LiquidAI LFM2: The Non-Transformer Model Worth Running Locally

February 25, 2026 · by Mark Bartlett

Download this guide as PDF

Quick Answer: LiquidAI's LFM2 is a hybrid architecture that combines short convolutions, grouped query attention, and mixture-of-experts routing. The flagship LFM2-24B-A2B has 24 billion total parameters but only activates 2.3 billion per token, meaning it fits in 32GB RAM at Q4 quantization (13.5GB file size) and runs at 112 tok/s on AMD CPU. It's not a transformer and not a pure state-space model. It's a novel architecture with day-one GGUF support for llama.cpp and Ollama. The smaller LFM2.5-1.2B-Thinking variant hits 235 tok/s on CPU and runs in the browser via WebGPU. Benchmarks are competitive with models 2-3x its active parameter count, though community testing of the 24B is still thin. Worth trying if you want to see what post-transformer local inference looks like. Not yet a replacement for Qwen or Llama as your daily driver.

Related: Beyond Transformers: 5 Architectures · VRAM Requirements · Model Formats Explained · What Can You Run on 8GB VRAM · Planning Tool

Every open-weight model you've downloaded in the past two years has been a transformer. Llama, Qwen, Mistral, DeepSeek, Phi, Gemma. Different sizes, different training data, different fine-tunes, same fundamental architecture. Attention layers stacked on attention layers, with a KV cache that grows linearly with context length.

LiquidAI's LFM2 is not that. It's a hybrid model built from convolutions, a handful of attention layers, and mixture-of-experts routing. The flagship LFM2-24B-A2B packs 24 billion total parameters but only activates 2.3 billion per token. It ships as GGUF with day-one llama.cpp support. It fits in 32GB of RAM.

Whether that matters to you depends on what you care about: raw benchmark scores (where it's competitive but not dominant), inference speed (where the small active parameter count pays off), or the question of whether transformers are the only architecture worth running locally.

What LFM2 actually is

LFM2 comes from LiquidAI, an MIT spinoff founded in 2023. Their original research was in Liquid Time-Constant Networks, a type of continuous-time neural network inspired by biological neurons. LFM2 is the commercial product of that research, evolved into something practical.

The architecture has three components:

Gated short convolutions. These are the workhorse blocks. Each one takes an input, runs it through a short convolution with multiplicative gates, and produces an output. They process sequential information with fixed memory cost regardless of sequence length. In the 24B model, 30 of the 40 layers are convolution blocks.

Grouped query attention (GQA). The same attention mechanism used in Llama and most modern transformers, but used sparingly. Only 10 of 40 layers are attention blocks. These handle the long-range dependencies that convolutions miss but the KV cache stays smaller because there are fewer attention layers.

Mixture of experts (MoE). The 24B model has 64 experts per MoE block with top-4 routing, meaning only 4 of 64 experts fire for any given token. This is why 24B total parameters yields only 2.3B active. The first two layers stay dense for training stability. Everything else is routed.

It doesn’t fit neatly into existing categories. It’s not a transformer (majority of layers are convolutions). It’s not a pure state-space model like Mamba or RWKV (it still uses attention). It’s not a standard MoE like Mixtral (the base blocks aren’t transformer layers). LiquidAI calls the underlying framework “Linear Input-Varying operators,” derived from their Liquid Time-Constant Networks research.

Why local users should care

Two things about this architecture matter for running inference on consumer hardware:

Smaller KV cache. A standard transformer builds a KV cache entry for every attention layer at every token position. LFM2-24B has only 10 attention layers instead of the 40+ you’d see in a comparable transformer. The convolution blocks use fixed-size state that doesn’t grow with sequence length. Less memory eaten by KV cache means more memory available for the model weights themselves.

Active parameter efficiency. At 2.3B active parameters per token, the per-token compute is comparable to a 2-3B dense model. The 24B of total parameters means the model “knows” more (more expert knowledge stored in weights), but each forward pass only uses a fraction. For single-user local inference, that means faster output.

The full LFM2 lineup

LiquidAI has released models across a wide range of sizes. The family splits into dense models and MoE models:

Model	Total params	Active params	Type	GGUF available
LFM2-350M	350M	350M	Dense	Yes
LFM2-700M	700M	700M	Dense	Yes
LFM2-1.2B	1.2B	1.2B	Dense	Yes
LFM2-2.6B	2.6B	2.6B	Dense	Yes
LFM2-8B-A1B	8.3B	1.5B	MoE	Yes
LFM2-24B-A2B	24B	2.3B	MoE	Yes

The dense models scale from 350M to 2.6B. These are straight inference: every parameter fires on every token. The MoE models (8B-A1B and 24B-A2B) activate only a fraction per token.

All models are trained on at least 10 trillion tokens (the 24B is still training at 17T and counting). The training mix is roughly 55% English, 25% multilingual, 20% code. Licensing is Apache 2.0-based, free for companies under $10M revenue. Commercial license required above that.

