

# LFM2 Technical Report

Liquid AI Team<sup>\*</sup>

 <https://huggingface.co/LiquidAI>

We present LFM2, a family of Liquid Foundation Models designed for efficient on-device deployment and strong task capabilities. Using hardware-in-the-loop architecture search under edge latency and memory constraints, we obtain a compact hybrid backbone that combines gated short convolutions with a small number of grouped query attention blocks, delivering up to  $2\times$  faster prefill and decode on CPUs compared to similarly sized models. The LFM2 family covers 350M–8.3B parameters, including dense models (350M, 700M, 1.2B, 2.6B) and a mixture-of-experts variant (8.3B total, 1.5B active), all with 32K context length. LFM2’s training pipeline includes a tempered, decoupled Top-K knowledge distillation objective that avoids support mismatch; curriculum learning with difficulty-ordered data; and a three-stage post-training recipe of supervised fine-tuning, length-normalized preference optimization, and model merging. Pre-trained on 10–12T tokens, LFM2 models achieve strong results across diverse benchmarks; for example, LFM2-2.6B reaches 79.56% on IFEval and 82.41% on GSM8K. We further build multimodal and retrieval variants: LFM2-VL for vision–language tasks, LFM2-Audio for speech, and LFM2-ColBERT for retrieval. LFM2-VL supports tunable accuracy–latency tradeoffs via token-efficient visual processing, while LFM2-Audio separates audio input and output pathways to enable real-time speech-to-speech interaction competitive with models  $3\times$  larger. LFM2-ColBERT provides a low-latency encoder for queries and documents, enabling high-performance retrieval across multiple languages. All models are released with open weights and deployment packages for ExecuTorch, llama.cpp, and vLLM, making LFM2 a practical base for edge applications that need fast, memory-efficient inference and strong task capabilities.

**Publication Date:** December 1, 2025

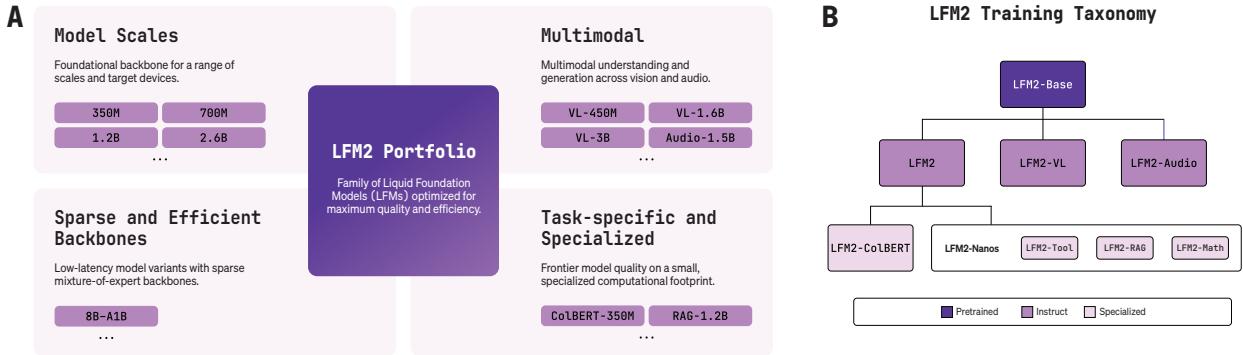
**Correspondence:** [mathias@liquid.ai](mailto:mathias@liquid.ai)

## 1 Introduction

Generative models that can be natively deployed on-device have the potential to bring transformative impact across sectors and industries (Xu et al., 2024). Applications such as voice assistants, local copilots, and agentic workflows that plan, call tools, and read sensors in tight loops must run under strict latency, memory, and energy budgets on phones, tablets, laptops, and system-on-chip (SoC) platforms. In these settings, time-to-first-token (TTFT), stable inter-token latency, and private or offline execution are not optional features, but rather hard constraints. Although recent open model families have improved efficiency at larger scales, targeting tens of billions of parameters and accelerator-heavy deployments (Gemma Team et al., 2025; Liu et al., 2024; Yang et al., 2025a), significant needs remain in the small-model, edge-first regime. We

---

\*Please cite the author as “Liquid AI (2025)”. See Section 10 (Authors) for the list of contributors.



**Fig. 1: The LFM2 Portfolio.** (A) We present a family of Liquid Foundation Models (LFMs) across a suite of scales, modalities, and edge capabilities. (B) The taxonomy of the LFM2 portfolio steps from co-designed architectures and pre-trained base models, to multi-modality, and specialized downstream tasks.

seek models that lead in quality, speed, and memory efficiency on CPUs and heterogeneous NPUs, while remaining practical to pre-train, post-train, and deploy widely.

Here, we introduce **LFM2**, the second generation of Liquid Foundation Models (LFMs) that are optimized explicitly for on-device deployment (Figure 1). The LFM2 design is *edge-first*: we co-design architecture, pre-training, and post-training around the objective of maximizing downstream quality subject to device-side latency and peak memory constraints. We release dense model checkpoints (350M–2.6B) with 32K context, an 8.3B mixture-of-experts (MoE) model with 1.5B active parameters, multimodal vision-language and audio-language variants, as well as a late-interaction retrieval model. We also release the initial series of LFM2-Nanos: LFM2 models that were further pre-trained and post-trained for domain or task-specific use cases such as data extraction, tool/function calling, retrieval augmented generation (RAG), and mathematical reasoning. All LFM2 models ship with open weights and deployment guides for Transformers (Wolf et al., 2020), 11ama.cpp, ExecuTorch, and vLLM (Kwon et al., 2023), along with quantized variants suited to edge runtimes<sup>1</sup>. Key aspects of the LFM2 family include:

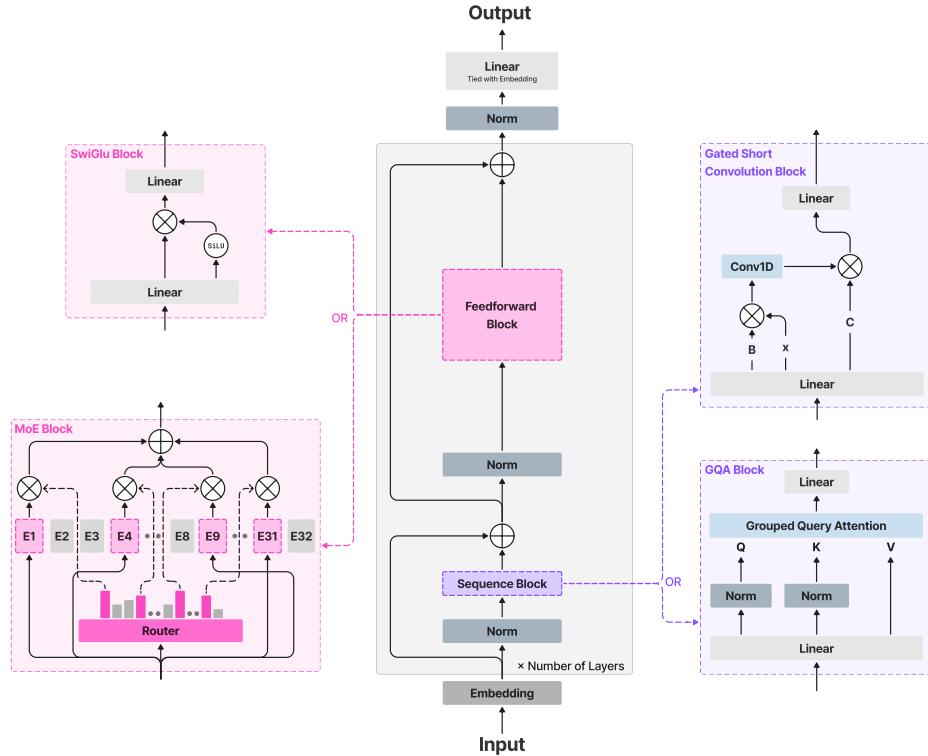
- **Edge-first backbone for small, fast models.** A hardware-in-the-loop architecture search selects a minimal hybrid architecture that combines short-range, input-aware gated convolutions with grouped-query attention (GQA) in a layout tuned for quality under strict speed and memory budgets. Using this backbone, dense LFM2 models from 350M to 2.6B parameters (32K context) and an MoE variant (LFM2-8B-A1B) cover a range of quality-latency-memory targets for on-device applications.
- **Pre-training and post-training for small model, on-device workflows.** LFM2 checkpoints undergo a 10-12T token pre-training phase and a long-context mid-training phase. To improve small-model quality without prohibitive cost, we introduce a tempered, decoupled Top-K distillation objective that avoids support mismatch. A three-stage post-training pipeline improves performance and robustness at small model scales.
- **Native multimodality.** The LFM2 vision-language model, **LFM2-VL**, augments the language backbone with a SigLIP2 (Tschanne et al., 2025) vision encoder and a lightweight connector, designed to enable flexible accuracy-latency trade-offs on device. The audio-language model, **LFM2-Audio**, turns the backbone into an end-to-end audio-text model that ingests continuous audio features and autoregressively generates either text or discrete audio tokens, enabling low-latency speech assistants and transcription/translation in a single model.
- **Multi- and cross-lingual information retrieval.** **LFM2-CoBERT** adds a dense module on top of the backbone, yielding a low-latency encoder for both queries and documents. It follows a late-interaction paradigm that uses a max-similarity operator (Khattab and Zaharia, 2020), enabling high-performance retrieval across multiple languages.

<sup>1</sup>See the LiquidAI organization on Hugging Face: <https://huggingface.co/LiquidAI>.

- **Strong performance with top efficiency.** Across sizes and applications, LFM2 models achieve strong tradeoffs between quality, latency, and memory, all critical considerations for edge deployment. On CPUs, LFM2 dense checkpoints deliver up to  $\sim 2\times$  prefill and decode speedups versus similarly sized baselines while maintaining or improving accuracy on benchmarks. LFM2-8B-A1B attains 3–4B-class quality at  $\sim 1.5$ B active parameters.

This report first details the architecture and hardware-in-the-loop search procedure (Section 2) used to design the LFM2 backbone. Detailed descriptions of the pre-training pipeline, including the long-context mid-training and decoupled Top-K distillation objective, and the three-stage post-training pipeline are provided in Sections 3 and 4, respectively, followed by results on model performance. Moving from the LFM2 language backbone to specific applications, we introduce the LFM2-VL (Section 5) and LFM2-Audio (Section 6) multimodal models, covering their design, training, and evaluation, as well as the LFM2-ColBERT-350M model for information retrieval. We conclude with an extended discussion of related work (Section 8) and a discussion of LFM2’s limitations, opportunities for development, and impact on edge deployment (Section 9).

## 2 Architecture



**Fig. 2: LFM2 architecture.** The LFM2 architecture supports both dense and mixture-of-experts (MoE) variants. Note the MoE experts are also SwiGLU blocks.

The LFM2 family instantiates the edge-first design introduced in Section 1 with a minimal hybrid backbone designed via hardware-in-the-loop architecture search. We release 4 dense models: LFM2-350M, LFM2-700M, LFM2-1.2B, and LFM2-2.6B, as well as an MoE variant, LFM2-8B-A1B, which has 8.3B total parameters and 1.5B active parameters. Figure 2 illustrates the shared backbone and the MoE extension. In this section, we describe the architecture search procedure (Section 2.1), the resulting dense backbone (Section 2.2) and MoE (Section 2.3) configurations, and their inference performance (Section 2.4).

---

## 2.1 Architecture Optimization Process

The design objective for LFM2 is edge-first: maximize downstream quality while staying within tight device-side budgets on latency and peak memory on CPUs and heterogeneous NPUs. Recent efficient architectures (Blakeman et al., 2025; Kimi Team et al., 2025; Waleffe et al., 2024) typically combine three ingredients: (i) alternative sequence operators such as linear attention variants or state space models (SSMs) (Dao and Gu, 2024; Yang et al., 2025b), (ii) short-range convolutions (generally integrated in the alternative sequence block), and (iii) a non-trivial fraction of standard softmax attention layers to recover quality deficits (e.g., long-range retrieval abilities) observed in models without softmax attention layers (Park et al., 2024; Waleffe et al., 2024; Wen et al., 2025). We explicitly search this design space with a hardware-in-the-loop search procedure and find that, under realistic edge latency and memory budgets, a minimal hybrid suffices: gated short convolutions for most layers plus a small minority of grouped-query attention (GQA) layers, without additional SSM or linear-attention operators.

**Objectives and constraints.** We frame architecture selection as a Pareto optimization over three axes:

1. **Quality**: performance on an internal suite of 50+ evaluations (spanning knowledge recall, multi-hop reasoning, instruction following, multilingual robustness, tool use, math, and long context performance) after training each candidate architecture on a reference dataset.
2. **Latency**: time-to-first-token (TTFT) and p50/p95 decode latency (ms/token) at batch= 1, as well as prefill throughput (tokens/s) on representative prompts.
3. **Peak memory**: measured as maximum resident set size (RSS) during prefill and decode at target context windows (4K and 32K).

Candidate architectures that violate device-side budgets on TTFT, decode latency, or peak memory are discarded. The remaining candidates are ranked by hypervolume improvement on the quality–latency–memory Pareto frontier.

**Search space.** We consider decoder-only stacks built from the following block families:

- **Local context and subquadratic blocks**: gated short convolution blocks with varying kernel sizes, sliding-window attention (Child et al., 2019), and a family of sub-quadratic sequence blocks including linear attention variants (Katharopoulos et al., 2020; Yang et al., 2024a, 2025b); state-space variants such as S4 (Gu et al., 2022), Liquid-S4 (Hasani et al., 2023), S5 (Smith et al., 2023b), RTF (Parnichkun et al., 2024), Mamba (Gu et al., 2022), and Mamba2 (Dao and Gu, 2024); Liquid-Time Constant networks such as CfC (Hasani et al., 2022), as well as internal variants of efficient sequence blocks. These blocks typically combine a depthwise short convolution with a longer-range linear attention/SSM component (Fu et al., 2023; Gu and Dao, 2024; Poli et al., 2023). The search space includes variants that keep only the short convolution submodule (i.e., the gated short convolution block) as well as variants that retain the full hybrid operator. This allows the search to attribute performance gains to specific computational units within the overall operator. While linear attention and SSM variants can perform global processing of the input, we consider them to be in the same class as local context blocks due to their limitations in retrieval-intensive tasks (Arora et al., 2024b; Bluir et al., 2024; Parnichkun et al., 2025; Wen et al., 2025) (see Section 8.2 for a broader discussion).
- **Global context blocks**: grouped-query attention (GQA) (Ainslie et al., 2023) with varying group counts and head dimensions, augmented with QK-Norm (Dehghani et al., 2023a).
- **Position-wise blocks**: SwiGLU feed-forward blocks (Shazeer, 2020) with expansion ratios chosen by search.
- **Layout**: interleaving patterns of local context blocks, global context blocks, position-wise blocks, and overall block counts under fixed parameter budgets, including options for shared weights and cache reuse.
- **MoE options**: per-layer sparse FFNs with varying width and expert granularity.

---

**On-device profiling.** Every candidate is compiled to the deployment stacks with identical settings (batch=1, fixed context windows at 4K/32K, and matched quantization/backends) and benchmarked on target devices:

- **CPU path:** ExecuTorch (8da4w) and `11ama.cpp` (Q4.0) on Samsung Galaxy S24 Ultra (Qualcomm Snapdragon 8 Gen 3 SoC) and AMD Ryzen HX 370.
- **Accelerator path:** vLLM for single-request and online batching (used for sanity checks; the primary target remains on-device CPU deployment).

We record TTFT, prefill throughput (tokens/s), decode ms/token (p50/p95), and peak memory with identical prompts and tokenizer settings.

**Evaluation and selection** We consider quality estimation and deployability checks. Quality is measured via targeted training experiments scored on the internal evaluation suite. Deployability is assessed via the on-device profiling described above. Training experiments are run at scales large enough to surface meaningful quality differences, while device measurements are taken on representative edge hardware using the same runtimes at which we plan to ship. Configurations that preserve their advantage across both quality and deployability checks advance, while others are dropped early. Among feasible models, we retain those on the quality-latency-memory Pareto frontier and carry them forward in the search.

Our earlier academic prototype (STAR) (Thomas et al., 2024) explored a specific design space of operator/layout choices with an evolutionary search heuristic optimized on proxy signals (i.e., perplexity for quality, cache size for efficiency). In practice, these proxies do not transfer reliably to downstream task scores or device-level latency and memory, limiting their utility as optimization objectives. By contrast, the LFM2 pipeline centers the *objective*: downstream task scores and hardware-in-the-loop TTFT/latency/memory on release runtimes. In practice, we found this has a much larger impact than the particulars of the search space or choice of search heuristic.

**Outcomes.** Across size targets, the hardware-in-the-loop search repeatedly selects a *minimal hybrid* architecture where most blocks are inexpensive gated short convolution blocks, interleaved with a small minority of GQA blocks. Under identical on-device performance budgets, augmenting these stacks (as in recent hybrid variants) with linear-attention, state-space, or additional convolution operators does not improve aggregate quality on the evaluation suite and typically worsens device metrics. Empirically, the selected hybrids:

- match or exceed the aggregate quality of attention-heavier and mixed (conv+linear/SSM/conv) baselines at the same budget;
- reduce decode latency (p50/p95) and increase prefill throughput at batch= 1 under identical tokenizer, prompt, quantization, and backend settings;
- lower peak RSS at long context (4K/32K), consistent with reduced KV-cache versus attention-heavy layouts.

These results suggest that, in the on-device regime, most of the benefits attributed to recent hybrid SSM/linear-attention blocks can be captured by their short convolutional submodules plus a small number of global attention layers. We therefore carry forward designs that minimize global blocks while prioritizing inexpensive, gated short convolution blocks elsewhere. The released dense backbones (Section 2) and the sparse-FFN placement in LFM2-8B-A1B (Section 2) instantiate this recipe. Detailed speed and memory measurements appear in Section 2.4.

## 2.2 LFM2 Dense Models

**Block choices and layout.** The LFM2 dense models instantiate the minimal hybrid repeatedly selected by the hardware-in-the-loop search (Section 2.1): a majority of inexpensive, gated short convolution blocks interleaved with a minority of grouped-query attention (GQA) blocks, plus SwiGLU (Shazeer, 2020) position-wise multi-layer perceptrons (MLPs). All layers use pre-norm RMSNorm (Zhang and Sennrich, 2019) and attention blocks use RoPE (Su et al., 2024) with QK-Norm (Dehghani et al., 2023a).

---

Model	Params T/A	Backbone						MoE		
		Layers	$d_{\text{model}}$	FF dim	H/KV/H <sub>size</sub>	Attn. Blocks	Conv k	E	Top-k	FF <sub>MoE</sub>
LFM2-350M	350M/—	16	1024	4608	16/8/64	6	3	—	—	—
LFM2-700M	700M/—	16	1536	6912	24/8/64	6	3	—	—	—
LFM2-1.2B	1.2B/—	16	2048	8192	32/8/64	6	3	—	—	—
LFM2-2.6B	2.6B/—	30	2048	10752	32/8/64	8	3	—	—	—
LFM2-8B-A1B	8.3B/1.5B	24	2048	7168	32/8/64	6	3	32	4	1792

**Table 1:** LFM2 model hyperparameters. “Backbone” columns list hyperparameters shared between dense and MoE models. T/A: total and active parameter counts. H/KV/H<sub>size</sub>: Number of attention heads / KV groups / head size. “MoE” columns list experts per layer (E), active experts (Top-k), and per-expert FF size. For LFM2-8B-A1B, all layers except the first two are MoE.

**Gated short convolution block.** Given an input hidden sequence  $\mathbf{h} \in \mathbb{R}^{L \times d}$  (batch dimension omitted for clarity), each gated short convolution block applies input-dependent multiplicative gating around a depthwise short convolution:

$$(\mathbf{B}, \mathbf{C}, \tilde{\mathbf{h}}) = \text{Linear}(\mathbf{h}), \quad \mathbf{y} = \mathbf{B} \odot \tilde{\mathbf{h}}, \quad \mathbf{z} = \text{Conv}_k(\mathbf{y}), \quad \mathbf{o} = \text{Linear}_{\text{out}}(\mathbf{C} \odot \mathbf{z}).$$

Here  $\text{Linear} : \mathbb{R}^d \rightarrow \mathbb{R}^{3d}$  is a linear map applied position-wise across the sequence length,  $L$ , and whose output channels are split along the feature dimension into  $(\mathbf{B}, \mathbf{C}, \tilde{\mathbf{h}})$  with  $\mathbf{B}, \mathbf{C}, \tilde{\mathbf{h}} \in \mathbb{R}^{L \times d}$ . The intermediate tensors  $\mathbf{y}, \mathbf{z}, \mathbf{o}$  also lie in  $\mathbb{R}^{L \times d}$ .  $\text{Conv}_k : \mathbb{R}^{L \times d} \rightarrow \mathbb{R}^{L \times d}$  is a depthwise 1D convolution along the sequence with kernel size  $k$ , and  $\odot$  denotes element-wise multiplication.  $\text{Linear}_{\text{out}} : \mathbb{R}^d \rightarrow \mathbb{R}^d$  is the linear output projection. This operator provides fast local mixing with excellent cache behavior on CPUs.

