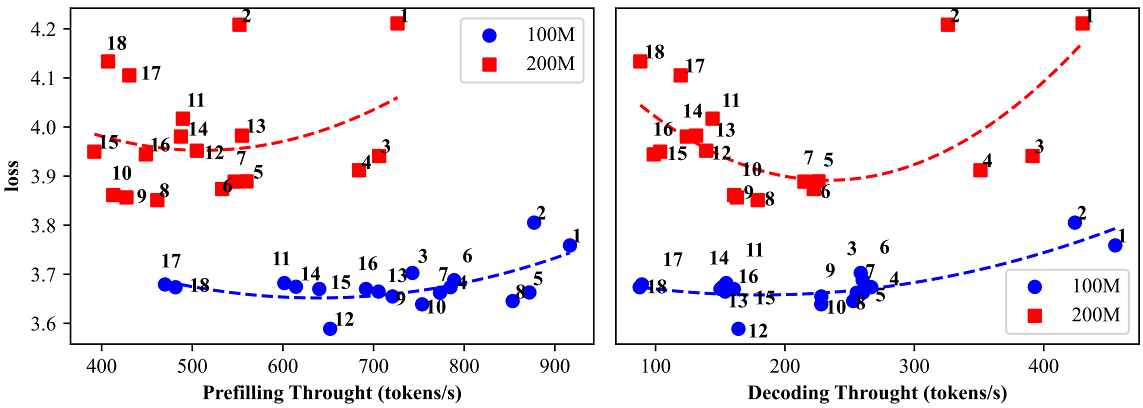


Figure 3: The comparison of the throughput and ability of the models with parameter quantities of 100M and 200M. More details of these model architecture are shown in appendix A

hidden	intermediate	layers	prefilling (tokens/s)	decoding (tokens/s)
2048	12288	16	70.75	55.12
2560	7680	18	64.98	60.60
2560	**6816**	**19**	**81.47**	**58.08**
2048	10240	19	68.52	54.48
1792	10752	21	65.42	50.18
2048	8192	22	67.10	54.04
1792	8960	25	63.29	48.63

Table 2: The throughput of models with multiple structures of 1.5B parameters on the Xiaomi 14 CPU (Snapdragon 8Gen3).

ters), as detailed in Table 1. `PhoneLM` featuring a context length of 2,048 tokens and utilize the tokenizer from SmolLM (HuggingFace, 2024.07), which supports a vocabulary size of 49,152. The models employ Rotary Position Embedding (RoPE) and multi-head attention mechanisms. The model adopts RMSNorm in place of LayerNorm as used in the traditional Transformer architecture. In their feed-forward components, they incorporate Gated Linear Units (GLU) mechanisms alongside ReLU activation functions.

Hardware-specific Hyperparameter Search for Resource Efficiency. To optimize `PhoneLM` for smartphone deployment, we conducted an exhaustive hyperparameter search on smartphone hardware. This search aimed to identify configurations that maximize runtime efficiency. Specifically, we explored a range of parameters including the number of layers, which varied from 15 to 25. We also examined the use of multi-head attention (MHA) with 16 heads and Grouped Query Attention (GQA) with 4 groups. Finally, we evaluated models with different ratios of intermediate hidden size to hidden size, ranging between 2 and 5.

Table 2 summarizes the throughput results for various 1.5B model structures tested on the Xiaomi 14 CPU (Snapdragon 8Gen3). Based on these experiment, we selected the configuration with the highest inference speed as the final structure for `PhoneLM`.

Activation Function Selection. Unlike recent SLMs that utilize SiLU or GELU, `PhoneLM` employs ReLU as its activation function. This choice is driven by two main factors. First, calculating ReLU is more efficient on smartphones, particularly for NPUs optimized for integer calculations. This efficiency makes ReLU a preferable choice for mobile devices where computational resources are limited. Second, ReLU introduces sparsity into the feed-forward network, which facilitates faster inference through techniques such as coefficient calculation. These techniques, discussed in detail by Song et al. (Song et al., 2023) and Alizadeh et al. (Alizadeh et al., 2023), leverage the sparsity introduced by ReLU to accelerate computations on mobile platforms.

Pre-quantized positional embedding. To further enhance computational efficiency on mobile devices, we apply INT8 quantization to the sin and cos values of RoPE. This linear quantization process scales floating-point values to the INT8 range [-128, 127]. Specifically, we first determine the maximum absolute values of sine and cosine functions, then scale the original values by dividing them by their respective maximum values and multiplying by 127, followed by rounding to the nearest integer. This approach minimizes accuracy loss while significantly improving computational efficiency on mobile accelerators such as NPUs.

type	dataset	token
web	DCLM-baseline (Li et al., 2024)	1.35T
code	StarCoderData (Li et al., 2023b)	112.75B
math	OpenWebMath (Paster et al., 2023)	13.25B
	Dolma-algebraic (Soldaini et al., 2024)	12.75B
academic	Dolma-arxiv (Soldaini et al., 2024)	29B
total		**1.5T**

(a) Stable Training Stage

type	dataset	token
web	DCLM-baseline (Li et al., 2024)	10B
code	StarCoderData (Li et al., 2023b)	1.575B
	The Stack Smol	0.95B
acadamic	Dolma-arxiv (Soldaini et al., 2024)	2.325B
	Dolma-pes2o (Soldaini et al., 2024)	2.35B
math instruct	MathInstruct (Yue et al., 2023)	65.25M
chat instruct	UltraChat (Ding et al., 2023)	1.775B
	OpenAssistant 2 (Köpf et al., 2024)	42.25M
	OpenHermes (Teknium, 2023)	77.25M
code instruct	Magicoder Evol Instruct (ise uiuc, 2024)	30.25M
	CommitPackFT (Muennighoff et al., 2023)	0.35B
	Magicoder OSS Instruct (Wei et al., 2023)	43.5M
	SlimOrca (Lian et al., 2023)	209.75M
function calling instruct	APIGen (Liu et al., 2024)	48.25M
	Glaive Function Calling (glaiveai, 2024)	57.5M
total		**20B**

(b) Decay Stage

type	dataset	token
math instruct	MathInstruct (Yue et al., 2023)	65.25M
chat instruct	UltraChat (Ding et al., 2023)	1.775B
	OpenAssistant 2 (Köpf et al., 2024)	42.25M
	OpenHermes (Teknium, 2023)	77.25M
code instruct	Magicoder Evol Instruct (ise uiuc, 2024)	30.25M
	CommitPackFT (Muennighoff et al., 2023)	0.35B
	Magicoder OSS Instruct (Wei et al., 2023)	43.5M
	SlimOrca (Lian et al., 2023)	209.75M
total		**2.59B**

(c) Instruct Turning Stage

Table 3: The classification of the datasets used in each stage and the number of their tokens. The description of the datasets is in appendix C.