The LFM2.5 series extends the base models with additional post-training:

Model	What it adds	Standout feature
LFM2.5-1.2B-Instruct	Instruction tuning	General-purpose chat at 1.2B
LFM2.5-1.2B-Thinking	Chain-of-thought reasoning	88% on MATH-500 at 1.2B params
LFM2.5-VL-1.6B	Vision-language	Image understanding under 2B params

Running LFM2 locally

GGUF and llama.cpp

The 24B model ships as GGUF with multiple quantization options:

Quantization	File size	Minimum RAM
Q4_0	13.5 GB	~16 GB
Q4_K_M	14.4 GB	~18 GB
Q5_K_M	16.9 GB	~20 GB
Q6_K	19.6 GB	~24 GB
Q8_0	25.4 GB	~30 GB
F16	47.7 GB	~52 GB

LiquidAI says the model is designed to "fit in 32GB of RAM," which checks out at Q4_K_M (14.4GB file + overhead for KV cache and runtime). If you have 32GB of system RAM and a GPU with 16GB+ VRAM, you can run the Q4_K_M quant with partial GPU offloading. With a 24GB GPU (RTX 3090/4090), Q4_0 or Q4_K_M should fit entirely in VRAM.

Quick start with llama.cpp:

```
llama-cli -hf LiquidAI/LFM2-24B-A2B-GGUF
```

With Ollama:

```
ollama run lfm2
```

LFM2 is in the Ollama library with the smaller variants. The 24B GGUF can also be imported manually.

Speed benchmarks

LiquidAI's published numbers for the 24B-A2B:

Hardware	Decode speed	Notes
AMD Ryzen AI Max+ 395 (CPU)	112 tok/s	Q4_K_M, llama.cpp
NVIDIA H100 (GPU)	293 tok/s	bfloat16, vLLM
H100, 1024 concurrent	26,800 tok/s total	vLLM, batched serving

112 tok/s decode on CPU is fast. For context, a Llama 3.2-3B on a comparable CPU typically does 30-60 tok/s. The 24B model is matching or beating 3B models on speed because it's only activating 2.3B parameters per forward pass, and the convolution blocks are computationally cheaper than attention.

Community benchmarks are still thin for the 24B. It launched the same day I'm writing this (February 25, 2026), so independent tok/s numbers on consumer GPUs will take a few days to show up. I'll update this when r/LocalLLaMA chimes in.

The smaller models are faster

The 1.2B variants are where the speed gets interesting for low-end hardware:

Hardware	Model	Decode speed	Memory
AMD Ryzen AI Max 395+ CPU	LFM2.5-1.2B-Thinking	235 tok/s	853 MB
Apple M4 Pro (INT8)	LFM2.5-1.2B-Thinking	96 tok/s	722 MB
Samsung Galaxy S25 Ultra	LFM2.5-1.2B-Thinking	70 tok/s	720 MB
Snapdragon 8 Elite (NPU)	LFM2.5-1.2B-Thinking	82 tok/s	900 MB
AMD Ryzen NPU	LFM2.5-1.2B-Thinking	60 tok/s	1,600 MB
Browser (WebGPU)	LFM2.5-1.2B-Thinking	Runs in-browser	~900 MB

235 tok/s on CPU with a reasoning model under 1GB of RAM. That's not a typo. The convolution-heavy architecture is naturally efficient on CPUs because short convolutions map well to CPU vector instructions.

The interesting variants

LFM2.5-1.2B-Thinking

A 1.2B reasoning model that fits under 900MB. It generates chain-of-thought traces before answering, similar to how DeepSeek-R1 or Qwen3 thinking mode works but at a fraction of the size.

The benchmarks are hard to argue with at this parameter count:

Benchmark	LFM2.5-1.2B-Thinking	Qwen3-1.7B (thinking)
MATH-500	87.96%	81.92%
GSM8K	85.60%	85.60%
GPQA Diamond	37.86%	36.93%
IFEval	88.42%	71.65%
MMLU-Pro	49.65%	56.68%

It beats Qwen3-1.7B on math, ties on GSM8K, and wins on instruction following. Qwen wins on MMLU-Pro (general knowledge). Both are thinking models, but LFM2.5 does it with 30% fewer parameters.

You can run this in your browser right now via LiquidAI's WebGPU demo. No install, no backend. It runs client-side in Chrome, Edge, or Safari.

LFM2.5-VL-1.6B

A vision-language model built on the LFM2 1.2B backbone with a SigLIP2 vision encoder. It handles image understanding tasks (OCR, visual QA, diagram reading) at 1.6B total parameters.

LiquidAI claims 2x faster inference than comparable VLMs. The model processes images at native resolution up to 512x512, splitting larger images into patches. A typical photo generates 96-240 tokens of visual context.

The practical application here is on-device vision. People in the community have been using it for real-time video captioning via webcam, and the WebGPU demo streams from your camera with live captions running entirely on your hardware. Think security camera analysis, document scanning, or accessibility tools that never leave your machine.

