LiquidAI LFM2: The First Hybrid Model Built for Your Hardware

February 26, 2026 · by Mark Bartlett

Download this guide as PDF

> **Quick Answer:** LiquidAI's LFM2-24B-A2B is a hybrid model (not a transformer) that packs 24 billion parameters but only activates 2.3 billion per token. The Q4_K_M quant is a 14.4GB file that fits in 32GB RAM and decodes at 112 tok/s on an AMD Ryzen AI CPU -- faster than most 3B transformers. It runs on llama.cpp, Ollama (v0.17.1-rc0+ required for MoE models), and LM Studio with standard GGUF files. The architecture mixes convolutions, sparse attention, and MoE routing. It's faster than Qwen3-30B-A3B at serving throughput on a single H100, and the smaller LFM2.5-1.2B-Thinking variant hits 235 tok/s on CPU in under 900MB. Creative writing and general knowledge are weak spots. Worth trying for speed-sensitive local inference. Not yet a replacement for Qwen or Llama as your daily driver.

More on this topic: Beyond Transformers: 5 Architectures | VRAM Requirements | Model Formats Explained | MoE Models Explained | What Can You Run on 8GB VRAM

Every model you've pulled through Ollama or loaded in LM Studio is a transformer. Llama, Qwen, Mistral, DeepSeek, Phi, Gemma – different training data, different sizes, same fundamental architecture. Attention all the way down, with a KV cache that scales with context length.

LFM2 is not a transformer. LiquidAI built it from short convolutions, a handful of attention layers, and mixture-of-experts routing. The flagship LFM2-24B-A2B has 24 billion total parameters, activates 2.3 billion per token, and decodes at 112 tok/s on a Ryzen AI CPU. The Q4 GGUF file is 14.4GB. It has day-one support in llama.cpp, Ollama, and LM Studio.

The question for local users isn't whether the architecture is interesting (it is). The question is whether you should download it. That depends on what you need it for and whether 32K context is enough for your work.

What makes LFM2 different from every other model

The architecture in plain terms

A standard transformer is a stack of attention layers. Each layer looks at every token that came before the current one, which gives it excellent recall but makes the KV cache grow with every token you add to the context.

LFM2 replaces most of those attention layers with gated short convolutions. In the 24B model, 30 of 40 layers are convolution blocks. Only 10 are grouped query attention (GQA) – the same attention mechanism Llama uses, but used sparingly. On top of that, the model uses mixture-of-experts routing with 64 experts per block, only 4 of which fire per token.

Three practical consequences:

1. **Smaller KV cache.** Only 10 attention layers build KV cache entries instead of 40+. The convolution blocks use fixed-size state that doesn't grow with context length. You use less memory on long conversations.

2. **Fast per-token compute.** At 2.3B active parameters, each forward pass does the work of a 2-3B dense model despite having 24B of total expert knowledge stored in weights.

3. **CPU-friendly.** Short convolutions map well to CPU vector instructions. The 24B model hits 112 tok/s decode on an AMD Ryzen AI Max+ 395 CPU. Most 3B transformers top out at 30-60 tok/s on comparable hardware.

LiquidAI calls the underlying framework "Linear Input-Varying operators," derived from their Liquid Time-Constant Networks research (an MIT spinoff, $250M raised). The company has partnerships with AMD, Qualcomm, Intel, and Samsung for hardware-optimized deployment. This isn't a research paper. It ships in GGUF.

How it fits (and doesn't) into existing categories

LFM2 is not a transformer (most 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).

vs Mamba and RWKV. Pure SSM and recurrent models use no attention at all, giving them constant memory usage regardless of context length. LFM2 keeps 10 attention layers out of 40, so its KV cache does grow with context, 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 requiring precise recall from earlier in the context. LFM2's approach is pragmatic – use attention where it matters, 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. Phi tends to win on coding benchmarks. LFM2 tends to win on math and instruction following. Both target edge deployment, but LFM2's MoE routing gives it access to more expert knowledge at the same inference cost.

The model family

LFM2 spans from 350M to 24B parameters. The dense models are conventional single-path inference. The MoE models activate a fraction of their parameters per token.

