MobileLLM-Flash: Latency-Guided On-Device LLM Design for Industry Scale Deployment

Hanxian Huang*, **Igor Fedorov***, **Andrey Gromov, Bernard Beckerman, Naveen Suda, David Eriksson, Maximilian Balandat, Rylan Conway, Patrick Huber, Chinnadhurai Sankar, Ayushi Dalmia, Zechun Liu, Lemeng Wu, Tarek Elgamal, Adithya Sagar, Vikas Chandra, Raghuraman Krishnamoorthi**

Meta AI

*Co-first authors

Real-time AI experiences call for on-device large language models (OD-LLMs) optimized for efficient deployment on resource-constrained hardware. The most useful OD-LLMs produce near-real-time responses and exhibit broad hardware compatibility, maximizing user reach. We present a methodology for designing such models using hardware-in-the-loop architecture search under mobile latency constraints. This system is amenable to industry-scale deployment: it generates models deployable without custom kernels and compatible with standard mobile runtimes like Executorch. Our methodology avoids specialized attention mechanisms and instead uses attention skipping for long-context acceleration.

Our approach jointly optimizes model architecture (layers, dimensions) and attention pattern. To efficiently evaluate candidates, we treat each as a pruned version of a pretrained backbone with inherited weights, thereby achieving high accuracy with minimal continued pretraining. We leverage the low cost of latency evaluation in a staged process: learning an accurate latency model first, then searching for the Pareto-frontier across latency and quality.

This yields MobileLLM-Flash, a family of foundation models (350M, 650M, 1.4B) for efficient on-device use with strong capabilities, supporting up to 8k context length. MobileLLM-Flash delivers up to $\mathbf{1.8}\times$ and $\mathbf{1.6}\times$ faster prefill and decode on mobile CPUs with comparable or superior quality. Our analysis of Pareto-frontier design choices offers actionable principles for OD-LLM design.

Date: April 29, 2026

Correspondence: hanxianhuang@meta.com, ifedorov@meta.com

Meta

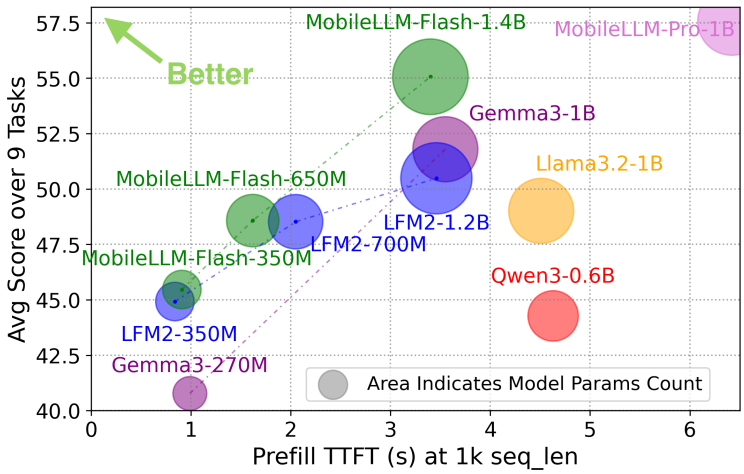

Figure 1 Comparison between MobileLLM-Flash and state-of-the-art OD-LLMs. MobileLLM-Flash achieves up to $\mathbf{1.8}\times$ /$\mathbf{1.6}\times$ faster prefill/decode on mobile CPUs with superior accuracy than LFM2 Amini et al. (2025). Evaluation details are in Sec. 4.2.

1 Introduction

Deployment of efficient on-device large language models (OD-LLMs) on resource-constrained devices, e.g., mobile phones and smart glasses, is critical for enabling real-time AI experiences. The two distinguishing features of real-world on-device AI assistants serving large scale industry-grade traffic are: 1) They must operate **near-real-time**, with a specific emphasis on time-to-first-token (TTFT, i.e., time from receiving the request to outputting the first generated token) since the decode rate can often be hidden by streaming the model output back to the user; 2) They must be close to **general-purpose** from a runtime perspective, avoiding complex building blocks that require specialized kernels. These requirements ensure models operate across diverse software and hardware stacks (Android, iPhone, wearables).

The real-time requirement places an upper bound on TTFT, and therefore, on the number of prefill tokens. While a 4s TTFT can still yield a reasonable user experience, a 10s TTFT does not Nielsen (1994); Kim et al. (2026). As such, models should be optimized for the most practical use cases. We find that $\sim$ 2k tokens is a practical sweet spot where OD-LLMs can both perform useful tasks and achieve TTFT $\leq$ 4s (Tab. 6).

Many existing efficient LLM designs often optimize proxy metrics, such as parameter count or floating-point operations per second (FLOPs) rather than measured on-device latency, or focus on server-side optimizations Fu et al. (2025); Gu et al. (2025); Yang et al. (2025b). However, these methods do not adequately capture the practical constraints and performance bottlenecks encountered in real-world on-device deployment, leaving a gap for methodologies grounded in empirical latency and hardware-in-the-loop evaluation. Furthermore, some designs relying on specialized kernels for efficient attention Gu et al. (2025); Yang et al. (2025c); Dao and Gu (2024) face barriers to large-scale adoption due to limited portability.

Our main contributions are:

- To our knowledge, the first joint architecture and attention pattern NAS for OD-LLMs, directly optimizing mobile CPU latency via a novel two-stage Bayesian optimization flow.
- A pruning-based search that inherits pretrained weights, using only 35% of the training tokens required by from-scratch training, making our hardware-in-the-loop LLM NAS practical.
- MobileLLM-Flash (350M, 650M, 1.4B) (Fig. 1), a family of deployment-ready OD-LLMs with up to 1.8$\times$ prefill and 1.6$\times$ decode speedup on mobile CPUs, runnable out of the box without specialized kernels.
- We present an analysis of the latency-accuracy Pareto frontier, deriving actionable design principles to guide future OD-LLM development. Our search reveals that interleaved skip-attention consistently outperforms sliding-window attention on mobile CPUs.

2 Related Work

OD-LLM design and architecture search. While prior works Liu et al. (2024); Huber et al. (2025) achieve parameter efficiency, their deep-and-thin structures often fail to improve on-device latency. Recent studies Acun et al. (2025); Gu et al. (2025); Fu et al. (2025); Yang et al. (2025b); Cowsik et al. (2025) use neural architecture search (NAS) to discover OD-LLMs, but focus on either parameter efficiency or server-side GPU optimization. LFM2 Amini et al. (2025) introduces an edge-optimized model family based on a new block gated short convolution attention, but the underlying `conv1d` operator can perform poorly at small batch sizes or sequence lengths Heinecke et al. (2017). Latency-aware NAS has been explored for server-scale 50–70B LLMs (Puzzle Bercovich et al. (2025)) and diffusion models (NanoSD Sanyal et al. (2026)), but neither searches attention patterns nor targets mobile CPU latency for on-device scale LLMs, which is the focus of our work.

Efficient attention alternatives and hybrid models. Sub-quadratic attention modules like Mamba2 Dao and Gu (2024), Gated DeltaNet Yang et al. (2025c) and JetBlock Gu et al. (2025), built for server-side GPUs, reduce compute and memory costs but lack on-device runtime support (e.g., Executorch). Hybrid models Lenz et al. (2025); Glorioso et al. (2024); Ren et al. (2025); Pilault et al. (2023); De et al. (2024); Yang et al. (2025c) combines linear and quadratic attentions to improve recall and reasoning, but they typically rely on tedious manual design. Our work automates the selection of efficient attention patterns in hybrid models, utilizing runtime-supported mechanisms such as skip attention and sliding-window attention (SWA), enabling more scalable development and optimal design choices. Automatic search with sub-quadratic modules is performed in Amini et al. (2025); however, it does not adopt such hybrids, possibly due to the absence of highly optimized kernels they automatically lose by latency. A comparison of our paper with the most relevant related works is shown in Tab. 1.

Table 1 Comparison of methodologies.

