

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

Anonymous ACL submission

## Abstract

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  $1.8\times$  and  $1.6\times$  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.

## 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

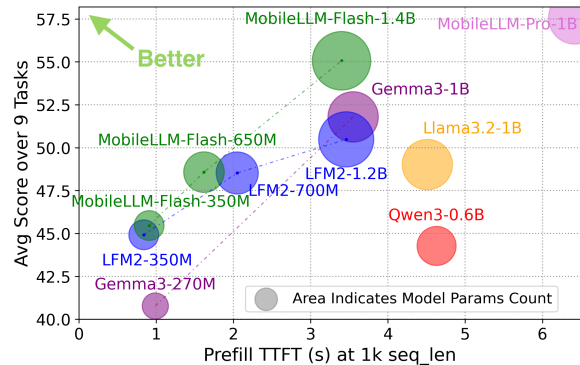


Figure 1: Comparison between MobileLLM-Flash and state-of-the-art OD-LLMs. MobileLLM-Flash achieves up to  $1.8\times$  /  $1.6\times$  faster prefill / decode on mobile CPUs with superior accuracy than LFM2 (Amini et al., 2025). Evaluation / benchmarking details are in Sec. 4.2.

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) 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 prefill latency, 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 sweet spot where OD-LLMs can both perform useful tasks and achieve  $TTFT \leq 4s$  (Tab. 6).

Existing efficient LLM designs often prioritize proxy metrics, such as parameter count or floating-point operations per second (FLOPs), or focus on server-side optimizations (Fu et al., 2025; Gu et al., 2025; Yang et al., 2025a). However, these meth-

ods 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, designs relying on specialized kernels for efficient attention (Gu et al., 2025; Yang et al., 2025b; Dao and Gu, 2024) face significant barriers to large-scale adoption due to limited portability.

**Our main contributions are:** **1)** We present a hardware-in-the-loop, pruning-based architecture search that directly optimizes mobile prefill latency. This yields deployment-ready models, compatible with ExecuTorch. **2)** We introduce the MobileLLM-Flash family (350M, 650M, 1.4B) (Fig. 1), featuring a compact hybrid backbone with skip attention and grouped query attention. This yields significant speedups in prefill and decode on mobile CPUs and runs out of the box without specialized kernels. **3)** We present an analysis of the latency-accuracy Pareto frontier, deriving actionable design principles to guide future OD-LLM development.

## 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., 2025a; Cowsik et al., 2025) use architecture search 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).

**Efficient attention alternatives and hybrid models.** Sub-quadratic attention modules like Mamba2 (Dao and Gu, 2024), Gated DeltaNet (Yang et al., 2025b) 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., 2025b) 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 pat-

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	✓	✓	✗

terns 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.

**Bayesian optimization (BO)** is a sample-efficient framework for optimizing expensive, noisy black-box functions  $f : \mathcal{S} \rightarrow \mathbb{R}$  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  $f_1, \dots, f_M$ , 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).

## 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 prefill latency*. 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, prefill latency,  $T_{\text{prefill}}(a) < \tau_{\text{prefill}}$ . The threshold  $\tau_{\text{prefill}}$  is product-dependent and may change as the use-case evolves; because it is not known *a priori*, we optimize both  $Q$  and  $T_{\text{prefill}}$  jointly. Because improvements in latency often re-

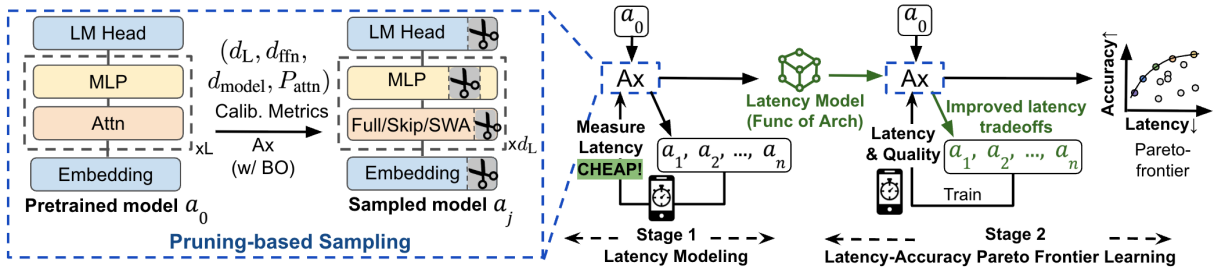


Figure 2: Overview of our two-stage OD-LLM design. We jointly search the architecture and attention pattern by pruning a pretrained model  $a_0$  (Sec. 3.3) using Bayesian Optimization (BO) with Ax. In Stage 1, we sample pruned architectures and measure their latency on phone to cheaply learn a latency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

duce 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  $\tau_{\text{prefill}}$ .