Model	Total params	Active params	Type	GGUF file (Q4)	Context
LFM2-350M	350M	350M	Dense	~200 MB	32K
LFM2-700M	700M	700M	Dense	~400 MB	32K
LFM2-1.2B	1.2B	1.2B	Dense	~700 MB	32K
LFM2-2.6B	2.6B	2.6B	Dense	~1.5 GB	125K
LFM2-8B-A1B	8.3B	1.5B	MoE	~4.5 GB	32K
LFM2-24B-A2B	24B	2.3B	MoE	13.5 GB	32K

All models are trained on at least 10 trillion tokens (the 24B is at 17T and counting). The training mix is roughly 55% English, 25% multilingual, 20% code. Licensing is LFM Open License v1.0, which is free for companies under $10M revenue. Commercial license required above that.

The LFM2.5 series adds post-training on top of the base models:

Model	What it adds	Why it matters
LFM2.5-1.2B-Instruct	Instruction tuning, 28T tokens	General chat, 116 tok/s on CPU
LFM2.5-1.2B-Thinking	Chain-of-thought reasoning	88% on MATH-500, under 900MB
LFM2.5-VL-1.6B	Vision-language	Image understanding at 696MB (Q4)
LFM2.5-Audio-1.5B	Audio input/output	8x faster audio decoding vs LFM2
LFM2.5-1.2B-JP	Japanese optimization	JMMLU: 50.7

The ones worth paying attention to

LFM2-24B-A2B is the flagship. It's the biggest model, the most capable, and the one you'll compare against Qwen3-30B-A3B. The tradeoff: it's also the newest (released February 25, 2026) and the least community-tested.

LFM2.5-1.2B-Thinking punches way above its weight. A reasoning model under 900MB that generates chain-of-thought traces before answering, similar to DeepSeek-R1 or Qwen3 thinking mode but at a fraction of the size. It runs at 235 tok/s on a Ryzen AI CPU, 82 tok/s on a phone NPU, and in your browser via WebGPU with no install. If you want to see what LFM2 can do without committing to a 14GB download, start here.

Head-to-head with Qwen3-1.7B (the closest competitor in this size class):

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 by a wide margin. Qwen wins on MMLU-Pro (general knowledge). Both are thinking models, but LFM2.5 does it with 30% fewer parameters. You can try it right now via LiquidAI's WebGPU demo – no install, runs client-side in Chrome, Edge, or Safari.

LFM2.5-VL-1.6B is the vision model, built on the LFM2 1.2B backbone with a SigLIP2 vision encoder. It processes images at native resolution up to 512x512, splitting larger images into non-overlapping patches. A typical photo generates 96-240 tokens of visual context. At Q4 quantization, the entire model is 696MB. People are already using it for live webcam captioning and security camera analysis – the WebGPU demo streams from your camera and generates captions locally in the browser.

How to run it

llama.cpp (direct GGUF)

The fastest path to running LFM2-24B:

```
# Pulls the default quant and runs it
llama-cli -hf LiquidAI/LFM2-24B-A2B-GGUF

# Specify a quant
llama-cli -hf LiquidAI/LFM2-24B-A2B-GGUF:Q4_K_M
```

LFM2 support landed in llama.cpp build b8145+ with the `lfm2moe` architecture label. If your llama.cpp is older than that, update it.

For the smaller models:

```
# 8B MoE
llama-cli -hf LiquidAI/LFM2-8B-A1B-GGUF

# 1.2B Thinking
llama-cli -hf LiquidAI/LFM2.5-1.2B-Thinking-GGUF

# Vision model
llama-cli -hf LiquidAI/LFM2.5-VL-1.6B-GGUF:Q4_0
```

Ollama

The smaller LFM2 models are in the Ollama library:

```
ollama run lfm2
```

For the 24B or LFM2.5 variants, pull directly from HuggingFace:

```
ollama run hf.co/LiquidAI/LFM2-24B-A2B-GGUF
ollama run hf.co/LiquidAI/LFM2.5-1.2B-Instruct-GGUF
```

Compatibility warning: Ollama v0.17.0 (the current stable release) fails on LFM2 MoE models with a `missing tensor 'output_norm.weight'` error. You need v0.17.1-rc0 or later. Check your version with `ollama --version` and grab the release candidate from GitHub if needed. The dense models (350M through 2.6B) work on v0.17.0 fine. The MoE models (8B-A1B and 24B-A2B) require the fix.