Feature	MobileLLM-Flash	LFM2 Amini et al. (2025)	Jet-Nemotron Gu et al. (2025), Nemotron-Flash Fu et al. (2025)
Directly optimizes mobile CPU latency	✓	✓	✗
Unified architecture (shape + attention) search	✓	✗	✗
Efficient pruning-based search	✓	✗	✗
Compatible with any hardware / runtime	✓	✓	✗

Bayesian optimization (BO) is a sample-efficient framework for optimizing expensive, noisy black-box functions $f : \mathcal{S} \to \mathbb{R}$ by leveraging a surrogate model and an acquisition function to balance exploration and exploitation Frazier (2018). Multi-objective Bayesian Optimization (MOBO) extends BO to optimize multiple objectives $f_1, \ldots, f_M$, aiming to efficiently approximate the Pareto frontier with respect to a user-specified reference point r. MOBO leverages specialized acquisition functions, such as Noisy Expected Hypervolume Improvement (NEHVI), to guide the search towards solutions that maximize the expected improvement in hypervolume Daulton et al. (2021); Eriksson et al. (2021).

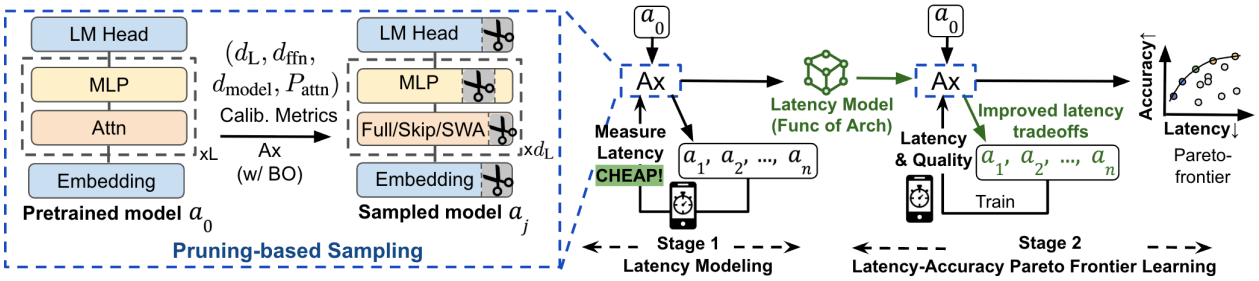

Figure 2 Overview of our two-stage OD-LLM design. We jointly search the architecture and attention pattern by pruning a pretrained model a_0 (Sec. 3.3) using Bayesian Optimization (BO) with Ax. In Stage 1, we sample pruned architectures and measure their latency on phone to cheaply learn a latency model. In Stage 2, leveraging the latency model, we efficiently search the space to generate the Accuracy-Latency Pareto-frontier (Sec. 3.4).

3 Latency-Aware OD-LLM Design

Our methodology addresses the challenge of designing OD-LLMs for resource-constrained devices. We target the most common hardware platform and *maximize model quality while minimizing TTFT*. Fig. 2 provides an overview.

3.1 Latency-Aware Optimization

We search candidate architectures $a \in \mathcal{A}$ for a configuration that maximizes model quality $Q(a)$ while meeting a constraint on our primary latency metric, TTFT, $T_{\text{prefill}}(a) < \tau_{\text{prefill}}$. We use loss as a proxy for $Q(a)$, based on the established findings that pretraining loss reliably predicts downstream quality Kaplan et al. (2020); Hoffmann et al. (2022). The threshold τ_{prefill} is product-dependent and may change as the use-case evolves; because it is not known *a priori*, we optimize both Q and T_{prefill} jointly. Because improvements in latency often reduce quality, and vice versa, we target the Pareto frontier where no solution can improve one objective without worsening another. This frontier represents the set of optimal tradeoffs between the objectives as shown in Fig. 3. An optimal point can be selected from this frontier given any τ_{prefill}.

3.2 Evaluating Model Configurations

To reduce the cost of the search relative to our target full training budget of 500B tokens, we train each candidate architecture (obtained by pruning the base model) for only 2.6B tokens. Empirically, we find that by $\sim$ 2.6B tokens the relative ordering of architectures is already predictive of their ordering after the full 500B-token run (Fig. 4). We refer to this early-but-predictive ordering as a *stable ranking*. Due to resource constraints, we do *not* sweep the optimizer hyperparameters.

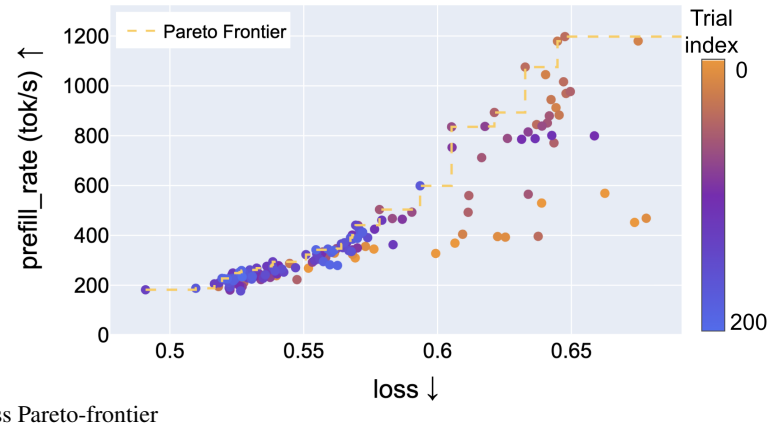

Figure 3 A latency-loss Pareto-frontier

We emphasize that our focus is the on-device latency, and it is motivated by practical considerations. While total parameter counts and total FLOPs per token are important metrics, they do not fully determine the on-device latency. To be more concrete, in Tab. 2 we present the Kendall tau correlation coefficient between these quantities *for our search space* across 100 architectures exported with Executorch on a Samsung Galaxy S25 for 2k sequence length. See Appendix A for visual representation of these correlations.

Table 2 Correlation coefficients between real measured prefill / decode speed vs. parameter count / FLOPs.

Kendall tau correlation (Kendall (1938))*	prefill	decode
#params vs. latency per tok	0.40	0.40
FLOPs vs. latency per tok	0.46	0.55

* The closer to 1, the higher correlation.

Key Insight-1 Model parameter count and FLOPs are suboptimal proxies for latency; hardware-in-the-loop optimization is necessary for accurate on-device latency improvements.

3.3 Hybrid Architecture Search Space

Structured Pruning-based Search: Instead of training candidates from scratch, we obtain candidates by pruning a larger pretrained model and inheriting its weights. Our approach is similar to weight-sharing approaches Fedorov et al. (2022); Cai et al. (2019); Kusupati et al. (2022) as the cost of training is amortized. However, we train candidates in isolation and are therefore not subject to the inductive bias of weight-sharing approaches, which assume that all candidates can be simultaneously trained in a nested fashion. We can think of a pruned pretrained model as a more data-efficient initialization for the architecture at hand. We observe empirically that the rank ordering of models trained from scratch aligns with that of pruned models (with the same architecture and inherited weights) under continued pretraining (CPT), with a Kendall tau correlation of 0.74 across 20 candidates. Despite this alignment, CPT achieves a significantly lower loss and approaches final convergence values much faster than random initialization. Furthermore, we need fewer tokens to get to a stable ranking of models; as shown in Fig. 4, the candidate ranking established at 10k steps matches the ranking at 120k steps.

Pruning can also be viewed as a natural method to efficiently explore the architecture search-space $\mathcal{S}$, consisting of the following parameters: number of transformer layers d_{L}, feed-forward network (FFN) hidden dimension d_{ffn}, residual stream dimension d_{model}, and efficient attention pattern $\mathbf{P}_{\text{attn}}$ (i.e., attention type for each transformer layer).