This operator is closely related to the short-range components that appear inside many recent efficient sequence blocks (Dao and Gu, 2024; Fu et al., 2023; Gu and Dao, 2024; Poli et al., 2023; Yang et al., 2025b). The search results from Section 2.1 can be viewed as an ablation of these hybrids in the on-device setting. Once a handful of GQA blocks are available to handle long-range retrieval, the inexpensive gated short convolution alone is sufficient to reach the best quality–latency–memory trade-off we observe, without additional linear attention/SSM/long convolution branches.

**Attention and MLP.** GQA reduces KV traffic by sharing keys/values across head groups while preserving multi-head queries (Ainslie et al., 2023). Position-wise MLPs use SwiGLU with a size-dependent expansion ratio chosen by the search.

**Tokenizer and special tokens.** We use a byte-level BPE tokenizer (Sennrich et al., 2016) with a 65,536-token vocabulary. We used the same pre-training dataset discussed in Section 3 to train the LFM2 tokenizer, with a focus on encoding efficiency of the English, Japanese, Arabic, Korean, Spanish, French, and German languages. We additionally include JSON and other code-like data to improve tokenization of structured formats.

The tokenizer includes special tokens for fill-in-the-middle training objectives (Bavarian et al., 2022), tool calling, and the ChatML chat template.

**Released sizes.** We release dense checkpoints at 350M, 700M, 1.2B, and 2.6B parameters, all with a 32,768 token context window. Table 1 summarizes the main model hyperparameters.

### 2.3 LFM2 MoE

LFM2-8B-A1B (8.3B total, 1.5B active) targets the on-device regime where compute per token, not weight storage, dominates perceived latency. By activating only a sparse subset of experts per token, the model attains 3–4B-class quality (Table 7) at roughly 1.5B-class decode cost. We keep the fast LFM2 hybrid backbone (Section 2.2) and replace dense MLPs with sparse MoE MLPs in most layers. For stability, the first two

---

layers remain dense while all subsequent layers include an MoE block. Experts themselves are SwiGLU MLPs. Each MoE layer has 32 experts and selects the Top-k=4 experts per token with a normalized sigmoid router and adaptive routing biases for load balancing (Liu et al., 2024). See Table 1 for a summary of the main LFM2-8B-A1B settings.

## 2.4 Inference Performance

**Experimental setup.** We evaluate the inference efficiency of both dense and MoE LFM2 models on two representative on-device CPU targets: a Snapdragon 8 Elite based Samsung Galaxy S25 smartphone (a newer model than we used for the architecture search in Section 2.1) and an AMD Ryzen AI 9 HX 370 laptop CPU. In both settings, we focus on latency-sensitive, single-stream usage with batch size equal to 1. All models are run with `llama.cpp` (Gerganov and contributors, 2023) using the Q4\_0 quantization format. For ease of comparison and reproducibility, we measure *prefill throughput* (prompt tokens processed per second) for 1K and 4K-token prompts and *decode throughput* (tokens generated per second) when producing 100 continuation tokens from 1K and 4K-token prefixes. Tables 2 and 3 summarize the results for the LFM2 family alongside contemporary open baselines (Bakouch et al., 2025; Gemma Team et al., 2025; Grattafiori et al., 2024; IBM Research, 2025; Yang et al., 2025a) at comparable parameter scales.

Model	Prefill throughput (tokens/s)		Decode throughput (tokens/s)	
	1K	4K	1K	4K
LFM2-350M	1,067	657	194.1	143.8
Granite-4.0-350M	528	210	132.9	70.7
Granite-4.0-H-350M	784	594	150.7	119.3
LFM2-700M	522	341	104.2	80.2
Qwen3-0.6B	318	136	76.7	41.8
LFM2-1.2B	335	222	69.8	55.5
Llama-3.2-1B	229	130	54.6	37.8
Qwen3-1.7B	140	98	39.7	26.9
Gemma-3-1B	377	295	67.5	67.1
Granite-4.0-1B	147	79	41.3	25.0
Granite-4.0-H-1B	186	159	46.1	44.1
LFM2-2.6B	143	116	33.8	30.0
LFM2-8B-A1B	85	76	48.6	41.9
Llama-3.2-3B	79	51	24.2	15.8
Qwen3-4B	57	35	17.2	11.4
Gemma-3-4B	72	63	18.3	17.9
SmolLM3-3B	98	66	26.1	19.2
Granite-4.0-Micro	65	38	20.2	12.1
Granite-4.0-H-Micro	83	77	24.9	23.5
Granite-4.0-H-Tiny	86	86	39.2	37.4

**Table 2:** Prefill and decode throughput on the Samsung Galaxy S25 device with a Qualcomm Snapdragon 8 Elite SoC (batch size 1). Prefill columns report tokens per second when processing prompts of length 1K/4K tokens. Decode columns report tokens per second when generating 100 tokens starting from prefixes of length 1K/4K tokens. All models are run with `llama.cpp` and Q4\_0 format with CPU backend.

**Smartphone-class CPU (Samsung Galaxy S25 with a Qualcomm Snapdragon 8 Elite SoC).** On the S25 device (Table 2), LFM2 models are generally faster than similarly sized baselines across scales.

In the 350M class, LFM2-350M yields about 2–3× higher prefill throughput than Granite-4.0-350M and improves decode throughput by roughly 1.5–2.0× across 1K and 4K contexts. Relative to the hybrid Granite-4.0-H-350M, LFM2-350M remains ~10–35% faster in both prefill and decode. At the 700M scale, LFM2-700M outperforms Qwen3-0.6B across all four settings. Prefill throughput improves by about 1.6–2.5×, while decode throughput improves by roughly 1.4–2×.

At the 1B scale, LFM2-1.2B improves over Granite-4.0-1B and Qwen3-1.7B by 2.3–2.8× on prefill and 1.7–2.2× on decode across both context lengths. Gemma-3-1B is the strongest 1B-class baseline in terms of raw throughput; however, LFM2-1.2B attains 0.75 – 0.9× of its prefill throughput, while offering slightly higher 1K decode throughput and remaining within 20% on 4K decode. In contrast, relative to Llama-3.2-1B, LFM2-1.2B is faster across all four measurements, with roughly 1.5–1.7× higher prefill throughput and 1.3–1.5× higher decode throughput.

In the 2–4B regime, the dense LFM2-2.6B model delivers about 2–3× higher throughput than Qwen3-4B across prefill and decode, and also outperforms Gemma-3-4B, Granite Micro/H variants, SmoILM3-3B, and Llama-3.2-3B in both metrics. The MoE model LFM2-8B-A1B attains prefill throughput comparable to Granite-4.0-H-Tiny (another hybrid MoE), but with 1.1–1.3× higher decode throughput, while providing roughly 2.8–3.7× higher decode throughput than dense 4B-class baselines such as Qwen3-4B. We have found that there is still room for large improvements for MoEs on CPU, and we are developing improved kernels that better utilize CPU hardware.

**Laptop-class CPU (AMD Ryzen HX 370).** On the Ryzen AI 9 HX 370 CPU (Table 3), absolute throughputs increase substantially compared to the S25, and the relative trends largely mirror the smartphone results.

Model	Prefill throughput (tokens/s)		Decode throughput (tokens/s)	
	1K	4K	1K	4K
LFM2-350M	7,534	5,540	207.7	170.5
Granite-4.0-350M	5,499	2,938	225.0	166.2
Granite-4.0-H-350M	4,568	4,250	134.7	123.2
LFM2-700M	4,214	3,311	134.6	118.4
Qwen3-0.6B	3,594	2,204	116.0	53.4
LFM2-1.2B	2,784	2,302	99.7	89.0
Llama-3.2-1B	2,912	1,890	97.3	73.5
Qwen3-1.7B	2,019	1,491	60.8	38.5
Gemma-3-1B	3,972	3,809	103.6	99.3
Granite-4.0-1B	1,823	1,256	64.5	43.6
Granite-4.0-H-1B	1,565	1,523	59.6	57.1
LFM2-2.6B	1,335	1,171	50.3	46.8
LFM2-8B-A1B	1,320	1,185	74.9	69.1
Llama-3.2-3B	1,179	916	38.3	28.2
Qwen3-4B	861	628	31.0	22.2
Gemma-3-4B	1,119	1,054	33.0	31.6
SmoILM3-3B	1,248	982	42.4	33.4
Granite-4.0-Micro	929	608	36.7	28.9
Granite-4.0-H-Micro	932	896	34.7	33.8
Granite-4.0-H-Tiny	1,050	1,046	55.0	52.1

**Table 3:** Prefill and decode throughput on HX 370 CPU (batch size 1). Prefill columns report tokens per second when processing prompts of length 1K/4K tokens. Decode columns report tokens per second when generating 100 tokens starting from prefixes of length 1K/4K tokens. All models are run with `llama.cpp` and Q4.0 format with CPU backend.

**Summary.** Across both smartphone and laptop class CPUs, LFM2 models generally achieve substantial latency and throughput improvements over similarly sized open baselines. Together with the quality measurements presented in Section 4.5, these results support the edge-first design of the LFM2 backbone and its suitability for latency-sensitive on-device deployment (Figure 4).

### 3 Pre-Training

We pre-train LFM2 using a multi-stage pipeline designed for small, on-device models. During pre-training, we combine standard next-token prediction with a tempered, decoupled Top-K knowledge distillation objective (Section 3.3), using an internal version of LFM1-7B as a teacher. This section details the pre-training data mixture (Section 3.1), the training stages (Section 3.2), and the decoupled Top-K distillation objective (Section 3.3).

#### 3.1 Training Data

The LFM2 dense models are pre-trained on a mixture comprising roughly 75% English text, 20% multilingual text, and 5% code. We prioritize Japanese, Arabic, Korean, Spanish, French, and German for multilingual coverage, with additional base support for Chinese, Italian, and Portuguese. The MoE model LFM2-8B-A1B uses a similar mixture, but with a heavier emphasis on code (60% English, 25% multilingual, 15% code). For code data, 50% of examples use a fill-in-the-middle (FIM) objective (Bavarian et al., 2022).

#### 3.2 Training Stages

The released dense LFM2 model checkpoints are pre-trained for 10T tokens at a context length of 4,096 tokens. We then perform a mid-training phase on an additional 1T higher-quality tokens, including sources with naturally long context, using a 32,768-token context window and an accelerated learning-rate decay schedule. The released LFM2-8B-A1B checkpoint follows the same two-stage recipe but is trained for 12T tokens in the initial phase before the 1T-token long-context mid-training stage.

#### 3.3 Decoupled Top-K Knowledge Distillation

During pre-training, we leverage LFM1-7B as a teacher model in a knowledge distillation (KD) framework (Hinton et al., 2015). To reduce storage and bandwidth, we distill the teacher distribution  $P_T(x | x_c)$  for each token  $x$  given its context  $x_c$  to a student  $P_S(x | x_c)$ , using only the Top-K=32 teacher logits per token. Naively applying the Kullback–Leibler (KL) divergence to a Top-K truncated teacher distribution, especially under temperature scaling, can cause support mismatch and unstable losses. We address this by decomposing the KL via the chain rule into (i) a binary term that matches the total probability mass assigned to the Top-K set, and (ii) a conditional KL within the Top-K, to which we apply temperature. This “decoupled, tempered Top-K” objective avoids support mismatch while preserving the benefits of temperature-scaled distillation.

Let  $\mathcal{A}$  be the vocabulary and  $\mathcal{T}(x_c) \subset \mathcal{A}$  be the teacher’s Top-K set for context  $x_c$ . Define

$$P_T(\mathcal{T} | x_c) = \sum_{x \in \mathcal{T}(x_c)} P_T(x | x_c), \quad P_S(\mathcal{T} | x_c) = \sum_{x \in \mathcal{T}(x_c)} P_S(x | x_c),$$

$$P_T(x | \mathcal{T}, x_c) = \frac{P_T(x | x_c)}{P_T(\mathcal{T} | x_c)}, \quad P_S(x | \mathcal{T}, x_c) = \frac{P_S(x | x_c)}{P_S(\mathcal{T} | x_c)} \quad (x \in \mathcal{T}).$$

We denote applying a temperature  $\tau$  as

$$p^{(\tau)}(x) = \frac{p(x)^{1/\tau}}{\sum_y p(y)^{1/\tau}}, \quad D_{\text{KL}}^{(\tau)}(p \| q) = \tau^2 D_{\text{KL}}(p^{(\tau)} \| q^{(\tau)}).$$

The decoupled, tempered Top-K objective (per token) is

$$\mathcal{L}_{\text{DTK}}(x_c) = \underbrace{D_{\text{KL}}(\text{Bern}(P_T(\mathcal{T} | x_c)) \| \text{Bern}(P_S(\mathcal{T} | x_c)))}_{\mathcal{L}_B} \quad (1)$$

$$+ \underbrace{P_T(\mathcal{T} | x_c) D_{\text{KL}}^{(\tau)}(P_T(\cdot | \mathcal{T}, x_c) \| P_S(\cdot | \mathcal{T}, x_c))}_{\mathcal{L}_T} \quad (2)$$

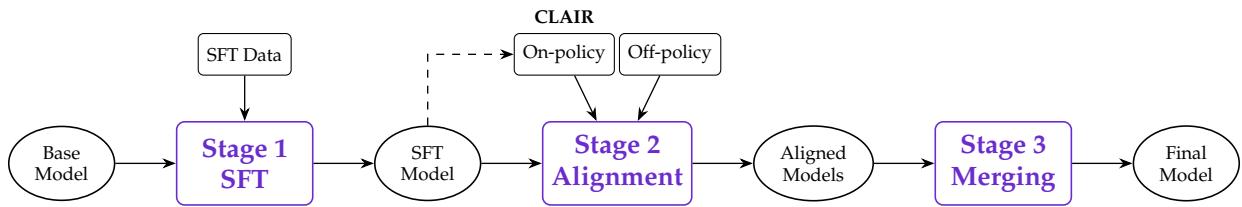
where  $\text{Bern}(p)$  denotes the Bernoulli distribution with success probability  $p$ . The first term  $\mathcal{L}_B$  is a binary KL that trains the student to put the same *total probability mass* into the teacher’s Top-K set as the teacher does. The second term  $\mathcal{L}_T$  is a conditional Top-K KL divergence that, given the next token lies in the Top-K, trains the student to match the teacher’s *relative probabilities* among those  $K$  tokens.

We apply temperature **only** to the conditional Top-K distributions inside  $\mathcal{L}_T$ . The Bernoulli membership term  $\mathcal{L}_B$  and the Top-K mass  $P_T(\mathcal{T} | x_c)$  remain untempered, which prevents the support mismatch that arises when tempering a Top-K truncated distribution over the full vocabulary. Note that with  $\tau = 1$ ,  $\mathcal{L}_{\text{DTK}}$  is a lower bound to the full forward KL because it omits the teacher tail where logits are unavailable. For  $\tau > 1$ , this can be viewed as optimizing a tempered surrogate of this bound. Please see Appendix A for the full derivation and more details. Finally, the overall  $\mathcal{L}_{\text{DTK}}$  term is balanced with next-token cross-entropy on hard labels. See Section 8.3 for an expanded discussion on related approaches such as Zhao et al. (2022) and Anshumann et al. (2025).

## 4 Post-Training

### 4.1 Overview

Following the 10-12T token pre-training phase, each LFM2 checkpoint undergoes a three-stage post-training pipeline (Figure 3) with two main objectives. The first goal is teaching the chat template to convert the base model into a conversational assistant that can reliably answer questions, follow instructions, and maintain multi-turn coherence. The second goal is to improve downstream capabilities that are relevant to end users, such as Retrieval-Augmented Generation (RAG) and function calling.



**Fig. 3: Overview of the three-stage post-training pipeline.**

The three-stage pipeline consists of:

1. **Supervised Fine-Tuning (SFT):** Large general-purpose mixture to provide good capabilities on downstream tasks and teach the chat template.
2. **Preference Alignment:** Enabling efficient preference optimization through direct alignment on offline data including on-policy samples generated from the SFT checkpoint.
3. **Model Merging:** Systematic evaluation and combination of candidate models to increase robustness and optimize performance.

The following subsections detail data recipes, algorithmic innovations, filtering heuristics, and hyper-parameter configurations used in the LFM2 post-training pipeline.

### 4.2 Stage 1: Supervised Fine-Tuning

#### 4.2.1 SFT Data Mixture

The SFT corpus we use comprises approximately 5.39 million samples for the 350M, 700M, 1.2B, and 2.6B models, and 9.24 million samples for the 8B-A1B model (which includes additional math and code-focused data). The two datasets integrate 67 and 79 carefully curated sources, respectively, spanning open-source datasets, licensed data, and targeted synthetic generations optimized for on-device deployment scenarios,

Category	Share (%)	Source	Category	Share (%)	Source
General-purpose	26.6	O	Mathematical reasoning	26.2	P,L,O
Instruction following	17.1	P	General-purpose	16.3	O
RAG	13.2	P	Instruction following	11.3	P
Real-world chats	10.1	O	Code	10.2	O,P
Tool use	10.1	P	Real-world chats	10.0	O
Off-policy preferences	8.2	P	RAG	8.3	P
Mathematical reasoning	7.4	P,L,O	Tool use	7.3	P
Long context	6.7	P	Off-policy preferences	5.3	P
Other	0.6	O,P	Long context	3.2	P
			Other	1.9	O,P

(a) Dense models (350M, 700M, 1.2B, 2.6B)

(b) LFM2-8B-A1B

**Table 4: SFT data mixture composition for LFM2 models.** Sources: P = proprietary, L = licensed, O = open-source.

with multilingual samples integrated throughout each category. Table 4 details the composition of the instruction datasets for the LFM2 dense and MoE models.

In particular, general-purpose denotes a broad mixture of conversational and task-oriented data (including some code, reasoning, and instruction-like instances) that does not cleanly fall into any single specialized bucket and serves to balance the overall domain coverage. Off-policy preferences correspond to the off-policy component of the preference dataset: we include only the “chosen” responses from preference pairs that we ultimately want the model to imitate during SFT, so as to make the supervised data distribution closer to the policy targeted by preference optimization.

**Skill-specific mixture design.** We develop the final SFT composition through an iterative process targeting individual capabilities in isolation before combining them into a balanced mixture. This approach allows us to approximate the upper bound performance for each evaluation domain given the training setup. The resulting mixture emphasizes diverse conversational data while maintaining strong performance across technical domains.

**Multilingual data.** Rather than treating multilingual capabilities as a separate domain, we systematically integrate multilingual samples across most task categories. The corpus is 80% English, with the remaining 20% uniformly distributed across seven languages: Arabic, Chinese, French, German, Japanese, Korean, and Spanish. This approach ensures that the models develop strong multilingual capabilities organically within each skill domain, providing greater diversity and more natural cross-lingual transfer compared to traditional approaches that rely on distinct multilingual datasets. Each category in the mixture contains representative samples in multiple languages, ensuring that specialized capabilities such as mathematical reasoning and tool use are developed multilingually from the ground up.

**Data quality pipeline.** We implement a comprehensive multi-stage filtering process to ensure dataset quality and prevent evaluation contamination. As a first-stage quality control step, we perform human evaluation on a subset of each candidate dataset, where the quality of the samples and the diversity of the mixture are assessed to decide which datasets are eligible for downstream use. Only datasets that pass this initial human screen are processed by the subsequent automated stages.

The filtering process starts with per-dataset quality assessment, where an ensemble of judge LLMs scores the factual accuracy, relevance, and helpfulness of the answers. Quality thresholds vary significantly across sources, with curated proprietary datasets requiring minimal filtering while some open-source datasets underwent more aggressive removal. We then apply validation to eliminate malformed or invalid samples, followed by refusal filtering to remove samples containing refusal patterns or unhelpful responses.

Stylistic filtering uses rule-based methods to eliminate samples with overused phrases, clichés, and undesirable patterns. We perform exact deduplication through direct matching, then apply near-duplicate

---

detection using CMinHash locality-sensitive hashing to catch similar but non-identical content. Semantic deduplication proved most impactful, using sentence embeddings with a high similarity threshold to remove semantically similar content and ensure diverse training signals.

Finally, we perform decontamination through exact n-gram matching against evaluation benchmarks, with additional semantic similarity filtering to catch paraphrased content. The complete pipeline removes a substantial portion of the initial corpus, with semantic deduplication contributing the largest reduction and ensuring non-repetitive, high-quality training data.

**Curriculum learning.** We implement a data-driven curriculum learning strategy across the entire dataset to optimize the training progression from simpler to more complex examples. Given the dataset  $\mathcal{D} = \{x_i\}_{i=1}^N$  with verifiable rewards and a diverse ensemble of 12 language models  $\mathcal{M} = \{m_j\}_{j=1}^{12}$  spanning multiple scales and architectures, we record binary outcomes  $r_{ij} \in \{0, 1\}$  indicating whether model  $m_j$  answered item  $x_i$  correctly.

The model ensemble includes: LFM2-350M, LFM2-700M, Hunyuan-500M (Sun et al., 2024), LFM2-1.2B, LFM2-2.6B, Phi-4-Mini (Abouelenin et al., 2025), Qwen3-4B-Instruct-2507 (Yang et al., 2025a), Qwen3-Klear-8B (Zhang et al., 2025), ERNIE-4.5-21B-A3B (Baidu, 2025), Qwen3-30B-A3B-Instruct-2507, Qwen3-30B-A3B-Coder-Instruct, and Qwen3-235B-A22-Instruct-2507. This diverse set spans parameter counts from 350M to 235B and includes both dense and mixture-of-experts architectures.