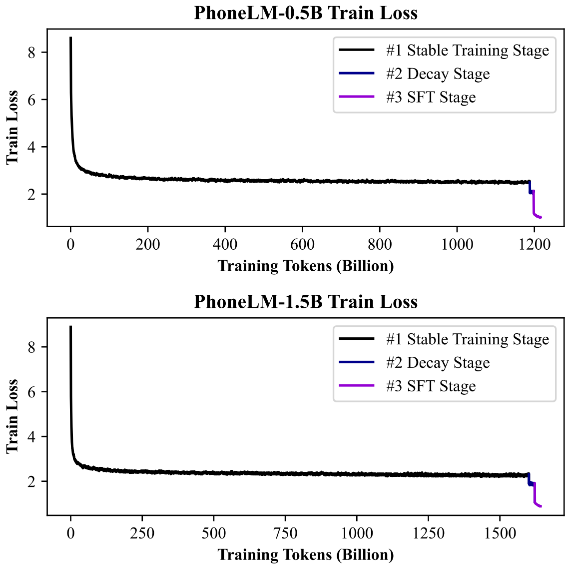


Figure 4: Training loss of PhoneLM-0.5B and PhoneLM-1.5B. This figure includes the loss in the Pre-training stage in Section 3.2 and the loss during Instruct Tuning stage in Section 3.3.

3.2 Pre-training

The training of PhoneLM has been set up as follows: (1) The optimizer is AdamW (Loshchilov, 2017) with β_1 of 0.9, β_2 of 0.95, and ϵ of 1e-8. (2) We use Fully Sharded Data Parallel (FSDP) to leverage multi-GPU and multi-node setups efficiently. (3) Another critical improvement is the integration of Flash Attention 2, an optimized attention mechanism. (4) We also use Zero Redundancy Optimizer(ZeRO), a memory optimization technique that reduces the models's memory footprint. (5) We use BF16 to accelerate the training process. The details of the setting of pre-training stage are shown in table 4.

We use a dataset sourced from open datasets. For PhoneLM-0.5B, we use 1.1 trillion tokens, and for PhoneLM-1.5B, we use 1.5 trillion tokens. In pre-training stage, we apply the weight decay, a learning rate warmup, and a cosine learning rate decay schedule.

PhoneLM is totally trained on open-sourced datasets without any manipulation, as shown in table 3. In the stable training stage, several open-source datasets are used, including DCLM-baseline, StarCoderData, OpenWebMath, Dolma. The details of the training datasets are shown in appendix C. The pre-training loss of PhoneLM family on the pretraining dataset is shown in figure 4 with black line.

3.3 Fine-tuning

The fine-tuning of PhoneLM base model is similar to MiniCPM (Hu et al., 2024) and Llama 3 (Dubey et al., 2024), which includes two stages: decay stage and Fine-tuning stage. (1) Decay Stage. We use a mixture of the pre-training data and high-quality supervised fine- tuning data, which is about 20 billion tokens. In this stage, we use a linear learning rate decay schedule. (2) Fine-tuning Stage. We find it still necessary to conduct a separate Fine-tuning stage. We utilize fine-tuning data similar to that in the decay phase but excludes pre-training data, totaling approximately 2.59 billion tokens. The learning rate for fine-tuning is set to match the final learning rate from the decay stage. The optimizer in the Fine-tuning stage is the same as that in the pre-training stage for acceleration, but with different hyperparameter settings, which are shown in the table 4.

Instruct Tuning. In the decay stage, the data mixture contains some dataset from stable training stage, including DCLM-baseline, StarCoderData,

stage	Stable	Decay	SFT
Datasets (tokens)	1.1TB	20B	2.59B
Learning Rate Scheduler	Cosine	Linear	None
Max Learning Rate	4e-04	8e-05	4e-05
Min Learning Rate	8e-05	4e-05	4e-05
Batch Size	13.5M	1.5M	32M
Epoch	1	1	7
Training Days (A100)	72×10	16×0.6	16×1

(a) PhoneLM-0.5B

stage	Stable	Decay	SFT
Datasets (tokens)	1.5TB	20B	2.59B
Learning Rate Scheduler	Cosine	Linear	None
Max Learning Rate	4e-04	4e-05	2e-05
Min Learning Rate	4e-05	2e-05	2e-05
Batch Size	9M	9M	128M
Epoch	1	1	8
Training Days (A100)	64×35	64×0.2	64×1

(b) PhoneLM-1.5B

Table 4: Training settings

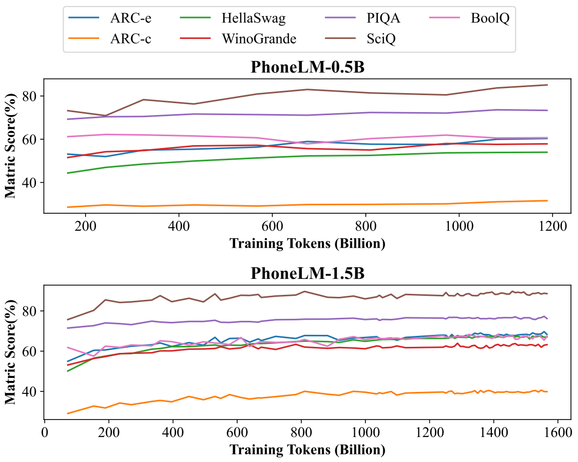


Figure 5: PhoneLM's performance across training iterations on standard zero-shot tasks

and Dolma. Then it contains some high-quality fine-tuning data,which is used in Fine-tuning stage. The fine-tuning datasets are shown in table 3, including APIGen, Stack Smol, UltraChat, MathInstruct, OpenAssistant 2, OpenHermes, CommitPackFT, OSS-Instruct, and SlimOrca. The details of these datasets are shown in appendix C. The pre-training loss of Decay Stage and Fine-tuning Stage is shown in figure 4 Since we continue fine-tuning the model after the decay stage, the loss drops significantly at the beginning of each epoch.