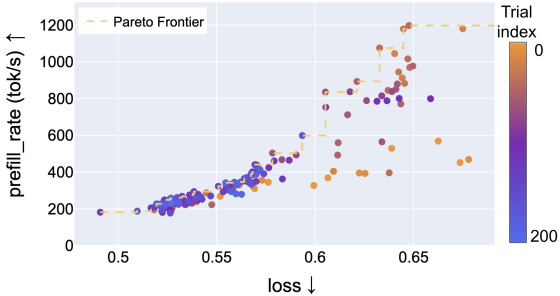


Figure 3: A latency-accuracy Pareto-frontier

### 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  $\sim 2.6$ B 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.

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.

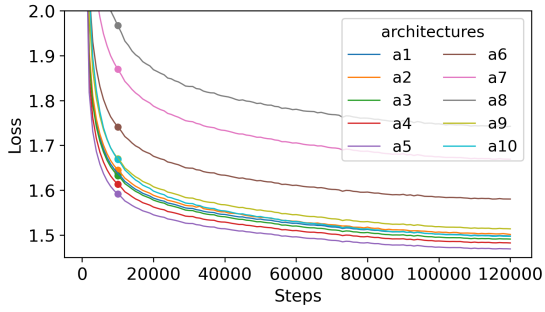


Figure 4: Pruned model loss evolution.

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  $d_{\text{ffn}}$ , residual stream dimension  $d_{\text{model}}$ , 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 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  $d_{\text{ffn}}$  and  $d_{\text{model}}$  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  $\mathbf{P}_{\text{attn}} = \langle p_1, p_2, \dots, p_L \rangle$ ,  $i \in L$ ,  $p_i$  specifies the attention type for the transformer layer  $i$  in a model with  $L$  layers. Using the importance metrics, we define a formal operator  $\text{Prune}(\cdot) : \mathcal{S} \rightarrow \mathcal{A}$ . This operator takes  $s = (d_L, d_{\text{ffn}}, d_{\text{model}}, \mathbf{P}_{\text{attn}})$  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}
$d_{\text{ffn}}$	{2048, 2304, 2560, ... 8192}
$d_{\text{model}}$	{1024, 1152, 1280 ... 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 densely sample the latency landscape of  $\mathcal{S}$  to build a high-quality Gaussian Process surrogate model. 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 prefill latency of 5 seconds, focusing on configurations that outperform a hand-tuned baseline model.

### 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).

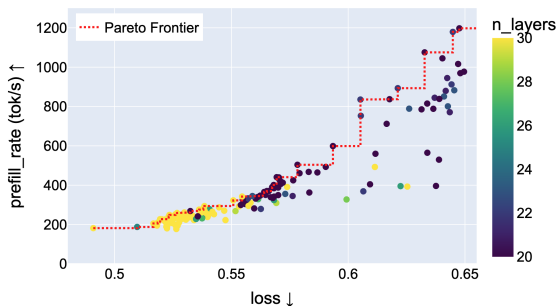


Figure 5: A Pareto curve with different model depths.

**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

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

Model	Layers	$d_{\text{model}}$	$d_{\text{FFN}}$	H/KV/H <sub>size</sub>	#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 <sup>†</sup>
MobileLLM-Flash-650M*	13	1280	6144	32/8/64	8 <sup>†</sup>
MobileLLM-Flash-1.4B*	16	2048	8192	32/8/64	16

<sup>†</sup> The remaining layers skip attention.

\* Shared weights between input embeddings and output projection.