We employ activation energy-based metrics, measured on a small calibration dataset to decide what to prune (similar to Fedorov et al. (2024)). Specifically, the hidden size and MLP metrics quantify the average activation L2 norm magnitude (energy) across batch and sequence dimensions: $\text{FFNMetric} = \frac{1}{N}\sum_{i=1}^{N}\|\mathbf{x}_i\|$ and $\text{ModelDimMetric} = \frac{1}{N}\sum_{i=1}^{N}\|\text{LN}(\mathbf{x}_i)\|$, while the layer metric captures the functional transformation energy via cosine similarity between input and output activations (as in Gromov et al. (2024)): $\text{LayerMetric} = 1 - \frac{1}{N}\sum_{i=1}^{N}\frac{\mathbf{x}_i \cdot \mathbf{x}_{i+1}}{\|\mathbf{x}_i\|\|\mathbf{x}_{i+1}\|}$, where the $\mathbf{x}_i$ is the input vector at position i, N is the total number of positions (batch size $\times$ sequence length), and LN is LayerNorm Ba

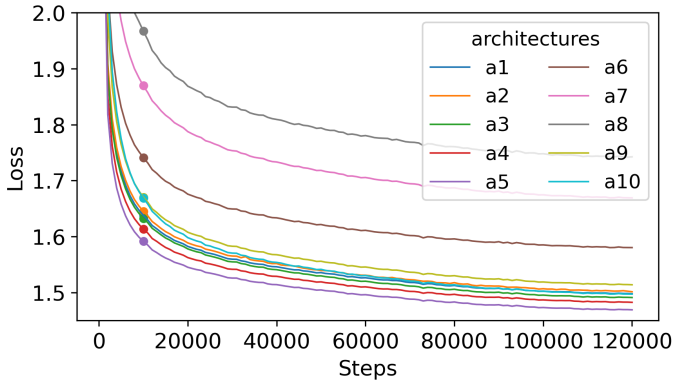

Figure 4 Pruned model loss evolution.

et al. (2016). Guided by the the FFNMetric and ModelDimMetric, we structurally prune the FFN hidden and residual stream dimensions for all decoder layers to the same target size d_{ffn} and d_{model} in block units (e.g., 128). The setup of block size makes the resulting architecture compatible with group-wise quantization in post-training. We rank all decoder layers by LayerMetric and remove those with the lowest contribution until d_{L} layers remain. After pruning, zeroed blocks and layers are removed, and the remaining architecture is concatenated into a compact and dense checkpoint.

Key Insight-2 Pruning offers a more data-efficient approach to architecture search than training candidates from scratch.

Efficient Attention Pattern: Although many efficient attention mechanisms exist (Sec. 2), most prior work targets server-side GPU optimization. Our focus is on real-world, on-device deployment, requiring each candidate to be compiled and benchmarked within production-grade deployment stacks such as Executorch. This ensures our evaluation reflects practical constraints and operational realities of mobile and edge devices. Consequently, we restrict our search to efficient attention operations that are natively supported in Executorch, specifically (1) skip attention, which allows certain layers to bypass attention computation; (2) global attention and (3) sliding window attention (SWA) Child et al. (2019). Both global attention and SWA use grouped query and QK-Norm.

Formally, the architecture space $\mathcal{A}$ consists of the model architecture and attention pattern $\mathbf{P}_{\text{attn}}$, where $\mathbf{P}_{\text{attn}} =< p_1, p_2, \ldots, p_{\text{L}} >$, $i \in \text{L}$, p_i specifies the attention type for the transformer layer i in a model with L layers. Using the importance metrics, we define a formal operator Prune$(\cdot) : \mathcal{S} \rightarrow \mathcal{A}$. This operator takes $s = (d_{\text{L}}, d_{\text{ffn}}, d_{\text{model}}, \mathbf{P}_{\text{attn}})$ and calibration metrics (FFNMetric, ModelDimMetric, LayerMetric) as input, and produces a unique architecture $a \in \mathcal{A}$ that optimizes the activation metrics. $\mathcal{A} = \{a \mid a = \text{Prune}(\text{Calib. Metrics}, s)\}$. Our search space (with $\sim 70B$ possible options) is shown in Tab. 3.

Table 3 Search space $\mathcal{S}$.

Parameters	Parameter Choices
d_{L}	$\{10, 11, 12, 13, 14, 15, 16\}$
d_{ffn}	$\{2048, 2304, 2560, \ldots 8192\}$
d_{model}	$\{1024, 1152, 1280 \ldots 2048\}$
p_i	{full_attn, SWA, skip_attn}

3.4 Optimization Strategy

We leverage Bayesian optimization (BO) with the Ax platform Olson et al. (2025) to optimize for the latency–quality Pareto frontier by searching over $\mathcal{S}$. We use a *two-stage BO* approach to leverage the fact that latency is nearly instantaneous while model quality requires substantial training resources. In the first stage, we use Sobol quasi-random sampling Sobol' (1967) to densely cover the latency landscape of $\mathcal{S}$, followed by BO. Since latency evaluation is significantly cheaper than quality evaluation, we collect $\sim$800 on-device latency measurements to build a high-quality Gaussian Process surrogate, achieving a cross-validation $R^2 = 0.97$. This allows the second stage to rely on predicted latency, focusing expensive quality evaluations on architectures in latency-favorable regions. The second stage optimizes

both objectives, accelerating the multi-objective search by focusing expensive model-quality evaluations on regions more likely to exhibit favorable latency tradeoffs. While BO methods such as HVKG Daulton et al. (2023) support objectives with different evaluation costs, our setting assumes latency is essentially free to evaluate. We leverage the NEHVI acquisition function and set the reference point r to a loss of 0.6 and a TTFT of 4 seconds, focusing on configurations that outperform a hand-tuned baseline model. The loss threshold of 0.6 is a heuristic calibrated by training a small number of candidates and validating against downstream tasks, and based on our own rank-ordering stability analysis (Sec 3.3).

3.5 OD-LLM Tuning Principles

By analyzing the optimal candidates along the Pareto-frontier, we distill the following efficiency principles for latency-aware OD-LLM design:

(1) **Model architecture**: We find that (at a fixed parameter count) deeper models generally have lower loss and higher latency, while shallow-and-wide models have higher loss and lower latency. Yet, at sufficiently low latency it is preferable to switch to shallow models. As shown in Fig. 5, on the Pareto-frontier curve, the 30-layer models (yellow dots) achieve the best model quality with the slowest prefill speed, while the shallower architectures (dark blue dots) offer a better balance on accuracy–latency trade-off. This aligns with observations in prior work Fu et al. (2025).

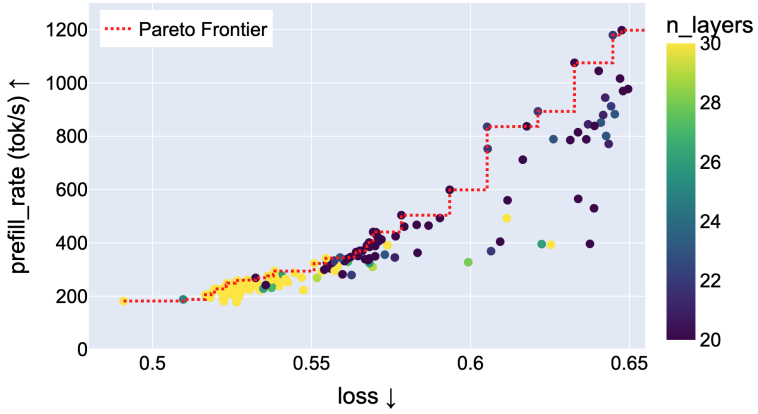

Figure 5 A Pareto curve with different model depths.

Efficiency Principle-1 For on-device deployment, shallow-and-wide models better balance accuracy and latency than deeper models.

(2) **Attention pattern**: We find, surprisingly, that our latency-guided search consistently favors skip attention over SWA, indicating that skipping a module provides a better latency-quality trade-off than SWA. We provide a high-level explanation for the inefficiency of SWA in our settings in Appendix B. Moreover, we observe that skipping too many attention layers consecutively (empirically, more than 3) degrades model quality under a fixed latency constraint and can even impair performance on harder generative tasks, as shown in Tab. 4. Consequently, we find the optimal attention pattern interleaves skip attention and global attention blocks. Therefore, we add a constraint in the final search: don't include 3 or more consecutive efficient attention types (either SWA or skip attention).