Function Call Tuning. To enhance the model's capability in smartphone operation, we fine-tuned the PhoneLM on the DroidCall (Xie et al., 2024) dataset, a synthetic dataset specifically focused on Android intent invocations generated by GPT-4. The DroidCall dataset includes 10k samples covering simple, parallel, and nested function call patterns for common Android operations. We use LoRA to fine-tune PhoneLM, adding adapter to all linear layers within both the attention layers and MLP layers The fine-tuning process was configured with an initial learning rate of 1.41e-5, utilizing a rank (r) of 8 and an alpha value of 16. A linear learning rate scheduler was employed with a warmup ratio of 0.1. To ensure a minimal computational load and to increase inference speed, we used a minimalist prompt, which essentially only included function information and user queries. The final function calling model was derived from the optimal checkpoint of the fine-tuning process. The details of prompt construction are shown in appendix E.

4 Experiment Results

We evaluate PhoneLM on a wide range of commonsense reasoning and problem-solving tasks and compare it to several existing open-source language models with similar model sizes.

4.1 Baselines and Tasks

We compare the PhoneLM family models with several existing open-source language models of similar model sizes. Table 5 lists all models used in the experiments. Gray text indicates models trained on datasets that are not publicly available, while black text denotes models trained on publicly available datasets.

To evaluate the capabilities of PhoneLM, we used 7 datasets from two domains: commonsense reasoning and problem solving. The commonsense reasoning datasets are HellaSwag (Zellers et al., 2019), Winogrande (Sakaguchi et al., 2020), PIQA (Bisk et al., 2020), SciQ (Welbl et al., 2017), and BoolQ (Clark et al., 2019). The problem solving ones are ARC Easy and ARC Challenge (Clark et al., 2018). Detailed descriptions are in the Appendix B.

We adopt the benchmark *lm_eval* (EleutherAI, 2024) to evaluate the models after the stable training stage. The primary evaluation metric is *accuracy*, which is the ratio of correct predictions to the total number of examples. For commonsense reasoning and problem-solving tasks, accuracy reflects the model's ability to choose correct options or offer accurate solutions. In line with previous practices, the models are evaluated in a zero-shot setting for these tasks. Our findings indicate that

Name	Size	Date	Training tokens	HellaSwag	WinoGrande	PIQA	SciQ	BoolQ	ARC Easy	ARC Challenge	Average
Pythia (EleutherAI, 2023.03b)	410M	23.03	207B	40.6	53.7	66.9	72.4	60.3	45.9	24.5	52.04
OPT (Facebook, 2022.05a)	350M	22.05	180B	36.8	52.3	64.3	68.5	57.6	40.1	23.7	49.04
BLOOM (BigScience, 2022.11a)	560M	22.11	350B	36.9	51.7	65.0	71.7	53.3	41.8	23.7	49.16
MobiLlama (MBZUAI, 2024.02)	500M	24.02	1.25T	51.1	53.4	70.9	76.4	55.7	46.0	26.6	54.30
OpenELM (Apple, 2024.04)	450M	24.04	1.5T	54.0	58.0	72.3	79.4	55.8	48.1	27.6	56.46
SmolLM (HuggingFace, 2024.07)	360M	24.07	600B	53.5	56.8	71.5	84.2	55.4	63.8	36.0	60.17
SmolLM2 (Allal et al., 2024)	360M	24.12	4T	56.3	58.6	71.9	86.4	61.4	68.3	37.7	62.94
Qwen1.5 (Alibaba, 2024.02)	500M	24.02	2.4T	49.2	55.7	69.5	82.5	49.5	52.3	29.4	55.44
Qwen2.5 (Team, 2024)	500M	24.09	12T	52.2	56.3	70.0	90.5	61.7	58.3	31.8	60.11
Cerebras-GPT (Cerebras, 2023.03b)	590M	23.03	371B	32.3	49.8	62.8	68.2	59.2	41.2	23.5	48.14
PhoneLM	**500M**	24.11	1.1T	**54.0**	**57.9**	**73.3**	**85.1**	**60.7**	**60.4**	**31.6**	**60.43**

(a) 0.5B

Name	Size	Date	Training tokens	HellaSwag	WinoGrande	PIQA	SciQ	BoolQ	ARC Easy	ARC Challenge	Average
Pythia (EleutherAI, 2023.03a)	1.4B	23.03	207B	52.0	57.2	71.1	79.2	63.2	53.9	28.3	57.84
OPT (Facebook, 2022.05b)	1.3B	22.05	180B	53.7	59.0	71.0	78.1	57.2	51.3	28.0	56.90
BLOOM (BigScience, 2022.11b)	1.1B	22.11	350B	43.0	54.9	67.2	74.6	59.1	45.4	25.6	52.83
TinyLlama (Unknown, 2023.12)	1.1B	23.12	3B	59.1	58.9	73.0	82.3	58.6	55.7	31.0	59.80
MobileLLaMA (Meituan, 2023.12)	1.4B	23.12	1.3T	56.1	59.4	73.0	81.9	56.7	55.8	30.3	59.03
MobiLlama (MBZUAI, 2024.02)	1B	24.02	1.25T	62.2	59.3	74.8	82.8	60.3	56.4	31.7	61.07
OpenELM (Apple, 2024.04)	1.1B	24.04	1.5T	64.8	61.7	75.6	83.6	63.6	55.4	32.3	62.43
DCLM (Toyota, 2024.08)	1.4B	24.08	4.3T	53.6	66.3	77.0	94.0	71.4	74.8	41.2	68.33
SmolLM (HuggingFace, 2024.07)	1.7B	24.07	1T	49.6	60.9	75.8	93.2	66.0	76.4	43.5	66.49
SmolLM2 (Allal et al., 2024)	1.7B	24.12	11T	71.5	65.9	77.5	90.9	72.4	73.3	47.6	71.30
Qwen1.5 (Alibaba, 2024.02)	1.8B	24.02	2.4T	60.9	60.5	74.2	89.4	66.5	59.1	34.7	63.61
Qwen2.5 (Team, 2024)	1.5B	24.09	7T	50.0	64.9	76.3	72.7	94.2	80.9	49.2	69.74
Galactica (Facebook, 2022.11)	1.3B	22.11	106B	41.0	54.4	63.8	87.7	62.0	58.6	30.5	56.86
StableLM2 (StabilityAI, 2024.01)	1.6B	24.01	2T	68.8	64.1	75.1	76.9	80.0	60.3	39.2	66.34
Cerebras-GPT (Cerebras, 2023.03a)	1.3B	23.03	371B	38.4	51.9	66.8	73.0	59.3	45.8	25.3	51.50
MiniCPM (OpenBMB, 2024.04)	1B	24.04	1.2T	67.5	63.7	75.1	91.0	70.5	62.9	38.1	66.97
MiniCPM (OpenBMB, 2024.04)	2B	24.04	1.2T	67.2	63.9	76.1	92.5	74.6	69.0	42.7	69.43
Gemma (Google, 2024.02)	2B	24.02	3T	71.4	65.2	78.4	91.4	69.9	72.3	42.0	70.09
Gemma2 (Google, 2024.07)	2B	24.07	2T	55.0	68.7	78.7	96.0	73.6	80.3	46.9	71.31
Llama3.2 (Dubey et al., 2024)	1B	24.09	9T	63.7	59.9	74.5	88.5	63.5	60.4	36.4	63.84
PhoneLM	**1.5B**	24.11	1.5T	**66.9**	**63.0**	**77.3**	**88.8**	**65.5**	**69.7**	**39.9**	**67.31**