For each question, we compute the empirical probability of success across the sampled models:

$$p_i = \frac{1}{J} \sum_{j=1}^J r_{ij}.$$

High values of  $p_i$  correspond to easier questions, while low values identify harder ones. We then train a model to predict and order the probabilities  $p_i$  based on prompt features, providing a ranking from easiest to most challenging examples. This allows SFT to proceed in a principled way: starting with items that most models can solve, and gradually introducing those that require deeper reasoning or knowledge.

#### 4.2.2 Training Protocol

The training protocol maintains the 32,768 token context length established during mid-training to ensure consistency across the training pipeline. Training is conducted over 3 epochs using a micro-batch size of 1 per GPU with gradient accumulation, resulting in an effective global batch size optimized for each model size.

The learning rate schedule employs cosine decay, starting from a maximum rate of  $3 \times 10^{-5}$  and decaying to a minimum of  $1 \times 10^{-7}$  over the training duration. We incorporate a linear warm-up phase spanning 500 steps, beginning from an initial rate of  $1 \times 10^{-5}$  with a start factor of 1.0. For optimization, we use the AdamW optimizer configured with  $\beta_1 = 0.9$ ,  $\beta_2 = 0.95$ , and weight decay  $\lambda = 0.1$ , along with gradient clipping at 1.0 and an epsilon value of  $1 \times 10^{-8}$ .

To balance model performance with computational efficiency, we apply 10% input dropout on embeddings during training only, while avoiding internal dropout layers to preserve inference latency. The training utilizes mixed precision arithmetic with bfloat16 for both activations and gradients, with float32 master weights maintained by the optimizer. The distributed training infrastructure leverages ZeRO-2 sharding with strided activation checkpointing across 8 H100-80GB GPUs per training run to optimize memory usage and training throughput while maintaining fine-grained control over the optimization process.

### 4.3 Stage 2: Preference Alignment

We leverage direct alignment methods to achieve a strong trade-off between model capabilities and iteration speed. In particular, we combine a family of length-normalized direct alignment objectives with an offline dataset consisting of both on-policy samples generated from the SFT checkpoint and off-policy reference responses (Lambert et al., 2024; Li and Khashabi, 2025; Meng et al., 2024; Rafailov et al., 2023).

### 4.3.1 Preference Dataset Creation

The dataset construction focuses on in-distribution samples and combines on-policy generation with skill-specific refinement through larger models. The base dataset consists of open-source as well as proprietary instruction and preference data that constitute approximately 1 million conversations. For each conversation, we sample  $N = 5$  responses from the selected SFT checkpoint to recover  $N + 1$  total responses for instruction datasets and  $N + 2$  total responses for preference datasets. We then score individual responses via an LLM jury, selecting the highest scored sample as the chosen response and the lowest scored sample as the rejected response. Here, we favor on-policy responses over instruction / preference samples in case of a tie to mitigate distribution shift. For instruction following data, we further refine the chosen responses via Contrastive Learning from AI Revisions (CLAIR) (D’Oosterlinck et al., 2025). We apply quantitative score-based filtering to remove low quality preference pairs as well as qualitative filters to mitigate potential undesirable behavior modes. Lastly, we filter out partial samples that exceed the context length. The resulting preference dataset consists of approximately 700,000 conversations.

### 4.3.2 Length-Normalized Direct Alignment

We implement a family of length-normalized alignment objectives with generalized loss function given by

$$\mathcal{L}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}} [\omega \cdot f(\Delta(x, y_w, y_l) - m) + \lambda \cdot g(\delta(x, y_w, y_l))], \quad (3)$$

evaluated on prompts  $x$ , chosen responses  $y_w$ , and rejected responses  $y_l$  sampled from dataset  $\mathcal{D}$ . Here, we define length-normalized relative  $\Delta(x, y_w, y_l)$  as well as absolute  $\delta(x, y_w, y_l)$  loss components based on the implicit reward  $r_\theta(x, y)$ , such that

$$\Delta(x, y_w, y_l) = \frac{r_\theta(x, y_w)}{|y_w|} - \frac{r_\theta(x, y_l)}{|y_l|}, \quad \delta(x, y_w, y_l) = \sigma\left(\frac{r_\theta(x, y_w)}{|y_w|}\right) - \sigma\left(\frac{r_\theta(x, y_l)}{|y_l|}\right), \quad (4)$$

$$r_\theta(x, y) = \beta \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}, \quad (5)$$

where  $|y|$  denotes token count and  $\sigma$  is the sigmoid function. The terms are transformed via functions  $f(\cdot)$  and  $g(\cdot)$ , and composed via weights  $\omega$  and  $\lambda$ . In the above,  $\beta$  denotes the standard KL coefficient and  $m$  the margin. From the general formulation, we recover length-normalized versions of DPO  $\{\omega = 1, f(x) = \log \sigma(x), m = 0, \lambda = 0, g(x) = 0\}$  (Rafailov et al., 2023) and APO-zero  $\{\omega = 0, f(x) = 0, m = 0, \lambda = 1, g(x) = x\}$  (D’Oosterlinck et al., 2025) as special cases. We can further construct joint objectives, e.g.  $\{\omega = 1, f(x) = \log \sigma(x), m = 0.1, \lambda = 0.2, g(x) = x\}$ , to leverage the strengths of both objectives. Notably, the inclusion of the margin  $m$  in the relative component aligns with the SimPO objective (Meng et al., 2024), while the absolute component can act as a regularizer to anchor generation probabilities. Reference hyperparameters for the direct alignment runs are provided in Table 5.

Parameter	Value
KL coefficient $\beta$	5.0
Learning rate schedule	Cosine
max $\eta_{\text{max}}$	$8 \times 10^{-7}$
min $\eta_{\text{min}}$	$8 \times 10^{-8}$
warm-up	0.01
Global batch size	2048
Context length	1024
Number of epochs	2

**Table 5:** Hyperparameters for direct alignment.

---

## 4.4 Stage 3: Model Merging

We evaluate LFM2 models on a custom harness with selected public benchmarks, covering knowledge, instruction following, mathematical reasoning, and multilingual domains.

We use multiple parameter-space merging techniques to combine fine-tuned models without requiring additional training. Since merging is computationally inexpensive, we apply different techniques in parallel and evaluate their results to select the best checkpoints. The techniques we test (model soup, task arithmetic, TIES-Merging, DARE, and DELLA) are described in detail in Appendix B.

## 4.5 Evaluation

Small models often fail evaluations because they struggle to output answers in the expected format. Unlike larger models, they have weaker instruction-following and generalization abilities, which creates a gap between their actual answers and what evaluation parsers can extract. To measure their true capabilities more accurately, we build a custom evaluation harness with robust parsing logic. This harness accounts for output quirks specific to models like LFM2, Qwen3, Llama 3.2, Gemma 3, and Granite 4.0, allowing us to extract answers correctly and measure each model’s performance fairly. More information about individual benchmarks and their implementations are provided in Appendix C.

Tables 6 and 7 present benchmark results for the LFM2 models compared to similarly-sized open-source alternatives. LFM2 models demonstrate competitive performance across multiple evaluation dimensions, particularly excelling in instruction following (IFEval, IFBench, Multi-IF) and mathematical reasoning (GSM8K, GSMPplus, MATH 500, MGSM) tasks relative to model size.

Several notable patterns emerge from the evaluation:

- **Knowledge Capabilities:** All LFM2 dense models demonstrate competitive performance in MMLU, MMLU-Pro, and GPQA relative to baselines matched in size. Furthermore, LFM2-8B-A1B achieves a 11.46 point improvement over LFM2-2.6B on MMLU-Pro (37.42%) while maintaining high inference throughput.
- **Instruction Following:** LFM2-1.2B achieves 74.89% on IFEval, surpassing the 30% larger Qwen3-1.7B (73.98%). LFM2-2.6B further improves to 79.56%, significantly outperforming Llama-3.2-3B (71.43%) and SmolLM3-3B (72.44%). Additionally, LFM2-8B-A1B scores 25.85% on IFBench. These strong results underline the efficacy of refining on-policy instruction following data via CLAIR.
- **Mathematical Reasoning:** LFM2-1.2B demonstrates strong multilingual reasoning capabilities, scoring 58.30% on GSM8K and 55.04% on MGSM. Despite a modest 7.4% data allocation for math reasoning, the curriculum-ordered training further scales performance of LFM2-2.6B to 82.41% on GSM8K and 74.32% on MGSM. With increased reasoning data allocation, LFM2-8B-A1B yields substantial gains over the 2.6B model, including +10.6 points on MATH 500 and +8 points on MATH Level 5.
- **Multilingual Transfer:** Evaluation on MGSM (multilingual GSM8K) and MMMLU (multilingual MMLU) indicates efficient cross-lingual transfer arising from the integrated multilingual training approach. LFM2-1.2B retains 94.4% of its English GSM8K performance on MGSM (58.30% vs. 55.04%) and 84.5% of its MMLU performance on MMMLU (55.23% vs. 46.73%).

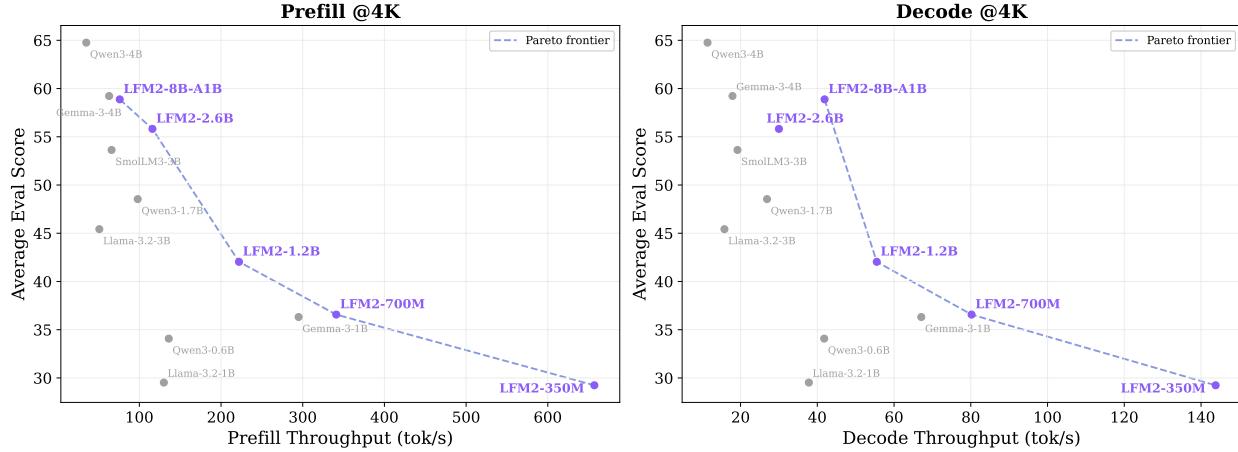
Benchmark	LFM2-350M	LFM2-700M	LFM2-1.2B	Qwen3-0.6B	Llama-3.2-1B	Gemma-3-1B	Qwen3-1.7B
# Total Params	0.35B	0.70B	1.2B	0.6B	1.2B	1B	1.7B
# Trained Tokens	11T	11T	11T	36T	9T	2T	36T
<i>Knowledge</i>							
MMLU	43.43	49.90	55.23	44.93	46.60	40.08	59.11
MMLU-Pro	15.85	18.65	19.71	27.02	19.37	13.72	43.04
GPQA	27.46	28.48	31.47	22.14	28.84	21.07	27.72
<i>Instruction Following</i>							
IFEval	65.12	72.23	74.89	64.24	52.39	62.90	73.98
IFBench	16.41	20.56	20.70	19.75	16.86	17.72	21.27
Multi-IF	32.85	40.92	45.28	45.13	29.43	44.49	56.28
<i>Mathematical Reasoning</i>							
GSM8K	30.10	46.40	58.30	36.47	35.71	59.59	51.40
GSMPlus	22.16	29.99	36.09	9.10	18.57	40.16	29.62
MATH 500	24.20	31.80	42.40	48.60	25.00	43.40	70.00
MATH Lvl 5	5.79	11.27	18.61	19.43	14.20	14.64	36.99
<i>Multilingual</i>							
MMMLU	37.99	43.28	46.73	30.84	38.15	34.43	46.51
MGSM	29.52	45.36	55.04	41.28	29.12	43.60	66.56

**Table 6:** Performance of tiny language models (350M–2B parameters) across knowledge, instruction following, and mathematical reasoning benchmarks. All results are obtained using our internal evaluation harness, and may differ from other reported scores.

Benchmark	LFM2-8B-A1B	LFM2-2.6B	Llama-3.2-3B	SmollLM3-3B	Gemma-3-4B	Qwen3-4B	Granite-4.0-H
# Total Params	8.3B	2.6B	3.2B	3.1B	4B	4B	7B
# Active Params	1.5B	2.6B	3.2B	3.1B	4B	4B	1B
# Trained Tokens	13T	11T	9T	11T	4T	36T	15T
<i>Knowledge</i>							
MMLU	64.84	64.42	60.35	59.84	58.35	72.25	66.79
MMLU-Pro	37.42	25.96	22.25	23.90	34.76	52.31	32.03
GPQA	29.29	26.57	30.60	26.31	29.51	34.85	26.46
<i>Instruction Following</i>							
IFEval	77.58	79.56	71.43	72.44	76.85	85.62	81.06
IFBench	25.85	22.19	20.78	17.93	23.53	30.28	18.37
Multi-IF	58.19	60.26	50.91	58.86	66.61	75.54	52.99
<i>Mathematical Reasoning</i>							
GSM8K	84.38	82.41	75.21	81.12	89.92	68.46	82.64
GSMPlus	64.76	60.75	38.68	58.91	68.38	56.16	59.14
MATH 500	74.20	63.60	41.20	73.60	73.20	85.60	58.20
MATH Lvl 5	62.38	54.38	24.06	51.93	52.18	73.62	36.11
<i>Multilingual</i>							
MMMLU	55.26	55.39	47.92	50.02	50.14	60.67	56.13
MGSM	72.40	74.32	61.68	68.72	87.28	81.76	73.68

**Table 7:** Performance of small language models (2B–8B parameters) across knowledge, instruction following, and mathematical reasoning benchmarks. All results are obtained using our internal evaluation harness, and may differ from other reported scores.

The strong performance across diverse benchmarks, paired with superior inference speed, allow LFM2 models to dominate the Pareto frontier of throughput versus average eval score, as shown in Figure 4.



**Fig. 4: Pareto frontier of Average Evaluation Score vs prefill (left) and decode (right) throughput.** LFM2 models dominate the Pareto frontier. Each point corresponds to a single model configuration profiled on the Samsung S25 device (Section 2.4). We chart prefill throughput (tok/s) when processing prompts of length 4k tokens, and decode throughput (tok/s) with a 4k-token prefix. Average Eval Score is computed as the unweighted mean of all evaluation scores reported in Table 6 and Table 7.

## 5 Vision-Language LFM2

We extend the LFM2 family to the vision-language setting by augmenting the language backbone with a vision encoder and a lightweight connector, yielding the **LFM2-VL** models. The overall design and training protocol are similar to other open-source vision-language models (VLMs) (An et al., 2025; Deitke et al., 2024; Marafioti et al., 2025; Wang et al., 2024a, 2025) and, at the same time, preserve the efficiency and deployment characteristics of LFM2.

LFM2-VL is instantiated in three sizes. **LFM2-VL-450M** combines a Base-86M SigLIP2 (Tschannen et al., 2025) encoder with an LFM2-350M backbone. **LFM2-VL-1.6B** and **LFM2-VL-3B** combine a So400M SigLIP2 encoder with LFM2-1.2B and LFM2-2.6B backbones, respectively. All variants are designed to support flexible image resolutions, efficient tokenization, and a controllable accuracy vs. latency trade-off for on-device deployment.

In this section, we introduce the LFM2-VL architecture (Section 5.1), describe its training recipe (Section 5.2), detail the data mixtures used at each training stage (Section 5.3), and present evaluation results across a wide range of benchmarks (Section 5.4).

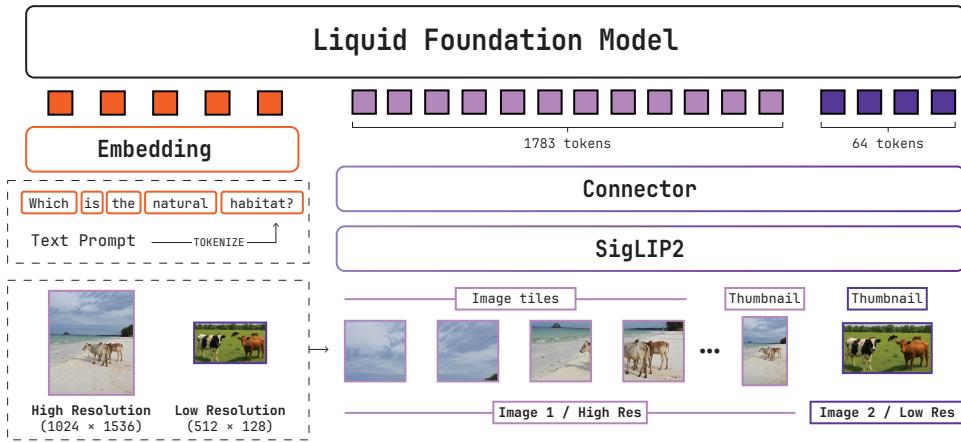
### 5.1 Architecture

All LFM2-VL variants retain the decoder-only LFM2 backbone and attach a SigLIP2 image encoder through a small connector that maps visual features into the language token space as visualized in Figure 5.

#### 5.1.1 Vision Encoder and Connector

For visual inputs, we adopt the NaFlex variant of SigLIP2 (Tschannen et al., 2025) similar to NaViT (Dehghani et al., 2023b), which supports variable input resolutions and native aspect ratios. The encoder outputs patch-level embeddings that are projected into language space via a lightweight connector. This connector first applies a PixelUnshuffle (Shi et al., 2016) operation that lowers the number of visual tokens, and then an MLP that maps image embeddings into the LFM2 hidden dimension. The PixelUnshuffle

"The second image shows the cows in their natural habitat."



**Fig. 5: LFM2-VL model architecture.** A SigLIP2 image encoder processes images either at native resolution for small inputs or with dynamic tiling for high-resolution inputs. A lightweight connector (PixelUnshuffle + MLP) reduces the number of vision tokens and projects visual embeddings into the LFM2 language token space, enabling unified multimodal processing.

step reduces the number of image tokens by a factor of four, yielding substantial savings in memory and compute with only a slight degradation in downstream multimodal performance in our experiments. For LFM2-VL-1.6B, we use the second-to-last hidden state representation as the image embedding. However, subsequent ablations showed no performance benefit, so for LFM2-VL-450M and LFM2-VL-3B, we use the last hidden state for simplicity.

### 5.1.2 Image Preprocessing

We preprocess images in two regimes depending on their effective resolution: a *single-frame* regime for images up to a resolution of  $512 \times 512$  pixels (e.g.,  $1024 \times 256$  or  $380 \times 680$ ), and a *tiled* regime for larger inputs.

For images in the single-frame regime, we first resize the input so that the SigLIP2 encoder sees a target range of patch counts. The resized height and width are chosen so that both spatial dimensions are divisible by the product of the encoder patch size and the PixelUnshuffle downsampling factor. This ensures alignment with the encoder’s patch grid and removes the need for additional padding prior to PixelUnshuffle.

For higher-resolution inputs, we switch to dynamic tiling. When the pixel count of an image exceeds a preset threshold, the image is reshaped to the closest supported aspect ratio and partitioned into tiles of size  $512 \times 512$  pixels. The number of tiles ranges from 2 to 10, depending on the original resolution. We optionally append a single thumbnail frame, processed through the same single-frame pipeline as above, to provide a coarse global representation of the scene. Tiles are interleaved with special positional tokens, and each multimodal input sequence is wrapped by image start and image end tokens to clearly delineate visual content from text tokens. The optional thumbnail has its own dedicated special token, allowing the model to distinguish global context from local tile information.

Depending on whether the tiling is active and on the size of the image, the number of image tokens ranges from 128 to 256 for untiled images to around 2,800 tokens for heavily tiled inputs. These ranges provide fine-grained control over the image token budget and, consequently, the compute cost during both training and inference.

## 5.2 Training Protocol

We train LFM2-VL in three stages, mirroring the language-only pipeline and introducing multimodal alignment at appropriate points. Unless otherwise indicated, the hyperparameters in this section refer to the

---

LFM2-VL-3B configuration; the smaller models follow a similar procedure with appropriately scaled batch sizes and learning rates.

### 5.2.1 Stage 1: Connector Pre-training

The first stage focuses on aligning the visual and textual embedding spaces while keeping the image encoder and language backbone frozen, establishing a mapping from image patch embeddings into the language embedding space before full multimodal training. In this phase, we use a higher learning rate for the connector parameters, a context length of 2,048 tokens, and pad each sample to maximum context length.

### 5.2.2 Stage 2: Multimodal Mid-training

After language mid-training, we introduce a multimodal mid-training stage that jointly optimizes the language backbone, pre-trained connector, and vision encoder. All components are trainable with a learning rate ratio of 5:5:1 for the text backbone, connector, and encoder, respectively. We found that using a smaller weight decay ( $1 \times 10^{-6}$ ) than in the language training benefits vision training performance. The context length is kept at 32,768 tokens, and samples are packed for efficient training. High-resolution tiling is disabled in this phase to maximize sample efficiency, and the learning rate schedule is restarted for approximately 50B additional tokens. In some multimodal mid-training runs, we incorporate additional higher-resolution document OCR data. For these configurations, we set the maximum number of tiles to 4, effectively extending the usable input resolution for more fine-grained OCR while keeping the overall image token budget conservative.