Table 4 Candidates with identical latency / identical architecture and skip attention counts can differ in performance due to the consecutive attention patterns.

	TQA	NQ	WinoG
With >3 consecutive skip attention	8.8%	2.5%	59.7%
Without >3 consecutive skip attention	33.2%	10.0%	61.2%

Efficiency Principle-2 Skip attention is more efficient than SWA; optimal patterns interleave skip attention and global attention to balance long-range modeling and latency.

4 MobileLLM-Flash: A New Family of Fast On-Device LLMs

4.1 Implementation

Our pruning-based architecture search is implemented as follows. **(1) Starting checkpoint**: As Sec. 3.5 highlights the benefits of shallow architectures, we select MobileLLM-Pro-Shallow-1.8B (Tab. 5) as the starting point – a shallow variant of MobileLLM-Pro Huber et al. (2025) trained with the same recipe and data. We use the same pretraining data, instruction fine-tuning (IFT) data and tokenizer (202k vocabulary size) as MobileLLM-Pro Huber et al. (2025) throughout our experiment. **(2) Calibration**: We calibrate the activation-based importance metrics (Sec. 3.3) on a 600M-token ($\sim$0.1% of the pretraining tokens) calibration set, following common practice for structured pruning Fedorov et al. (2024); Gromov et al. (2024). **(3) Sampling and pruning**: Ax samples 8 candidates per iteration, determined by available GPU resources for parallel evaluation within each BO round. We prune the initial model based on these configurations, applying efficient attention patterns and inheriting weights to create hybrid small, dense models. **(4) Candidate evaluation**: Candidate architectures are trained for 2.6B tokens to ensure stable rank-ordering of architectures (as discussed in Sec. 3.3) and the loss is used to measure model quality. We then export models with ExecuTorch (v1.1.0) and measure TTFT on a Samsung Galaxy S25 at a 2k sequence length. **(5) Candidate selection and final training**: We run 200 trials to maximize coverage of the latency–quality tradeoff space within our compute budget. After 200 trials, we select Pareto-optimal candidates that satisfy our specific latency constraints. Then we CPT the best candidates for 500B tokens using knowledge distillation with MobileLLM-Pro-Shallow-1.8B as teacher.

Our approach only requires lightweight CPT rather than training from scratch: we use $\sim$ 35% of the tokens used to pretrain MobileLLM-Pro. Finally, the models are IFTed for 800B tokens to prepare them for downstream tasks. During CPT, we set the sequence length to 2048, and the SWA window size to 256. During IFT, we set the sequence length to 8192, enabling the model to generalize to longer-sequence downstream tasks. Note that the total search cost is $200 \times 2.6\text{B} = 520\text{B}$ tokens across all candidates. Only 3 final Pareto-optimal candidates undergo full training: 500B tokens of CPT and 800B tokens of IFT each. This makes our method practical for OD-LLM development, compared to MobileLLM-Pro (1.6T) Huber et al. (2025) and LFM2 (10-12T) Amini et al. (2025).

Table 5 MobileLLM-Pro variants and MobileLLM-Flash model architectures.

Model	Layers	d_{model}	d_{FFN}	H/KV/H_{size}	#Attn blocks
MobileLLM-Pro-1B*	30	1280	6144	20/4/64	16
MobileLLM-Pro-Shallow-1.8B	16	2048	8192	32/8/64	16
MobileLLM-Flash-350M*	12	1024	4096	32/8/64	7†
			full_attn_idxs: [0,1,3,6,7,9,11]†		
MobileLLM-Flash-650M*	13	1280	6144	32/8/64	8†
			full_attn_idxs: [0,2,3,5,7,8,9,10]†		
MobileLLM-Flash-1.4B*	16	2048	8192	32/8/64	16

† The remaining layers skip attention.
* Shared weights between input embeddings and output projection.

4.2 Experimental Results

4.2.1 Model Quality

We evaluate our pre-trained models across reasoning, retrieval, and knowledge-intensive tasks: HellaSwag Zellers et al. (2019), BoolQ Clark et al. (2019), PIQA Bisk et al. (2019), SIQA Sap et al. (2019), *WinoG*rande Sakaguchi et al. (2021), *ARC E*asy Clark et al. (2018) in 0-shot settings, *T*rivia*QA* Joshi et al. (2017) and *N*at*Q* Kwiatkowski et al. (2019) with 5 shots and *ARC C*hallenge with 25 shots. Using lm-eval-harness Gao et al. (2023), we report exact-match rate for TriviaQA/NQ and character-level accuracy for other tasks. We compare model accuracy and efficiency with OD-LLM baselines LFM2-350M/700M/1.2B Amini et al. (2025), Nemotron-FLash-1B Fu et al. (2025), Qwen3 0.6B Yang et al. (2025a), Llama3.2-1B Grattafiori et al. (2024) and Gemma3-270M/1B Team et al. (2025), as shown in Tab. 6. MobileLLM-Flash achieves the highest average scores across all size regimes, establishing it as the most capable on-device model. For reference, Tab. 6 includes the unpruned MobileLLM-Pro-Shallow-1.8B base model. MobileLLM-Flash-1.4B trades a modest 1.1% average accuracy for up to $1.2\times$ / $1.3\times$ faster prefill/decode, illustrating the favorable latency-quality tradeoff identified by our search.

Table 6 Comparison of model quality and efficiency performance across various on-device scale models

Model (Parameter Count)	HellaSwag Zellers et al. (2019)	BoolQ Clark et al. (2019)	PIQA Bisk et al. (2019)	SocialIQA Sap et al. (2019)	TQA Joshi et al. (2017)	NQ Kwiatkowski et al. (2019)	ARC-c Clark et al. (2018)	ARC-e Clark et al. (2018)	WinoG Sakaguchi et al. (2021)	Avg ↑	Prefill TTFT(s) ↓			Decode rate (tok/s) ↑		
											1k	2k	4k	1k	2k	4k
Gemma3 270M Team et al. (2025)	41.38	58.17	68.34	39.71	15.4	4.04	29.0	57.32	53.59	40.77	0.99	3.13	5.64	123.71	82.85	52.40
LFM2 350M Amini et al. (2025)	49.00	**64.37**	69.48	35.01	14.97	4.96	**44.54**	**66.04**	55.96	44.92	**0.84**	**2.18**	**4.52**	147.54	96.90	63.60
MobileLLM-Flash 350M	**49.16**	62.39	**70.08**	**43.50**	**19.84**	**5.90**	38.89	63.26	**56.09**	**45.46**	0.91	2.78	5.33	**165.56**	**112.58**	**95.55**
Qwen3 0.6B Yang et al. (2025a)	53.80	69.39	69.86	43.14	2.41	4.32	38.57	58.08	58.88	44.27	4.63	11.59	25.40	44.56	28.67	18.48
LFM2 700M Amini et al. (2025)	45.01	**71.38**	71.11	37.41	22.50	6.90	**49.40**	**74.60**	58.40	48.52	2.05	6.01	**8.24**	86.30	53.57	44.36
MobileLLM-Flash 650M	**54.60**	64.77	**71.82**	**45.45**	**24.49**	**7.56**	42.58	66.55	**59.35**	**48.57**	**1.62**	**3.34**	8.48	**96.64**	**85.35**	**60.74**
Nemotron-Flash 1B Fu et al. (2025)	45.80	–	75.41	–	–	–	41.47	74.83	59.67	–	–	–	–	–	–	–
Gemma3 1B Team et al. (2025)	62.30	63.20	73.80	**48.90**	**39.80**	9.48	38.40	73.00	58.20	51.79	3.55	9.29	18.86	58.58	**43.11**	**36.29**
Llama3.2 1B Grattafiori et al. (2024)	65.69	62.51	75.14	45.60	23.81	5.48	38.28	63.47	61.09	49.01	4.51	15.09	26.95	39.01	18.13	14.34
LFM2 1.2B Amini et al. (2025)	45.26	66.09	74.27	37.72	33.50	8.60	**52.20**	**77.90**	58.80	50.48	3.46	**8.41**	16.88	**61.14**	42.15	29.14
MobileLLM-Flash 1.4B	**66.87**	**71.07**	**75.52**	47.34	36.06	**11.83**	50.56	72.26	**64.01**	**55.06**	**3.40**	9.08	**16.50**	60.52	42.65	34.01
Base model (before pruning)																
MobileLLM-Pro-Shallow-1.8B	67.74	72.63	76.82	47.34	39.00	12.41	51.33	74.12	64.48	56.20	3.82	9.20	19.93	46.52	35.88	25.69