(b) 1.5B

Table 5: Benchmark Score of PhoneLM. Models with gray text indicate that their training datasets are not publicly available.

PhoneLM outperforms the baselines in many tasks and achieves the highest average scores among most open-source models.In the Appendix B, we also evaluate these models on other tasks.

4.2 Capability

The capability for 7 standard zero-shot tasks of PhoneLM are presented in table 5. It can be seen from table 5(a) that PhoneLM-0.5B achieves the highest average accuracy on these 7 tasks. Except for the two tasks of ARC-e and ARC-c, where PhoneLM-0.5B performs lower than SmolLM, PhoneLM-0.5B demonstrates the strongest performance on other tasks among models with similar parameter counts. For PhoneLM-1.5B, which is shown in table 5(b), it performs better than other open-source models on most tasks. Combining all the tasks, it can be seen that PhoneLM performs better than other models with the same number of parameters in commonsense reasoning tasks and problem solving tasks.

In figure 5, the accuracy of PhoneLM-0.5B and PhoneLM-1.5B are plotted against training iterations for 7 standard zero-shot tasks. We observe an overall increase in accuracy with longer training durations across most tasks.

4.3 Instruction and Function Call

Instruction Following Evaluation. We have attached examples of PhoneLM-1.5B-Instruction in several scenarios, including "Reasoning", "Knowledge", "Programming and Logic Building", "Innovative Thinking", "Translation", and "Creativity and Imagination" in Appendix D.

Function call Evaluation. To assess the model's intrinsic function calling capabilities, we designed structured prompts to systematically guide the chat model in executing function calls. Following the fine-tuning methodology outlined in Section 3.3, we adapted the PhoneLM and evaluated multiple mainstream models on the DroidCall benchmark (Xie et al., 2024). The experimental results, as shown in Table 6, demonstrate the effectiveness of our approach.

4.4 On-device Runtime Cost

Hardware and framework. To benchmark PhoneLM models on the Android smartphone, we used a Xiaomi 14 with a Qualcomm Snapdragon 8 Gen 3 SoC and 16GiB of RAM. The smartphone was set to performance mode for stable results. The inference engine is *mllm* (Yi et al., 2023). For CPU experiments, 4 threads were used. The weights of

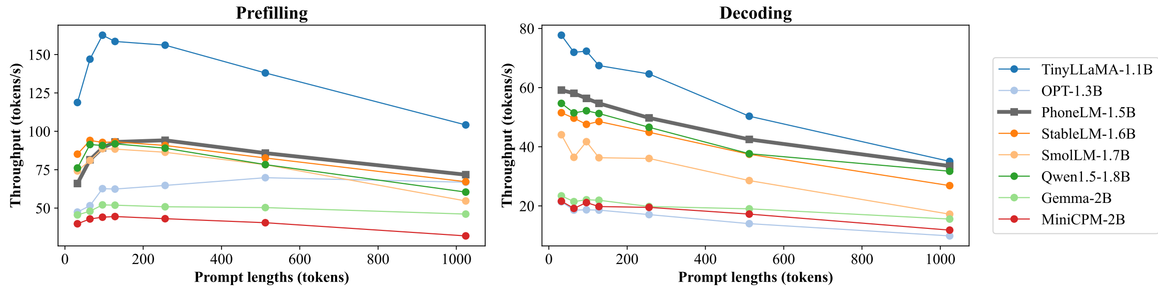


Figure 6: PhoneLM's throughput. This figure illustrates the comparison of throughput between PhoneLM and other models with similar parameter sizes on the Xiaomi 14 mobile phone, under varying input prompt lengths. All models were inferred using the mobile phone's CPU with 4 threads.

Model	Accuracy	Soft Accuracy
Qwen2.5-Coder-1.5B	50.0	63.5
Qwen2.5-1.5B-Instruct	58.5	75.3
Phi-3.5-mini-instruct	62.0	77.7
MiniCPM3-4B	70.0	85.7
Gemma-2-2b-it	56.5	75.8
TinyLlama-1.1B-Chat-v1.0	18.0	18.7
Llama-3.2-1B-Instruct	36.0	43.8
Llama-3.2-3B-Instruct	47.5	57.9
GPT-4o-mini	71.0	86.1
PhoneLM-1.5B-Instruct	**17.5**	**17.8**
PhoneLM-1.5B-Call	**75.0**	**86.1**

Table 6: Performance comparison of different models on the DroidCall test set. *Accuracy*: A sample is correct only if all predicted functions and parameters exactly match ground-truth calls; accuracy is the ratio of fully correct samples to the total. *Soft Accuracy*: It evaluates partial correctness by averaging per-call scores, where each score reflects the ratio of correctly predicted parameters to the total required.

the linear and embedding layers were quantized to 4-bit, while activation values remained in fp32 during runtime. For NPU experiments, we used Qualcomm's QNN (Inc., 2024c) framework and methods from mllm-NPU (Xu et al., 2025).