Our pruning-based architecture search is implemented as follows. (1) **Starting checkpoint:** We select a pretrained model to initialize our search. Sec. 3.5 highlights the benefits of shallow architectures, so we pretrained a shallow version of MobileLLM-Pro 1B (Huber et al., 2025) (Tab. 5) to serve as the starting point for the search. We use the same pretraining data, instruction fine-tuning (IFT) data and tokenizer (202k vocabulary size) as MobileLLM-Pro (Huber et al., 2025). (2) **Calibration:** We calibrate the activation-based importance metrics (Sec. 3.3) on a 600M-token calibration set. (3) **Sampling and pruning:** Ax samples 8 candidates per iteration. 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 and the loss is used to measure model quality. We then export models with ExecuTorch (v1.1.0) and measure prefill latency on a Samsung Galaxy S25 at a 2k sequence length. (5) **Candidate selection and final training:** 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

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

Model (Parameter Count)	HellaSwag	BoolQ	PIQA	SocialQA	TQA	NQ	ARC-c	ARC-e	WinoG	Avg ↑	Prefill TTFT(s) ↓			Decode rate (tok/s) ↑		
											1k	2k	4k	1k	2k	4k
Gemma3 270M	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	49.00	<b>64.37</b>	69.48	35.01	14.97	4.96	<b>44.54</b>	<b>66.04</b>	55.96	44.92	<b>0.84</b>	<b>2.18</b>	<b>4.52</b>	147.54	96.90	63.60
MobileLLM-Flash 350M	<b>49.16</b>	62.39	<b>70.08</b>	<b>43.50</b>	<b>19.84</b>	<b>5.90</b>	38.89	63.26	<b>56.09</b>	<b>45.46</b>	0.91	2.78	5.33	<b>165.56</b>	<b>112.58</b>	<b>95.55</b>
Qwen3 0.6B	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	45.01	<b>71.38</b>	71.11	37.41	22.50	6.90	<b>49.40</b>	<b>74.60</b>	58.40	48.52	2.05	6.01	<b>8.24</b>	86.30	53.57	44.36
MobileLLM-Flash 650M	<b>54.60</b>	64.77	<b>71.82</b>	<b>45.45</b>	<b>24.49</b>	<b>7.56</b>	42.58	66.55	<b>59.35</b>	<b>48.57</b>	<b>1.62</b>	<b>3.34</b>	8.48	<b>96.64</b>	<b>85.35</b>	<b>60.74</b>
Nemotron-Flash 1B	45.80	–	75.41	–	–	–	41.47	74.83	59.67	–	–	–	–	–	–	–
Gemma3 1B	62.30	63.20	73.80	<b>48.90</b>	<b>39.80</b>	9.48	38.40	73.00	58.20	51.79	3.55	9.29	18.86	58.58	<b>43.11</b>	<b>36.29</b>
Llama3.2 1B	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	45.26	66.09	74.27	37.72	33.50	8.60	<b>52.20</b>	<b>77.90</b>	58.80	50.48	3.46	<b>8.41</b>	16.88	<b>61.14</b>	42.15	29.14
MobileLLM-Flash 1.4B	<b>66.87</b>	<b>71.07</b>	<b>75.52</b>	47.34	36.06	<b>11.83</b>	50.56	72.26	<b>64.01</b>	<b>55.06</b>	<b>3.40</b>	9.08	<b>16.50</b>	60.52	42.65	34.01

Table 7: Comparison of downstream task results

	MMLU	MBPP	HumanEval	Open Rewrite	TLDR9+
Gemma3-1B	29.90	35.20	41.50	–	–
Llama3.2-1B	<b>49.30</b>	<b>39.60</b>	37.80	41.60	16.80
MobileLLM-Flash 650M	36.00	33.00	<b>45.12</b>	<b>48.72</b>	15.18
MobileLLM-Flash 1.4B	47.89	35.60	<b>45.12</b>	40.10	<b>16.89</b>

distillation with MobileLLM-Pro-Shallow-1.8B as teacher. Our approach only requires lightweight CPT rather than training from scratch: we use 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.