The language mid-training text mixture is reused and augmented with multimodal data (described in Sec 5.3). We apply a data annealing schedule that starts with 100% text-only data and, over the first 5% of steps, linearly decreases its fraction to 30%, which is then kept fixed for the remainder of training. Ablations showed that the annealing data schedule removes the need for connector pre-training, though we kept the connector pre-training stage as it did not hurt performance. This stage emphasizes robust modality fusion and the introduction of visual knowledge into the model, while maintaining the strong language performance of the mid-trained LFM2 checkpoint.

### 5.2.3 Stage 3: Joint Multimodal SFT

The final stage performs supervised fine-tuning on multimodal instruction-following data, analogous to the language-only SFT pipeline (Section 4.2). At this point, high-resolution tiling is enabled to expose the model to realistic deployment resolutions. We train several variants on about 50B tokens with diverse text–image ratios, ranging from 5% to 30% of total training samples, in order to explore trade-offs between stronger vision and language skills. The context length remains 32,768 tokens. This stage aligns the model to downstream use cases such as visual question answering, document understanding, and multi-image reasoning, while simultaneously reinforcing the language-only instruction-following and conversational capabilities.

### 5.2.4 Checkpoint Selection and Merging

For LFM2-VL-3B, we train several candidate runs that vary slightly in training procedure or data composition, for example, through different data sampling ratios, curriculum schedules, OCR-heavy variants, or SFT mixtures with a higher proportion of text-only data. Each candidate is evaluated on an internal multimodal benchmark suite, and the highest-performing checkpoints are combined with a simple linear merge in weight space (as discussed in Section 4.4). The merged model inherits a balanced combination of text-centric and vision-language capabilities and serves as the final checkpoint.

## 5.3 VLM Training Data Mixtures

**Connector pre-training mixture.** The connector pre-training stage uses a captioning corpus designed to provide diverse yet well aligned image-text pairs, with the primary objective of creating an initial alignment signal between the SigLIP2 visual features and the LFM2 language space.

---

**VLM mid-training mixture.** During multimodal mid-training, we employ a broad mixture of captioning datasets covering diverse scenes and concepts, interleaved image–text documents, OCR-focused data, document visual question answering (VQA), and general VQA datasets. To improve mid-training data quality, we re-capture existing open-source image datasets and generate additional VQA samples specifically targeted at visual understanding, including fine-grained reasoning about objects, layouts, and attributes. Some large-scale image datasets used at this stage are filtered for safe-for-work content using automated filters and LLM-based screening; this process removes roughly 0.1% of samples from the original pools.

We incorporate additional document OCR data with visual augmentations such as blur, noise, and geometric distortions to make the OCR behavior more robust to real-world document scans. We observe that some lower-quality SFT subsets degrade performance when used directly in the SFT stage. To still benefit from their coverage and to familiarize the model with multimodal conversational patterns early in the training stage, we move these datasets to the mid-training stage instead of using them in SFT.

**VLM SFT mixture.** The multimodal SFT mixture spans a wide range of tasks and interaction styles, including both single- and multi-image inputs, single- and multi-turn conversations, and tasks ranging from free-form captioning to multiple-choice questions, short answer tasks, and complex visual reasoning tasks. Around 5% of the data involves multi-image inputs, and roughly 70% consists of multi-turn samples, which include both sequences of independent questions and multi-turn dialogues. Most instruction tuning vision datasets are drawn from open-source repositories, with additional synthetic subsets used to strengthen specific downstream behaviors. Notably, we do not employ explicit grounding supervision such as bounding boxes or point coordinates.

**VLM multilingual data.** To strengthen the multilingual capabilities of LFM2-VL-3B, across training stages we integrate vision-language data translated into several target languages, including Arabic, Chinese, French, German, Italian, Japanese, Korean, Portuguese, and Spanish. The multilingual subset of the captioning and VQA mid-training data enhances the image alignment phase, ensuring that the model sees visually grounded supervision in other languages rather than primarily English. In addition, a portion of the OCR datasets is inherently multilingual, covering a broad range of document types.

Beyond mid-training data, we also translate a variety of English SFT data to target languages, which helps extend multilingual support across the entire task spectrum during the instruction tuning phase. While the translated datasets significantly improve visual capabilities across languages, we acknowledge that further gains will require more culturally contextual and locally sourced data to better capture region-specific visual knowledge.

## 5.4 Evaluation

We evaluate LFM2-VL across a diverse set of multimodal benchmarks, including general visual understanding, visual reasoning, OCR-heavy tasks, math reasoning, and multilingual visual understanding. As shown in Tables 8 and 9, the LFM2-VL family achieves consistently strong performance relative to similarly sized open-source VLMs. Multilingual scores are based on the average of benchmarks translated by GPT-4.1-mini from English to Arabic, Chinese, French, German, Italian, Japanese, Korean, Portuguese, and Spanish.

### Key performance insights.

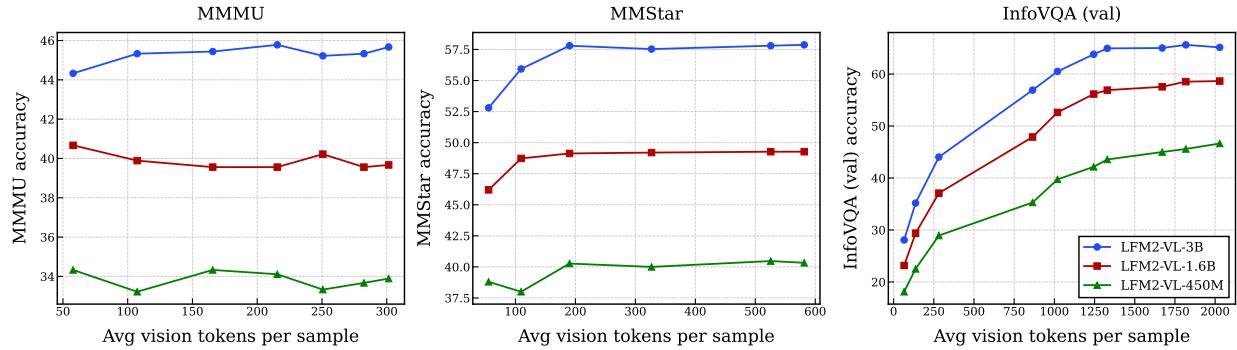
- **General Vision Understanding:** Across standard multimodal benchmarks such as MMBench, MMStar, SEEDBench, and RealWorldQA, LFM2-VL models exhibit strong parameter-efficient performance. LFM2-VL-450M outperforms SmolVLM2-500M on MMStar (+3.14) and MMBench (+4.47), while LFM2-VL-1.6B surpasses InternVL3.5-1B on RealWorldQA (+8.63). At the 3B scale, LFM2-VL-3B scores 76.55 on SEEDBench and 71.37 on RealWorldQA, outperforming all baselines, including the larger Qwen2.5-VL-3B model. The models also maintain low object hallucination rate, with LFM2-VL-3B achieving 89.01 on POPE.

Benchmark	LFM2-VL-450M	LFM2-VL-1.6B	SmoIVLM2-500M	InternVL3.5-1B
# Total Params	0.45B	1.6B	0.5B	1.1B
<i>General (vision)</i>				
MMStar	40.87	49.87	37.73	50.27
MMBench (dev)	55.76	69.16	51.29	70.79
RealWorldQA	52.03	65.75	51.37	57.12
MME	1,230	1,757	1,456	1,894
SEEDBench	63.62	72.00	60.34	72.67
POPE	83.68	87.17	85.83	86.30
<i>Capability-focused (vision)</i>				
MMMU	34.44	39.67	33.89	41.89
MathVista	45.30	51.70	34.50	53.10
BLINK	42.61	44.50	40.45	44.19
InfoVQA (val)	44.56	58.35	24.64	60.99
OCRBench	657	729	562	790
MM-IFEval	33.09	46.35	11.56	36.17

**Table 8:** Performance of VLMs with fewer than 2B parameters on general image understanding and capability-focused benchmarks (multimodal reasoning, mathematical reasoning, multi-image, high resolution input, OCR, and instruction following). All results are obtained using VLMEvalKit (Duan et al., 2024) and may differ from reported scores of other models.

Benchmark	LFM2-VL-3B	SmoIVLM2-2.2B	InternVL3.5-2B	Qwen3-VL-2B	Qwen2.5-VL-3B
# Total Params	3.0B	2.2B	2.3B	2.1B	3.75B
<i>General (vision)</i>					
MMStar	57.73	46.00	57.67	49.00	56.13
MMBench (dev)	79.81	69.24	78.18	77.58	80.41
RealWorldQA	71.37	57.50	60.78	65.75	65.23
MME	2,051	1,793	2,129	2,045	2,163
SEEDBench	76.55	71.30	75.41	74.08	73.88
POPE	89.01	85.10	87.17	89.05	86.17
<i>Capability-focused (vision)</i>					
MMMU	45.33	41.60	51.78	45.78	51.67
MathVista	62.20	51.50	61.60	63.50	62.50
BLINK	51.03	42.30	50.97	54.87	48.97
InfoVQA (val)	67.37	37.75	69.29	71.68	76.12
OCRBench	822	725	834	864	824
MM-IFEval	51.83	19.42	47.31	51.35	38.62
<i>Multilingual (vision)</i>					
MMBench (dev)	75.84	44.07	70.03	69.52	74.33
MMMB	81.52	57.82	75.86	74.11	77.60
<i>Knowledge (language)</i>					
MMLU	62.70	26.89	60.89	59.87	64.92
MMMLU	53.37	32.16	49.32	42.20	54.23

**Table 9:** Performance of 2–4B VLMs on general image understanding, capability-focused and multilingual vision evaluations, and language benchmarks. All results are obtained using VLMEvalKit (Duan et al., 2024) or our internal language eval suite, and may differ from reported scores of other models.

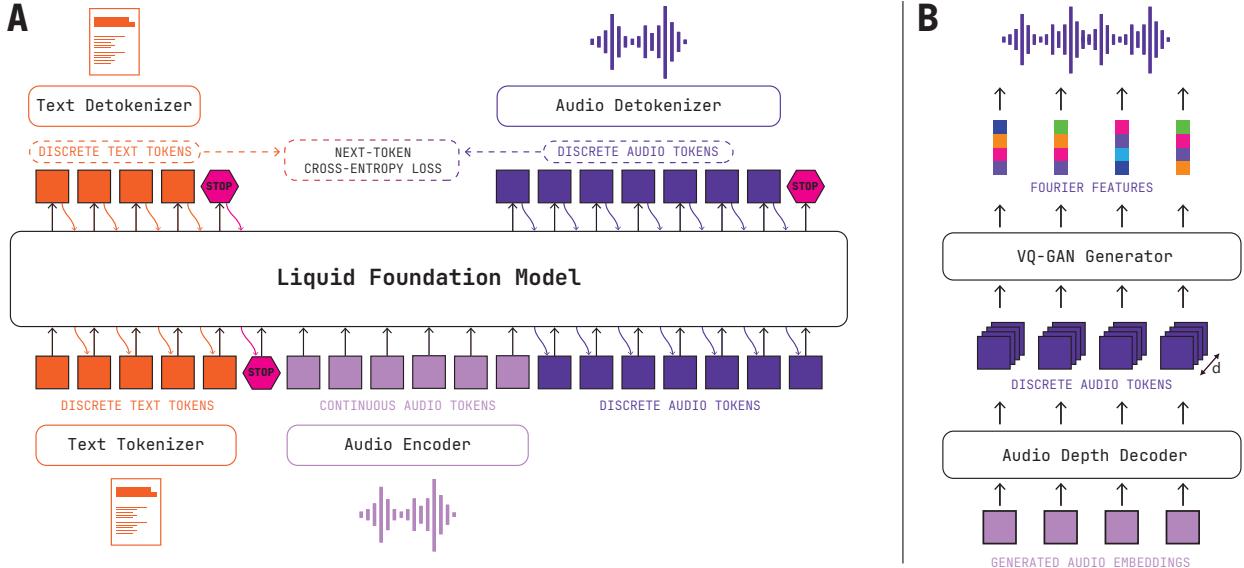


**Fig. 6: Accuracy of LFM2-VL on selected benchmarks under varying vision token budgets.** We vary the maximum number of vision tokens per image at inference by adjusting the preprocessing pipeline (downsizing single-frame inputs or limiting the number of tiles for high-resolution images), using the token budget as a proxy for on-device latency. Across multimodal reasoning, general vision understanding, and high-resolution perception benchmarks, LFM2-VL retains strong performance under compression, with performance degrading gracefully, especially on high-resolution inputs.

- **Image Grounded Instruction Following:** The LFM2-VL family performs well in instruction-following tasks. LFM2-VL-450M outperforms SmoVLML-500M by a large margin, demonstrating strong capability even at small scales. LFM2-VL-1.6B achieves 46.35 on MM-IFEval, outperforming InternVL3.5-1B (36.17), while LFM2-VL-3B reaches 51.83, matching or exceeding all open-source models below 4B.
- **Text-Rich and OCR Tasks:** LFM2-VL models demonstrate competitive document understanding and OCR capabilities. The 450M model achieves 657 on OCRCBench, substantially higher than SmoVLML-500M (562), while LFM2-VL-3B scores 822, reaching performance of other similarly sized models.
- **Multilingual Visual Understanding:** LFM2-VL-3B surpasses Qwen2.5-VL-3B, with an average score of 75.84 (+1.51) on translated MMBench and 81.52 (+3.92) on translated MMMB, outperforming all other open-source models under 4B parameters. A detailed breakdown of scores by language are in Table 16 in Appendix D.
- **Language-Only Tasks:** Despite being optimized for multimodal understanding, LFM2-VL-3B maintains strong language capabilities. The model achieves 62.70 on MMLU and 53.37 on MMMLU, outperforming SmoVLML-2.2B by large margins (+35.81 on MMLU, +21.21 on MMMLU) and closely matching the performance of larger baselines such as Qwen2.5-VL-3B. These results demonstrate that the training protocol produces strong VLMs without sacrificing language-only performance.

These results highlight strong multimodal performance of LFM2-VL models across scales, while also revealing headroom for further gains through reinforcement learning, particularly for more complex multimodal reasoning, such as in the MMMU eval.

**Accuracy-vs-latency.** To illustrate the flexibility of LFM2-VL, Figure 6 shows performance as a function of the number of vision tokens at inference time, which serves as a proxy for latency on-device. Reducing the token budget introduces minimal degradation on general vision benchmarks, indicating that the model retains strong vision understanding even under aggressive compression. In contrast, tasks requiring high-resolution perception exhibit more pronounced drops as fewer tokens limit the model’s ability to capture fine-grained detail. These results demonstrate that LFM2-VL offers adaptable performance vs. latency scaling, enabling users to tune computational cost to match visual difficulty and resource constraints.



**Fig. 7: LFM2-Audio model architecture.** (A) The LFM2 backbone operates on a single sequence of embeddings that can interleave text and audio tokens, producing next-token predictions. (B) The audio generation pipeline. Continuous LFM2 embeddings are turned into discrete code sequences, which are subsequently turned into a waveform by a Fourier based GAN generator.

## 6 LFM2-Audio

**LFM2-Audio** extends the LFM2 language model backbone with dedicated components for audio input and output. In contrast to several contemporary audio-language models, LFM2-Audio explicitly separates *continuous* audio input features from *discrete* audio output codes. Audio input is handled via log-mel features and an encoder stack, while audio generation is implemented through a discrete Residual Vector Quantization (RVQ) code path and a lightweight audio detokenizer. This separation preserves a rich representation for speech recognition and understanding while avoiding artifacts introduced by audio tokenization and keeping generation latency low enough for real-time speech-to-speech interaction on edge hardware.

In this section we describe the architecture of **LFM2-Audio-1.5B** (Section 6.1), the inference modes it exposes (Section 6.2), the training procedure and losses (Section 6.3), the data mixtures (Section 6.4), and evaluation results on audio chat and ASR (Automatic Speech Recognition) benchmarks (Section 6.5).

### 6.1 Architecture

LFM2-Audio-1.5B is built around the decoder-only LFM2-1.2B language backbone (Section 2), augmented with an audio encoder on the input side and a discrete audio code path plus detokenizer on the output side. The main parts of the architecture are sketched in Figure 7.

#### 6.1.1 Encoder

The audio encoder produces a sequence of continuous embeddings that can be consumed by the LFM2 backbone in the same way as text token embeddings. Audio inputs are converted to LLM embeddings by an audio encoder followed by an MLP connector. Raw 16 kHz waveforms are first transformed into 128-dimensional log-mel features using a 0.025s window and 0.01s stride. A small stack of strided convolutions performs  $8\times$  temporal downsampling on the log-mel features, after which 17 FastConformer layers (Rekesh et al., 2023) process the resulting 512-dimensional continuous features. The MLP connector projects these encoder features to the LFM2-1.2B hidden dimension of 2048. Thus, each resulting embedding corresponds to 0.08 seconds of audio, so the operates at a temporal resolution of 12.5 embeddings per second of input audio.

---

### 6.1.2 Audio Decoder

For audio output, LFM2-Audio generates *discrete* audio codes rather than continuous waveforms. Specifically, the model predicts 8-codebook Mimi RVQ codes (Défossez et al., 2024), which are later converted to a waveform by a detokenizer. The 8 codebooks consist of one semantic codebook and seven acoustic codebooks, following the Mimi design. Each 8-code frame corresponds to 0.08 seconds of audio, matching the temporal resolution of the encoder.

The combined audio vocabulary contains 2049 tokens per codebook (2048 Mimi tokens plus a special token that marks end-of-audio for that codebook). Once an 8-code frame has been generated, the eight code tokens are embedded and summed to form a single 2048 dimensional embedding, which is fed back into the LFM2 backbone for autoregressive generation. In parallel, the corresponding 8-code frame is streamed to the detokenizer for real-time playback.

To keep inference latency low, we do not ask the LFM2 backbone to predict all eight codes in sequence. Instead, we use a much smaller RQ-Transformer (Lee et al., 2022), similar to the one used in Moshi (Défossez et al., 2024), as a decoder for code generation. At each time step, the LFM2 backbone produces a single 2048-dimensional output embedding. This embedding is used to condition the RQ-Transformer, which runs for 8 steps to autoregressively generate the 8 codes in the frame (one per codebook). Concretely, when the model is in *audio generation state* (Section 6.2), each backbone step is followed by 8 RQ-Transformer steps. Because the RQ-Transformer has a much smaller parameter count than the LFM2 backbone (115M vs. 1.2B parameters, respectively), this schedule yields low time-to-first-audio and real-time generation on commodity CPUs compared to other RVQ generation strategies, such as delay patterns (Copet et al., 2023).

### 6.1.3 Audio Detokenizer

Since LFM2-Audio generates Mimi codes, the conversion from codes to a waveform can in principle be done with the Mimi decoder itself or any other compatible model. In practice, we found the Mimi decoder too slow for the target hardware, so we instead train a compact, LFM2-style detokenizer optimized for edge deployment.

The detokenizer closely follows the Vocos architecture (Siuzdak, 2023). Its job is to map a sequence of discrete 8-code Mimi frames to a sequence of complex short-time Fourier transform (STFT) coefficients, which are then converted to 24 kHz audio by an inverse STFT. Concretely, the sequence of 8-code frames is first embedded into a sequence of vectors, one embedding per frame. These embeddings are upsampled along the time axis by a factor of 6 using nearest-neighbor upsampling so that the resulting sequence length matches the target number of STFT frames. The upsampled sequence is processed by a small 35M-parameter LFM2 variant with 8 layers (5 gated short convolution blocks and 3 sliding-window attention blocks) and hidden dimension 512. This backbone produces one hidden vector per STFT frame. A final linear projection maps each hidden vector to the complex STFT domain, jointly predicting the log-magnitude and phase for all frequency bins at that frame. Thus, the *network* outputs a sequence of complex STFT coefficients, and the *final* audio waveform is obtained by applying an inverse STFT to these coefficients. This design yields a detokenizer that is fast enough for real-time synthesis on edge hardware while maintaining good perceptual quality.

## 6.2 Inference

Unlike other speech-to-speech models, LFM2-Audio supports two complementary and distinct inference modes, *interleaved* and *sequential*, that share the same backbone but differ in how they schedule text and audio generation. The choice of mode depends on the downstream task. Interleaved generation is suited to settings that require both non-trivial reasoning and low-latency speech output (e.g., conversational assistants), where the model can emit text and audio in real time. Sequential generation instead produces one modality at a time, with the ability to switch between modalities as needed. This is useful for tasks such as Automatic Speech Recognition (ASR; audio-in, text-out), Text-to-Speech (TTS; text-in, audio-out), or mixed text/audio responses with variable proportions.

We further note that both generation modes are *stateful*, which allows the separation of text and audio vocabularies. The state determines which modality that is being generated at each timestep. In the *text*

---

*generation state*, the model behaves exactly like the underlying LFM2 base model, using the output embedding to compute next-token probabilities over the text vocabulary. In the *audio generation state*, the model sends the output embedding to the RQ-Transformer and computes next-code probabilities over the audio vocabulary. The different generation modes merely control the state-changing logic for outputs consisting of multiple modalities.

### 6.2.1 Interleaved Generation

In interleaved generation mode, LFM2-Audio generates the assistant response in both text and audio, with a predetermined interleaved pattern. This acts as a kind of chain-of-thought, where the model generates the semantic content in text, leveraging knowledge learned during large-scale text-only pre-training, and subsequently translates the generated text into audio. In our implementation, we generate 6 text tokens followed by 12 audio tokens and repeat until all text tokens are exhausted. When this happens, the model outputs a special `<|end_of_text|>` token, after which it outputs all remaining audio tokens until a special `<|end_of_audio|>` token is generated.