Evaluation. We provide two separate measurements for token throughput: prefilling and decoding. In the benchmark experiments of the model, we set different prompt lengths ranging from 32 to 1024 tokens and generate 100 tokens in an autoregressive manner to measure the throughput in the prefilling stage and the decoding stage. We repeat 5 times for each model and take the average result. We use key-value caching in all experiments.

Results. Figure 6 shows CPU benchmark results. PhoneLM-0.5B has higher prefilling throughput than most 0.5B models except SmolLM-360M, and its decoding throughput surpasses all models. PhoneLM-1.5B outperforms models larger than 1.3B in both prefilling and decoding throughput. Throughput decreases as prompt length increases due to higher self-attention computational load. Figure 2 compares throughputs at 64 tokens prompt length against average metrics. Models in the upper right corner exhibit better performance and speed. Notably, PhoneLM-1.5B achieves 654 tokens/second prefilling throught on Xiaomi 14 NPU, outpacing Qwen2.5-1.5B at 602 tokens/second.

4.5 An End-to-end Android Demo

We also have an end-to-end Android demo application for PhoneLM-1.5B based on *mllm*. This demo contains two invocations: chat and Android intent invocation. The screenshots of this application are shown in figure 1. Figure 1(a) shows an example of a user having a conversation with an Android application with PhoneLM-1.5B-Instruct built in. Figure 1(b) shows the Android intent invocation ability of the PhoneLM-1.5B-Call model. In this example, after understanding the user's "Wake me up at 8:00", the model uses the Android alarm-setting Intent to set an alarm for 8 o'clock.

5 Conclusions

This work presents PhoneLM, an efficient, capable, and fully open-sourced small language family. PhoneLM is built atop a unique principle: searching for a runtime-efficient transformer architecture ahead of pre-training. We also release an end-to-end demo using PhoneLM for intent invocations on Android OS in a fast and accurate performance. The goal of PhoneLM is to advance the development and research on small language models towards more practical on-device deployment.

6 Limitations

Our approach tested 14 metrics (7 in the main text and 7 in the appendix), but only covered two types of tasks: commonsense reasoning and problem-solving. Metrics for other tasks applicable to language models were not experimented upon.

The Instruct model proposed in this paper lacks quantitative experimental metrics, primarily due to the absence of an effective method to evaluate the model's instruction-following capability.

Additionally, the third-party models compared in our experiments were SLMs released up to December 2024, thus excluding several SLMs launched in 2025.

References

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

Keivan Alizadeh, Iman Mirzadeh, Dmitry Belenko, Karen Khatamifard, Minsik Cho, Carlo C Del Mundo, Mohammad Rastegari, and Mehrdad Farajtabar. 2023. Llm in a flash: Efficient large language model inference with limited memory. *arXiv preprint arXiv:2312.11514*.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Lewis Tunstall, Agustín Piqueres, Andres Marafioti, Cyril Zakka, Leandro von Werra, and Thomas Wolf. 2024. Smollm2 - with great data, comes great performance.

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

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. 2023.03a. cerebras/cerebras-gpt-1.3b. https://huggingface.co/cerebras/Cerebras-GPT-1.3B.

Cerebras. 2023.03b. cerebras/cerebras-gpt-590m. https://huggingface.co/cerebras/Cerebras-GPT-590M.

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*.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. 2023. Enhancing chat language models by scaling high-quality instructional conversations. *arXiv preprint arXiv:2305.14233*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

EleutherAI. 2023.03a. Eleutherai/pythia-1.4b. https://huggingface.co/EleutherAI/pythia-1.4b.

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

EleutherAI. 2024. Language model evaluation harness.

Stefan Elfwing, Eiji Uchibe, and Kenji Doya. 2018. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural networks*, 107:3–11.

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

Facebook. 2022.05b. facebook/opt-1.3b. https://huggingface.co/facebook/opt-1.3b.

Facebook. 2022.11. facebook/galactica-1.3b. https://huggingface.co/facebook/galactica-1.3b.

glaiveai. 2024. glaive function calling.

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

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

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*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, et al. 2024. Minicpm: Unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*.

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Jiayi Lei, Yao Fu, Maosong Sun, and Junxian He. 2023. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. In *Advances in Neural Information Processing Systems*.

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

Apple Inc. 2024a. ios 18. https://www.apple.com/ios/ios-18/.

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

Qualcomm Inc. 2024c. Qualcomm neural processing sdk for ai. https://www.qualcomm.com/developer/software/neural-processing-sdk-for-ai.

ise uiuc. 2024. Magicoder evol instruct.

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*.

Denis Kocetkov, Raymond Li, Loubna Ben Allal, Jia Li, Chenghao Mou, Carlos Muñoz Ferrandis, Yacine Jernite, Margaret Mitchell, Sean Hughes, Thomas Wolf, et al. 2022. The stack: 3 tb of permissively licensed source code. *arXiv preprint arXiv:2211.15533*.

Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Duc Nguyen, Oliver Stanley, Richárd Nagyfi, et al. 2024. Openassistant conversations-democratizing large language model alignment. *Advances in Neural Information Processing Systems*, 36.

Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. 2023a. Cmmlu: Measuring massive multitask language understanding in chinese. *Preprint*, arXiv:2306.09212.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, Hritik Bansal, Etash Guha, Sedrick Keh, Kushal Arora, Saurabh Garg, Rui Xin, Niklas Muennighoff, Reinhard Heckel, Jean Mercat, Mayee Chen, Suchin Gururangan, Mitchell Wortsman, Alon Albalak, Yonatan Bitton, Marianna Nezhurina, Amro Abbas, Cheng-Yu Hsieh, Dhruba Ghosh, Josh Gardner, Maciej Kilian, Hanlin Zhang, Rulin Shao, Sarah Pratt, Sunny Sanyal, Gabriel Ilharco, Giannis Daras, Kalyani Marathe, Aaron Gokaslan, Jieyu Zhang, Khyathi Chandu, Thao Nguyen, Igor Vasiljevic, Sham Kakade, Shuran Song, Sujay Sanghavi, Fartash Faghri, Sewoong Oh, Luke Zettlemoyer, Kyle Lo, Alaaeldin El-Nouby, Hadi Pouransari, Alexander Toshev, Stephanie Wang, Dirk Groeneveld, Luca Soldaini, Pang Wei Koh, Jenia Jitsev, Thomas Kollar, Alexandros G. Dimakis, Yair Carmon, Achal Dave, Ludwig Schmidt, and Vaishaal Shankar. 2024. Datacomp-lm: In search of the next generation of training sets for language models. *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, et al. 2023b. Starcoder: may the source be with you! *arXiv preprint arXiv:2305.06161*.

