MiniCPM-V 4.7: *27B-class on short work, behind on long work.*

openbmb's MiniCPM-V 4.7 comes in two sizes: a 35B mixture of experts with about 3B active per token, and a 1B. We put both through the suite we use for every model, ran both down a quantization ladder, and timed them on seven GPUs from an RTX PRO 6000 to a 12 GB RTX 5070, with and without openbmb's draft model.

Bottom line

- **The 35B matches a dense 27B on short work.** Against Qwen3.8-27B: HumanEval 160 vs 160 of 164, Global-MMLU-Lite 84.9–85.0 vs 85.0%, and 120 of 120 tool calls copied byte-exact.
- **It falls behind on long work.** Tracking state through a long log: 24–27 of 60 walks fully right against Qwen's 40. Hard agentic tasks: 18 of 34 against 26. It stops early rather than looping.
- **It quantizes very well.** Every GGUF from F16 down to IQ3_M (15.4 GB, 22% of BF16) scores inside run-to-run noise. At 4 bits the recipe decides: openbmb's AWQ file loses 9.2 points of multilingual knowledge, its GPTQ file 0.6. The cliff is IQ2_XXS (9.5 GB).
- **It is fast on any card, and long context barely slows it.** One user, Q4_K_M on llama.cpp: 367 tok/s on an RTX PRO 6000, 339 on an RTX 5090, 185 on an RTX 3090. On vLLM in BF16 it still decodes 171 tok/s with 120K tokens of context (218 at 1K), and one card serves 2,566 tok/s across 64 users.
- **openbmb's DSpark draft model does not pay in real use.** Run as its model card says (15 drafted tokens), it is slower than no draft on all seven cards (0.56–0.88×). It wins only on short prompts with greedy decoding (2.9× on short JSON on vLLM) and falls behind no draft from 16K tokens of context.
- **The 1B's thinking mode is broken as served.** It often never closes its <think> block, so the server returns an empty answer: 267 of 500 on GSM8K, and 236 of those hidden answers are right. In instruct mode it trails Bonsai 2 2B on code and tool calls (HumanEval 81 vs 98, tool calls 19 vs 82 of 120).

1 · What we ran

- **Models.** openbmb/MiniCPM-V-4.7-35B-A3B @ 0bbda784: a mixture of experts, 70.6 GB in BF16, 40 layers of which 30 use linear attention, 8 of 256 experts active per token. And openbmb/MiniCPM-V-4.7-1B @ 286817f1, 3.24 GB in BF16. Both Apache-2.0. Served text-only: we did not measure vision.
- **Scores.** BF16 on a pinned vLLM nightly (0.32.0rc1.dev190), one GPU per copy, several copies in parallel. The GGUF quants ran on mainline llama.cpp with the same chat template. Two settings. **Thinking on**: the 35B's own sampling (temperature 1.0, top-p 0.95, top-k 20). **Instruct**: thinking off, the model card's instruct sampling (temperature 0.7, top-p 0.95, top-k 100, repetition penalty 1.03). 32,768 tokens of room per answer, a seed per item.
- **Sets.** HumanEval (164) and MBPP (500), both executed, with two rounds of self-repair on the failures. GSM8K (500). The ledger walk: 60 chains of 12–40 hops through 8K–32K-token logs, scored on walks with every value

right. Tool-call payload fidelity: 120 write_file calls that must copy text byte-exact. Agentic tier 2 (81 repository tasks) and tier 3 (34 held-out trap tasks with hidden tests). MGSM (11 languages × 250) and Global-MMLU-Lite (23 languages × 100).

- **Quant ladders.** 18 files of the 35B (openbmb's own plus community quants), every set at every rung, plus instruct mode on BF16, FP8 and Q4_K_M. For the 1B, 8 GGUF files in instruct mode.
- **Speed.** The 35B and the 1B as Q4_K_M on seven cards with llama.cpp b11696: an engine benchmark, then real use (one stream, 24 prompts × 1,024 tokens) with and without the DSpark draft model. Separately, the 35B in BF16 on vLLM on one RTX PRO 6000: by context length to 120K tokens and by load to 64 users.
- **Reference rows.** Qwen3.8-27B and Bonsai 2 2B come from our 15–16 October runs on llama.cpp (Qwen as Q8_0 at medium reasoning effort, with a presence penalty of 1.0); their multilingual rows ran Qwen in BF16 on vLLM. They are context, not a controlled pair.