Table 7 Comparison of downstream task results

	MMLU	MBPP	HumanEval	Open Rewrite	TLDR9+
Gemma3-1B	29.90	35.20	41.50	–	–
Llama3.2-1B	**49.30**	**39.60**	37.80	41.60	16.80
MobileLLM-Flash 650M	35.37	33.00	**45.12**	**46.84**	14.93
MobileLLM-Flash 1.4B	47.89	35.60	**46.34**	40.10	**16.89**

Table 8 MobileLLM-Flash maintains competitive or superior TTFT across devices.

TTFT 1k (s) ↓	S25	iPhone 17
LFM2 350M	**0.84**	**1.47**
MobileLLM-Flash 350M	0.91	1.55
LFM2 700M	2.05	3.40
MobileLLM-Flash 650M	**1.62**	**2.81**
LFM2 1.2B	3.46	5.51
MobileLLM-Flash 1.4B	**3.40**	**4.96**

We evaluate our IFTed models across knowledge (MMLU), coding (MBPP, HumanEval), rewriting (OpenRewrite), and summarization (TLDR9+) tasks (Tab. 7). All tasks are formatted as user-assistant conversations and evaluated on the final response. The results demonstrate that MobileLLM-Flash achieves superior or comparable performance, making it as the leading on-device instruction-tuned model for assistant applications.

4.2.2 Model Efficiency

For latency evaluation, we export models using ExecuTorch (v1.1.0) and optimize them for mobile CPUs via the XNNPACK backend. All models are quantized to 4-bit weights (group size 32) and 8-bit dynamic activations, with a quantized KV cache. Nemotron-Flash-1B is excluded because its custom JetBlock operators are not supported by ExecuTorch. We conduct latency benchmarking on the Samsung Galaxy S25, a flagship Android phone powered by the Snapdragon 8 Elite chipset with an octa-core CPU and 12 GB of memory. We evaluated models at context lengths of 1k, 2k, and 4k using 4 CPU threads. Reported metrics are averaged over three runs following a warmup. The results demonstrate that MobileLLM-Flash yields $\mathbf{1.8\times}$ / $\mathbf{1.6\times}$ faster prefill / decode speeds than LFM2, establishing MobileLLM-Flash as the fastest OD-LLM at this scale.

Our models use a 202k vocabulary size. While this increases the embedding parameters, it enhances information density per token. By representing common words more efficiently, the model requires fewer tokens to encode the same semantic content compared to models with smaller vocabulary sizes (e.g., 32k), leading to cheaper inference. Besides, this vocabulary is shared with Llama4-Scout Meta AI (2025), enabling effective knowledge distillation that benefits both our shallow and deep backbones Huber et al. (2025).

MobileLLM-Flash uses only standard operators natively supported by ExecuTorch and thus deploys on any compatible backend — including Apple devices (CoreML dispatches to CPU, GPU, Apple Neural Engine) — without specialized kernels. Retargeting the search to a new platform requires only substituting the benchmark device; the methodology itself is hardware-agnostic. We note that Pareto-optimal architectures for one accelerator class (e.g., CPU) are not necessarily optimal for another (e.g., Apple Neural Engine) due to differing execution characteristics. Nevertheless, we observe that latency rankings tend to transfer well across mobile CPUs: architectures searched on a Samsung Galaxy S25 preserve their relative advantage on an iPhone 17 (Tab. 8).

5 Conclusion

We introduce MobileLLM-Flash, a model family optimized for mobile latency through a novel 2-stage hardware-in-the-loop architecture search. By prioritizing shallow-and-wide structures and interleaved skip-attention patterns, we achieve state-of-the-art quality with significant speedups over strong baselines. Our methodology is compatible with Executorch and establishes a practical, scalable framework for delivering efficient, real-time AI on edge devices without specialized kernels.

Limitations

Due to the high computational cost of training candidate architectures, our Bayesian Optimization search focused exclusively on architectural parameters (depth, width, attention patterns). We did not perform a co-optimization of training hyperparameters (e.g., learning rate schedules, optimizer settings) via Ax. It is possible that specific architectural candidates could achieve higher quality with bespoke hyperparameter tuning, which was outside the scope of this study.

To ensure immediate industry-scale deployability and compatibility with standard runtimes like ExecuTorch, in this paper we did not explore novel sub-quadratic attention mechanisms (such as SSMs or linear attention variants mentioned in Sec. 2) that currently lack mature runtime support. Extending the search space to include these emerging architectures remains a direction for future work as their software support matures.

Ethical Considerations

Our work contributes to "Green AI" by focusing on efficiency. By optimizing for lower latency and smaller model sizes, MobileLLM-Flash reduces the computational energy required for inference. Furthermore, our pruning-based search method is data-efficient, requiring significantly fewer (only 35%) training tokens (and thus less GPU energy) to discover optimal architectures compared to training candidates from scratch.

References

Bilge Acun, Prasoon Sinha, Newsha Ardalani, Sangmin Bae, Alicia Golden, Chien-Yu Lin, Meghana Madhyastha, Fei Sun, Neeraja J. Yadwadkar, and Carole-Jean Wu. Composer: A search framework for hybrid neural architecture design, 2025. https://arxiv.org/abs/2510.00379.

Alexander Amini, Anna Banaszak, Harold Benoit, Arthur Böök, Tarek Dakhran, Song Duong, Alfred Eng, Fernando Fernandes, Marc Härkönen, Anne Harrington, Ramin Hasani, Saniya Karwa, Yuri Khrustalev, Maxime Labonne, Mathias Lechner, Valentine Lechner, Simon Lee, Zetian Li, Noel Loo, Jacob Marks, Edoardo Mosca, Samuel J. Paech, Paul Pak, Rom N. Parnichkun, Alex Quach, Ryan Rogers, Daniela Rus, Nayan Saxena, Bettina Schlager, Tim Seyde, Jimmy T. H. Smith, Aditya Tadimeti, and Neehal Tumma. Lfm2 technical report, 2025. https://arxiv.org/abs/2511.23404.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016. https://arxiv.org/abs/1607.06450.

Akhiad Bercovich, Tomer Ronen, Talor Abramovich, Nir Ailon, Nave Assaf, Mohammed Dabbah, Ido Galil, Amnon Geifman, Yonatan Geifman, Izhak Golan, Netanel Haber, Ehud Dov Karpas, Roi Koren, Itay Levy, Pavlo Molchanov, Shahar Mor, Zach Moshe, Najeeb Nabwani, Omri Puny, Ran Rubin, Itamar Schen, Ido Shahaf, Oren Tropp, Omer Ullman Argov, Ran Zilberstein, and Ran El-Yaniv. Puzzle: Distillation-based NAS for inference-optimized LLMs. In *Forty-second International Conference on Machine Learning*, 2025. https://openreview.net/forum?id=RY5MMBHRqo.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language, 2019. https://arxiv.org/abs/1911.11641.

Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2019.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers, 2019. https://arxiv.org/abs/1904.10509.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1300. https://aclanthology.org/N19-1300/.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. https://arxiv.org/abs/1803.05457.

Aditya Cowsik, Tianyu He, and Andrey Gromov. Towards distributed neural architectures. *arXiv preprint arXiv:2506.22389*, 2025.