Wing Lian, Guan Wang, Bleys Goodson, Eugene Pentland, Austin Cook, Chanvichet Vong, and "Teknium". 2023. Slimorca: An open dataset of gpt-4 augmented flan reasoning traces, with verification.

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

Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Re, et al. 2023. Deja vu: Contextual sparsity for efficient llms at inference time. In *International Conference on Machine Learning*, pages 22137–22176. PMLR.

Zuxin Liu, Thai Hoang, Jianguo Zhang, Ming Zhu, Tian Lan, Shirley Kokane, Juntao Tan, Weiran Yao, Zhiwei Liu, Yihao Feng, et al. 2024. Apigen: Automated pipeline for generating verifiable and diverse function-calling datasets. *arXiv preprint arXiv:2406.18518*.

I Loshchilov. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Zhenyan Lu, Xiang Li, Dongqi Cai, Rongjie Yi, Fangming Liu, Xiwen Zhang, Nicholas D Lane, and Mengwei Xu. 2024. Small language models: Survey, measurements, and insights. *arXiv preprint arXiv:2409.15790*.

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

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

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

Niklas Muennighoff, Qian Liu, Armel Zebaze, Qinkai Zheng, Binyuan Hui, Terry Yue Zhuo, Swayam Singh, Xiangru Tang, Leandro von Werra, and Shayne Longpre. 2023. Octopack: Instruction tuning code large language models. *arXiv preprint arXiv:2308.07124*.

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

Keiran Paster, Marco Dos Santos, Zhangir Azerbayev, and Jimmy Ba. 2023. Openwebmath: An open dataset of high-quality mathematical web text. *Preprint*, arXiv:2310.06786.

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*.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. 2019. Socialiqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*.

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*.

Yixin Song, Zeyu Mi, Haotong Xie, and Haibo Chen. 2023. Powerinfer: Fast large language model serving with a consumer-grade gpu. *arXiv preprint arXiv:2312.12456*.

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

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Teknium. 2023. openhermes: An open dataset of synthetic data for generalist llm assistants.

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

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

Yuxiang Wei, Zhe Wang, Jiawei Liu, Yifeng Ding, and Lingming Zhang. 2023. Magicoder-oss-instruct-75k: An open dataset generated by gpt-3.5-turbo-1106.

Johannes Welbl, Nelson F Liu, and Matt Gardner. 2017. Crowdsourcing multiple choice science questions. *arXiv preprint arXiv:1707.06209*.

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. 2025. Fast on-device llm inference with npus.

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

Rongjie Yi, Xiang Li, Qichen Qiu, Zhenyan Lu, Hao Zhang, Daliang Xu, Liming Yang, Weikai Xie, Chenghua Wang, and Mengwei Xu. 2023. mllm: fast and lightweight multimodal llm inference engine for mobile and edge devices.

Wangsong Yin, Mengwei Xu, Yuanchun Li, and Xuanzhe Liu. 2024. Llm as a system service on mobile devices. *arXiv preprint arXiv:2403.11805*.

Jinliang Yuan, Chen Yang, Dongqi Cai, Shihe Wang, Xin Yuan, Zeling Zhang, Xiang Li, Dingge Zhang, Hanzi Mei, Xianqing Jia, et al. 2024. Mobile foundation model as firmware. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, pages 279–295.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. 2023. Mammoth: Building math generalist models through hybrid instruction tuning. *arXiv preprint arXiv:2309.05653*.

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*.

A Setting of 100M and 200M models

Tested the speed and performance of 100M and 200M models, training on data with 20 billion tokens. The settings are shown in the table 7.

B Matrics

We also evaluate the models on other following tasks, which contains the following tasks: SocialIQA, TruthfulQA, MMLU, CMMLU and C-Eval. The following are the datasets used in all experiments:

- **Commonsense Reasoning Datasets**:
  - **HellaSwag** (Zellers et al., 2019): Tests narrative understanding through plausible sentence completion.
  - **Winogrande** (Sakaguchi et al., 2020): Evaluates pronoun ambiguity resolution using commonsense reasoning.
  - **PIQA** (Bisk et al., 2020): Focuses on physical commonsense reasoning and object interactions.

ID	size(M)	hidden	intermediate	layers	activation	q heads	kv heads	loss	prefilling (tokens/s)	decoding (tokens/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

(a) 100M

ID	size(M)	hidden	intermediate	layers	activation	q heads	kv heads	loss	prefilling (tokens/s)	decoding (tokens/s)
1	201.32	2048	5460	2	relu	16	16	4.21	726.44	430.06
2	201.32	2048	5460	2	silu	16	16	4.21	552.06	325.93
3	188.76	1536	4096	4	relu	16	16	3.94	706.14	391.36
4	188.76	1536	4096	4	silu	16	16	3.91	683.97	351.09
5	199.78	1024	2688	12	relu	16	16	3.89	559.80	225.88
6	199.78	1024	2688	12	relu	4	4	3.87	533.00	222.27
7	199.78	1024	2688	12	relu	16	4	3.89	546.76	215.04
8	199.78	1024	2688	12	silu	16	16	3.85	461.42	178.95
9	199.78	1024	2688	12	silu	4	4	3.86	427.38	162.85
10	199.78	1024	2688	12	silu	16	4	3.86	412.81	160.71
11	182.20	704	1856	25	relu	16	16	4.02	489.62	144.05
12	182.20	704	1856	25	relu	4	4	3.95	505.01	139.14
13	182.20	704	1856	25	relu	16	4	3.98	554.88	131.29
14	182.20	704	1856	25	silu	16	16	3.98	487.49	124.17
15	182.20	704	1856	25	silu	4	4	3.95	391.94	103.51
16	182.20	704	1856	25	silu	16	4	3.94	448.85	98.58
17	187.61	576	1536	40	relu	16	16	4.11	430.52	119.42
18	187.61	576	1536	40	silu	16	16	4.13	407.08	88.21

(b) 200M

Table 7: 100M and 200M models' setting

  - **SciQ** (Welbl et al., 2017): a dataset of 13.7K multiple choice science exam questions.
  - **BoolQ** (Clark et al., 2019): Tests commonsense and factual reasoning with yes/no questions.
  - **TruthfulQA** (Lin et al., 2022): Assesses the model's ability to avoid providing false information.
  - **SocialIQA** (Sap et al., 2019): A dataset of 13.7K multiple choice science exam questions.

