NariQ Tested: Better Than IQ3, Nowhere Near BF16

November 25, 2026 · by Mark Bartlett

Download this guide as PDF

> **Quick Answer:** Nari Labs says its 10 GB Qwen3.8-27B is "BF16-level". I measured it against a full BF16 reference on text written after the model was finished. It beats the stock 3-bit file of similar size by 22%. A plain 4-bit file is still six times closer to BF16. On a 12 GB card the speed is a wash, prompts are slower, and the quality gain is real, so the question is whether it's worth running a fork for. Here's who should and who shouldn't.

📚 **More on this topic:** Quantization Explained · What Can You Run on 12GB VRAM · Bonsai 2 vs Qwen3.8-27B on an RTX 3090 · Model Formats Explained

On a 12 GB card, Qwen3.8-27B is a squeeze. The 3-bit file everyone reaches for, Unsloth's UD-IQ3_XXS, is 10.9 GB, and you can feel the quality give in a 3-bit 27B. So when Nari Labs put up a 10.1 GB NariQ build labelled "BF16-level", it picked up 195 points on Hacker News and went straight onto my bench.

NariQ is Nari Labs' trellis-coded quantization (the paper is arXiv 2611.26157). The files come in 3.0 and 2.5 bits per weight, and they only run on Nari Labs' own fork of llama.cpp, because they use tensor types upstream llama.cpp doesn't have.

I tested it in two parts. Speed and fit on my budget RTX 3060 12 GB box, then speed again plus a quality measurement against a full BF16 reference on an RTX 3090. Both parts were pre-registered: the predictions, rules and file hashes were published before the first measurement.

The bottom line: NariQ 3.0 is measurably better than UD-IQ3_XXS: 22% less drift from BF16, with the margin well outside the noise. It's nowhere near BF16. An ordinary 4-bit file, UD-Q4_K_XL, drifts about a sixth as much. NariQ 2.5 is worse than IQ3.

What "BF16-level" means in Nari Labs' numbers

Nari Labs backs the claim with task scores (DeepSWE, Terminal-Bench 2.1, AIME, MGSM and RealWorldQA) plus KL divergence measured on tokens the model wrote itself during agent tasks. Those are fair ways to test a quant. They're also friendly ones. Task scores are coarse: a quant can change plenty of individual tokens and still pass the same tests. And KL on the model's own output scores the text it finds easiest to predict.

I measured something narrower and harsher: KL divergence against BF16 on human-written text the model has never seen. The model ran over 131,072 tokens of text, and at every scored position I compared each file's full next-token distribution with the BF16 model's. Mean KLD is how far the quant's predictions drift from the original's on average. Zero means identical.

The text was held out by date. Qwen's repo was last modified on 2026-11-07, so every document was first published on or after 2026-11-08:

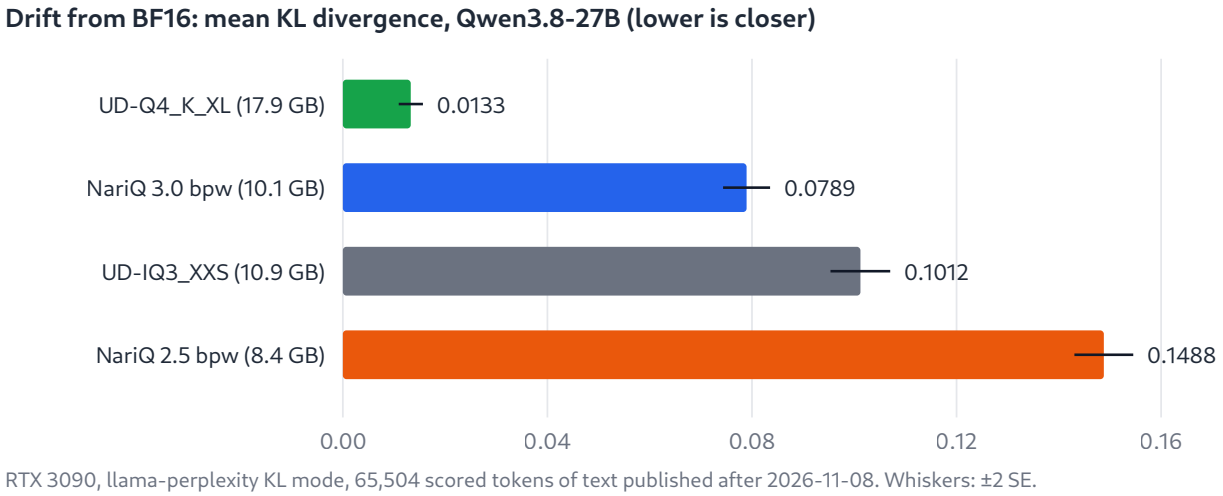
- **Code half:** 14 source files added to llama.cpp after that date. Files split out of older code were filtered out (no file has more than 20% of its lines in the pre-date tree).
- **Prose half:** 47 English Wikipedia articles created after that date, with copies, splits and translations excluded.