Tri Dao and Albert Gu. Transformers are ssms: generalized models and efficient algorithms through structured state space duality. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Sam Daulton, Maximilian Balandat, and Eytan Bakshy. Hypervolume knowledge gradient: A lookahead approach for multi-objective Bayesian optimization with partial information. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 7167–7204. PMLR, 23–29 Jul 2023. https://proceedings.mlr.press/v202/daulton23a.html.

Samuel Daulton, Maximilian Balandat, and Eytan Bakshy. Parallel bayesian optimization of multiple noisy objectives with expected hypervolume improvement. *CoRR*, abs/2105.08195, 2021. https://arxiv.org/abs/2105.08195.

Soham De, Samuel L. Smith, Anushan Fernando, Aleksandar Botev, George Cristian-Muraru, Albert Gu, Ruba Haroun, Leonard Berrada, Yutian Chen, Srivatsan Srinivasan, Guillaume Desjardins, Arnaud Doucet, David Budden, Yee Whye Teh, Razvan Pascanu, Nando De Freitas, and Caglar Gulcehre. Griffin: Mixing gated linear recurrences with local attention for efficient language models, 2024. https://arxiv.org/abs/2402.19427.

David Eriksson, Pierce I-Jen Chuang, Samuel Daulton, Peng Xia, Akshat Shrivastava, Arun Babu, Shicong Zhao, Ahmed A Aly, Ganesh Venkatesh, and Maximilian Balandat. Latency-aware neural architecture search with multi-objective bayesian optimization. In *8th ICML Workshop on Automated Machine Learning (AutoML)*, 2021. https://openreview.net/forum?id=0ciyfd4SvbI.

Igor Fedorov, Ramon Matas, Hokchhay Tann, Chuteng Zhou, Matthew Mattina, and Paul Whatmough. Udc: Unified dnas for compressible tinyml models for neural processing units. *Advances in Neural Information Processing Systems*, 35:18456–18471, 2022.

Igor Fedorov, Kate Plawiak, Lemeng Wu, Tarek Elgamal, Naveen Suda, Eric Smith, Hongyuan Zhan, Jianfeng Chi, Yuriy Hulovatyy, Kimish Patel, Zechun Liu, Changsheng Zhao, Yangyang Shi, Tijmen Blankevoort, Mahesh Pasupuleti, Bilge Soran,

Zacharie Delpierre Coudert, Rachad Alao, Raghuraman Krishnamoorthi, and Vikas Chandra. Llama guard 3-1b-int4: Compact and efficient safeguard for human-ai conversations, 2024. https://arxiv.org/abs/2411.17713.

Peter I Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

Yonggan Fu, Xin Dong, Shizhe Diao, Matthijs Van keirsbilck, Hanrong Ye, Wonmin Byeon, Yashaswi Karnati, Lucas Liebenwein, Maksim Khadkevich, Alexander Keller, Jan Kautz, Yingyan Celine Lin, and Pavlo Molchanov. Nemotron-flash: Towards latency-optimal hybrid small language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. https://openreview.net/forum?id=KTDAbnFsQj.

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. A framework for few-shot language model evaluation, 12 2023. https://zenodo.org/records/10256836.

Paolo Glorioso, Quentin Anthony, Yury Tokpanov, James Whittington, Jonathan Pilault, Adam Ibrahim, and Beren Millidge. Zamba: A compact 7b ssm hybrid model, 2024. https://arxiv.org/abs/2405.16712.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim

Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024. https://arxiv.org/abs/2407.21783.

Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A Roberts. The unreasonable ineffectiveness of the deeper layers. *arXiv preprint arXiv:2403.17887*, 2024.

Yuxian Gu, Qinghao Hu, Haocheng Xi, Junyu Chen, Shang Yang, Song Han, and Han Cai. Jet-nemotron: Efficient language model with post neural architecture search. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. https://openreview.net/forum?id=WZQXaTNYEB.

Alexander Heinecke, Evangelos Georganas, Kunal Banerjee, Dhiraj Kalamkar, Narayanan Sundaram, Anand Venkat, Greg Henry, and Hans Pabst. Understanding the performance of small convolution operations for cnn on intel architecture. In *Poster in the International Conference for High Performance Computing, Networking, Storage, and Analysis*, 2017.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack W. Rae, and Laurent Sifre. Training compute-optimal large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Patrick Huber, Ernie Chang, Wei Wen, Igor Fedorov, Tarek Elgamal, Hanxian Huang, Naveen Suda, Chinnadhurai Sankar, Vish Vogeti, Yanghan Wang, Alex Gladkov, Kai Sheng Tai, Abdelrahman Elogeel, Tarek Hefny, Vikas Chandra, Ahmed Aly, Anuj Kumar, Raghuraman Krishnamoorthi, and Adithya Sagar. Mobilellm-pro technical report, 2025. https://arxiv.org/abs/2511.06719.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In Regina Barzilay and Min-Yen Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1147. https://aclanthology.org/P17-1147/.

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

M. G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938. ISSN 00063444. http://www.jstor.org/stable/2332226.

Kaeun Kim, Ghazal Shams, and Kawon Kim. From seconds to sentiments: differential effects of chatbot response latency on customer evaluations. *International Journal of Human–Computer Interaction*, 42(1):597–612, 2026.

Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder,

Kaifeng Chen, Sham Kakade, Prateek Jain, et al. Matryoshka representation learning. *Advances in Neural Information Processing Systems*, 35:30233–30249, 2022.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. https://aclanthology.org/Q19-1026/.

Barak Lenz, Opher Lieber, Alan Arazi, Amir Bergman, Avshalom Manevich, Barak Peleg, Ben Aviram, Chen Almagor, Clara Fridman, Dan Padnos, Daniel Gissin, Daniel Jannai, Dor Muhlgay, Dor Zimberg, Edden M. Gerber, Elad Dolev, Eran Krakovsky, Erez Safahi, Erez Schwartz, Gal Cohen, Gal Shachaf, Haim Rozenblum, Hofit Bata, Ido Blass, Inbal Magar, Itay Dalmedigos, Jhonathan Osin, Julie Fadlon, Maria Rozman, Matan Danos, Michael Gokhman, Mor Zusman, Naama Gidron, Nir Ratner, Noam Gat, Noam Rozen, Oded Fried, Ohad Leshno, Omer Antverg, Omri Abend, Or Dagan, Orit Cohavi, Raz Alon, Ro'i Belson, Roi Cohen, Rom Gilad, Roman Glozman, Shahar Lev, Shai Shalev-Shwartz, Shaked Haim Meirom, Tal Delbari, Tal Ness, Tomer Asida, Tom Ben Gal, Tom Braude, Uriya Pumerantz, Josh Cohen, Yonatan Belinkov, Yuval Globerson, Yuval Peleg Levy, and Yoav Shoham. Jamba: Hybrid transformer-mamba language models. In *The Thirteenth International Conference on Learning Representations*, 2025. https://openreview.net/forum?id=JFPaD7lpBD.

Zechun Liu, Changsheng Zhao, Forrest Iandola, Chen Lai, Yuandong Tian, Igor Fedorov, Yunyang Xiong, Ernie Chang, Yangyang Shi, Raghuraman Krishnamoorthi, et al. Mobilellm: Optimizing sub-billion parameter language models for on-device use cases. *arXiv preprint arXiv:2402.14905*, 2024.

Meta AI. Llama 4 maverick and scout. https://huggingface.co/meta-llama/Llama-4-Scout-17B-16E, 2025.

Jakob Nielsen. *Usability engineering*. Morgan Kaufmann, 1994.

Miles Olson, Elizabeth Santorella, Louis C. Tiao, Sait Cakmak, David Eriksson, Mia Garrard, Sam Daulton, Maximilian Balandat, Eytan Bakshy, Elena Kashtelyan, Zhiyuan Jerry Lin, Sebastian Ament, Bernard Beckerman, Eric Onofrey, Paschal Igusti, Cristian Lara, Benjamin Letham, Cesar Cardoso, Shiyun Sunny Shen, Andy Chenyuan Lin, and Matthew Grange. Ax: A platform for adaptive experimentation. In *AutoML 2025 ABCD Track*, 2025. https://openreview.net/forum?id=U1f6wHtG1g.