## 4.2 Experimental Results

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), WinoGrande (Sakaguchi et al., 2021), ARC Easy (Clark et al., 2018) in 0-shot settings, TriviaQA (Joshi et al., 2017) and NatQ (Kwiatkowski et al., 2019) with 5 shots and ARC Challenge with 25 shots. Using Im-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), Qwen 0.6B, 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 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  $1.8 \times / 1.6 \times$  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.

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.

## 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.

## 437 Limitations

438 Due to the high computational cost of training candi-  
439 date architectures, our Bayesian Optimization  
440 search focused exclusively on architectural param-  
441 eters (depth, width, attention patterns). We did  
442 not perform a co-optimization of training hyperpa-  
443 rameters (e.g., learning rate schedules, optimizer  
444 settings) via Ax. It is possible that specific architec-  
445 tural candidates could achieve higher quality with  
446 bespoke hyperparameter tuning, which was outside  
447 the scope of this study.

448 To ensure immediate industry-scale deployabil-  
449 ity and compatibility with standard runtimes like  
450 ExecuTorch, in this paper we did not explore novel  
451 sub-quadratic attention mechanisms (such as SSMS  
452 or linear attention variants mentioned in Sec. 2) that  
453 currently lack mature runtime support. Extending  
454 the search space to include these emerging archi-  
455 tectures remains a direction for future work as their  
456 software support matures.

## 457 Ethical Considerations

458 Our work contributes to "Green AI" by focusing  
459 on efficiency. By optimizing for lower latency and  
460 smaller model sizes, MobileLLM-Flash reduces  
461 the computational energy required for inference.  
462 Furthermore, our pruning-based search method is  
463 data-efficient, requiring significantly fewer (only  
464 35%) training tokens (and thus less GPU energy) to  
465 discover optimal architectures compared to training  
466 candidates from scratch.

## 467 References

468 Bilge Acun, Prasoon Sinha, Newsha Ardalani, Sang-  
469 min Bae, Alicia Golden, Chien-Yu Lin, Meghana  
470 Madhyastha, Fei Sun, Neeraja J. Yadwadkar, and  
471 Carole-Jean Wu. 2025. [Composer: A search frame-  
472 work for hybrid neural architecture design](#). *Preprint*,  
473 arXiv:2510.00379.

474 Alexander Amini, Anna Banaszak, Harold Benoit,  
475 Arthur Böök, Tarek Dakhran, Song Duong, Al-  
476 fred Eng, Fernando Fernandes, Marc Härkönen,  
477 Anne Harrington, Ramin Hasani, Saniya Karwa,  
478 Yuri Khrustalev, Maxime Labonne, Mathias Lech-  
479 ner, Valentine Lechner, Simon Lee, Zetian Li, Noel  
480 Loo, and 14 others. 2025. [Lfm2 technical report](#).  
481 *Preprint*, arXiv:2511.23404.

482 Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E.  
483 Hinton. 2016. [Layer normalization](#). *Preprint*,  
484 arXiv:1607.06450.

485 Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng  
486 Gao, and Yejin Choi. 2019. [Piq: Reasoning about](#)

[physical commonsense in natural language](#). *Preprint*,  
arXiv:1911.11641.

Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang,  
and Song Han. 2019. [Once-for-all: Train one net-  
work and specialize it for efficient deployment](#). *arXiv  
preprint arXiv:1908.09791*.

Rewon Child, Scott Gray, Alec Radford, and Ilya  
Sutskever. 2019. [Generating long sequences with  
sparse transformers](#). *Preprint*, arXiv:1904.10509.

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 Chap-  
ter of the Association for Computational Linguistics:  
Human Language Technologies, Volume 1 (Long and  
Short Papers)*, pages 2924–2936, Minneapolis, Min-  
nesota. Association for Computational Linguistics.

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](#).  
*Preprint*, arXiv:1803.05457.

Aditya Cowsik, Tianyu He, and Andrey Gromov. 2025.  
[Towards distributed neural architectures](#). *arXiv  
preprint arXiv:2506.22389*.

Tri Dao and Albert Gu. 2024. [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.

Sam Daulton, Maximilian Balandat, and Eytan Bakshy.  
2023. [Hypervolume knowledge gradient: A looka-  
head approach for multi-objective Bayesian optimiza-  
tion with partial information](#). In *Proceedings of the  
40th International Conference on Machine Learning*,  
volume 202 of *Proceedings of Machine Learning  
Research*, pages 7167–7204. PMLR.