The 6 : 12 ratio of text and audio tokens is selected to keep text generation ahead of audio generation and to minimize time-to-first-audio token. Keeping the text-to-audio ratio fixed and separating text and audio vocabularies enables independence from extra control tokens, as is required by other models generating interleaved tokens (Li et al., 2025; Zeng et al., 2024).

### 6.2.2 Sequential Generation

In sequential generation mode, LFM2-Audio can change its own generation state by generating special control tokens. By default, the model always starts in text generation mode, where only the text vocabulary is used. When a special `<|start_of_audio|>` is encountered, the model switches to the audio generation state and starts using the RQ-Transformer and audio vocabulary to generate outputs. This continues until a special `<|end_of_audio|>` token is generated, after which it switches back to the text state. This design enables generation of responses consisting of both text and audio of arbitrary lengths mixed in arbitrary ways. This mode is suitable for tasks with text-only outputs, such as ASR, since by default the model generates only text. Sequential generation is also used in TTS, by first generating the `<|start_of_audio|>` token, and stopping generation when `<|end_of_audio|>` is generated.

## 6.3 Training details

As a decoder-only model, LFM2-Audio is trained with autoregressive cross-entropy over discrete targets, with separate losses for text and audio tokens.

**Token-level loss.** Let  $N$  be the total number of (discrete) tokens in a batch,  $V$  the vocabulary size under the current head, and  $y_n$  the reference token at position  $n$  with corresponding logits  $l_{n,i}$ . The standard cross-entropy loss is

$$L = -\frac{1}{N} \sum_{n=1}^N \log \frac{\exp(l_{n,y_n})}{\sum_{i=1}^V \exp(l_{n,i})}.$$

We compute this loss separately for the text vocabulary, yielding  $L_T$ , and each of the 8 Mimi codebooks, yielding  $L_1, \dots, L_8$ .

**Codebook weighting.** We combine the 8 audio codebook losses into a single  $L_A$  via a weighted average. Concretely,  $L_1$  is assigned weight 100, and other weights receive exponentially decaying weights such that  $L_8$  gets a weight 1. This emphasizes semantic and low-index acoustic codes in the loss, which we found empirically to improve perceived audio quality.

The final loss value is the mean of  $L_T$  and  $L_A$ , weighted by the number of tokens of each modality present in a batch. We do not compute losses on continuous audio input embeddings.

Dataset type	# Text tokens	Input audio (hours)	Output audio (hours)
Transcription	1.5B	69,163	–
Text To Speech	2.6B	–	90,826
Language classification	175M	4,728	851
Audio Chat Instruction Tuning	4.7B	72,130	234,113
Text Chat Instruction Tuning	3.7B	–	–
<b>Total</b>	<b>~13B</b>	<b>146,021</b>	<b>325,790</b>

**Table 10:** Dataset composition per epoch across text and audio modalities.

**Training phases.** Training proceeds in three phases: alignment, mid-training, and post-training. In the alignment phase, we use ASR data to train the MLP connector, while keeping the LFM2 backbone and encoder frozen. When the loss plateaus, we move on to mid-training. Here, we unfreeze the entire model and train on a large-scale mixture of text and audio datasets. In total, the mid-training phase sees around 100 billion tokens, of which approximately 62% are text tokens, 25% are audio output frames (8-codebook Mimi tokens), and 13% are audio input embeddings (log-mel features).

**Detokenizer training.** The detokenizer is trained separately as a GAN, using both a multi-period discriminator (Kong et al., 2020) and a multi-resolution discriminator (Jang et al., 2021). The training objective is a combination of log-mel loss (L1), discriminator loss (hinge loss on each frame), and feature matching loss. We train the detokenizer on batches of  $16 \times 2.4$  second audio clips for a total of 1,000,000 steps.

## 6.4 Audio Training Data Mixtures

In the mid- and post-training phases, we mix datasets spanning transcription, language classification, text-to-speech, and audio chat instruction tuning across multiple domains. Table 10 presents an overview of the data mixture. The datasets range from established open-source collections such as CommonVoice and LibriSpeech to in-house synthetic datasets generated for specialized audio chat capabilities. For each dataset, we apply augmentation to improve robustness and filtering to ensure quality. Filtering combines rule-based validation (e.g., duration constraints, transcription alignment, and pattern-based filtering) with LLM-judging techniques, where the text or audio quality is evaluated for coherence, correctness, and stylistic naturalness.

The total audio volume used per epoch (approximately 472,000 hours across input and output modalities) is comparable in scale to large-scale speech models such as Whisper, which was trained on around 680,000 hours of audio (Radford et al., 2023). In aggregate, the audio data corresponds to roughly 21.5 billion audio embeddings, providing the model with a dense and diverse representation of spoken language and sound across multiple acoustic and conversational domains. This scale ensures that the model is exposed to sufficient variability in speech, style, and content to generalize effectively across downstream audio-language tasks.

A large focus on synthetic conversation data is employed to generate a high volume of multi-turn audio conversations intended to generalize the model for potential downstream use cases such as instruction following, ideation, and general reasoning. These conversations are generated using proprietary data factories that chain together open-source language and audio models to produce natural-sounding exchanges between a user and an audio assistant. Broadly, these pipelines come in two types: a) pipelines specialized for turning existing text SFT datasets into audio conversations, and b) pipelines that generate audio conversations following a pre-determined theme using randomly sampled seed specifications.

An important aspect in both pipelines is the introduction of paralinguistic and discourse features characteristic of natural spoken dialogue. This involves the synthetic generation of speech cues and conversational behaviors such as intonation patterns, pauses, filler words (e.g., “uh,” “you know”), turn-taking signals, and backchannel responses (“mm-hmm,” “right”), making interactions sound human and spontaneous. Additionally, we sample user voices via voice-cloning techniques to teach the model a wide distribution of user voice characteristics (e.g., accent, pitch, timbre, and speaking style). These features are programmati-

Benchmark	LFM2- <b>Audio-1.5B</b>	Qwen2.5-Omni-3B	Moshi	Ultravox-v0.5-Llama-3.2-1b	Mini-Omni2
# Total Params	1.5B	5B	7B	0.7B	0.6B
AlpacaEval	3.78	3.72	2.01	4.02	2.32
CommonEval	3.48	3.51	1.6	3.53	2.18
WildVoice	3.12	3.42	1.3	3.48	1.79
SD-QA	34.81	44.94	15.64	46.38	9.31
MMSU	33.99	55.29	24.04	29.99	24.27
OBQA	45.49	76.26	25.93	33.41	26.59
BBH	51.2	61.3	47.4	51.4	46.4
IFEval	30.13	32.9	10.12	43.51	11.56
AdvBench	98.85	88.46	44.23	97.88	57.5

**Table 11:** Performance on VoiceBench. VoiceBench is a collection of 9 speech-in, text-out chatbot tasks. AlpacaEval, CommonEval, and WildVoice are scored on a scale from 0 to 5, whereas the rest are scored on a scale from 0 to 100. Higher scores are better.

Benchmark	LFM2- <b>Audio-1.5B</b>	Qwen2.5-Omni-3B	Whisper-large-v3-turbo
# Total Params	1.5B	5B	1.5B
AMI	15.36	15.05	16.13
Earnings22	19.75	14.81	11.63
Gigaspeech	10.63	11.76	10.14
LS Clean	2.03	2.14	2.10
LS Other	4.39	4.52	4.24
SPGISpeech	4.17	3.24	2.97
Tedlium	3.56	5.08	3.57
Voxpopuli	9.93	6.59	11.87

**Table 12:** Performance on ASR tasks, as measured by Word Error Rate (WER). Lower is better.

cally varied to capture the rhythm, nuance, and acoustic diversity of real conversation, thereby enhancing realism and robustness in downstream audio chat models.

The resulting mixture provides broad coverage of tasks and styles, ensuring the trained model demonstrates robustness across varied speech patterns, dialogue structures, and user voice characteristics.

## 6.5 Evaluation

We evaluate LFM2-Audio on two sets of tasks: general-purpose speech-to-speech conversational abilities and ASR.

Audio chat capabilities are evaluated using VoiceBench (Chen et al., 2024), which evaluates models on freeform answers (using a GPT4o judge), multiple choice questions, instruction following, and refusals on adversarial prompts. In Table 11 we compare LFM2-Audio-1.5B to other similarly sized models. We note that LFM2-Audio-1.5B remains competitive even against the more than three times larger Qwen2.5-Omni-3B<sup>2</sup>. The closest model in total parameter count, Ultravox-v0.5-Llama-3.2-1b, performs similarly but does not support audio generation, a must-have for real-time, low-latency speech-to-speech chatbots.

For ASR, we adopt the framework from the Hugging Face Open ASR Leaderboard (Srivastav et al., 2023), testing ASR performance over 8 different datasets. A comparison between LFM2-Audio and other models can be found in Table 12. Once again, we see that LFM2-Audio-1.5B performs extremely competitively against Qwen2.5-Omni-3B, and approaches the world error rate (WER) numbers of ASR-only models such as Whisper. Together with the model’s strong performance in conversational speech-to-speech interaction, these results make LFM2-Audio well-suited for real-time, low-latency audio-text applications.

<sup>2</sup>Qwen2.5-Omni-3B contains 5B parameters, including a 3B parameter backbone.

---

## 7 LFM2-ColBERT

We extend the LFM2 backbone to information retrieval through **LFM2-ColBERT-350M**, a late interaction retriever optimized for multilingual and cross-lingual semantic search. Late interaction models occupy a unique position in the retrieval landscape: they preserve much of the expressivity of cross-encoders (re-rankers) while retaining the efficiency of bi-encoders, enabling both large-scale retrieval and effective ranking in a single model (Khattab and Zaharia, 2020). LFM2-ColBERT-350M demonstrates best-in-class multilingual performance and achieves inference speeds comparable to models with 2.37 times fewer parameters, thanks to the computational efficiency of the LFM2 hybrid architecture.

### 7.1 Architecture and Design

**Late interaction paradigm.** Unlike bi-encoders that compress entire documents and queries into single dense vectors, or cross-encoders that perform expensive token-level interactions, late interaction models compute fine-grained token representations independently for queries and documents, deferring the interaction computation until retrieval time. This design enables pre-computation and efficient indexing of document representations while preserving rich semantic matching capabilities.

**Model architecture.** LFM2-ColBERT-350M builds on the LFM2-350M backbone (16 layers, 1024 hidden dimensions) with an additional 9 task-specific layers, resulting in a 17-layer architecture comprising 10 convolutional blocks, 6 attention blocks, and 1 linear layer. The model processes inputs through the LFM2 transformer to produce contextualized token embeddings, which are then projected to a lower-dimensional space (128 dimensions) optimized for similarity computation:

$$\text{ColBERT}(x) = \text{Linear}(\text{LFM2-Transformer}(x)) \quad (6)$$

where  $x$  represents either a query or document, and the linear layer projects from 1024 to 128 dimensions without bias or additional activation. The total parameter count is 353M, distributed across the hybrid backbone and projection head.

**Similarity computation.** Following the ColBERT framework (Khattab and Zaharia, 2020), we compute similarity between a query  $q$  and document  $d$  using the MaxSim operator:

$$S(q, d) = \sum_i \max_j \text{sim}(q_i, d_j) \quad (7)$$

where  $q_i$  and  $d_j$  are the  $i$ -th and  $j$ -th token embeddings of the query and document respectively, and  $\text{sim}(\cdot, \cdot)$  denotes cosine similarity. This operation allows each query token to attend to its most relevant document token, enabling fine-grained semantic matching while maintaining computational efficiency through pre-computed document representations.

**Configuration parameters.** The model supports a maximum context length of 32,768 tokens inherited from the LFM2 backbone. For retrieval tasks, we constrain document inputs to 512 tokens and query inputs to 32 tokens, balancing retrieval quality with computational efficiency. The vocabulary size is 65,536 tokens, using the same byte-level BPE tokenizer as the base LFM2 models.

### 7.2 Training Methodology

We continued to pre-train LFM2-350M to 25T tokens, and LFM2-ColBERT-350M is an extension of this 350M checkpoint. Here we discuss the training method used to convert this LFM2-350M checkpoint to a ColBERT model.

---

**Knowledge distillation framework.** We train LFM2-ColBERT-350M using knowledge distillation through the PyLate framework (Chaffin and Sourty, 2024), leveraging pre-computed relevance scores from a powerful cross-encoder teacher model. This approach enables the student model to learn fine-grained semantic matching patterns from the teacher’s outputs while maintaining the computational efficiency of late interaction retrieval. Knowledge distillation has been shown to produce superior ColBERT models compared to pure contrastive learning approaches (Khattab and Zaharia, 2020), as it provides richer training signals through continuous relevance scores rather than binary positive/negative labels.

**Training objective.** We employ the distillation loss implemented in PyLate, which minimizes the mean squared error between student and teacher relevance scores. For a query  $q$  with associated documents  $\{d_1, \dots, d_m\}$  and teacher scores  $\{s_1^T, \dots, s_m^T\}$ , the loss is:

$$\mathcal{L}_{\text{distill}} = \frac{1}{m} \sum_{i=1}^m \left( s_i^T - S(q, d_i) \right)^2 \quad (8)$$

where  $S(q, d_i)$  is the student’s MaxSim score (Equation 2). The PyLate framework applies min-max normalization to teacher scores to ensure numerical stability and consistent gradient magnitudes across different teacher model scales. We train with the normalized scores following best practices from JaColBERTv2.5 (Clavié, 2025).

**Optimization details.** We initialize the transformer backbone from the pre-trained LFM2-350M checkpoint and add 9 randomly initialized layers for the task-specific retrieval head. The model is trained end-to-end using the AdamW optimizer with a learning rate of  $3 \times 10^{-5}$ ,  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$ , and weight decay of 0.01. Training runs for approximately 200,000 steps with a global batch size of 128, distributed across 8 NVIDIA H100 GPUs using BF16 mixed precision. We employ a cosine learning rate schedule with 10% warmup steps, decaying from the maximum learning rate to  $1 \times 10^{-7}$ .

During the training process, each instance contains a query and 8 documents with their corresponding teacher scores. Document inputs are truncated to 512 tokens and query inputs to 32 tokens, matching the target deployment constraints.

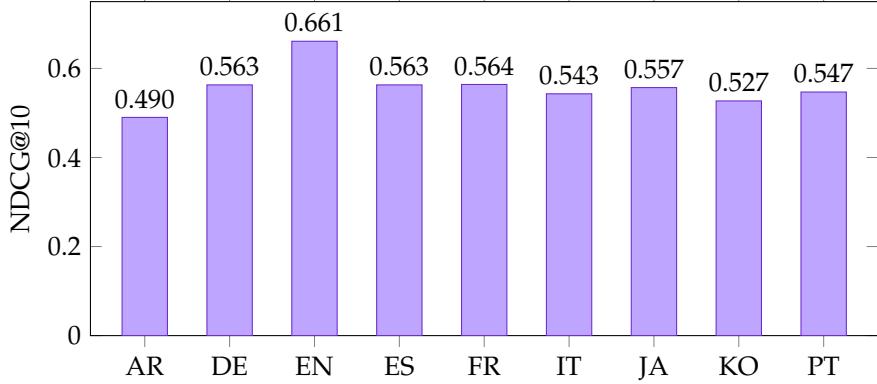
### 7.3 Evaluation

**Benchmarks.** We evaluate LFM2-ColBERT-350M on NanoBEIR Multilingual (Sourty, 2025), a condensed version of the BEIR benchmark (Thakur et al., 2021), extended to include Japanese and Korean languages. We release this multilingual extension as LiquidAI/nanobeir-multilingual-extended for reproducibility. The benchmark spans 13 diverse retrieval tasks, including fact verification (FEVER, Climate FEVER), question answering (Natural Questions, HotpotQA, FiQA), and entity retrieval (DBPedia, Quora).

**Monolingual retrieval.** Figure 8 presents NDCG@10 scores across languages when queries and documents are in the same language. LFM2-ColBERT-350M achieves a mean NDCG@10 of 0.661 across all tasks and languages, with particularly strong performance in English (0.661), German (0.563), Spanish (0.563), and French (0.564). The model demonstrates robust multilingual capabilities, maintaining competitive performance even in lower-resource languages like Arabic (0.490), Japanese (0.557), and Korean (0.527).

Compared to GTE-ModernColBERT-v1 (Chaffin, 2025), a similarly-sized baseline with 149M parameters, LFM2-ColBERT-350M shows substantial improvements in multilingual coverage. While the baseline GTE-ModernColBERT-v1 achieves 0.680 NDCG@10 in English, it drops to 0.309 in Arabic, 0.459 in Japanese, and 0.368 in Korean. LFM2-ColBERT-350M maintains more consistent performance across all languages, with the gap between English and lower-resource languages being significantly smaller (0.661 to 0.490 for Arabic, a 26% reduction vs. 55% for the baseline).

**Cross-lingual retrieval.** Table 13 presents NDCG@10 scores for all language pairs, where rows represent document languages and columns represent query languages. LFM2-ColBERT-350M is particularly strong



**Fig. 8: Monolingual retrieval performance of LFM2-ColBERT-350M.** Documents and queries are in the same language and measured by NDCG@10 on NanoBEIR across languages.

in cross-lingual transfer for European languages. For instance, querying in English against French documents achieves 0.551 NDCG@10 (97.7% of monolingual French performance), and querying in Spanish against Portuguese documents achieves 0.547 (100% of monolingual Portuguese performance).

Doc / Query	AR	DE	EN	ES	FR	IT	JA	KO	PT	Avg
AR	<b>0.490</b>	0.288	0.339	0.303	0.304	0.286	0.357	0.338	0.291	33.30
DE	0.383	<b>0.563</b>	0.547	0.498	0.502	0.489	0.424	0.368	0.486	47.33
EN	0.416	0.554	<b>0.661</b>	0.553	0.551	0.522	0.477	0.395	0.535	51.82
ES	0.412	0.514	0.578	<b>0.563</b>	0.547	0.529	0.436	0.394	0.547	50.21
FR	0.408	0.527	0.573	0.552	<b>0.564</b>	0.537	0.450	0.388	0.549	50.53
IT	0.395	0.512	0.554	0.535	0.535	<b>0.543</b>	0.439	0.386	0.529	49.20
JA	0.375	0.365	0.409	0.358	0.345	0.337	<b>0.557</b>	0.491	0.330	39.63
KO	0.326	0.274	0.310	0.282	0.265	0.266	0.440	<b>0.527</b>	0.271	32.89
PT	0.402	0.499	0.558	0.545	0.528	0.529	0.436	0.382	<b>0.547</b>	49.17
<b>Avg</b>	40.07	45.51	50.32	46.54	46.00	44.86	44.62	40.78	45.38	-

**Table 13:** Cross-lingual retrieval performance (NDCG@10) on NanoBEIR for LFM2-ColBERT-350M. Rows represent document language, columns represent query language. Diagonal elements (**bold**) represent monolingual retrieval. Compare to the results for GTE-ModernColBERT-v1 in Table 14.

Table 14 shows that GTE-ModernColBERT-v1 suffers from more degradation in cross-lingual scenarios. Querying in German against English documents yields only 0.408 NDCG@10 (60% of monolingual English performance), and most cross-lingual combinations involving Arabic, Japanese, or Korean fall below 0.200.

**NanoBEIR task breakdown.** Table 15 presents detailed results on individual NanoBEIR tasks, showing consistent performance across diverse retrieval scenarios. The model excels in fact verification tasks (FEVER: 0.949 NDCG@10, HotpotQA: 0.895 NDCG@10) and maintains strong performance in question answering (Natural Questions: 0.746 NDCG@10) and domain-specific retrieval (SciFact: 0.804 NDCG@10, FiQA: 0.591 NDCG@10).

**Multilingual capability inheritance.** Despite training exclusively on English data, the resulting model demonstrates strong multilingual retrieval capabilities (Figure 8) and robust cross-lingual transfer (Table 13). During pre-training, the LFM2-350M checkpoint we use here was exposed to text in several languages, including English, Arabic, Chinese, French, German, Japanese, Korean, Spanish, across 25 trillion tokens. Afterwards, during the knowledge distillation phase, the model was only exposed to English to-

<b>Doc / Query</b>	<b>AR</b>	<b>DE</b>	<b>EN</b>	<b>ES</b>	<b>FR</b>	<b>IT</b>	<b>JA</b>	<b>KO</b>	<b>PT</b>	<b>Avg</b>
AR	<b>0.309</b>	0.089	0.107	0.089	0.094	0.092	0.070	0.049	0.087	10.96
DE	0.039	<b>0.499</b>	0.454	0.362	0.393	0.367	0.133	0.061	0.361	29.66
EN	0.042	0.408	<b>0.680</b>	0.446	0.484	0.420	0.167	0.073	0.438	35.09
ES	0.044	0.360	0.485	<b>0.525</b>	0.465	0.437	0.149	0.061	0.487	33.48
FR	0.044	0.381	0.505	0.455	<b>0.546</b>	0.428	0.136	0.057	0.467	33.54
IT	0.043	0.369	0.449	0.446	0.451	<b>0.516</b>	0.143	0.054	0.448	32.43
JA	0.031	0.169	0.250	0.172	0.177	0.169	<b>0.459</b>	0.059	0.165	18.34
KO	0.030	0.134	0.169	0.127	0.133	0.125	0.090	<b>0.368</b>	0.124	14.44
PT	0.043	0.368	0.479	0.492	0.467	0.448	0.138	0.062	<b>0.530</b>	33.63
<b>Avg</b>	6.94	30.86	39.76	34.60	35.67	33.36	16.50	9.38	34.52	-

**Table 14:** Cross-lingual retrieval performance (NDCG@10) on NanoBEIR for GTE-ModernColBERT-v1. Rows represent document language, columns represent query language. Diagonal elements (bold) represent monolingual retrieval. Compare to the results for LFM2-ColBERT-350M in Table 13.