Jonathan Pilault, Mahan Fathi, Orhan Firat, Christopher Pal, Pierre-Luc Bacon, and Ross Goroshin. Block-state transformers. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. https://openreview.net/forum?id=XRTxIBs2eu.

Liliang Ren, Yang Liu, Yadong Lu, yelong shen, Chen Liang, and Weizhu Chen. Samba: Simple hybrid state space models for efficient unlimited context language modeling. In *The Thirteenth International Conference on Learning Representations*, 2025. https://openreview.net/forum?id=bIlnpVM4bc.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: an adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106, August 2021. ISSN 0001-0782. doi: 10.1145/3474381. https://doi.org/10.1145/3474381.

Subhajit Sanyal, Srinivas Soumitri Miriyala, Akshay Janardan Bankar, Manjunath Arveti, Sowmya Vajrala, Shreyas Pandith, Sravanth Kodavanti, Abhishek Ameta, Harshit, and Amit Satish Unde. Nanosd: Edge efficient foundation model for real time image restoration, 2026. https://arxiv.org/abs/2601.09823.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social IQa: Commonsense reasoning about social interactions. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4463–4473, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1454. https://aclanthology.org/D19-1454/.

I.M Sobol'. On the distribution of points in a cube and the approximate evaluation of integrals. *USSR Computational Mathematics and Mathematical Physics*, 7(4):86–112, 1967. ISSN 0041-5553. doi: https://doi.org/10.1016/0041-5553(67)90144-9. https://www.sciencedirect.com/science/article/pii/0041555367901449.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gaël Liu, Francesco Visin, Kathleen Kenealy, Lucas Beyer, Xiaohai Zhai, Anton Tsitsulin, Robert Busa-Fekete, Alex Feng, Noveen Sachdeva, Benjamin Coleman, Yi Gao, Basil Mustafa, Iain Barr, Emilio Parisotto, David Tian, Matan Eyal, Colin Cherry, Jan-Thorsten Peter, Danila Sinopalnikov, Surya Bhupatiraju, Rishabh Agarwal, Mehran Kazemi, Dan Malkin, Ravin Kumar, David Vilar, Idan Brusilovsky, Jiaming Luo, Andreas Steiner, Abe Friesen, Abhanshu Sharma, Abheesht Sharma, Adi Mayrav Gilady, Adrian Goedeckemeyer, Alaa Saade, Alex Feng, Alexander Kolesnikov, Alexei Bendebury, Alvin Abdagic, Amit Vadi, András György, André Susano Pinto, Anil Das, Ankur Bapna, Antoine Miech,

Antoine Yang, Antonia Paterson, Ashish Shenoy, Ayan Chakrabarti, Bilal Piot, Bo Wu, Bobak Shahriari, Bryce Petrini, Charlie Chen, Charline Le Lan, Christopher A. Choquette-Choo, CJ Carey, Cormac Brick, Daniel Deutsch, Danielle Eisenbud, Dee Cattle, Derek Cheng, Dimitris Paparas, Divyashree Shivakumar Sreepathihalli, Doug Reid, Dustin Tran, Dustin Zelle, Eric Noland, Erwin Huizenga, Eugene Kharitonov, Frederick Liu, Gagik Amirkhanyan, Glenn Cameron, Hadi Hashemi, Hanna Klimczak-Plucińska, Harman Singh, Harsh Mehta, Harshal Tushar Lehri, Hussein Hazimeh, Ian Ballantyne, Idan Szpektor, Ivan Nardini, Jean Pouget-Abadie, Jetha Chan, Joe Stanton, John Wieting, Jonathan Lai, Jordi Orbay, Joseph Fernandez, Josh Newlan, Ju yeong Ji, Jyotinder Singh, Kat Black, Kathy Yu, Kevin Hui, Kiran Vodrahalli, Klaus Greff, Linhai Qiu, Marcella Valentine, Marina Coelho, Marvin Ritter, Matt Hoffman, Matthew Watson, Mayank Chaturvedi, Michael Moynihan, Min Ma, Nabila Babar, Natasha Noy, Nathan Byrd, Nick Roy, Nikola Momchev, Nilay Chauhan, Noveen Sachdeva, Oskar Bunyan, Pankil Botarda, Paul Caron, Paul Kishan Rubenstein, Phil Culliton, Philipp Schmid, Pier Giuseppe Sessa, Pingmei Xu, Piotr Stanczyk, Pouya Tafti, Rakesh Shivanna, Renjie Wu, Renke Pan, Reza Rokni, Rob Willoughby, Rohith Vallu, Ryan Mullins, Sammy Jerome, Sara Smoot, Sertan Girgin, Shariq Iqbal, Shashir Reddy, Shruti Sheth, Siim Põder, Sijal Bhatnagar, Sindhu Raghuram Panyam, Sivan Eiger, Susan Zhang, Tianqi Liu, Trevor Yacovone, Tyler Liechty, Uday Kalra, Utku Evci, Vedant Misra, Vincent Roseberry, Vlad Feinberg, Vlad Kolesnikov, Woohyun Han, Woosuk Kwon, Xi Chen, Yinlam Chow, Yuvein Zhu, Zichuan Wei, Zoltan Egyed, Victor Cotruta, Minh Giang, Phoebe Kirk, Anand Rao, Kat Black, Nabila Babar, Jessica Lo, Erica Moreira, Luiz Gustavo Martins, Omar Sanseviero, Lucas Gonzalez, Zach Gleicher, Tris Warkentin, Vahab Mirrokni, Evan Senter, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, Yossi Matias, D. Sculley, Slav Petrov, Noah Fiedel, Noam Shazeer, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Jean-Baptiste Alayrac, Rohan Anil, Dmitry, Lepikhin, Sebastian Borgeaud, Olivier Bachem, Armand Joulin, Alek Andreev, Cassidy Hardin, Robert Dadashi, and Léonard Hussenot. Gemma 3 technical report, 2025. https://arxiv.org/abs/2503.19786.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025a. https://arxiv.org/abs/2505.09388.

Mingyu Yang, Mehdi Rezagholizadeh, Guihong Li, Vikram Appia, and Emad Barsoum. Zebra-llama: Towards extremely efficient hybrid models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b. https://openreview.net/forum?id=l42UGsdrNn.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In *The Thirteenth International Conference on Learning Representations*, 2025c. https://openreview.net/forum?id=r8H7xhYPwz.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In Anna Korhonen, David Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1472. https://aclanthology.org/P19-1472/.

Appendix

A Efficiency Proxies

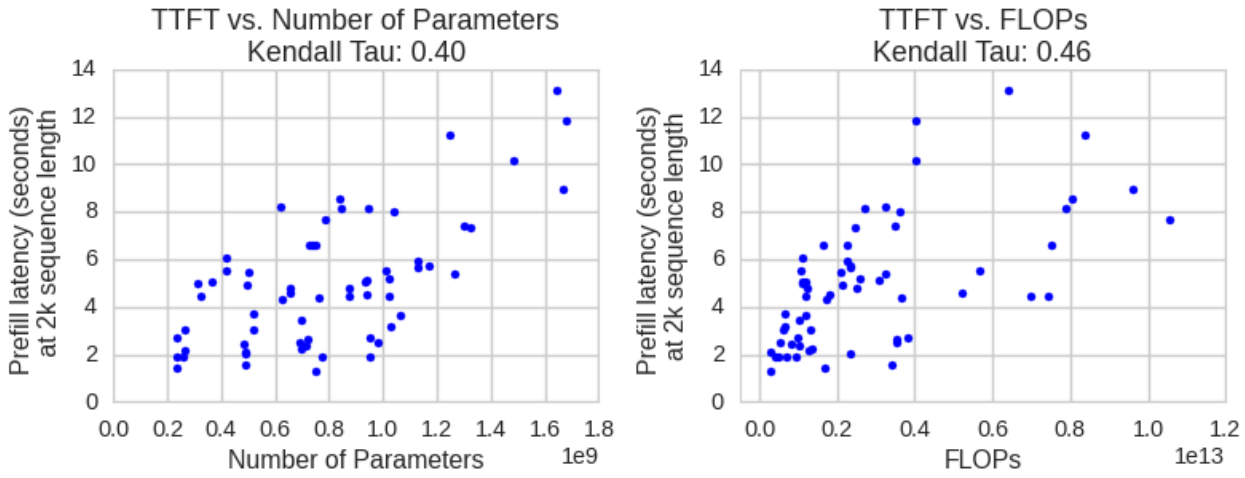

Figure 6 Kendall Tau correlation between TTFT at 2k sequence length vs. model parameter count and FLOPs.