Samuel Daulton, Maximilian Balandat, and Eytan Bak-  
shy. 2021. [Parallel bayesian optimization of multiple  
noisy objectives with expected hypervolume improve-  
ment](#). *CoRR*, abs/2105.08195.

Soham De, Samuel L. Smith, Anushan Fernando, Alek-  
sandar Botev, George Cristian-Muraru, Albert Gu,  
Ruba Haroun, Leonard Berrada, Yutian Chen, Sri-  
vatsan Srinivasan, Guillaume Desjardins, Arnaud  
Doucet, David Budden, Yee Whye Teh, Razvan Pas-  
canu, Nando De Freitas, and Caglar Gulcehre. 2024.  
[Griffin: Mixing gated linear recurrences with local  
attention for efficient language models](#). *Preprint*,  
arXiv:2402.19427.

David Eriksson, Pierce I-Jen Chuang, Samuel Daulton,  
Peng Xia, Akshat Shrivastava, Arun Babu, Shicong

540	Zhao, Ahmed A Aly, Ganesh Venkatesh, and Maximilian Balandat. 2021. <a href="#">Latency-aware neural architecture search with multi-objective bayesian optimization</a> . In <i>8th ICML Workshop on Automated Machine Learning (AutoML)</i> .	597
541		598
542		
543		
544		
545	Igor Fedorov, Ramon Matas, Hokchhay Tann, Chuteng Zhou, Matthew Mattina, and Paul Whatmough. 2022. Udc: Unified dnas for compressible tinymt models for neural processing units. <i>Advances in Neural Information Processing Systems</i> , 35:18456–18471.	
546		
547		
548		
549		
550	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. 2024. <a href="#">Llama guard 3-1b-int4: Compact and efficient safeguard for human-ai conversations</a> . <i>Preprint</i> , arXiv:2411.17713.	
551		
552		
553		
554		
555		
556		
557		
558		
559		
560	Peter I Frazier. 2018. A tutorial on bayesian optimization. <i>arXiv preprint arXiv:1807.02811</i> .	
561		
562	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. 2025. <a href="#">Nemotron-flash: Towards latency-optimal hybrid small language models</a> . In <i>The Thirty-ninth Annual Conference on Neural Information Processing Systems</i> .	
563		
564		
565		
566		
567		
568		
569		
570	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, and 5 others. 2023. <a href="#">A framework for few-shot language model evaluation</a> .	
571		
572		
573		
574		
575		
576		
577		
578	Paolo Glorioso, Quentin Anthony, Yury Tokpanov, James Whittington, Jonathan Pilault, Adam Ibrahim, and Beren Millidge. 2024. <a href="#">Zamba: A compact 7b ssm hybrid model</a> . <i>Preprint</i> , arXiv:2405.16712.	
579		
580		
581		
582	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, and 542 others. 2024. <a href="#">The llama 3 herd of models</a> . <i>Preprint</i> , arXiv:2407.21783.	
583		
584		
585		
586		
587		
588		
589		
590	Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A Roberts. 2024. The unreasonable ineffectiveness of the deeper layers. <i>arXiv preprint arXiv:2403.17887</i> .	
591		
592		
593		
594	Yuxian Gu, Qinghao Hu, Haocheng Xi, Junyu Chen, Shang Yang, Song Han, and Han Cai. 2025. <a href="#">Jet-nemotron: Efficient language model with post neural</a>	
595		
596		
	<a href="#">architecture search</a> . In <i>The Thirty-ninth Annual Conference on Neural Information Processing Systems</i> .	597
		598
	Alexander Heinecke, Evangelos Georganas, Kunal Banerjee, Dhiraj Kalamkar, Narayanan Sundaram, Anand Venkat, Greg Henry, and Hans Pabst. 2017. Understanding the performance of small convolution operations for cnn on intel architecture. In <i>Poster in the International Conference for High Performance Computing, Networking, Storage, and Analysis</i> .	599
		600
		601
		602
		603
		604
		605
	Patrick Huber, Ernie Chang, Wei Wen, Igor Fedorov, Tarek Elgamal, Hanxian Huang, Naveen Suda, Chin-nadhurai 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. 2025. <a href="#">Mobilellm-pro technical report</a> . <i>Preprint</i> , arXiv:2511.06719.	606
		607
		608
		609
		610
		611
		612
		613
	Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. <a href="#">TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension</a> . In <i>Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)</i> , pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.	614
		615
		616
		617
		618
		619
		620
	M. G. Kendall. 1938. <a href="#">A new measure of rank correlation</a> . <i>Biometrika</i> , 30(1/2):81–93.	621
		622
	Kaeun Kim, Ghazal Shams, and Kawon Kim. 2026. From seconds to sentiments: differential effects of chatbot response latency on customer evaluations. <i>International Journal of Human-Computer Interaction</i> , 42(1):597–612.	623
		624
		625
		626
		627
	Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, and 1 others. 2022. <a href="#">Matryoshka representation learning</a> . <i>Advances in Neural Information Processing Systems</i> , 35:30233–30249.	628
		629
		630
		631
		632
		633
	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. 2019. <a href="#">Natural questions: A benchmark for question answering research</a> . <i>Transactions of the Association for Computational Linguistics</i> , 7:452–466.	634
		635
		636
		637
		638
		639
		640
		641
		642
	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, and 42 others. 2025. <a href="#">Jamba: Hybrid transformer-mamba language models</a> . In <i>The Thirteenth International Conference on Learning Representations</i> .	643
		644
		645
		646
		647
		648
		649
		650
		651
	Zechun Liu, Changsheng Zhao, Forrest Iandola, Chen Lai, Yuandong Tian, Igor Fedorov, Yunyang Xiong,	652
		653