That's 65,504 scored positions at 4,096 context. The set is public, with its sha256, in the record.

This doesn't reproduce Nari Labs' test, and it can't show that their task scores are wrong. It answers a different question: how close are NariQ's predictions to the full model's on fresh text?

The quality numbers

Reference: Qwen3.8-27B at full BF16 (Unsloth's two-shard GGUF, 54.7 GB), run once on the 3090 with partial offload to produce the reference logits. Each quant then ran over the same stored tokens.



RTX 3090, llama-perplexity KL mode, 65,504 scored tokens of text published after 2026-11-08. Whiskers: ±2 SE.

File	Size	Mean KLD ± SE	Same top token	PPL vs BF16	Code-half KLD	Prose-half KLD
UD-Q4_K_XL	17.9 GB	**0.0133** ± 0.0012	97.5%	+0.8%	0.0040	0.0226

File	Size	Mean KLD ± SE	Same top token	PPL vs BF16	Code-half KLD	Prose-half KLD
NariQ 3.0 bpw	10.1 GB	**0.0789** ± 0.0023	92.3%	+1.9%	0.0232	0.1347
UD-IQ3_XXS	10.9 GB	0.1012 ± 0.0029	91.5%	+7.5%	0.0377	0.1647
NariQ 2.5 bpw	8.4 GB	0.1488 ± 0.0029	89.2%	+8.0%	0.0462	0.2514

"Same top token" is how often the quant's most likely next token matches BF16's. "PPL vs BF16" is the perplexity ratio (BF16 perplexity on this set is 3.42).

What I take from it:

- **NariQ 3.0 beats IQ3_XXS cleanly.** 0.78x the drift, and the error bars aren't close to touching. It's also 0.8 GB smaller. On a 12 GB card, that's a real improvement in the 3-bit slot.
- **It's not 4-bit quality, let alone BF16.** UD-Q4_K_XL drifts 5.9x less. In fairness, that file is 17.9 GB and won't fit a 12 GB card. But "BF16-level" should at least mean "close to Q4", and it isn't.
- **Perplexity flatters NariQ.** On perplexity alone, NariQ 3.0 sits 1.9% above BF16, against 7.5% for IQ3, which looks almost lossless. KLD compares the whole distribution, not just the probability of the right answer, and there the gap to IQ3 is only 22%. If you've seen "within 2% of BF16 perplexity" quoted for a quant, this is why that number can mislead.
- **NariQ 2.5 is a size play.** It's worse than IQ3 on every quality column. You'd pick it for the 2.5 GB it saves, and only if you need those 2.5 GB.
- **Every file drifts more on prose than on code**, four to six times more. Fresh Wikipedia articles are harder on a quant than fresh llama.cpp source.