- **Problem Solving Datasets**:
  - **ARC Easy** (Clark et al., 2018): Contains simple science questions testing general knowledge and reasoning.
  - **ARC Challenge** (Clark et al., 2018): Presents complex science exam questions requiring knowledge integration.
  - **MMLU** (Hendrycks et al., 2021): Evaluates problem-solving across diverse academic disciplines.
  - **CMMLU** (Li et al., 2023a): Evaluates problem-solving across diverse academic disciplines in Chinese.
  - **C-Eval Valid** (Huang et al., 2023): A comprehensive Chinese evaluation suite for foundation models. It consists of 13948 multi-choice questions spanning 52 diverse disciplines and four difficulty levels.

The results are presented in table 8.

name	SocialIQA	TruthfulQA MC1	TruthfulQA MC2	TruthfulQA zh MC2	MMLU	CMMLU	C-Eval Valid
Pythia-410M	32.9	23.7	41.2	47.9	23.6	25.3	23
OPT-350M	32.9	23.3	40.8	47.3	23.1	25.4	22.5
BLOOM-560M	34.2	25	42.4	41.6	23	25.3	23
MobiLlama-0.5B	32.9	23.3	37.5	42	24.9	25.3	21.6
OpenELM-450M	32.8	24.8	40.2	47.5	25.9	24.9	22.7
SmolLM-360M	32.9	24.6	37.9	47.4	25.8	25.4	25.7
SmolLM2-360M	40.9	21.5	33.5	44.2	25.6	24.7	22.4
Qwen1.5-0.5B	33.3	23.6	38.3	41.3	36.5	47.4	49.9
Qwen2.5-0.5B	44.4	25.1	40.0	42.8	47.6	48.7	51.8
Cerebras-GPT-590M	33.1	25.1	44.1	47.5	23.1	25.3	22.9
PhoneLM-0.5B	**42.5**	**21.9**	**36.5**	**43**	**25.4**	**24.6**	**23.6**

(a) 0.5B

name	SocialIQA	TruthfulQA MC1	TruthfulQA MC2	TruthfulQA zh MC2	MMLU	CMMLU	C-Eval Valid
Pythia-1.4B	33.6	22.8	38.9	44.9	24.4	25.3	23
OPT-1.3B	32.7	24.1	38.7	47	25.2	25.3	23
BLOOM-1.1B	33.5	25.3	41.8	40.5	24	25.4	24.1
TinyLlama-1.1B	32.9	22	37.3	42.6	24.9	24.7	24.2
MobileLLaMA-1.4B	33	21.7	34.8	43.5	24.5	25.2	23.1
MobiLlama-1B	32.9	21.7	35.2	41.6	25.4	25.4	25.3
OpenELM-1.1B	32.7	22.2	37	47.3	25.3	25.3	23.4
DCLM-1B	44.3	22.8	36.5	43.8	46.5	30.6	29.3
SmolLM-1.7B	43.6	24.4	38.5	44.8	27.7	25.2	24.5
SmolLM2-1.7B	44.5	25.1	36.6	42.9	45.9	30	32.3
Qwen1.5-1.8B	32.9	25.8	39.4	42.3	45.4	59.1	61.1
Qwen2.5-1.5B	49.1	30.2	47.6	46.4	60.9	66.5	67.7
Galactica-1.3B	32.5	24.8	41.4	47.1	27.7	25.2	22.7
StableLM2-1.6B	32.9	30.6	45.1	48.8	41.1	30.3	31.6
Cerebras-GPT-1.3B	32.8	24.5	42.7	46.2	23	25.3	23
MiniCPM-1B	32.6	23.1	36.9	37.4	44.9	47.4	47
MiniCPM-2B	32.9	25.2	40.5	41.8	45.6	44.4	43.2
Gemma-2B	33	22.2	33.1	43.6	32.9	28.4	26.1
Gemma2-2B	51.2	24.1	36.2	41.3	49.6	34.5	35
Llama3.2-1B	43.0	23.6	37.6	42.7	36.2	29.2	29.9
PhoneLM-1.5B	**43.2**	**20.9**	**33.3**	**46.1**	**26.5**	**25.0**	**25.7**

(b) 1.5B

Table 8: Some Benchmark Score of PhoneLM. Models with gray text indicate that their training datasets are not publicly available.

C Training Dataets

DCLM-baseline (Li et al., 2024) is a 4T token and 3B document pretraining dataset that achieves strong performance on language model benchmarks.PhoneLM only uses a maximum of 1.5T among it. The code is publiced in https://huggingface.co/datasets/mlfoundations/dclm-baseline-1.0-parquet.

StarCoderData (Li et al., 2023b) contains 783GB of code in 86 programming languages, and includes 54GB GitHub Issues + 13GB Jupyter notebooks in scripts and text-code pairs, and 32GB of GitHub commits, which is approximately 250 Billion tokens. The code is publiced in https://huggingface.co/datasets/bigcode/starcoderdata.

OpenWebMath (Paster et al., 2023) is a dataset containing the majority of the high-quality, mathematical text from the internet. It is filtered and extracted from over 200B HTML files on Common Crawl down to a set of 6.3 million documents containing a total of 14.7B tokens. The code is publiced in https://huggingface.co/datasets/open-web-math/open-web-math.

Dolma (Soldaini et al., 2024) is a dataset of 3 trillion tokens from a diverse mix of web content, academic publications, code, books, and encyclopedic materials. The code is publiced in https://huggingface.co/datasets/allenai/dolma.

APIGen (Liu et al., 2024) contains 60,000 data collected by APIGen, an automated data generation pipeline designed to produce verifiable high-quality datasets for function-calling applications. The code is publiced in https://huggingface.co/datasets/Salesforce/xlam-function-calling-60k.

The Stack Smol (Kocetkov et al., 2022) is a small subset of the-stack dataset, each programming language has 10,000 random samples from the original dataset. The code is publiced in https://huggingface.co/datasets/bigcode/the-stack-smol.

UltraChat (Ding et al., 2023) is an open-source, large-scale, and multi-round dialogue data powered by Turbo APIs. The code is publiced in https://huggingface.co/datasets/stingning/ultrachat.