654	Ernie Chang, Yangyang Shi, Raghuraman Krishnamoorthi, and 1 others. 2024. MobileLLM: Optimizing sub-billion parameter language models for on-device use cases. <i>arXiv preprint arXiv:2402.14905</i> .	708
655		709
656		710
657		711
658	Jakob Nielsen. 1994. <i>Usability engineering</i> . Morgan Kaufmann.	712
659		713
660	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, and 2 others. 2025. <i>Ax: A platform for adaptive experimentation</i> . In <i>AutoML 2025 ABCD Track</i> .	714
661		715
662		716
663		717
664		718
665		
666		
667		
668		
669	Jonathan Pilault, Mahan Fathi, Orhan Firat, Christopher Pal, Pierre-Luc Bacon, and Ross Goroshin. 2023. <i>Block-state transformers</i> . In <i>Thirty-seventh Conference on Neural Information Processing Systems</i> .	719
670		720
671		721
672		722
673	Liliang Ren, Yang Liu, Yadong Lu, yelong shen, Chen Liang, and Weizhu Chen. 2025. <i>Samba: Simple hybrid state space models for efficient unlimited context language modeling</i> . In <i>The Thirteenth International Conference on Learning Representations</i> .	723
674		724
675		725
676		726
677		727
678	Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. <i>Winogrande: an adversarial winograd schema challenge at scale</i> . <i>Commun. ACM</i> , 64(9):99–106.	728
679		729
680		730
681		731
682	Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. <i>Social IQa: Commonsense reasoning about social interactions</i> . In <i>Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)</i> , pages 4463–4473, Hong Kong, China. Association for Computational Linguistics.	732
683		733
684		734
685		735
686		736
687		737
688		738
689		
690		
691	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, and 197 others. 2025. <i>Gemma 3 technical report</i> . <i>Preprint</i> , arXiv:2503.19786.	
692		
693		
694		
695		
696		
697		
698		
699	Mingyu Yang, Mehdi Rezagholizadeh, Guihong Li, Vikram Appia, and Emad Barsoum. 2025a. <i>Zebra-llama: Towards extremely efficient hybrid models</i> . In <i>The Thirty-ninth Annual Conference on Neural Information Processing Systems</i> .	
700		
701		
702		
703		
704	Songlin Yang, Jan Kautz, and Ali Hatamizadeh. 2025b. <i>Gated delta networks: Improving mamba2 with delta rule</i> . In <i>The Thirteenth International Conference on Learning Representations</i> .	
705		
706		
707		
	Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. <i>HellaSwag: Can a machine really finish your sentence?</i> In <i>Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics</i> , pages 4791–4800, Florence, Italy. Association for Computational Linguistics.	
	<b>A Efficiency Proxies</b>	
	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>B Explanation for the inefficiency of SWA</b>	
	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.	