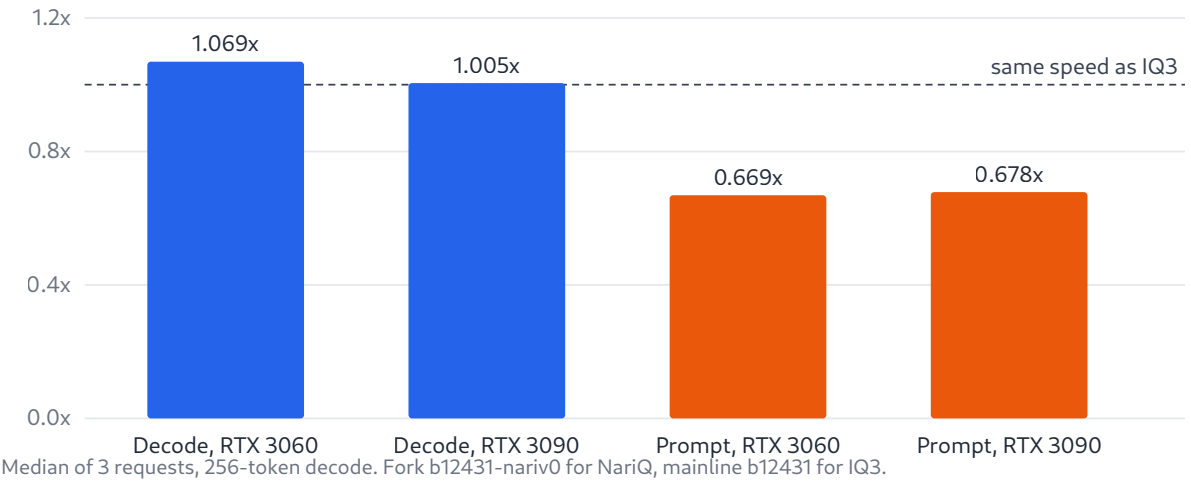
Speed: a wash on decode, slower on prompts

All speed numbers are a median of 3 requests, 256-token decode, thinking off, with fresh prompts so no cache helps. IQ3 ran on mainline llama.cpp at the fork's base commit (b12431), which is what you'd actually run. NariQ ran on the fork (`b12431-nariv0`).

Card	File	Decode, 512 / 4,096 prompt	Prompt processing, 512 / 4,096
RTX 3060 12 GB	UD-IQ3_XXS (mainline)	20.6 / 20.3 tok/s	526 / 581 tok/s

Card	File	Decode, 512 / 4,096 prompt	Prompt processing, 512 / 4,096
RTX 3060 12 GB	NariQ 3.0	22.0 / 21.7 tok/s	369 / 389 tok/s
RTX 3060 12 GB	NariQ 2.5	23.1 / 22.7 tok/s	368 / 389 tok/s
RTX 3090 24 GB	UD-IQ3_XXS (mainline)	50.7 / 49.7 tok/s	1,244 / 1,495 tok/s
RTX 3090 24 GB	NariQ 3.0	51.2 / 50.0 tok/s	896 / 1,014 tok/s
RTX 3090 24 GB	NariQ 2.5	54.5 / 53.2 tok/s	898 / 1,014 tok/s
RTX 3090 24 GB	UD-Q4_K_XL (mainline)	42.0 / 41.4 tok/s	1,266 / 1,512 tok/s

NariQ 3.0 bpw speed relative to UD-IQ3_XXS on mainline (4,096-token prompt)



Median of 3 requests, 256-token decode. Fork b12431-nariv0 for NariQ, mainline b12431 for IQ3.

The fork's own release notes say NariQ files "decode slower than comparable stock GGUFs" on Ampere. I tested that claim directly, and on both of my Ampere cards it's wrong, in NariQ's favour. On the 3060, NariQ 3.0 decodes 7% faster than IQ3. On the 3090 the edge shrinks to under 1%, which is technically faster and practically the same.

The cost is prompt processing, about 30% slower on both cards (0.67-0.72x of IQ3). At a 4,096-token prompt on the 3060, that took time to first token from about 7.1 seconds to 10.6. If you paste long documents or run RAG with big contexts, you'll feel that more than the decode gain.