LM Studio

LM Studio supports GGUF files, so LFM2 works through the standard model import flow. Search for "LFM2" in the model browser or download a GGUF from HuggingFace and drag it in. The MLX backend on Mac may or may not work (MLX support for LFM2 is listed on HuggingFace, but MLX-format weights aren't as widely tested as GGUF).

Recommended generation parameters

LiquidAI recommends these settings for best output quality:

```
temperature: 0.3
min_p: 0.15
repetition_penalty: 1.05
```

These are lower temperature and tighter sampling than most Llama/Qwen defaults. The model tends to ramble or repeat at higher temperatures.

Memory requirements

The "24B model with 2B active parameters" framing is accurate for compute, but misleading for memory. All 24 billion parameters still need to live in RAM or VRAM. The 2.3B active count means inference is fast, not that the model is small.

LFM2-24B-A2B GGUF file sizes

Quantization	File size	Minimum RAM	GPU VRAM (full offload)
Q4_0	13.5 GB	~16 GB	16 GB (tight)
Q4_K_M	14.4 GB	~18 GB	16-24 GB
Q5_K_M	16.9 GB	~20 GB	24 GB
Q6_K	19.6 GB	~24 GB	24 GB
Q8_0	25.4 GB	~30 GB	24 GB (tight)
F16	47.7 GB	~52 GB	48 GB+

Add 2-3GB overhead for KV cache and runtime. A 32GB system runs Q4_K_M with room to spare. A 16GB GPU (RTX 4060 Ti 16GB, RTX 5060 Ti 16GB) can fit Q4_0 with the model fully offloaded. An RTX 3090 or 4090 with 24GB handles Q6_K.

Comparison: LFM2-24B vs Qwen3.5-35B-A3B

Both are MoE models targeting the “big model knowledge, small model speed” niche:

	LFM2-24B-A2B	Qwen3.5-35B-A3B
Total parameters	24B	35B
Active parameters	2.3B	3B
Q4 GGUF size	13.5-14.4 GB	~22 GB
Architecture	Hybrid (conv + attn + MoE)	Transformer + MoE
Context length	32K	262K
Minimum RAM	~16 GB	~24 GB
llama.cpp support	Yes (b8145+)	Yes
Ollama support	Yes (v0.17.1-rc0+)	Yes

LFM2-24B is smaller and faster per token. Qwen3.5-35B-A3B has a much longer context window (262K vs 32K) and more community testing. If you need long documents, Qwen wins by default. If you want fast inference on shorter tasks and have less RAM, LFM2 fits where Qwen doesn’t.

Speed benchmarks

LFM2-24B-A2B

Hardware	Framework	Decode speed	Notes
AMD Ryzen AI Max+ 395 (CPU)	llama.cpp	112 tok/s	Q4_K_M
Apple M4 Pro (CPU, INT8)	Cactus	27 tok/s	INT8 quant
Samsung Galaxy S25 Ultra (NPU)	Qualcomm SDK	35.4 tok/s	Snapdragon 8 Elite
NVIDIA H100 (GPU)	vLLM	293 tok/s	BF16
H100, 1024 concurrent	vLLM	26,800 tok/s total	Batched serving

112 tok/s on CPU is the headline. For context, Llama 3.2 3B on a similar CPU does 30-60 tok/s. The 24B model is keeping pace with models a tenth its size because the active parameter path is so small and convolutions are cheap on CPUs.

27 tok/s on Apple M4 Pro is more modest. The hybrid architecture doesn't benefit from MLX's transformer-specific optimizations the way Qwen or Llama does. On Mac, expect roughly Ollama-tier speed until MLX adds dedicated LFM2 kernels.

Smaller models

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
Snapdragon Gen4 (NPU)	LFM2.5-1.2B	82 tok/s	900 MB
Galaxy S25 Ultra (CPU)	LFM2.5-1.2B	70 tok/s	719 MB
AMD Ryzen AI 9 HX 370 (CPU)	LFM2.5-1.2B	116 tok/s	856 MB
Browser (WebGPU)	LFM2.5-1.2B-Thinking	Real-time	~900 MB

235 tok/s on CPU for a reasoning model under 900MB. You can run the 1.2B-Thinking model in your browser right now through LiquidAI's WebGPU demo. No install, no backend. It runs client-side.