MathInstruct (Yue et al., 2023) is a meticulously curated instruction tuning dataset that is lightweight yet generalizable. The code is publiced in https://huggingface.co/datasets/TIGER-Lab/MathInstruct.

OpenAssistant 2 (Köpf et al., 2024) contains message trees. Each message tree has an initial prompt message as the root node, which can have multiple child messages as replies, and these child messages can have multiple replies. The code is publiced in https://huggingface.co/datasets/OpenAssistant/oasst2.

OpenHermes (Teknium, 2023) dataset is composed of 242,000 entries of primarily GPT-4 generated data, from open datasets across the AI landscape. The code is publiced in https://huggingface.co/datasets/teknium/openhermes.

CommitPackFT (Muennighoff et al., 2023) is a 2GB filtered version of CommitPack to contain only high-quality commit messages that resemble natural language instructions. The code is publiced in https://huggingface.co/datasets/bigcode/commitpackft.

OSS Instruct dataset (Wei et al., 2023) is generated by gpt-3.5-turbo-1106 developed by OpenAI. The code is publiced in https://huggingface.co/datasets/ise-uiuc/Magicoder-OSS-Instruct-75K.

SlimOrca (Lian et al., 2023) release provides an efficient means of reaching performance on-par with using larger slices of our data, while only including 500k GPT-4 completions. The code is publiced in https://huggingface.co/datasets/Open-Orca/SlimOrca.

D Instruct Following Examples

Reasoning:

Prompt: What are the key benefits of practicing mindfulness meditation?

PhoneLM-1.5B-Instruct:
There are many benefits of practicing mindfulness meditation, including:

1. Reduced stress and anxiety
2. Improved focus and concentration
3. Increased self-awareness and emotional regulation
4. Improved sleep quality
5. Reduced symptoms of depression
6. Increased feelings of well-being and happiness
7. Improved immune system function
8. Reduced symptoms of chronic pain
9. Improved relationships and communication skills
10. Increased feelings of compassion and empathy.

Knowledge:

Prompt: What are some ways to reduce the carbon footprint of your home?

PhoneLM-1.5B-Instruct:
There are several ways to reduce the carbon footprint of your home. Here are some suggestions:

1. Install energy-efficient appliances and lighting.
2. Use low-flow showerheads and faucets.
3. Install a programmable thermostat to regulate heating and cooling.
4. Use natural light and ventilation instead of artificial lighting and air conditioning.
5. Plant trees or install shading devices to block the sun's heat.
6. Use eco-friendly cleaning products and reduce the use of disposable products.
7. Reduce water usage by fixing leaks and installing low-flow fixtures.
8. Use renewable energy sources such as solar panels or wind turbines.
9. Reduce waste by recycling, composting, and using reusable products.
10. Choose sustainable building materials and design your home to be energy-efficient.

Programming and Logic Building:

Prompt: Generate a Python code snippet that implements a function to calculate the Fibonacci sequence using recursion. Bonus points if you can make it more efficient using memoization.

PhoneLM-1.5B-Instruct:

```
def fibonacci(n):
  if n <= 1:
    return n
  else:
    return fibonacci(n-1) +
           fibonacci(n-2)


# Test the function
print(fibonacci(0)) # 0
print(fibonacci(1)) # 1
print(fibonacci(2)) # 1
print(fibonacci(3)) # 2
print(fibonacci(4)) # 3
print(fibonacci(5)) # 5
print(fibonacci(6)) # 8
print(fibonacci(7)) # 13
print(fibonacci(8)) # 21
print(fibonacci(9)) # 34
print(fibonacci(10)) # 55
```

Innovative Thinking:

Prompt: Construct a sentence that uses every letter of the alphabet at least once.

PhoneLM-1.5B-Instruct:
Able was I ere I saw Elba, but ere I saw Elba, I heard a rumor that would change my life forever.

Translation:

Prompt: Translate this sentence into French: "Ming due to the circumstances, the knower system according to the matter."

PhoneLM-1.5B-Instruct:
"Ming, parce qu'il y a des circonstances, le système d'apprentissage selon le matériel."

Creativity and Imagination:

Prompt: Write a role-play with a tourist and a travel guide for a finctional planet.

PhoneLM-1.5B-Instruct:
Tourist: Hi, I'm looking for a planet that's not too crowded. Can you recommend one?

Travel Guide: Sure, I can recommend a few. One of the most popular planets is Zorion. It's a bit more remote, but it's still accessible.

Tourist: That sounds great. What's the best way to get there?

Travel Guide: You can take a shuttle from the nearest spaceport. It's about a 3-hour journey.

Tourist: That sounds doable. What's the atmosphere like on Zorion?

Travel Guide: It's a bit different from Earth. The atmosphere is mostly nitrogen and oxygen, but there's also a lot of carbon dioxide. It's a bit more humid too.

Tourist: That's interesting. What's the terrain like?

Travel Guide: It's mostly rocky and barren, but there are some forests and lakes. It's a bit like Mars, but with more vegetation.

Tourist: That sounds like a unique experience. Are there any popular tourist attractions on Zorion?

Travel Guide: Yes, there are a few. One of the most popular is the Zorion Museum, which showcases the planet's history and culture. Another is the Zorion Observatory, which is a great place to see the stars.

Tourist: That sounds amazing. Is there anything else I should know before I go?

Travel Guide: Just be prepared for the weather. It can get pretty hot and humid, so bring a lot of water and sunscreen. And don't forget your passport, as Zorion is a popular tourist destination.

Tourist: Thanks for the tips. I'm excited to visit Zorion.

E Function Calling fine-tuning details

We adopt a chat-based approach for function calling, where the requirements, use query, and function descriptions are encapsulated in the system prompt and user prompt sections, while the function calls are placed in the assistant output section. The prompt design for function calling fine-tuning thus focuses on structuring these three components: system prompt, user message, and assistant output. These components are subsequently formatted using a chat template, which is then utilized for model fine-tuning. These components are shown in 1, in which $function is the functions description information, which describes the function name, parameters, and other information, $user_query is the user input.

Listing 1: Minimalist prompt of function calling

```
System Prompt:

You are an expert in composing functions.

User message:

Here is a list of functions that you can
      invoke:
$functions
Now my query is: $user_query

Assistant output:

$result1 = func0(arg1="v1", arg2="v2", ...)
result2 = func1(arg1="v1", arg2=result1, ...)
... $
```