<b>Dataset</b>	<b>NDCG@10</b>	<b>MRR@10</b>	<b>MAP@100</b>	<b>Acc@1</b>	<b>Acc@10</b>	<b>Recall@10</b>
NanoFEVER	0.949	0.967	0.940	0.96	0.98	0.960
NanoHotpotQA	0.895	0.954	0.845	0.92	1.00	0.940
NanoQuoraRetrieval	0.882	0.863	0.843	0.80	1.00	0.979
NanoSciFact	0.804	0.771	0.771	0.70	0.92	0.910
NanoNQ	0.746	0.732	0.708	0.66	0.88	0.850
NanoDBPedia	0.714	0.898	0.575	0.86	0.98	0.418
NanoMSMARCO	0.686	0.644	0.656	0.58	0.82	0.820
NanoTouche2020	0.630	0.889	0.462	0.80	1.00	0.347
NanoFiQA2018	0.591	0.663	0.533	0.56	0.82	0.636
NanoSCIDOCs	0.384	0.613	0.297	0.50	0.86	0.381
NanoArguAna	0.551	0.449	0.451	0.28	0.88	0.880
NanoClimateFEVER	0.387	0.506	0.313	0.40	0.80	0.459
NanoNFCorpus	0.377	0.566	0.184	0.50	0.70	0.152
<b>Mean</b>	0.661	0.732	0.583	0.655	0.895	0.672

**Table 15:** Detailed results on NanoBEIR tasks for LFM2-ColBERT-350M. All metrics are reported on English queries and documents.

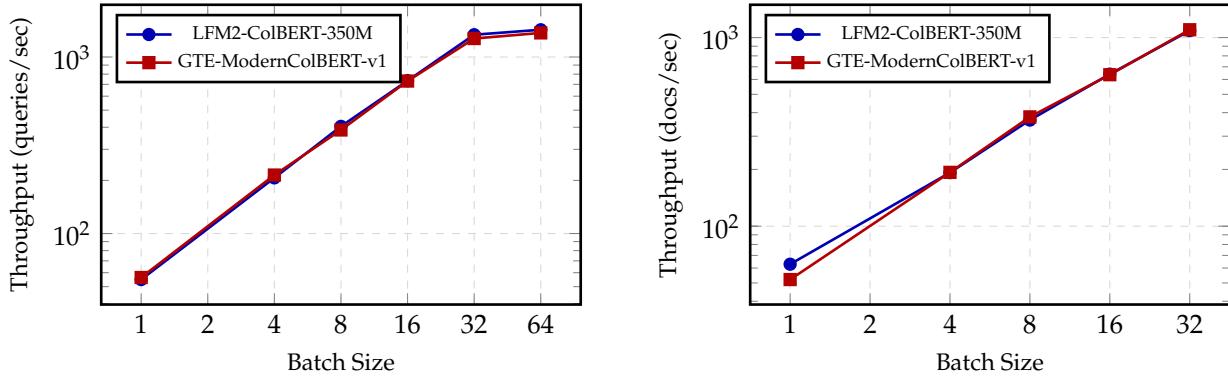
kens. This transfer mechanism enables zero-shot multilingual retrieval without requiring parallel training data in target languages.

The effectiveness of this approach depends on the similarity between the pre-training multilingual data distribution and the target retrieval tasks. The results show stronger cross-lingual transfer for European languages (0.50+ NDCG@10 cross-lingually) compared to more distant language pairs like Arabic-Japanese (0.375 NDCG@10), likely reflecting both linguistic distance and the relative volume of each language in the pre-training corpus. Future work could investigate whether targeted multilingual fine-tuning on translated English data or parallel retrieval corpora further improves cross-lingual performance, particularly for low-resource language pairs.

## 7.4 Inference Performance

**Experimental setup.** We benchmark encoding throughput on NVIDIA H100 GPUs across batch sizes from 1 to 64 for query encoding and 1 to 32 for document encoding. All measurements use identical quantization (BF16), tokenizer settings, and hardware configurations. Query inputs are constrained to 32 tokens following realistic patterns from MS MARCO (Bajaj et al., 2016) and Natural Questions (Kwiatkowski et al., 2019) datasets. Document inputs use 512 tokens sampled from diverse domains to capture realistic length

distributions. We report throughput in sequences per second, averaged over 1000 warmup iterations followed by 10,000 timed iterations.



(a) Query encoding throughput (queries/sec). Query length fixed at 32 tokens.

(b) Document encoding throughput (docs/sec). Document length fixed at 512 tokens.

**Fig. 9: Encoding throughput on NVIDIA H100 GPU across batch sizes.** LFM2-ColBERT-350M (353M parameters) matches the throughput of GTE-ModernColBERT-v1 (149M parameters).

**Query encoding results.** Figure 9a presents query encoding throughput across batch sizes. At batch size 1, both models achieve approximately 60 queries/second. As batch size increases, LFM2-ColBERT-350M demonstrates efficient scaling, reaching 1,350 queries/second at batch size 32 and 1,420 queries/second at batch size 64. GTE-ModernColBERT-v1 achieves 1,300 and 1,370 queries/second at the same batch sizes, respectively. LFM2-ColBERT-350M thus maintains a 3.8% throughput advantage at batch size 32 and a 3.6% advantage at batch size 64, despite being 2.37 times larger. The throughput curves show saturation around batch size 32, indicating optimal GPU utilization where memory bandwidth becomes the limiting factor rather than compute capacity.

**Document encoding results.** Figure 9b presents document encoding throughput. The performance curves are nearly indistinguishable, with both models achieving approximately 60 documents/second at batch size 1 and approximately 1,100 documents/second at batch size 32.

## 8 Related Work

Here, we further situate LFM2 relative to three threads: (i) modern open foundation model releases, (ii) architectural choices for efficiency, (iii) training methods for small models.

### 8.1 Modern Open Weight Foundation Model Families

Modern autoregressive language modeling generally follows the GPT-3 paradigm of large decoder-only Transformers trained on diverse web text (Brown et al., 2020), with capability gains predicted by scaling laws over data, compute, and parameters (Hoffmann et al., 2022; Kaplan et al., 2020). LFM2 adapts this lineage to an *edge-first* objective.

Recent general-purpose families (Gemma Team et al., 2025; Grattafiori et al., 2024; Liu et al., 2024; Yang et al., 2025a) optimize primarily for broad capability and efficient datacenter serving (high throughput, large batches, accelerator-rich environments), though sometimes also include smaller model checkpoints. Other notable model families that have been useful for open science include the OLMo series of models (Groeneveld et al., 2024; Muennighoff et al., 2024; OLMo Team et al., 2024). By contrast, LFM2 is edge-first and addresses an efficiency and quality gap for on-device models by co-designing architecture and training procedures specifically for on-device constraints while preserving the generalist capabilities that make foundation models valuable.

---

## 8.2 Alternative Architectures for Sequence Modeling

**Liquid Time-Constant Networks** Our work draws inspiration from Liquid Time-Constant (LTC) networks which introduce continuous-time neural dynamics with input-conditioned time constants for sequence modeling (Hasani et al., 2021, 2022; Lechner et al., 2020a). LTC networks demonstrate that neural ODEs with adaptive time constants can achieve superior performance on time-series tasks while maintaining interpretability and computational efficiency (Chahine et al., 2023; Hasani et al., 2020; Lechner et al., 2019, 2020b; Vorbach et al., 2021). The liquid formulation’s ability to dynamically adjust its effective temporal receptive field based on input characteristics inspired LFM2’s adaptive processing of local and global context.

**State space models, linear attention and convolutions** Another line of work that influences the LFM2 design is the development of efficient softmax attention alternatives that take the form of linear state space models (SSMs), linear attention, or convolutions. The S4 (Gu et al., 2022) model introduced a linear state space parameterization enabling efficient long-range sequence modeling by using a long convolution formulation for parallel processing and a fixed-state size recurrent formulation for autoregressive decoding. Further algorithmic improvements to this class of models were achieved by directly leveraging an alternative frequency-domain representation, the z-domain rational transfer function, which reduces the entire linear SSM parallel inference algorithm to a small number of FFT operations over the input and parameters (Parnichkun et al., 2024). However, a limitation of these approaches was that it required a linear time-invariant formulation, preventing the ability to use input-dependent dynamics. Liquid-S4 (Hasani et al., 2023) showed how a particular input-dependent state transition could be incorporated into S4 while maintaining the convolutional computation path, leading to improved performance. Meanwhile, S5 (Smith et al., 2023a,b) unlocked the ability to use time-varying SSMs by showing that fully recurrent SSMs could be computed using efficient parallel scans and diagonalized dynamics, obviating the need for the convolution formulation and LTI restriction. This SSM line of work led to the S6 formulation and Mamba block (Gu and Dao, 2024), which introduced a hardware-aware parallel scan implementation that enabled the ability to use input-dependent SSMs with extremely large states.

In parallel, linear attention variants (Choromanski et al., 2021; Katharopoulos et al., 2020) reparameterize or approximate the softmax activation and often take advantage of the associativity of matrix multiplications to admit more efficient parallel processing and fixed-state size recurrences. More recent versions improve on the gating and parameterization of the update rules to increase expressivity (Dao and Gu, 2024; Yang et al., 2024a,b, 2025b).

Attempts to perform language modeling with efficient SSMs or linear attention led to more complex computational blocks to improve expressivity. The Gated State Space (GSS) (Mehta et al., 2023) model introduced the idea of adding additional multiplicative gating around an SSM. The H3 model (Fu et al., 2023) augmented this design by adding an additional short-range shift SSM (equivalent to a short-range convolution) along with a long-range SSM and extra gating in-between. Hyena (Poli et al., 2023) replaced the long-range SSM with an implicitly parametrized long-range convolution (Romero et al., 2022; Sitzmann et al., 2020) and replaced the shift SSM with a short convolution. While the long-range convolution prevents efficient autoregressive decoding (Massaroli et al., 2023), the idea of augmenting SSM or linear attention blocks with a short convolution has remained in various other designs (Dao and Gu, 2024; Gu and Dao, 2024; Yang et al., 2024b).

**Hybrid architectures** While some studies have shown the ability of sub-quadratic architectures (SSMs, linear attention, long convolutions) to match softmax attention in perplexity and some public benchmarks, a growing body of work indicates that *pure* sub-quadratic architectures degrade on long-range in-context abilities (Park et al., 2024), such as long-range retrieval (Bluir et al., 2024; Wen et al., 2025), multi-query associative recall (Arora et al., 2024a,b), and copying (Jelassi et al., 2024). In addition, they exhibit structurally bounded memory capacity and often much lower effective state size on memory-intensive tasks compared to softmax attention (Parnichkun et al., 2025). These types of core skills are critical for LLMs that need to be able to generate coherent text, follow directions, aggregate information, and respond accurately to multiple queries.

A typical approach to address these weaknesses has been to formulate hybrid models that interleave

---

sub-quadratic layers with global attention layers. While previous works such as Longformer (Beltagy et al., 2020) Big Bird (Zaheer et al., 2020), and GPT-3 (Brown et al., 2020) explored hybrid architectures of various softmax attention variants (e.g., sliding window attention, sparse attention), to our knowledge, the GSS work was the first to propose interleaving softmax attention with sub-quadratic architectures (Mehta et al., 2023) to overcome performance deficits of pure sub-quadratic models. Other works followed suit (Fu et al., 2023; Park et al., 2024) to improve observed skill degradations. Hybrid strategies with varying amounts of attention have become standard for released alternative architectures (Chen et al., 2025; Kimi Team et al., 2025; Lieber et al., 2024; Waleffe et al., 2024).

LFM2 follows this general principle of hybridization, but adopts a minimal hybrid tuned for devices with the hardware-in-the-loop search: gated short convolutions handle most local mixing, and only a minority of GQA blocks provide global context processing. We find that for on-device settings under identical efficiency budgets, further augmenting these stacks with SSMs, linear attention, or additional convolution operations did not improve quality (2.1). We note that there are connections between these findings and the recently proposed Canon layers (Allen-Zhu, 2025).

### 8.3 Efficient Training for Small Models

The knowledge distillation approach introduced in Section 3.3 extends classical distillation (Hinton et al., 2015) through a decoupled Top-K objective that addresses support mismatch issues when distilling from larger models. This approach is closest to the *ghost token* device discussed in concurrent Sparse Logit Sampling (SLS) work (Anshumann et al., 2025), where the teacher’s tail probability is aggregated into one token and KL is computed on the augmented support. By the KL chain rule, at  $\tau = 1$  this is exactly a binary membership KL plus a conditional Top-K KL as in our approach. Our contribution makes this decomposition explicit for Top-K storage and introduces temperature *only* inside the conditional Top-K term, which avoids the support mismatch that arises when tempering is applied over the augmented support. We note that our objective is deterministic and complementary to the sampling-based path in SLS. More broadly, this decomposition echoes the spirit of Decoupled Knowledge Distillation (DKD) (Zhao et al., 2022), but unlike DKD we target the Top-K setting where tail logits are unavailable, and storage is the primary constraint.

## 9 Conclusion

We presented LFM2, the second generation of Liquid Foundation Models, designed for on-device deployment under tight latency and memory constraints. The model family includes dense variants from 350M to 2.6B parameters and an 8.3B MoE model with 1.5B active parameters, all supporting 32K context length. We extend this backbone to multimodal and retrieval domains with LFM2-VL for vision–language tasks, LFM2-Audio for speech, and LFM2-ColBERT-350M for late-interaction retrieval.

### 9.1 Key Contributions

LFM2 combines architectural design, training, and deployment choices around an edge-first objective.

- **Minimal hybrid backbone.** A hardware-in-the-loop search over architectures and layouts yields a compact hybrid that uses gated short convolutions for most layers and a small number of GQA blocks for global context. Under identical quantization and runtimes, this design delivers up to  $2\times$  faster prefill and decode on CPUs compared to similarly sized attention-heavy baselines, while matching or improving benchmark accuracy.
- **Efficient pre-training** A tempered, decoupled Top-K knowledge distillation objective reduces storage and bandwidth when distilling from a larger teacher while avoiding support mismatch and stabilizing losses.
- **Post-training for edge workflows** A three-stage post-training recipe including supervised fine-tuning, length-normalized direct preference alignment, and parameter-space model merging that improves instruction following, RAG, function calling, and multilingual robustness for small models.

- 
- **Multimodal and retrieval extensions** LFM2-VL adds a SigLIP2-based vision encoder with dynamic tiling and token reduction, enabling a flexible resolution-latency trade-off. LFM2-Audio separates continuous audio input from discrete audio output, enabling real-time speech-to-speech capabilities with an efficient model. LFM2-ColBERT-350M adapts the hybrid backbone to late-interaction retrieval, achieving strong multilingual and cross-lingual performance with throughput comparable to much smaller attention-only baselines.

Together, these components show that an architecture and training procedure co-designed with device constraints can deliver broadly capable models that remain practical to deploy on commodity edge hardware.

## 9.2 Limitations and Future Work

While LFM2 achieves strong results for its size class, several limitations and opportunities for future work remain.

**Hardware & deployment coverage.** The deployment recipes and architecture search considered here are tuned for batch size of 1, low-latency inference on a small set of CPU and mobile SoC configurations (Snapdragon-class phones and Ryzen-class laptops) using specific quantization schemes (8da4w in ExecuTorch and Q4\_0 in llama.cpp). LFM2 also runs competitively on modern NPUs and GPUs, but these accelerators were not central to the hardware-in-the-loop search, and we do not claim the resulting architectures or quantization schemes are optimal for large-batch server settings or any particular accelerator family. The trade-off between specializing model variants for specific hardware targets versus maintaining a single cross-platform backbone remains open. As edge runtimes, compiler stacks, and NPU/GPU kernel libraries continue to mature, a broader region of the model and quantization design space may become attractive. Revisiting LFM2’s architecture and deployment recipes under improved edge infrastructure is therefore an important direction for future work.

**Capacity and task coverage.** As small-scale models, LFM2 variants are inherently capacity-limited compared to frontier-scale systems: they cannot match the breadth and depth of capabilities across all open-ended reasoning, knowledge-intensive, and highly compositional tasks. In practice, we observe that LFM2 performs best on workloads that align with its design targets: short to medium context interactions, task-oriented applications, and edge deployments with tight latency and memory budgets. We view LFM2 as a step in an ongoing line of work to expand the capabilities of small models through improved data, objectives, and architectures. We are continuing to explore training and deployment techniques that narrow the gap with larger models for targeted use cases.

**Multimodal scope.** LFM2-VL and LFM2-Audio are the first multimodal extensions of the LFM2 backbone, aimed at vision and speech on edge devices. Their current capabilities reflect design choices made for on-device deployment.

LFM2-VL targets multi-image vision-language tasks. It supports images up to  $512 \times 512$  pixels directly and uses tiling plus token reduction for higher resolutions to trade off accuracy against latency and memory. The current model is trained without explicit grounding signals (e.g., bounding boxes, masks, and keypoints) and does not natively handle video or other continuous visual inputs, so we expect weaker performance on fine-grained localization and temporal reasoning. Future work includes adding lightweight grounding compatible with edge budgets, expanding the supported resolution range, exploring video extensions, and integrating sensor streams. We also plan post-training stages for the VLM with reinforcement-learning-style objectives to improve visual reasoning and long-horizon instruction following.

LFM2-Audio focuses on speech applications such as ASR, TTS, and conversational assistants, using a mixture of transcription, classification, TTS, and audio-chat data. We do not systematically evaluate non-speech audio (e.g., music or environmental sounds) or overlapping multi-speaker speech, and training is dominated by English and other high-resource languages, which likely reduces robustness for low-resource languages and accents. Future work includes expanding to richer multilingual and non-speech audio corpora, studying streaming behavior under real device constraints, and exploring on-device personalization.

---

**Late Interaction Models** While European languages show strong mutual transfer, Arabic and East Asian languages (JA, KO, ZH) exhibit weaker cross-lingual performance. Future work should investigate targeted training strategies for distant language pairs, such as multilingual contrastive objectives or language-adversarial training. The current model is trained on general-domain retrieval, however, specialized domains (e.g., biomedical literature, legal documents, scientific papers) are likely to benefit from domain-specific adaptation. Thanks to the modular architecture, this can be achieved by continuing to train the projection layer while keeping the LFM2 backbone frozen, reducing the cost of domain-specific deployments. Finally, the current configuration limits documents to 512 tokens, which is sufficient for most retrieval scenarios but restrictive for long-form content. Since the LFM2 backbone supports 32K tokens, extending document encoding to longer sequences could improve retrieval for technical documentation, research papers, and legal contracts, but would require careful management of MaxSim computation and index storage as sequence length grows.

### 9.3 Closing Remarks

LFM2 targets a specific regime: models small enough to run entirely on commodity CPUs and mobile SoCs while still supporting 32K contexts, multimodal inputs, and retrieval. By co-designing architecture, training, and post-training with hardware-in-the-loop, LFM2 models provide strong capabilities for their size, achieve substantially faster prefill and decode on CPUs, while fitting within the RAM budgets of contemporary phones, tablets, and embedded systems. This makes it practical to run assistants, RAG workflows, and speech interfaces fully on-device, without relying on continuous network connectivity or large accelerator clusters.

Shifting more inference from centralized data centers to edge devices has direct implications for privacy, reliability, and resource use. Processing text, images, and audio locally reduces the need to transmit raw user data to remote servers, allowing applications to function in low-connectivity settings. At the same time, reducing reliance on remote inference can substantially lower the aggregate cost of serving everyday interactions. We expect several of the techniques introduced here, from hardware-in-the-loop optimized architectures to improved training methods, lightweight multimodal extensions, and retrieval-focused variants, to be broadly applicable beyond this model family.

Together with the open weights and deployment recipes (Section 9.4), we hope LFM2 serves as a strong and practical baseline for building and studying edge-first foundation models.

### 9.4 Availability

All models, including the task-specific Liquid-Nanos family, are released with open weights at [HuggingFace.co/LiquidAI](https://HuggingFace.co/LiquidAI) with optimized deployment packages for ExecuTorch, llama.cpp, and vLLM. The models support deployment on Qualcomm Snapdragon SoCs, AMD Ryzen processors, and standard CPUs. Comprehensive deployment guides and quantization profiles are included to facilitate immediate adoption across diverse hardware platforms.

## 10 Authors

**Contributors (alphabetical by last name):** 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, Neehal Tumma

## References

Abdelrahman Abouelenin, Atabak Ashfaq, Adam Atkinson, Hany Awadalla, Nguyen Bach, Jianmin Bao, Alon Benhaim, Martin Cai, Vishrav Chaudhary, Congcong Chen, et al. Phi-4-mini technical report: Com-

---

pact yet powerful multimodal language models via mixture-of-loras. *arXiv preprint arXiv:2503.01743*, 2025.

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4895–4901, 2023.