What it's good at (and not)

Where LFM2 performs

Math and reasoning. The 8B-A1B scores 84.38% on GSM8K and 74.2% on MATH-500 with only 1.5B active parameters. Those numbers beat Llama 3.2 3B (75.2% and 41.2% respectively). The 1.2B-Thinking variant pushes to 88% on MATH-500 with chain-of-thought.

Instruction following. LFM2-8B-A1B scores 77.58% on IFEval. The 1.2B-Thinking version hits 88.42%. These are strong numbers for models this size.

Speed-sensitive tasks. If your workflow involves lots of short requests (API routing, classification, entity extraction, summarization), the small active parameter count means you get responses faster per token than a comparable transformer.

On-device and edge. The vision model (LFM2.5-VL-1.6B) at Q4 is 696MB. People are running it for live webcam captioning and security camera analysis entirely on-device. A 696MB vision model that runs on a phone is a different category of tool than a 4GB VLM that needs a GPU.

Where it falls short

Creative writing. The 8B-A1B scored 44.2% on Creative Writing v3 compared to 69% for Gemma-3-4B. The convolution-heavy architecture seems to trade fluency for efficiency. If chat or storytelling is your use case, stick with Qwen or Llama.

General knowledge. MMLU-Pro for the 8B-A1B is 37.42% versus Qwen3-4B's 52.31%. The model knows less trivia per parameter than transformer competitors. The 24B should close this gap but detailed benchmarks aren't published yet.

Context length. 32K tokens is the ceiling for all LFM2 models. Qwen3.5 offers 262K. If you're doing RAG over large documents or analyzing codebases, 32K is a real constraint.

Community ecosystem. No LoRA adapters on HuggingFace. No community fine-tunes worth running. If you want to customize LFM2 for your use case, LiquidAI provides fine-tuning notebooks for SFT and DPO, but the broader ecosystem is thin compared to Llama or Qwen.

The 24B is early. It launched February 25, 2026 with training ongoing at 17 trillion tokens. The eventual post-trained version (LFM2.5-24B equivalent) will be the real product. What's available now is a solid checkpoint, not the final model. The 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. 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.

Why this matters for local AI

LFM2 is the first non-transformer model with real local inference support. Not a paper, not a demo – actual GGUF files you can pull through Ollama and run on your hardware today.

That matters beyond LFM2 itself. If hybrid architectures prove competitive at scale (and the 24B scaling curves are encouraging), future models will be faster and cheaper to run locally. The convolution blocks that make LFM2 fast on CPUs could mean usable inference on hardware that can't touch a transformer of comparable quality. State-space components that don't grow KV cache with context length could make long conversations cheaper in memory.

None of that is guaranteed. Transformers have years of optimization in every inference engine, every quantization tool, every fine-tuning framework. LFM2 has llama.cpp support and a handful of Ollama tags. The gap in ecosystem maturity is massive.

But if you run AI locally and want to see what the next generation of architecture looks like, the barrier is a 14GB download and a single command. Mamba and RWKV never got this close to plug-and-play.

The bottom line

Try the 1.2B-Thinking first. It runs in your browser via WebGPU, needs under 900MB, and is the best way to see what the architecture can do without committing to anything. If you like what you see, download the 24B-A2B at Q4_K_M (14.4GB) and run it on whatever hardware you have with 18GB+ of RAM.

Don't replace your daily driver yet. Qwen3 and Llama are still better at general knowledge and creative work, and Qwen's 262K context window dwarfs LFM2's 32K. Where LFM2 wins is speed on math and instruction-following tasks, especially on CPU.

Watch this space. LiquidAI is well-funded ($250M), actively training larger models, and has hardware partnerships with AMD, Qualcomm, Intel, and Samsung. The LFM2.5 post-trained series is already stronger than the base LFM2 models. When the 24B gets its LFM2.5 treatment, it could close the gap with Qwen on quality while keeping its speed advantage.

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
- MoE Models Explained
- Qwen3 Complete Guide

Get notified when we publish new guides.

Subscribe — free, no spam