One reassurance: the fork doesn't touch stock files. IQ3 ran at the same speed on the fork as on mainline (within 1% on both cards), and its quality numbers came out identical to the last printed digit.

The 3060 fit test: where NariQ 2.5 earns its place

On 12 GB, how much context you can fit matters as much as file size. I tried each file at 16K and at 32K context:

File	16K context	32K context
UD-IQ3_XXS (mainline)	answers	**doesn't load** (out of memory on the recurrent-state cache)
NariQ 3.0	answers	loads, then **crashes** on the first request (out of memory in a fork kernel)
NariQ 2.5	answers	**answers**, 23.1 tok/s, 10,309 MiB peak

NariQ 2.5 is the only one of the three that gives you a 27B dense model at 32K context on a 12 GB card. That's its whole case. NariQ 3.0 loading at 32K is a trap: the server comes up and then dies at the first request.

How my predictions did

I wrote predictions before each part. Scored straight:

- **3060 speed: decode wrong, in NariQ's favour.** I predicted 0.50-0.85x of IQ3 from the fork's own warning; it came in at 1.07x. NariQ 2.5's edge over 3.0 (1.048x) fell just under my 1.05-1.20x band. Prompt processing and all three fit predictions hit.
- **3090 speed: 10 of 12 hit.** The miss was UD-Q4_K_XL. I predicted 0.55-0.75x of IQ3's decode, and it did 0.83x.
- **Quality: 7 of 9 hit.** Hits: Q4's KLD band, IQ3's band, NariQ 3.0 beating IQ3 (at 0.78x, just inside my 0.35-0.80x band), NariQ 3.0 losing to Q4, the perplexity ordering, the fork giving IQ3 identical numbers, and the code half drifting less than prose. Both misses were NariQ 2.5: I predicted it would be no worse than IQ3, and it was clearly worse on KLD and same-top-token.

One process note, because it's the kind of thing that should be public. The first BF16 reference run broke two of my own rules. It swapped 7.3 MiB to disk, and the log didn't print the line

proving the BOS setting had been applied. My harness wasn't passing the verbose flag. Before any quant was measured, I wrote and published an amendment: verbose logging on every run, and swap counts against timing runs only, since swap can't change logits. The failed run stays in the record as failed. The re-run passed, and its stored tokens were byte-identical to the first run's.

Who should use which file

- **12 GB card, short-to-medium context, and you're fine running a fork:** NariQ 3.0. It's measurably better than IQ3_XXS at a smaller size, and decode is the same or a bit faster. Accept the slower prompts.
- **12 GB card and you need 32K context:** NariQ 2.5 is the only file of these three that does it. You're trading quality below IQ3 for the context.
- **You live in long prompts (RAG, document Q&A, big coding contexts):** the 30% prompt-processing hit eats into NariQ's appeal. IQ3_XXS on mainline is the simpler choice.
- **Ollama, LM Studio, or anything else built on upstream llama.cpp:** they won't load NariQ files. They need the fork. Stick with UD-IQ3_XXS until upstream adds the tensor types.
- **24 GB card:** skip the 3-bit fight entirely. UD-Q4_K_XL is 17.9 GB, drifts a sixth as much as NariQ 3.0, and still decodes at 41 tok/s on a 3090.

Limits

This is one model, one fork build (`b12431-nariv0`), two Ampere cards, and one text set of 131K tokens. KLD on fresh human-written text isn't a task score, and Nari Labs could still be right that NariQ holds up on agent tasks. The fork is tuned for Blackwell (SM120), so a 50-series card may see a different speed picture, and the fork may well speed up on Ampere later. And "same size" is approximate: IQ3_XXS is 0.8 GB bigger than NariQ 3.0, but it's the comparison Nari Labs chose.

Every number here is in the public record, along with the eval set, the harness, and every log, including the failed run.

A weekly email with every new guide and measured benchmark.

Subscribe — free, no spam

Source: https://insiderllm.com/guides/nariq-qwen3-8-27b-tested-rtx-3060-3090/

Free guides for running AI locally