How it compares

vs Qwen3-30B-A3B

The closest MoE competitor. Qwen3-30B-A3B has 30.5B total parameters and 3.3B active, compared to LFM2-24B's 24B total and 2.3B active. Qwen activates 43% more parameters per token, which should mean better quality but slower inference.

On throughput, LiquidAI claims the 24B beats both Qwen3-30B-A3B and gpt-oss-20b on a single H100, hitting 26,800 tokens/sec at 1,024 concurrent requests. The smaller active parameter path and cheaper convolution blocks give it an edge on raw speed.

On quality: the 24B is still an early checkpoint (17T tokens, training ongoing). LiquidAI hasn't published detailed benchmark tables for it yet. The scaling curves show log-linear improvement from 350M to 24B, which is promising, but we'll need the community to run independent evals before drawing conclusions. Based on the 8B-A1B benchmarks (which are published), the architecture is competitive with models 2-3x its active parameter count on math and reasoning, but weaker on general knowledge (MMLU-Pro) and creative writing.

vs Llama/Qwen (standard transformers)

The fundamental difference: transformers are tested, optimized, and understood. Every quantization tool, every inference engine, every fine-tuning framework has been built for transformers first. LFM2 has llama.cpp support and GGUF, which covers the basics, but you won't find LFM2 LoRA adapters on HuggingFace. The fine-tuning ecosystem doesn't exist yet.

If you need a reliable daily driver for local inference, Llama 3.3 or Qwen3 is still the safer choice. LFM2 is for people who want to see what comes after transformers, not people who need production stability today.

vs Mamba and RWKV (pure SSMs)

Mamba and RWKV are pure state-space / recurrent models: no attention at all, constant memory usage regardless of context length. LFM2 is a hybrid that keeps some attention layers (10 of 40 in the 24B). This means LFM2's memory does grow with context (from the KV cache of those 10 attention layers), but much less than a full transformer.

The trade-off: Mamba's constant memory is elegant, but pure SSM models have struggled to match transformer quality on tasks that require precise recall of information from earlier in the context. LFM2's approach is pragmatic: use attention where it matters most, use convolutions everywhere else.

vs Phi-4 Mini

Both compete for the "small but smart" slot. Phi-4 Mini is a dense 3.8B transformer. LFM2-8B-A1B is an 8.3B MoE with 1.5B active. At the active parameter level, LFM2 is smaller, but the total parameter count gives it access to more expert knowledge. Phi tends to win on coding benchmarks. LFM2 tends to win on math and instruction following. Both are optimized for edge deployment.

The honest assessment

The architecture works. It's not a research prototype. The 1.2B models are fast and competitive. The WebGPU browser inference is real and usable. The 24B fits in consumer RAM with good quantization. The speed advantage from small active parameter counts is measurable.

LiquidAI also has actual partnerships with AMD, Qualcomm, Samsung, and Ollama. This isn't a paper on arXiv. It's shipping in production frameworks on production hardware.

But the 24B model is an early checkpoint. Training is ongoing at 17T tokens with more planned. The eventual LFM2.5-24B-A2B (with reinforcement learning and post-training) will be the real product. What's available now is a preview.

Community adoption is thin. The LFM2-8B-A1B has been tested more widely, and the reception is mixed: the Dubesor benchmark site describes it as "still a very weak model" relative to expectations, performing around Ministral 8B level on practical tasks. The official benchmarks tell a better story, but there's a gap between benchmark scores and vibes-based evaluation that the community hasn't fully resolved.

The fine-tuning ecosystem doesn't exist. No LoRA adapters, no community fine-tunes, no RLHF variants. If you want to customize the model for your use case, you're on your own. And creative writing is a weak spot: the 8B-A1B scored 44% on Creative Writing v3 versus 69% for Gemma-3-4B. If your use case is chat or creative work, this isn't the model for you.

So who should try it? If you're interested in alternative architectures and want to see what post-transformer inference looks like on your hardware, LFM2 is the most accessible option available. The GGUF support means you can try it with the tools you already have. The 1.2B-Thinking variant is worth running just to see a sub-1GB reasoning model hit 235 tok/s on CPU.

If you need a reliable local model for daily use, stick with Qwen3 or Llama for now. Come back when the 24B training finishes and the community has had a month to test it.

> Want to figure out which quant fits your GPU? Use the VRAM Calculator. For background on GGUF and quantization formats, see Model Formats Explained. And for the bigger picture on non-transformer architectures, read Beyond Transformers: 5 Architectures for Your $50 Mini PC.

Related guides

- Beyond Transformers: 5 Architectures for Your $50 Mini PC
- VRAM Requirements for Local LLMs
- Model Formats Explained: GGUF vs GPTQ vs AWQ vs EXL2
- What Can You Run on 8GB VRAM
- Qwen3 Complete Guide
- MoE Models Explained

Source: https://insiderllm.com/guides/liquidai-lfm2-local-guide/