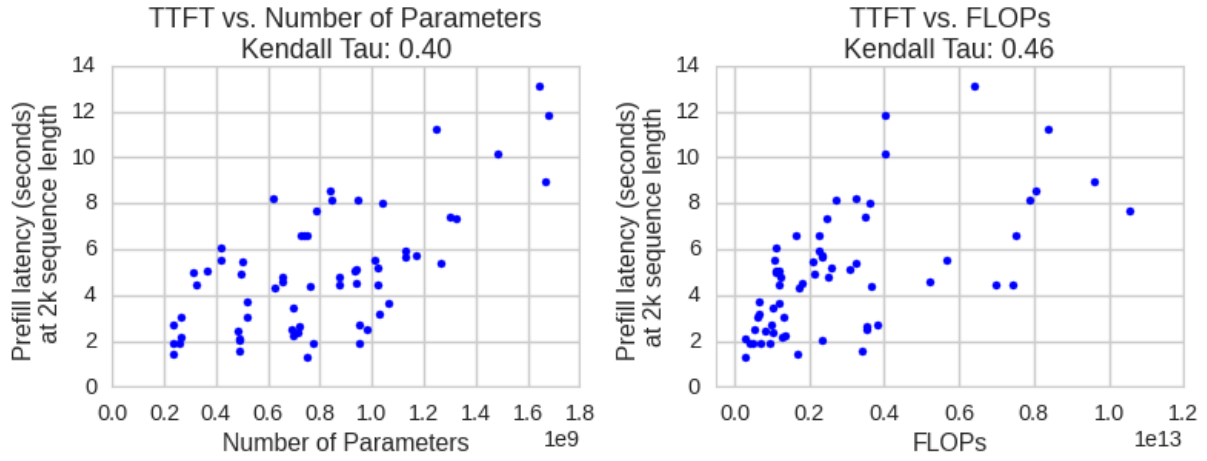


Figure 6: Kendall Tau correlation between TTFT at 2k sequence length vs. model parameter count and FLOPs.

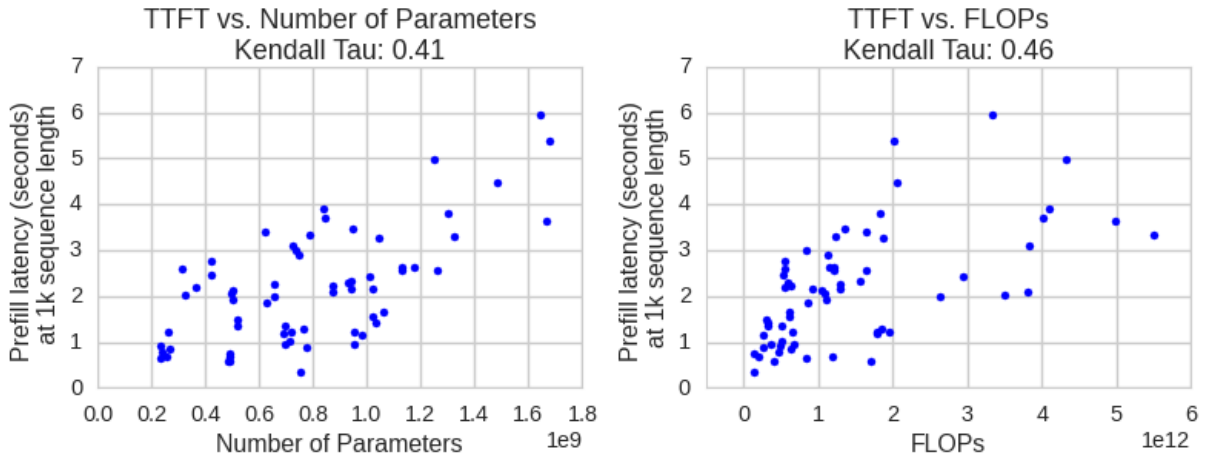


Figure 7: Kendall Tau correlation between TTFT at 1k sequence length vs. model parameter count and FLOPs.

