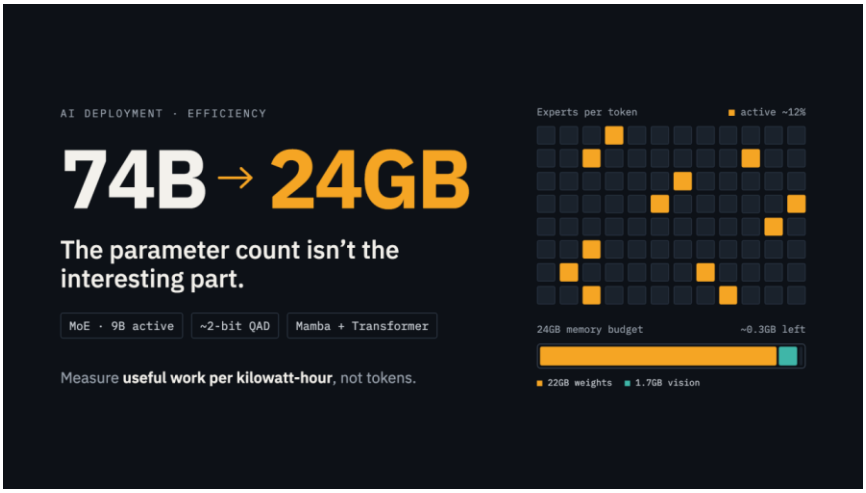



Raj Reddy: Founder | Building AI infrastructure | Physical AI systems

October 19, 2026

In early October 2026 NVIDIA released Nemotron 3.5 Bolt, a 74B parameter model designed to run within 24GB of memory. The headline is the big number. What matters more is how they got there and what it says about where AI deployment is going.

Three things make it work.

First, it's a MoE model. Only about 9 billion of the 74B parameters are active for each token, roughly 12%. You get the capacity of a large model while paying the compute cost of a much smaller one. But fewer active parameters don't mean less memory. All 74 billion parameters still have to sit in memory, even if only 9 billion are used for each token.

Second, it wasn't trained small. NVIDIA started from Nemotron 3.5 Super VL (120B total, 12B active), which is also where Bolt gets its vision capability. They compressed it using Star Elastic, their router-guided architecture search, then quantized the experts to roughly 2 bits and used quantization-aware distillation to recover accuracy. At exactly 2 bits, 74 billion parameters would need 18.5GB in theory. The actual weights come to about 22GB

because some components stay at higher precision. Keeping model quality intact at that level of compression is the real engineering achievement.

Third, it's a hybrid of Mamba and Transformer layers, with a 256K context window. People will skip over this, but they shouldn't. The usual catch with "fits in 24GB" is that the KV cache grows with context length and eats into available memory. Mamba layers don't build a conventional KV cache, although the attention layers still do. That makes long-context inference more memory-efficient than in a comparable pure Transformer.

Where I'd be careful

Fitting the model isn't the same as running your workload. The 22GB of weights plus a 1.7GB vision projector come to nearly the full 24GB. NVIDIA's own design target was batch size one at 128K context, which is a single-user configuration. Concurrency and the full 256K window are where the budget runs out.

The question isn't "can I load it?" It's "can I run my actual workload at acceptable quality, latency, and concurrency?"

It's also optimized for NVIDIA's RTX Spark unified-memory platform, where NVIDIA reports about 84 tokens per second. That's a different setup from a conventional 24GB discrete GPU, so don't assume the performance numbers transfer directly.

Independent evaluation is still limited, particularly for sustained production workloads. NVIDIA's figures are promising, but I'd want third-party measurements of latency, long-context performance, and energy per completed task before making deployment decisions.

Why I care about this

I've seen two sides of the same problem. One is squeezing perception and decision-making onto in-vehicle compute with strict power and thermal budgets. The other is AI data centers, where the constraint is increasingly megawatts, not GPUs.

Both end up asking the same question: how much useful work do you get per watt?

The AI industry still puts too much emphasis on parameter counts and leaderboard scores. Those describe the model, not the system running it.

In production, I want to know the time to first token, sustained throughput, performance degradation with longer contexts, cost per million tokens at realistic utilization, and the energy required to complete a task.

That last one is what people often miss. An agent that finishes a job correctly in 2,000 tokens and three tool calls beats one that generates 20,000 tokens faster but gets it wrong. Tokens per second is a throughput metric. Useful work per kilowatt-hour is a business metric.

MoE makes this more complicated, not less. You reduce arithmetic operations, but memory movement and expert routing can become bottlenecks. You only capture the gains when the model, inference engine, hardware, and scheduler are optimized together, and most deployments today aren't.

Open weights change the economics, not just the politics

The weights are open, and the license permits commercial use, subject to its terms.

That gives teams a real choice. They can keep sensitive inference on their own infrastructure, fine-tune for specialized tasks, or compare fixed hardware costs against per-token API pricing.

Self-hosting isn't automatically cheaper. Idle GPUs are expensive. But having the option changes the economics and gives organizations more control over their infrastructure.

What it unlocks, with caveats

The use cases I find most interesting are industrial. Picture a maintenance agent that analyzes equipment imagery, retrieves the right manual, and suggests a fix, or a robot that interprets a scene alongside its instructions.

A smaller memory footprint makes these applications easier to prototype and potentially deploy closer to the data. Coming from automotive, though, I'll say it plainly: a capable multimodal model isn't a safety architecture.

Retrieval, permissions, monitoring, and validation still have to be engineered. Lower memory requirements make these systems possible. They don't make them dependable.

The shift underneath all this

For the last few years, the story was who could train the biggest model. Bolt is another sign that the story is shifting toward who can deploy intelligence most efficiently. The winners won't necessarily be the ones with the most parameters or the most GPUs. They'll be the ones who deliver the most useful work per kilowatt-hour.

If you're running AI inference in production, do you actually measure cost and energy per completed task, or only per token? I'd be interested to hear how teams are approaching it.