A 74B model at 128K, on one used 3090

November 9, 2026 · by Mark Bartlett

Download this post as PDF

InsiderLLM Weekly issue 29 – November 9, 2026

NVIDIA built Nemotron 3.5 Bolt to fit 24 GB at 128K context, then left 24 GB cards off its own recipe table. So I put it on my 3090 and filled the context. First, the week's news.

New This Week

- **Nemotron 3.5 Bolt on One RTX 3090: 24 GB Holds at 128K** (2026-11-03). 128K context loads with 2,310 MiB spare, about 90 tok/s decode, 82.42 at 100K deep. The main piece, below.

Updated: What Can You Run on 24GB VRAM has new picks, including Nemotron Bolt at 128K, and the OpenClaw hardware guide dropped the Mac from its $1,000+ tier. Plus GPU-price and command fixes across 51 guides.

Quick Hits

- **Used 3090s got dearer.** Our pull of 226 eBay sales from October 25 to November 6 puts the median at $1,580, up from $1,499 in mid-October. If you're buying one, the last week ran higher still. (our price record)
- **llama.cpp 0.9.0 runs Clover, minus the diffusion.** Clover is a block-diffusion model built from Qwen3.6-35B-A3B, with GGUFs out. llama.cpp runs it one token at a time, so it fits where the 35B MoE fits, without the parallel-decode speedup. (release notes; Clover; ours: the 35B MoE on a 3090)
- **Ripple-65B is open, and dense.** Ricursive's coding model has run in llama.cpp since November 5, and its Q4_K_M file is 38.8 GB. Every token reads all of it, so on a 24 GB card the part in system RAM sets your speed. (Ricursive; llama.cpp support)
- **Kimi K3.1 Pro weights are out.** Moonshot put its 2.8-trillion-parameter model on Hugging Face. That's server-rack territory, and it doesn't change what a home rig can run. (Moonshot; ours: open weights you can't run)

Does 24 GB Hold? Yes.

Nemotron 3.5 Bolt is a 74B model with 9B active per token, in a 22.02 GB file. NVIDIA's model card says it was designed to fit under 24 GB at 128K context. Then its recipe table starts at 32 GB cards and gives no number for 24 GB.

On one RTX 3090 with llama.cpp v0.7.0, 128K loaded with 2,310 MiB to spare. Decode sat near 90 tok/s. With 102,400 tokens already in the cache, it still decoded at 82.42. The reason it fits: only 8 of its 88 layers keep a KV cache, so a full 128K costs 1,024 MiB.

The setting that costs you is the usual tight-card advice, a q8_0 KV cache. At short context it's 1.8% slower than f16. At 100K deep it decoded 60.61 tok/s against f16's 82.42, and it saved 479.99 MiB the card didn't need. On this model, use f16.

Quality is a different story. On my 47-item intent task Bolt scored 22, the same as the Qwen3.6-35B-A3B I used as the control. The same Qwen file scored 23 on an older llama.cpp, so one item either way is noise. What you get from Bolt is the context window, not a smarter model.

Two side results came out of the same run. llama.cpp v0.7.0 processes prompts 14-16% faster than v0.4.0, with decode flat. It's worth the upgrade, but the server's default port moved to 9931. And the new `--moe-cache-mib` flag, shown in its pull request at 1.6-2.2× decode, gave me +5% on a 35B MoE, at the cost of some prompt speed.

I locked 13 predictions in a commit before anything ran. Five were wrong, and they're all in the article.

📖 **Full breakdown here.**

📊 **Every number we've measured, with configs and raw logs: insiderllm.com/benchmarks/.**

That's the week. If long context on a 24 GB card feels slow, check your KV cache type before you blame the model.

New here, reading this on the web? Subscribe and the next one lands in your inbox.

— Mark, InsiderLLM

What context length do you actually use day to day? Reply, or hit me at hello@insiderllm.com. I read everything.

A weekly email with every new guide and measured benchmark.

Subscribe — free, no spam

Source: https://insiderllm.com/blog/newsletter-2026-11-09/

Free guides for running AI locally