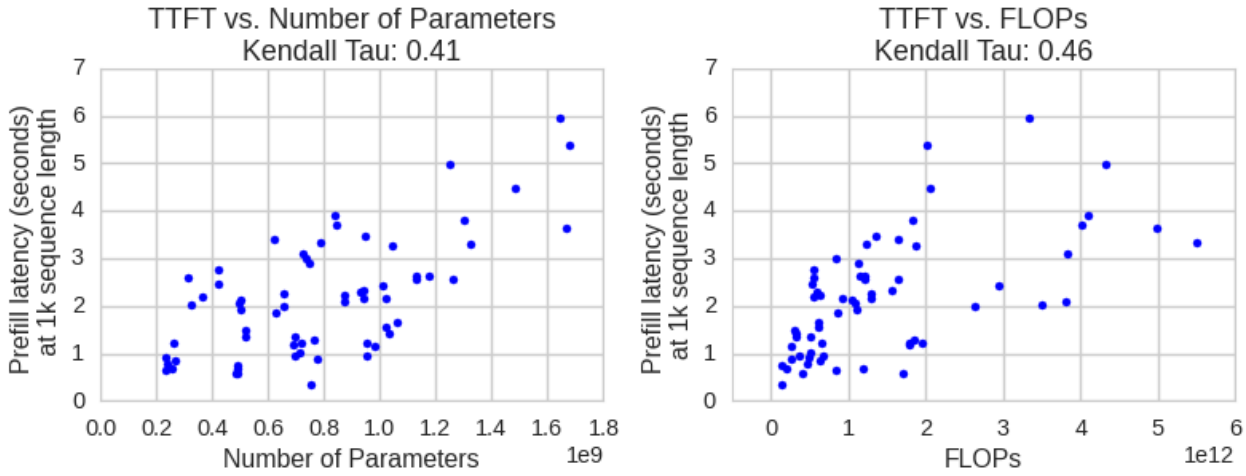

Figure 7 Kendall Tau correlation between TTFT at 1k sequence length vs. model parameter count and FLOPs.

We present visualization of correlation coefficients between real measured prefill / decode speed vs. parameter count / FLOPs in Fig. 6, Fig. 7,Fig. 8, and Fig. 9 for various context lengths.

B Explanation for the inefficiency of SWA

A high level explanation for the inefficiency of SWA in our settings is as follows. We find that a prefill chunk size of 1024 produces the lowest time-to-first-token (TTFT), making the best use of parallelism across tokens in the context window. At the same time, Executorch constrains the SWA to be greater than or equal to the prefill chunk size, such that the sliding window is lower-bounded by 1024. Therefore, when prefilling 2k tokens, only the second chunk benefits from the SWA. At the same time, the ring-buffer implementation of SWA in Executorch requires computation of the entire attention matrix, as opposed to just the lower triangular portion as done during standard attention computation. As a result, in the 2k prefill, 1024 chunk size setting, we find that the benefits of SWA are outweighed by the drawback of the slower attention calculation and the model overall achieves worse TTFT with SWA.

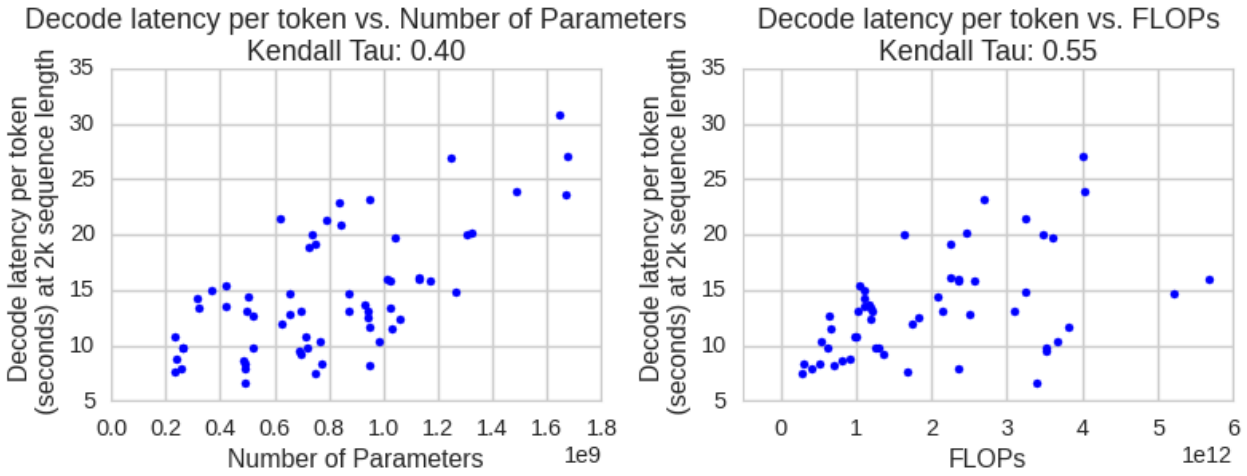

Figure 8 Kendall Tau correlation between decode latency at 2k sequence length vs. model parameter count and FLOPs.

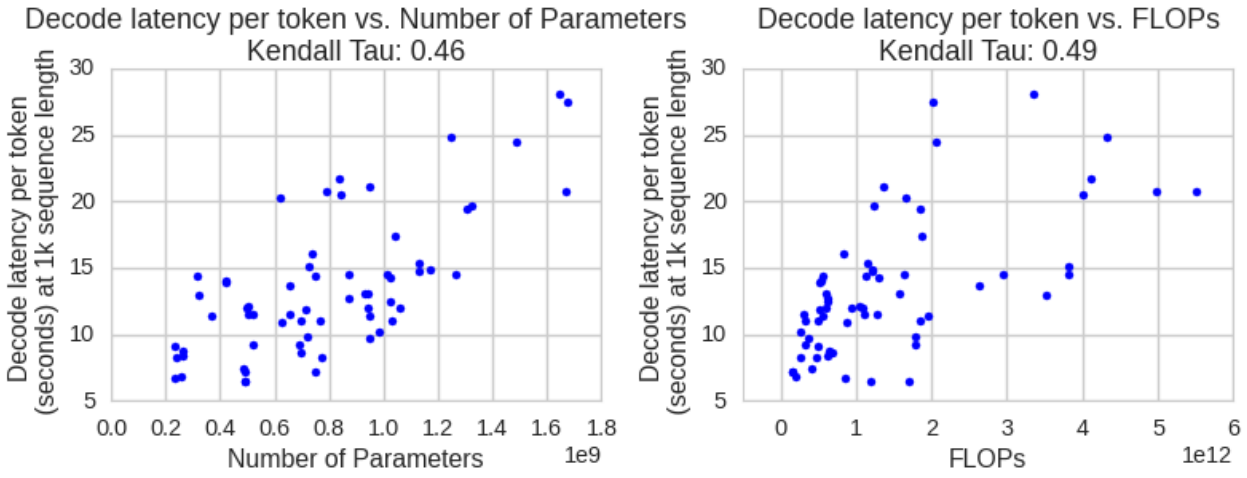

Figure 9 Kendall Tau correlation between decode latency at 1k sequence length vs. model parameter count and FLOPs.