I got the result I wanted. Then I paid $4.97 to run it nine more times.

September 1, 2026 · by Mark Bartlett

Download this post as PDF

InsiderLLM Weekly issue 19 – September 1, 2026

Twenty-four cents of API credit bought me a 91-line skill file, and it made the 27B on my desk 10.6 points more accurate at a job it does hundreds of times a day. That is precisely the result I was hoping for, and I got it on the first run.

Quick Hits

- **"MoE routing is flat" is correct arithmetic on the wrong quantity.** Pool all forty layers together and the busiest 10% of experts carry **17.2%**, barely above a coin flip. Look inside a layer and the top 10% carry **42 to 55%**. 📖 The full routing trace.
- **Ornith 1.5-35B generates about 9% faster than Qwen 3.6-35B on a 3090, and loses prefill by 3%.** Peak VRAM is a tie — **14 MiB apart**, 0.06%. Both files in one session, A-B-B-A. 📖 The full A-B-B-A bench.
- **The German sovereign-AI model declares NVIDIA's architecture string in its own file header.** Soofi S 30B-A3B is `nemotron_h_moe`. The technical report says so plainly; the model card doesn't mention it. 📖 Both technical reports, read.
- **We got two dated claims wrong on our own agent-incidents page and fixed them in public on 27 August.** OpenAI's own technical report, published 26 August after our page, is what corrects us. 📖 The corrected record.

Two stories below and they're the same story on different material: what it costs to install a skill into a model you didn't train, and how little anyone has checked whether either method works twice.

The Number Was the Top of Ten Draws

I handed Claude Opus 5 twenty-four real traces from my own intent classifier and asked for procedural rules that would make the 27B more accurate. It came back with 91 lines of ordered tests and tie-breakers, and it had worked out — correctly — that my deployed prompt treats "you"

and “your” and “we” as retrieval signals when they’re just how a person addresses an assistant, so the model fires RAG at anything introspective. Real bug. I’ve stared at that prompt for months without naming it that cleanly.

Baseline on my 47-item split was 48.94%. The compiled skill scored **59.57%**. Up 10.64 points, for 24 cents.

I only know that number is fake because I paid to do it nine more times.

Same bundle, same model, byte-identical input, ten runs. They range from **42.55% to 59.57%** — six points worse than no skill at all, up to ten points better. The mean is 50.85%, so the honest effect is **+1.91 points**, and the standard deviation across the ten, which is the compiler’s own noise floor, is **4.65**. My result sits 1.88 deviations above the mean. It isn’t a better skill. It’s the top of ten draws.

The pass conditions were written down before any of it ran, which is the only reason I trust the answer: the delta had to clear 9.29 points and managed 1.91, and the paired bootstrap straddled zero at **[−4.26, +8.30]**. Twenty compiled skills across three corpus sizes, not one beating the plain prompt.

The obvious defence doesn’t work either. A validation gate on 15 items moves **6.67 points** per item against a 4.65-point noise floor. Its smallest possible step is larger than the thing it exists to detect, so it can’t separate a good skill from a lucky draw. It launders one into the other.

Nobody runs a compiler ten times, and that’s rational rather than lazy: when the thing works, nine more runs buy you a bill. Which is exactly why this survives. If you have a result you’re pleased with and you generated it once, you don’t yet know what you have.

📖 **All twenty skills, the pre-registered conditions, and the $4.97, are here.**

The Other Substrate, Measured on One Side Only

That skill doesn’t just fail to help. It **costs 1,383.9 extra tokens on every call, forever** — 651.7 prompt tokens baseline, 2,035.6 with the skill appended. Compilation was 24 cents once; the tax runs for the life of the deployment.

Macaron-V1, out of Mind Lab on 10 August, takes the other road: freeze the base, train specialist LoRA adapters, select one per turn. No prompt tax by construction — the skill lives in the adapter, not the context window — and llama.cpp already has the serving side. It costs **7.55 GB** per adapter and a training run on collected trajectories instead of one API call.

The thing I went looking for isn't in the paper. Every `±` in it is evaluation-sampling spread, five eval seeds on one already-trained adapter. **Nobody has trained the same adapter ten times on identical trajectories to see whether LoRA compilation is reproducible.** Same omission as the prompt substrate, in a more expensive material: when a compiled prompt turns out to be a draw from a distribution you rerun it for 24 cents, and when a 7.55 GB adapter is, you find out after the GPU hours.

So I'm running it. Ten adapters from one recipe, differing only by seed. Two are trained and scored as of this morning, eight to go. The two-seed spread is striking and I'm not printing it, because publishing a two-sample result in the same issue as the story above would be self-parody. It goes out at ten or it doesn't go out.

📖 **The three substrates, the tax, and what Macaron admits, are here.**

Both rigs behind the routing trace live in one file, along with most of the numbers above. The open dataset is at **118 rows** now — every flag, error bars, repetition counts, whether a display was attached, and the configs that failed to load. It was 50 rows when I last pointed you at it in July. The Ornith numbers aren't in yet; they go in on the next pass, and I'd rather they arrive late than arrive wrong. CC BY 4.0, attribution the only ask.

📖 **The dataset, and the raw JSON, are here.**

---

277 GB in a File With No Name

The load test I ran to prove llama.cpp could open the Ornith GGUF filled the root disk and took the morning brief down with it.

Three defects stacked and each one individually looked handled. The build ignored `-no-cnv` and came up interactive, so with stdin at EOF it looped, writing 2.9 MB/s into a log. My kill reported success against the wrong PID, because `cd X && nohup Y &` backgrounds the AND-list and `$!` hands you the subshell. And I trimmed the log with `head > tmp && mv`, which orphaned the inode the still-live process was writing into. It grew to exactly the free space and stopped.

`ls`, `du` and `stat` all described an innocent 1,109-byte file. `df` said zero available and was the only one telling the truth. The tell was a cleanup that freed nothing: **when a disk is full and deleting things doesn't move the number, stop deleting.** `lsof +L1` lists open files with a link count below one, naming the PID and the bytes. The kill returned 257 GiB instantly. What it cost

was two zero-byte files — the 00:05 crawler split and the 07:30 morning brief each wrote a plausible nothing and neither said a word about it.

That's the week. If you have a compiled skill, a tuned prompt or a fine-tune you're pleased with and you produced it exactly once, go and produce it again before you deploy it — right now you don't know whether you measured an improvement or a draw.

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

— Mark, InsiderLLM

Run a skill compiler more than once, or trained the same LoRA twice and compared? Reply, or hit me at hello@insiderllm.com. I read everything — and this issue is mostly things that did not work.

Source: https://insiderllm.com/blog/newsletter-2026-09-01/