Zeyuan Allen-Zhu. Physics of language models: Part 4.1, architecture design and the magic of canon layers. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Xiang An, Yin Xie, Kaicheng Yang, Wenkang Zhang, Xiuwei Zhao, Zheng Cheng, Yirui Wang, Songcen Xu, Changrui Chen, Chunsheng Wu, et al. Llava-onevision-1.5: Fully open framework for democratized multimodal training. *arXiv preprint arXiv:2509.23661*, 2025.

Anshumann Anshumann, Mohd Abbas Zaidi, Akhil Kedia, Jinwoo Ahn, Taehwak Kwon, Kangwook Lee, Haejun Lee, and Joohyung Lee. Sparse logit sampling: Accelerating knowledge distillation in llms. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 18085–18108, 2025.

Simran Arora, Sabri Eyuboglu, Aman Timalsina, Isys Johnson, Michael Poli, James Zou, Atri Rudra, and Christopher Re. Zoology: Measuring and improving recall in efficient language models. In *The Twelfth International Conference on Learning Representations*, 2024a.

Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, James Zou, Atri Rudra, and Christopher Re. Simple linear attention language models balance the recall-throughput tradeoff. In *International Conference on Machine Learning*, pages 1763–1840. PMLR, 2024b.

Baidu. Ernie 4.5 technical report, 2025.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. MS MARCO: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*, 2016.

Elie Bakouch, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, Lewis Tunstall, Carlos Miguel Patiño, Edward Beeching, Aymeric Roucher, Aksel Joonas Reedi, Quentin Gallouédec, Kashif Rasul, Nathan Habib, Clémentine Fourrier, Hynek Kydlicek, Guilherme Penedo, Hugo Larcher, Mathieu Morlon, Vaibhav Srivastav, Joshua Lochner, Xuan-Son Nguyen, Colin Raffel, Leandro von Werra, and Thomas Wolf. SmolLM3: smol, multilingual, long-context reasoner. <https://huggingface.co/blog/smollm3>, 2025.

Mohammad Bavarian, Heewoo Jun, Nikolas Tezak, John Schulman, Christine McLeavey, Jerry Tworek, and Mark Chen. Efficient training of language models to fill in the middle. *arXiv preprint arXiv:2207.14255*, 2022.

Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Aaron Blakeman, Aarti Basant, Abhinav Khattar, Adithya Renduchintala, Akhiad Bercovich, Aleksander Ficek, Alexis BJORLIN, Ali Taghibakhshi, Amala Sanjay Deshmukh, Ameya Sunil Mahabaleshwarkar, et al. Nemotron-h: A family of accurate and efficient hybrid mamba-transformer models. *arXiv preprint arXiv:2504.03624*, 2025.

Sam Bluir, Jimmy T.H. Smith, Antonios Anastasopoulos, and Amarda Shehu. Birdie: Advancing state space models with reward-driven objectives and curricula. *arXiv preprint arXiv:2411.01030*, 2024.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.

Antoine Chaffin. Gte-moderncolbert, 2025. URL <https://huggingface.co/lightonai/GTE-ModernColBERT-v1>.

- 
- Antoine Chaffin and Raphaël Sourté. Pylate: Flexible training and retrieval for late interaction models, 2024. URL <https://github.com/lightonai/pylate>.
- Makram Chahine, Ramin Hasani, Patrick Kao, Aaron Ray, Ryan Shubert, Mathias Lechner, Alexander Amini, and Daniela Rus. Robust flight navigation out of distribution with liquid neural networks. *Science Robotics*, 8(77):eadc8892, 2023.
- Aili Chen, Aonian Li, Bangwei Gong, Binyang Jiang, Bo Fei, Bo Yang, Boji Shan, Changqing Yu, Chao Wang, Cheng Zhu, et al. Minimax-m1: Scaling test-time compute efficiently with lightning attention. *arXiv preprint arXiv:2506.13585*, 2025.
- Yiming Chen, Xianghu Yue, Chen Zhang, Xiaoxue Gao, Robby T. Tan, and Haizhou Li. Voicebench: Benchmarking llm-based voice assistants. *arXiv preprint arXiv:2410.17196*, 2024.
- Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.
- Krzysztof Marcin Choromanski, Valerii Likhoshesterov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J Colwell, and Adrian Weller. Rethinking attention with performers. In *International Conference on Learning Representations*, 2021.
- Benjamin Clavié. JaColBERTv2. 5: Optimising multi-vector retrievers to create state-of-the-art japanese retrievers with constrained resources. *Journal of Natural Language Processing*, 32(1):176–218, 2025.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- Jade Copet, Felix Kreuk, Itai Gat, Tal Remez, David Kant, Gabriel Synnaeve, Yossi Adi, and Alexandre Défossez. Simple and controllable music generation. *Advances in Neural Information Processing Systems*, 36:47704–47720, 2023.
- Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *Forty-first International Conference on Machine Learning*, 2024.
- Pala Tej Deep, Rishabh Bhardwaj, and Soujanya Poria. DELLA-merging: Reducing interference in model merging through magnitude-based sampling. *arXiv preprint arXiv:2406.11617*, 2024.
- Alexandre Défossez, Laurent Mazaré, Manu Orsimi, Amélie Royer, Patrick Pérez, Hervé Jégou, Edouard Grave, and Neil Zeghidour. Moshi: a speech-text foundation model for real-time dialogue. *arXiv preprint arXiv:2410.00037*, 2024.
- Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Peter Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, et al. Scaling vision transformers to 22 billion parameters. In *International Conference on Machine Learning*, pages 7480–7512. PMLR, 2023a.
- Mostafa Dehghani, Basil Mustafa, Josip Djolonga, Jonathan Heek, Matthias Minderer, Mathilde Caron, Andreas Steiner, Joan Puigcerver, Robert Geirhos, Ibrahim M Alabdulmohsin, et al. Patch n’ pack: Navit, a vision transformer for any aspect ratio and resolution. *Advances in Neural Information Processing Systems*, 36:2252–2274, 2023b.
- Matt Deitke, Christopher Clark, Sangho Lee, Rohun Tripathi, Yue Yang, Jae Sung Park, Mohammadreza Salehi, Niklas Muennighoff, Kyle Lo, Luca Soldaini, et al. Molmo and pixmo: Open weights and open data for state-of-the-art multimodal models. *arXiv e-prints*, pages arXiv–2409, 2024.
- Karel D’Oosterlinck, Winnie Xu, Chris Develder, Thomas Demeester, Amanpreet Singh, Christopher Potts, Douwe Kiela, and Shikib Mehri. Anchored preference optimization and contrastive revisions: Addressing underspecification in alignment. *Transactions of the Association for Computational Linguistics*, 13:442–460, 2025.

---

Haodong Duan, Junming Yang, Yuxuan Qiao, Xinyu Fang, Lin Chen, Yuan Liu, Xiaoyi Dong, Yuhang Zang, Pan Zhang, Jiaqi Wang, et al. Vlmevalkit: An open-source toolkit for evaluating large multi-modality models. In *Proceedings of the 32nd ACM international conference on multimedia*, pages 11198–11201, 2024.

Daniel Y Fu, Tri Dao, Khaled Kamal Saab, Armin W Thomas, Atri Rudra, and Christopher Re. Hungry hungry hippos: Towards language modeling with state space models. In *The Eleventh International Conference on Learning Representations*, 2023.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

Georgi Gerganov and contributors. llama.cpp. <https://github.com/ggml-org/llama.cpp>, 2023. LLM inference in C/C++.

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

Dirk Groeneveld, Iz Beltagy, Evan Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, et al. OLMo: Accelerating the science of language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (volume 1: Long papers)*, pages 15789–15809, 2024.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024.

Albert Gu, Karan Goel, and Christopher Re. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2022.

Ramin Hasani, Mathias Lechner, Alexander Amini, Daniela Rus, and Radu Grosu. A natural lottery ticket winner: Reinforcement learning with ordinary neural circuits. In *International Conference on Machine Learning*, pages 4082–4093. PMLR, 2020.

Ramin Hasani, Mathias Lechner, Alexander Amini, Daniela Rus, and Radu Grosu. Liquid time-constant networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 7657–7666, 2021.

Ramin Hasani, Mathias Lechner, Alexander Amini, Lucas Liebenwein, Aaron Ray, Max Tschaikowski, Gerald Teschl, and Daniela Rus. Closed-form continuous-time neural networks. *Nature Machine Intelligence*, 4(11):992–1003, 2022.

Ramin Hasani, Mathias Lechner, Tsun-Hsuan Wang, Makram Chahine, Alexander Amini, and Daniela Rus. Liquid structural state-space models. In *The Eleventh International Conference on Learning Representations*, 2023.

Yun He, Di Jin, Chaoqi Wang, Chloe Bi, Karishma Mandyam, Hejia Zhang, Chen Zhu, Ning Li, Tengyu Xu, Hongjiang Lv, et al. Multi-IF: Benchmarking llms on multi-turn and multilingual instruction following. *arXiv preprint arXiv:2410.15553*, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Saurav Kadavath, Dawn Arora, Eric Guo, Nicholas He, et al. Measuring mathematical problem solving with the MATH dataset. *arXiv preprint arXiv:2103.03874*, 2021a.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021b.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

---

---

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

IBM Research. Granite 4.0 language models. <https://github.com/ibm-granite/granite-4.0-language-models>, 2025.

Gabriel Ilharco, Marco Túlio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

Won Jang, Dan Lim, Jaesam Yoon, Bongwan Kim, and Juntae Kim. Univnet: A neural vocoder with multi-resolution spectrogram discriminators for high-fidelity waveform generation. In *Interspeech 2021*, pages 2207–2211, 2021. doi: 10.21437/Interspeech.2021-1016.

Samy Jelassi, David Brandfonbrener, Sham M Kakade, and Eran Malach. Repeat after me: Transformers are better than state space models at copying. In *International Conference on Machine Learning*, pages 21502–21521. PMLR, 2024.

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. *arXiv preprint arXiv:2001.08361*, 2020.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pages 5156–5165. PMLR, 2020.

Omar Khattab and Matei Zaharia. ColBERT: Efficient and effective passage search via contextualized late interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 39–48, 2020.

Kimi Team, Yu Zhang, Zongyu Lin, Xingcheng Yao, Jiaxi Hu, Fanqing Meng, Chengyin Liu, Xin Men, Songlin Yang, Zhiyuan Li, et al. Kimi linear: An expressive, efficient attention architecture. *arXiv preprint arXiv:2510.26692*, 2025.

Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. Hifi-gan: generative adversarial networks for efficient and high fidelity speech synthesis. In *Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS '20*, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

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.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

Mathias Lechner, Ramin Hasani, Manuel Zimmer, Thomas A Henzinger, and Radu Grosu. Designing worm-inspired neural networks for interpretable robotic control. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 87–94. IEEE, 2019.

Mathias Lechner, Ramin Hasani, Alexander Amini, Thomas A Henzinger, Daniela Rus, and Radu Grosu. Neural circuit policies enabling auditable autonomy. *Nature Machine Intelligence*, 2(10):642–652, 2020a.

---

Mathias Lechner, Ramin Hasani, Daniela Rus, and Radu Grosu. Gershgorin loss stabilizes the recurrent neural network compartment of an end-to-end robot learning scheme. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5446–5452. IEEE, 2020b.

Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11523–11532, 2022.

Qintong Li, Leyang Cui, Xueliang Zhao, Lingpeng Kong, and Wei Bi. GSM-Plus: A comprehensive benchmark for evaluating the robustness of LLMs as mathematical problem solvers. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL 2024), Volume 1: Long Papers*, pages 2961–2984, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.163.

Tianjian Li and Daniel Khashabi. Simplemix: Frustratingly simple mixing of off-and on-policy data in language model preference learning. In *Forty-second International Conference on Machine Learning*, 2025.

Tianpeng Li, Jun Liu, Tao Zhang, Yuanbo Fang, Da Pan, Mingrui Wang, Zheng Liang, Zehuan Li, Mingan Lin, Guosheng Dong, et al. Baichuan-audio: A unified framework for end-to-end speech interaction. *arXiv preprint arXiv:2502.17239*, 2025.

Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meiriom, Yonatan Belinkov, Shai Shalev-Shwartz, et al. Jamba: A hybrid transformer-mamba language model. *arXiv preprint arXiv:2403.19887*, 2024.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let’s verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Andrés Marafioti, Orr Zohar, Miquel Farré, Merve Noyan, Elie Bakouch, Pedro Cuenca, Cyril Zakka, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, et al. Smolvlm: Redefining small and efficient multimodal models. *arXiv preprint arXiv:2504.05299*, 2025.

Stefano Massaroli, Michael Poli, Dan Fu, Hermann Kumbong, Rom Parnichkun, David Romero, Aman Timalsina, Quinn McIntyre, Beidi Chen, Atri Rudra, et al. Laughing hyena distillery: Extracting compact recurrences from convolutions. *Advances in Neural Information Processing Systems*, 36:17072–17116, 2023.

Harsh Mehta, Ankit Gupta, Ashok Cutkosky, and Behnam Neyshabur. Long range language modeling via gated state spaces. In *The Eleventh International Conference on Learning Representations*, 2023.

Yu Meng, Mengzhou Xia, and Danqi Chen. SimPo: Simple Preference Optimization with a Reference-Free Reward. *Advances in Neural Information Processing Systems*, 37:124198–124235, 2024.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, et al. OLMoE: Open mixture-of-experts language models. *arXiv preprint arXiv:2409.02060*, 2024.

OLMo Team, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, et al. 2 OLMo 2 Furious. *arXiv preprint arXiv:2501.00656*, 2024.

OpenAI. MMMLU: Multilingual massive multitask language understanding. <https://huggingface.co/datasets/openai/MMMLU>, 2024. Dataset accessed 2025-11-26.

Jongho Park, Jaeseung Park, Zheyang Xiong, Nayoung Lee, Jaewoong Cho, Samet Oymak, Kangwook Lee, and Dimitris Papailiopoulos. Can Mamba learn how to learn? a comparative study on in-context learning tasks. In *International Conference on Machine Learning*, pages 39793–39812. PMLR, 2024.

---

- 
- Rom Parnichkun, Stefano Massaroli, Alessandro Moro, Jimmy T.H. Smith, Ramin Hasani, Mathias Lechner, Qi An, Christopher Re, Hajime Asama, Stefano Ermon, et al. State-free inference of state-space models: The transfer function approach. In *International Conference on Machine Learning*, pages 39834–39860. PMLR, 2024.
- Rom Parnichkun, Neehal Tumma, Armin W Thomas, Alessandro Moro, Qi An, Taiji Suzuki, Atsushi Yamashita, Michael Poli, and Stefano Massaroli. Quantifying memory utilization with effective state-size. In *Forty-second International Conference on Machine Learning*, 2025.
- Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Ré. Hyena hierarchy: Towards larger convolutional language models. In *International Conference on Machine Learning*, pages 28043–28078. PMLR, 2023.
- Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hannaneh Hajishirzi. Generalizing verifiable instruction following. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025.
- Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. In *International Conference on Machine Learning*, pages 28492–28518. PMLR, 2023.
- Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct Preference Optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741, 2023.
- David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level Google-proof Q&A benchmark. In *First Conference on Language Modeling*, 2024.
- Dima Rekesh, Nithin Rao Koluguri, Samuel Kriman, Somshubra Majumdar, Vahid Noroozi, He Huang, Oleksii Hrinchuk, Krishna Puvvada, Ankur Kumar, Jagadeesh Balam, et al. Fast conformer with linearly scalable attention for efficient speech recognition. In *2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 1–8. IEEE, 2023.
- David W. Romero, Anna Kuzina, Erik J Bekkers, Jakub Mikolaj Tomczak, and Mark Hoogendoorn. CK-Conv: Continuous kernel convolution for sequential data. In *International Conference on Learning Representations*, 2022.
- Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, 2016.
- Noam Shazeer. GLU variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.
- Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, Dipanjan Das, and Jason Wei. Language models are multilingual chain-of-thought reasoners. In *The Eleventh International Conference on Learning Representations*, 2023.
- Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1874–1883, 2016.
- Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33: 7462–7473, 2020.

- 
- Hubert Siuzdak. Vocos: Closing the gap between time-domain and fourier-based neural vocoders for high-quality audio synthesis. *arXiv preprint arXiv:2306.00814*, 2023.
- Jimmy T.H. Smith, Shalini De Mello, Jan Kautz, Scott Linderman, and Wonmin Byeon. Convolutional state space models for long-range spatiotemporal modeling. *Advances in Neural Information Processing Systems*, 36:80690–80729, 2023a.
- Jimmy T.H. Smith, Andrew Warrington, and Scott Linderman. Simplified state space layers for sequence modeling. In *The Eleventh International Conference on Learning Representations*, 2023b.
- Raphaël Sourty. Nanobeir-multilingual: Multilingual version of nanobeir for quick evaluation., 2025. URL <https://huggingface.co/datasets/lightonai/nanobeir-multilingual>.
- Vaibhav Srivastav, Somshubra Majumdar, Nithin Koluguri, Adel Moumen, Sanchit Gandhi, et al. Open automatic speech recognition leaderboard. [https://huggingface.co/spaces/hf-audio/open\\_asr\\_leaderboard](https://huggingface.co/spaces/hf-audio/open_asr_leaderboard), 2023.
- Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.
- Xingwu Sun, Yanfeng Chen, Yiqing Huang, Ruobing Xie, Jiaqi Zhu, Kai Zhang, Shuaipeng Li, Zhen Yang, Jonny Han, Xiaobo Shu, et al. Hunyuan-large: An open-source moe model with 52 billion activated parameters by tencent. *arXiv preprint arXiv:2411.02265*, 2024.
- Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. Beir: A heterogeneous benchmark for zero-shot evaluation of information retrieval models, 2021.
- Armin W Thomas, Rom Parnichkun, Alexander Amini, Stefano Massaroli, and Michael Poli. STAR: Synthesis of tailored architectures. In *The Thirteenth International Conference on Learning Representations*, 2024.
- Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, et al. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv preprint arXiv:2502.14786*, 2025.
- Charles Vorbach, Ramin Hasani, Alexander Amini, Mathias Lechner, and Daniela Rus. Causal navigation by continuous-time neural networks. *Advances in Neural Information Processing Systems*, 34:12425–12440, 2021.
- Roger Waleffe, Wonmin Byeon, Duncan Riach, Brandon Norick, Vijay Korthikanti, Tri Dao, Albert Gu, Ali Hatamizadeh, Sudhakar Singh, Deepak Narayanan, et al. An empirical study of mamba-based language models. *arXiv preprint arXiv:2406.07887*, 2024.
- Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-vl: Enhancing vision-language model’s perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024a.
- Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, et al. Internvl3. 5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *arXiv preprint arXiv:2508.18265*, 2025.
- Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhui Chen. MMLU-pro: A more robust and challenging multi-task language understanding benchmark. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024b.
- Kaiyue Wen, Xingyu Dang, and Kaifeng Lyu. RNNs are not transformers (yet): The key bottleneck on in-context retrieval. In *The Thirteenth International Conference on Learning Representations*, 2025.

---

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierrick Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pages 23965–23998. PMLR, 2022.

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

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models. *Advances in Neural Information Processing Systems*, 36:7093–7115, 2023.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengan Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. In *Forty-first International Conference on Machine Learning*, 2024a.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. *Advances in neural information processing systems*, 37:115491–115522, 2024b.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In *The Thirteenth International Conference on Learning Representations*, 2025b.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *Forty-first International Conference on Machine Learning*, 2024.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297, 2020.

Aohan Zeng, Zhengxiao Du, Mingdao Liu, Kedong Wang, Shengmin Jiang, Lei Zhao, Yuxiao Dong, and Jie Tang. Gbm-4-voice: Towards intelligent and human-like end-to-end spoken chatbot. *arXiv preprint arXiv:2412.02612*, 2024.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32, 2019.

Jingyuan Zhang, Kai Fu, Yang Yue, Chenxi Sun, Hongzhi Zhang, Yahui Liu, Xinguang Ji, Jia Fu, Tinghai Zhang, Yan Li, Qi Wang, Fuzheng Zhang, Guorui Zhou, and Kun Gai. Klear-qwen3-thinking-preview, 2025. URL <https://west-mask-4fa.notion.site/Klear-Qwen3-Thinking-Preview-23aab5f955ec8063b115da7d59dd9427>.

Borui Zhao, Quan Cui, Renjie Song, Yiyu Qiu, and Jiajun Liang. Decoupled knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11953–11962, 2022.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

## A Decoupled Top-K Knowledge Distillation

### A.1 Forward KL

**Notation.** Let  $\mathcal{A}$  be the vocabulary and  $\mathcal{T}(x_c) \subset \mathcal{A}$  the teacher's Top-K set for context  $x_c$ . Write  $\tilde{\mathcal{T}}(x_c) = \mathcal{A} \setminus \mathcal{T}(x_c)$ . Define

$$P_T(\mathcal{T} | x_c) = \sum_{x \in \mathcal{T}(x_c)} P_T(x | x_c), \quad P_S(\mathcal{T} | x_c) = \sum_{x \in \mathcal{T}(x_c)} P_S(x | x_c),$$

$$P_T(x | \mathcal{T}, x_c) = \frac{P_T(x | x_c)}{P_T(\mathcal{T} | x_c)}, \quad P_S(x | \mathcal{T}, x_c) = \frac{P_S(x | x_c)}{P_S(\mathcal{T} | x_c)} \quad (x \in \mathcal{T}).$$

Let  $\text{Bern}(p)$  denote a Bernoulli distribution with success probability  $p$ .

