60-second takeaway
Both models fit comfortably on one 24 GB RTX 3090 Ti, but fit did not predict usefulness. Qwen3-VL-4B passed 13 of 25 fixed text tasks; Nanbeige4.2-3B passed 8. Nanbeige passed one more tool-call task, while Qwen did better on arithmetic, logic, planning, and text transformations.
This is a narrow local probe with exact-match scoring. It is evidence for choosing between these two models for this task mix, not a general model ranking.

The practical question

Nanbeige4.2-3B is small enough to look attractive for a local agent. A model that fits into consumer GPU memory, however, is useful only if it follows the formats and solves the tasks the agent actually needs.

We compared the official Nanbeige4.2-3B BF16 checkpoint with the existing Qwen3-VL-4B-Instruct BF16 baseline on one RTX 3090 Ti. The question was narrow:

Which model was more usable on 25 short, deterministic agent-style text tasks under the same prompts and exact-match scorer?

The test was not designed to reproduce broad benchmark claims. Qwen is also a visual-language model, but this run supplied text only.

Results

Measurement	Nanbeige4.2-3B	Qwen3-VL-4B
Tasks passed	8/25	13/25
Peak allocated VRAM	8,045.6 MiB	8,486.2 MiB
Mean generation latency	3.470 s	0.176 s
Minimum generation latency	1.718 s	0.049 s
Maximum generation latency	3.987 s	0.336 s
Model load time	83.63 s	2.11 s

Do not compare the load times as model speed. Nanbeige loaded from an external hard drive, while Qwen loaded from NVMe storage. The generation measurements were collected inside the same harness after loading, so they are more directly comparable within this run.

The task split matters

The combined score hides a useful difference:

Task family	Nanbeige4.2-3B	Qwen3-VL-4B
Arithmetic	1/3	3/3
Logic	0/2	1/2
Planning	1/5	3/5
Tool calls	4/10	3/10
Transformations	2/5	3/5

These five mutually exclusive families account for all 25 tasks. A single average can still conceal the behavior that matters to an application.

Nanbeige's stronger tool-call result makes it worth revisiting if structured tool emission is the dominant requirement. Qwen's broader score made it the better default for this mixed task set. Neither conclusion should be carried into a different prompt format without another fixed comparison.

Nanbeige needed a compatibility shim

The official Nanbeige snapshot reached generation under Transformers 4.57.6, then its remote model code called `DynamicCache.get_max_length`, an API that is not present in that version.

Pinning Transformers 4.44.2 restored the method, but its older dynamic-module scanner then required `flash_attn` even though the import was conditional. The final harness stayed on Transformers 4.57.6 and added one narrow compatibility shim: an unbounded `DynamicCache.get_max_length()` returned `None`.

The shim did not change model weights, attention, prompts, or decoding. It did change the runtime environment, so the benchmark applies to the pinned checkpoint and this explicit compatibility path. A future upstream update may remove the need for it.

What the VRAM numbers establish

Both models used about 8 GiB of peak allocated VRAM. That establishes that the tested BF16 inference paths fit comfortably on this 24 GB card for these short prompts.

It does not establish:

- maximum context length that will fit;
- concurrency or batching capacity;
- sustained production throughput;
- multimodal memory use for Qwen;
- fine-tuning memory requirements; or
- behavior when another process shares the GPU.

That final limit mattered during the experiment. A separate root-owned vLLM process started after the first GPU preflight and consumed about 23.6 GiB. The resulting out-of-memory failure initially looked like Nanbeige memory use, but process attribution showed that it belonged to another session. A later clean window produced the 8,045.6 MiB measurement reported here.

The operational rule is simple: checking free VRAM while planning is not a reservation. Check again immediately before loading, attribute every process, and never stop another workload without knowing its owner.

What the score does and does not measure

Each task used deterministic decoding and a fixed expected pattern. This makes the run repeatable, but exact-match scoring is brittle. A correct answer can fail because it includes extra reasoning, uses a different valid JSON shape, or violates a formatting rule that matters more to the harness than to every real application.

The result directly supports this statement:

> Under the tested prompts, deterministic decoding, compatibility shim, and exact-match rules, Qwen3-VL-4B passed more of the 25 tasks than Nanbeige4.2-3B.

It does not support claims that Qwen is generally smarter, that Nanbeige is a poor model, or that either model is production-ready. Twenty-five tasks are too

few to estimate broad quality, and no independent human grading was performed.

Which model should you choose?

Choose **Qwen3-VL-4B as the default challenger** when your workload resembles this mixed set and you also want the option to add visual inputs later.

Keep **Nanbeige4.2-3B as a targeted candidate** when tool-call formatting is the main requirement, but first test more schemas, malformed arguments, retries, and multi-step calls. Its current Transformers compatibility boundary also needs to be part of deployment testing.

For either model, build a small evaluation from your real task distribution. Memory fit is only the first gate. The final choice should follow task success, latency, failure recovery, and operational stability.

The next useful tests

1. Repeat the 25 tasks across several fixed seeds or controlled sampling settings.
2. Add human review for answers rejected only by exact-match formatting.
3. Expand tool tests to nested arguments, invalid calls, correction, and multi-step plans.
4. Add longer contexts and record latency and peak VRAM by input length.
5. Test Qwen with visual evidence rather than treating a text-only run as a complete evaluation of a visual-language model.
6. Repeat Nanbeige after an upstream compatibility update without the shim.

A separate infrastructure question

The same research batch tested SIE as an on-demand embedding and reranking server. Model switching and 15-second idle eviction worked, and shutting down the server returned the GPU to its host baseline. That is useful operational evidence, but it is not yet a pressure-driven LRU result: the two small models could not reach SIE's enforced 50 percent pressure threshold safely.

We will keep that as a separate model-serving investigation after testing a larger approved model pair, concurrent requests, readiness retries, repeated eviction

cycles, and failed-load recovery. Combining it with this model-choice benchmark would blur two different reader questions.

Reproducibility boundary

The local evidence ledger records the pinned checkpoints, 25 prompts, expected patterns, responses, per-task latency, peak allocated VRAM, compatibility failure, storage placement, and process collision. The raw local paths are not part of this public article, but the reported figures come from that preserved ledger rather than reconstructed notes.

The strongest conclusion remains narrow: on this one RTX 3090 Ti and this 25-task agent-style probe, both models fit, but Qwen3-VL-4B was faster and passed more tasks overall. A different workload can produce a different choice.