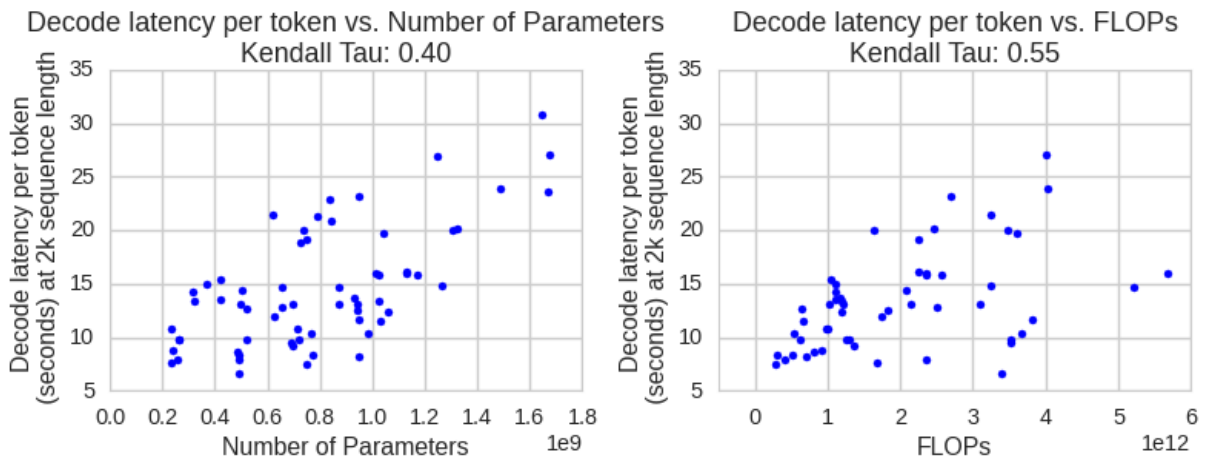


Figure 8: Kendall Tau correlation between decode latency at 2k sequence length vs. model parameter count and FLOPs.

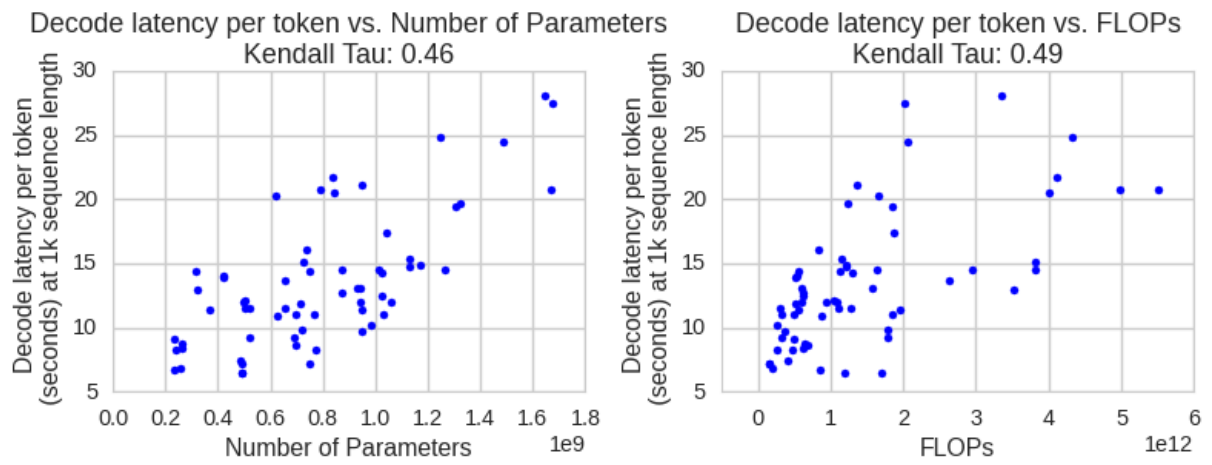


Figure 9: Kendall Tau correlation between decode latency at 1k sequence length vs. model parameter count and FLOPs.