**Chain-rule identity for forward KL.** Starting from

$$D_{\text{KL}}(P_T(\cdot | x_c) \| P_S(\cdot | x_c)) = \sum_{x \in \mathcal{A}} P_T(x | x_c) \log \frac{P_T(x | x_c)}{P_S(x | x_c)}, \quad (9)$$

split the sum over  $\mathcal{T}$  and  $\tilde{\mathcal{T}}$ , write  $P_T(x | x_c) = P_T(\mathcal{T} | x_c) P_T(x | \mathcal{T}, x_c)$  for  $x \in \mathcal{T}$  (and analogously for  $\tilde{\mathcal{T}}$ ), and add–subtract the corresponding membership terms. This yields

$$\begin{aligned} D_{\text{KL}}(P_T(\cdot | x_c) \| P_S(\cdot | x_c)) &= D_{\text{KL}}(\text{Bern}(P_T(\mathcal{T} | x_c)) \| \text{Bern}(P_S(\mathcal{T} | x_c))) \\ &\quad + P_T(\mathcal{T} | x_c) D_{\text{KL}}(P_T(\cdot | \mathcal{T}, x_c) \| P_S(\cdot | \mathcal{T}, x_c)) \\ &\quad + P_T(\tilde{\mathcal{T}} | x_c) D_{\text{KL}}(P_T(\cdot | \tilde{\mathcal{T}}, x_c) \| P_S(\cdot | \tilde{\mathcal{T}}, x_c)). \end{aligned} \quad (10)$$

**Decoupled Top-K objective (lower bound).** Because teacher logits on  $\tilde{\mathcal{T}}(x_c)$  are unavailable, we drop the last (non-negative) term and optimize the lower bound

$$\mathcal{L}_{\text{DTK}}^{LB}(x_c) = D_{\text{KL}}(\text{Bern}(P_T(\mathcal{T} | x_c)) \| \text{Bern}(P_S(\mathcal{T} | x_c))) + P_T(\mathcal{T} | x_c) D_{\text{KL}}(P_T(\cdot | \mathcal{T}, x_c) \| P_S(\cdot | \mathcal{T}, x_c)). \quad (11)$$

**Temperature placement.** We apply temperature only to the conditional Top-K distributions  $P_T(\cdot | \mathcal{T}, x_c)$  and  $P_S(\cdot | \mathcal{T}, x_c)$ , i.e., we replace the conditional KL in Eq. (11) by  $D_{\text{KL}}^{(\tau)}$ , while keeping both the Bernoulli term and the scalar weight  $P_T(\mathcal{T} | x_c)$  untempered. For  $\tau \geq 1$ ,

$$p^{(\tau)}(x) = \frac{p(x)^{1/\tau}}{\sum_y p(y)^{1/\tau}}, \quad D_{\text{KL}}^{(\tau)}(p \| q) = \tau^2 D_{\text{KL}}(p^{(\tau)} \| q^{(\tau)}), \quad (12)$$

giving

$$\mathcal{L}_{\text{DTK}}(x_c) = D_{\text{KL}}(\text{Bern}(P_T(\mathcal{T} | x_c)) \| \text{Bern}(P_S(\mathcal{T} | x_c))) + P_T(\mathcal{T} | x_c) D_{\text{KL}}^{(\tau)}(P_T(\cdot | \mathcal{T}, x_c) \| P_S(\cdot | \mathcal{T}, x_c)) \quad (13)$$

with the membership term untempered.

**Interpretation.** The first term ( $\mathcal{L}_B$ ) matches the *amount* of mass on the Top-K, while the second term ( $\mathcal{L}_T$ ) matches the *shape* within the Top-K.

---

**Why naive Top-K+temperature is unstable.** Let  $\hat{P}_T$  denote the teacher distribution truncated to  $\mathcal{T}(x_c)$  (zero mass outside  $\mathcal{T}$ ). If we first truncate the teacher to  $\hat{P}_T$  and then apply temperature over the full vocabulary  $\mathcal{A}$ , we obtain, as  $\tau \rightarrow \infty$ ,

$$\lim_{\tau \rightarrow \infty} P_S^{(\tau)}(x) = \frac{1}{|\mathcal{A}|} \quad (\forall x \in \mathcal{A}), \quad \lim_{\tau \rightarrow \infty} \hat{P}_T^{(\tau)}(x) = \begin{cases} \frac{1}{|\mathcal{T}|}, & x \in \mathcal{T}, \\ 0, & x \notin \mathcal{T}, \end{cases} \quad (14)$$

so  $D_{\text{KL}}(\hat{P}_T^{(\tau)} \| P_S^{(\tau)}) \rightarrow \log \frac{|\mathcal{A}|}{|\mathcal{T}|}$  and, by definition  $D_{\text{KL}}^{(\tau)} = \tau^2 D_{\text{KL}}(\hat{P}_T^{(\tau)} \| P_S^{(\tau)})$ , the tempered KL grows as  $\tau^2$ . In contrast, with full logits both tempered distributions converge to uniform on  $\mathcal{A}$  and the KL tends to zero. Conditioning both teacher and student on  $\mathcal{T}$  and tempering only these conditional distributions makes them both converge to uniform on  $\mathcal{T}$ , removing this support mismatch.

## A.2 Reverse KL variant

For completeness, we also include some thoughts on the derivation of a decoupled reverse KL.

**Reverse KL chain rule.** With  $Z = \mathbf{1}[x \in \mathcal{T}(x_c)]$ , the reverse KL decomposes as

$$\begin{aligned} D_{\text{KL}}(P_S(\cdot | x_c) \| P_T(\cdot | x_c)) &= D_{\text{KL}}(\text{Bern}(P_S(\mathcal{T} | x_c)) \| \text{Bern}(P_T(\mathcal{T} | x_c))) \\ &\quad + P_S(\mathcal{T} | x_c) D_{\text{KL}}(P_S(\cdot | \mathcal{T}, x_c) \| P_T(\cdot | \mathcal{T}, x_c)) \\ &\quad + P_S(\bar{\mathcal{T}} | x_c) D_{\text{KL}}(P_S(\cdot | \bar{\mathcal{T}}, x_c) \| P_T(\cdot | \bar{\mathcal{T}}, x_c)). \end{aligned} \quad (15)$$

**Lower bound with Top-K.** Because teacher logits on  $\bar{\mathcal{T}}(x_c)$  are unavailable, we drop the last (non-negative) term and optimize the computable lower bound

$$\mathcal{L}_{\text{rev}}^{(\text{LB})}(x_c) = D_{\text{KL}}(\text{Bern}(P_S(\mathcal{T} | x_c)) \| \text{Bern}(P_T(\mathcal{T} | x_c))) + P_S(\mathcal{T} | x_c) D_{\text{KL}}^{(\tau)}(P_S(\cdot | \mathcal{T}, x_c) \| P_T(\cdot | \mathcal{T}, x_c)). \quad (16)$$

As in the forward-KL case, we apply temperature  $\tau$  only to the conditional Top-K term:

$$p^{(\tau)}(x) = \frac{p(x)^{1/\tau}}{\sum_y p(y)^{1/\tau}}, \quad D_{\text{KL}}^{(\tau)}(p \| q) = \tau^2 D_{\text{KL}}(p^{(\tau)} \| q^{(\tau)}).$$

**Teacher-weighted surrogate.** To avoid incentivizing the student to down-weight  $P_S(\mathcal{T} | x_c)$ , we also consider a teacher-weighted surrogate:

$$\mathcal{L}_{\text{rev}}^{(\text{tw})}(x_c) = D_{\text{KL}}(\text{Bern}(P_S(\mathcal{T} | x_c)) \| \text{Bern}(P_T(\mathcal{T} | x_c))) + P_T(\mathcal{T} | x_c) D_{\text{KL}}^{(\tau)}(P_S(\cdot | \mathcal{T}, x_c) \| P_T(\cdot | \mathcal{T}, x_c))$$

(17)

This variant is no longer a strict lower bound, but empirically removes the incentive to shrink  $P_S(\mathcal{T} | x_c)$ . As before, the outer binary term is untempered and temperature is applied only to the Top-K conditional term.

## B Model Merging Techniques

All merging methods operate on model state dictionaries loaded from safetensors format and merge parameters tensor-by-tensor while preserving the original dtype of each parameter.

**Model Soup** The simplest approach performs weighted averaging of corresponding parameters across models (Wortsman et al., 2022). Given  $n$  models with parameters  $\theta_1, \dots, \theta_n$  and normalized weights  $w_1, \dots, w_n$  (where  $\sum_{i=1}^n w_i = 1$ ), the merged parameters are computed as:

$$\theta_{\text{merged}} = \sum_{i=1}^n w_i \theta_i \quad (18)$$

---

**Task Arithmetic** Task arithmetic (Ilharco et al., 2022) extends model soups by working with parameter deltas relative to a base model. For a base model with parameters  $\theta_0$  and fine-tuned models  $\theta_1, \dots, \theta_n$ , we first compute the task vectors  $\tau_i = \theta_i - \theta_0$  representing the adaptations learned during fine-tuning. The merged model is then constructed as:

$$\theta_{\text{merged}} = \theta_0 + \sum_{i=1}^n w_i \tau_i \quad (19)$$

This formulation allows us to control the contribution of each fine-tuned capability independently.

**TIES-Merging** TIES (Trim, Elect Sign & Merge) (Yadav et al., 2023) addresses parameter interference in task arithmetic through a three-step procedure. First, magnitude-based sparsification retains only the top  $k\%$  of parameters by absolute value in each task vector  $\tau_i$ , setting the remainder to zero. Second, sign election resolves conflicts by computing the majority sign for each parameter position across all task vectors. Third, the disjoint merge averages only those parameters that agree with the elected majority sign. Formally, given sparsified task vectors  $\tilde{\tau}_i$  and elected signs  $\gamma_j = \text{sign}(\sum_{i=1}^n \tilde{\tau}_{ij})$ , we construct a consensus mask  $M_{ij} = \mathbb{1}[\text{sign}(\tilde{\tau}_{ij}) = \gamma_j]$  and compute:

$$\theta_{\text{merged}} = \theta_0 + \sum_j \frac{\sum_{i=1}^n w_i \tilde{\tau}_{ij} M_{ij}}{\sum_{i=1}^n w_i M_{ij}} \quad (20)$$

where the denominator normalizes by the sum of weights that agree with the majority sign.

**DARE** DARE (Drop And REscale) (Yu et al., 2024) employs random sparsification as an alternative to magnitude-based pruning. Each parameter in a task vector is dropped with probability  $p$  (the drop rate) and retained otherwise. To maintain the expected magnitude of the merged updates, retained parameters are rescaled by  $1/(1-p)$ . For a task vector  $\tau_i$  and drop rate  $p$ , the sparsified vector is computed as:

$$m_i \sim \text{Bernoulli}(p), \quad \tilde{\tau}_i = (1 - m_i) \odot \tau_i, \quad \hat{\tau}_i = \frac{\tilde{\tau}_i}{1 - p} \quad (21)$$

where  $\odot$  denotes element-wise multiplication. We implement DARE in combination with both standard task arithmetic (DARE-linear) and with TIES-style consensus (DARE-TIES), where the random sparsification replaces the magnitude-based trimming step while preserving the sign election and disjoint merge procedures.

**DELLA** DELLA (Drop and rEscaLe via sampLing with mAgNitude) (Deep et al., 2024) extends DARE by replacing uniform random dropout with magnitude-aware sampling. The key insight is that lower-magnitude parameters contribute less to task-specific performance and can tolerate higher dropout rates. For each layer, parameters are ranked by absolute magnitude, and dropout probabilities are assigned inversely proportional to rank:

$$r_i = \text{rank}(|\tau_{ij}|), \quad p_i = p_{\min} + \frac{\epsilon}{n} \cdot r_i \quad (22)$$

where  $p_{\min} = p - \epsilon/2$  ensures the average dropout rate equals  $p$ , and  $\epsilon$  controls the spread of probabilities across ranks. Each parameter is then independently dropped and rescaled:

$$m_i \sim \text{Bernoulli}(p_i), \quad \hat{\tau}_i = \frac{(1 - m_i) \odot \tau_i}{1 - p_i} \quad (23)$$

The per-parameter rescaling by  $1/(1 - p_i)$  preserves the expected contribution of each parameter to the output embeddings. We combine DELLA’s magnitude-based sampling with TIES-style sign election and disjoint merging, where parameters agreeing with the majority sign are averaged after the stochastic pruning step.

---

## C Evaluation Details

The evaluation scores for the text-only models come from a custom internal harness. We designed the benchmark implementations to reliably extract final answers from model outputs, whereas open-source evaluation harnesses use more restrictive parsers that may miss correct answers due to formatting variations.

### MMLU

Based on Hendrycks et al., Measuring Massive Multitask Language Understanding ([Hendrycks et al., 2021b](#)).

- **Dataset:** `cais/mmlu`
- **Implementation:** 5-shot by default (uses first 5 examples from dev set per subject). We apply the chat template to build a multi-turn conversation with each shot as instruction-answer pair.
- **Parsing:** Takes the highest logit token, strips whitespace/newlines, falls back to constraint-based prediction (max probability among A/B/C/D) if the stripped token is not a valid choice. This makes it more robust to tokenizer differences.
- **Scoring:** Constrains to A/B/C/D token logits via softmax for probability distribution.

### MMLU-Pro

From Wang et al., MMLU-Pro: A More Robust and Challenging Multi-Task Language Understanding Benchmark ([Wang et al., 2024b](#)).

- **Dataset:** TIGER-Lab/MMLU-Pro
- **Implementation:** 5-shot by default using Chain-of-Thought (CoT) examples from validation set as a conversation with a chat format applied. We remove all “N/A” options from the dataset.
- **Prompt:** “The following are multiple-choice questions (with answers) about subject. Think step by step and then finish your answer with “The answer is (X)” where X is the correct letter choice.\n” as a system prompt.
- **Parsing:** Custom parser with priority order:
  - Letter patterns at beginning of last 5 lines (e.g., “A.”, “A)”, “A/”, “A:”)
  - “answer is (X)” patterns
  - Standalone parenthesized letters
  - “choice X” or “option X” patterns
  - “final answer: X” patterns
  - Fallback to any letter followed by delimiter
  - Excludes false positives like “A: Let’s think step by step”
- **Generation:** Greedy decoding, max 8192 tokens.
- **Scoring:** String matching of extracted letter vs ground truth.

---

## GPQA Diamond

From Rein et al., GPQA: A Graduate-Level Google-Proof Q&A Benchmark ([Rein et al., 2024](#)).

- **Dataset:** `Idavidrein/gpqa` (`gpqa_diamond` subset)
- **Implementation:** Each question evaluated 10 times with different answer orderings to reduce position bias. Chat template is applied.
- **Prompt:** “What is the correct answer to this question: {question}\n\n{choices}”
- **Parsing:** Same approach as in MMLU.
- **Scoring:** Final accuracy is average across all permuted evaluations.

## IFEval

From Zhou et al., Instruction-Following Evaluation for Large Language Models ([Zhou et al., 2023](#)).

- **Dataset:** `google/IFEval`
- **Implementation:** Uses official IFEval reference implementation.
- **Generation:** Greedy decoding, `max_tokens=4096`
- **Scoring:** Average of all 4 metrics (`strict_prompt`, `loose_prompt`, `strict_instruction`, `loose_instruction`)

## IFBench

From Pyatkin et al., Generalizing Verifiable Instruction Following ([Pyatkin et al., 2025](#)).

- **Dataset:** `allenai/IFBench_test`
- **Implementation:** Uses official IFBench reference implementation.
- **Generation:** Greedy decoding, max 4096 tokens.
- **Scoring:** Similar to IFEval - strict/loose modes, prompt-level and instruction-level accuracy

## Multi-IF

From He et al., Multi-IF: Benchmarking LLMs on Multi-Turn and Multilingual Instruction Following ([He et al., 2024](#)).

- **Dataset:** `facebook/Multi-IF`
- **Implementation:** Uses official Multi-IF reference implementation. Automatically truncates prompts exceeding model's max context length.
- **Generation:** Greedy decoding, 32768 tokens.
- **Scoring:** Average accuracy across all 3 turns.

---

## GSM8K

From Cobbe et al., Training Verifiers to Solve Math Word Problems ([Cobbe et al., 2021](#)).

- **Dataset:** openai/gsm8k
- **Implementation:** 5-shot examples randomly sampled from train set. No system message but the chat template is applied to the entire conversation.
- **Prompt:**
  - Gemma 3 uses an additional instruction to improve performance “Solve the following math problem and write the final solution in `\boxed{}`”
  - Llama 3.2 uses an additional instruction to improve performance “Given the following problem, reason and give a final answer to the problem.\nYour response should end with “The final answer is [answer]” where [answer] is the response to the problem.”
- **Parsing:** Custom parser with priority order:
  1. `\boxed{}` pattern (LaTeX)
  2. `###` or `####` patterns
  3. “ANSWER:” pattern
  4. “answer is {}” pattern
  5. Other patterns (final answer, etc.)
  6. Numeric fallback from end of text
- **Generation:** Greedy decoding, max 4096 tokens.
- **Scoring:** Reports both strict (exact match after extraction) and loose (substring match) accuracy. Loose score is used because of low performance with strict scoring with Gemma 3 models.

## GSMPlus

From Li et al., GSM-Plus: A Comprehensive Benchmark for Evaluating the Robustness of LLMs as Mathematical Problem Solvers ([Li et al., 2024](#)).

- **Dataset:** qintongli/GSM-Plus
- **Implementation:** Same as GSM8K.
- **Prompt:** Same as GSM8K.
- **Parsing:** Same as GSM8K.
- **Generation:** Greedy decoding, max 4096 tokens.
- **Scoring:** Strict matching only.

## MATH 500

MATH-500 is a 500-problem subset of MATH constructed in the Let’s Verify Step by Step paper ([Lightman et al., 2024](#)).

- **Dataset:** HuggingFaceH4/MATH-500
- **Implementation:** 4-shot fixed examples with chat template applied to the entire conversation.
- **Prompt:** “Problem:\n{problem}”
- **Parsing:** Same as GSM8K.
- **Generation:** Greedy decoding, max 4096 tokens.

---

## MATH Lvl 5

Level 5 problems come from the original MATH dataset ([Hendrycks et al., 2021a](#)).

- **Dataset:** EleutherAI/hendrycks\_math (Level 5 only)
- **Implementation:** 0-shot with chat template.
- **Prompt:** “Problem:\n{problem}”
- **Parsing:** Same as GSM8K.
- **Generation:** Greedy decoding, max 4096 tokens.

## MMMLU

MMMLU is a multilingual extension of MMLU released by OpenAI ([OpenAI, 2024](#)).

- **Dataset:** openai/MMMLU
- **Implementation:** 0-shot without chat template. Seven language subsets - Arabic (AR\_XY), German (DE\_DE), Spanish (ES\_LA), French (FR\_FR), Japanese (JA\_JP), Korean (KO\_KR), Chinese (ZH\_CN).
- **Parsing:** Same as MMLU.
- **Scoring:** Reports average across all 7 languages.

## MGSM

From Shi et al., Language Models are Multilingual Chain-of-Thought Reasoners ([Shi et al., 2023](#)).

- **Dataset:** juletxara/mgsm
- **Implementation:** 5-shot examples from training dataset per language. Six languages - Spanish (es), French (fr), German (de), Chinese (zh), Japanese (ja), Russian (ru).
- **Prompt:** Same as GSM8K.
- **Parsing:** Same as GSM8K.
- **Generation:** Greedy decoding, max 4096 tokens.
- **Scoring:** Reports average loose accuracy across all 6 languages. Both strict and loose accuracy per language.

## D Multilingual Vision Evaluations

We provide detailed per-language multilingual vision evaluation results in Table 16. All multilingual benchmarks were translated from English into Arabic, Chinese, French, German, Italian, Japanese, Korean, Portuguese, and Spanish using GPT-4.1-mini.

Language	SmoLVM2-2.2B	InternVL3.5-2B	Qwen3-VL-2B	Qwen2.5-VL-3B	LFM2-VL-3B
# Total Params	2.2B	2.3B	2.1B	3.75B	3.0B
<i>Multilingual MMBench</i>					
Arabic	21.56	60.78	62.33	69.87	73.16
Chinese	50.30	73.16	74.29	75.41	74.37
French	53.07	71.95	72.47	75.58	77.14
German	47.62	72.29	72.47	75.50	78.18
Italian	47.01	70.65	70.65	75.15	77.14
Japanese	40.78	68.92	69.26	73.94	74.98
Korean	28.23	68.23	67.97	71.95	74.46
Portuguese	52.29	71.86	63.64	74.72	76.36
Spanish	55.76	72.47	72.64	76.88	76.80
<b>Average</b>	44.07	70.03	69.52	74.33	75.84
<i>MMMB</i>					
Arabic	43.69	70.10	71.01	73.74	81.31
Chinese	64.75	79.49	77.73	79.90	80.40
French	64.49	77.32	73.13	78.74	82.63
German	59.44	75.66	75.66	78.84	82.63
Italian	61.06	76.57	73.69	78.43	82.22
Japanese	56.67	75.40	74.14	76.26	80.71
Korean	46.87	73.03	72.47	75.45	80.86
Portuguese	59.80	77.02	72.58	77.98	81.06
Spanish	63.64	78.18	76.57	79.04	81.87
<b>Average</b>	57.82	75.86	74.11	77.60	81.52

**Table 16:** Performance of 2–4B VLMs on multilingual MMBench and MMMB. All results are obtained using VLMEvalKit (Duan et al., 2024).