2 · The 35B against Qwen3.8-27B

set	35B thinking on	35B instruct	Qwen3.8-27B	
HumanEval (164)	160	159	160	level
MBPP (500)	464–467	408	475	thinking +56
GSM8K (500)	484	484	490	level
Self-repair (fixed of failed)	25 of 36	39 of 97	22 of 29	69% vs 76%
Ledger walk, fully right (60)	24–27	0	40	behind
Tool-call payload, byte-exact (120)	120	120	118	level
Agentic tier 3 (34)	18	17	26	behind
Agentic tier 2 (81)	77–79	77	81	−2 to −4
MGSM, % (11 languages)	91.7	90.9	93.2	−1.5 pts
Global-MMLU-Lite, % (23 languages)	84.9–85.0	83.8	85.0	level

Counts over all items: an answer cut off at the 32K cap or left empty counts wrong. Ranges are two BF16 runs with different seeds. Self-repair counts the failures each run fixed itself, so its base differs per run. Qwen3.8-27B: Q8_0 on llama.cpp at medium reasoning effort, multilingual in BF16 on vLLM.

Where it falls short is state and persistence. The ledger walk asks for a running value through 12–40 hops of a long log, and one slip spoils every later line: the 35B finishes 24–27 walks of 60 fully right, Qwen3.8-27B 40. On the hard agentic tier it passes every trap task when the rule is written down (10 of 10), but 2 of 10 when the same rule has to be inferred, and 2 of 8 spec builds. Every failure ended within 5–10 steps of a 25-step budget: it declares done or gives up early; it does not loop.

Thinking buys MBPP and the ledger, little else. Turning thinking off costs 56 MBPP problems and the whole ledger (24 → 0); HumanEval, GSM8K, tool calls and tier 3 move by at most 1 item, multilingual by at most 1.1 points. On code, instruct mode writes 18% of the tokens (a typical HumanEval answer 218 tokens instead of 624); on the multilingual sets it saves less (74% of the tokens).

3 · The 35B quant ladder

file	GB	HumanEval	MBPP	GSM8K	ledger	payload	tier 3 · 2	MGSM %	GMMLU %
BF16, two runs	70.6	160	464–467	484	24–27	120	18 · 77–79	91.7	84.9–85.0
FP8	38.0	159	464	485	35	120	17 · 77	91.6	85.1
GPTQ (W4A16)	22.7	158	467	484	33	120	18 · 78	91.5	84.3
int4, no calibration (W4A16)	22.7	161	458	482	29	120	16 · 79	90.9	83.2
AWQ (W4A16)	22.7	157	453	488	31	120	20 · 77	88.4	75.7
NVFP4 (community)	23.6	158	464	484	22	120	16 · 77	91.4	83.0
NVFP4, experts only (community)	24.2	160	465	485	28	120	18 · 78	91.7	83.0
F16 GGUF	69.4	159	461	483	34	120	17 · 79	91.9	84.8
Q8_0	36.9	158	464	485	34	120	17 · 79	91.9	85.0
Q6_K	28.5	159	466	485	30	120	17 · 77	91.7	84.6
Q5_K_M	24.7	157	462	482	26	120	18 · 77	91.8	84.2
Q4_K_M	21.2	160	461	484	31	120	18 · 78	91.5	84.8
IQ4_XS	18.7	159	465	484	28	120	19 · 77	91.6	84.4
mixed 3-bit	15.5	158	465	479	32	120	17 · 78	91.3	84.2
IQ3_M	15.4	159	458	484	28	120	18 · 77	91.7	84.8
IQ3_XXS (imatrix)	13.6	156	458	485	21	120	16 · 79	90.3	83.3
IQ2_M (imatrix)	11.7	159	458	481	9	118	17 · 75	90.0	82.4
IQ2_XXS (imatrix)	9.5	149	409	400	4	92	11 · 73	64.7	57.8
IQ1_S (imatrix)	7.5	106	251	340	0	19	2 · 6	45.9	46.8

Thinking on. Counts over all items (cut-offs and empty answers count wrong); multilingual as % of all items. vLLM for the safetensors files, llama.cpp for the GGUFs. One run per file.

file	GB	HumanEval	MBPP	GSM8K	ledger	payload	tier 3 · 2	MGSM %	GMMLU %
BF16	70.6	159	408	484	0	120	17 · 77	90.9	83.8
FP8	38.0	159	405	483	0	120	17 · 79	90.7	84.6
Q4_K_M	21.2	159	405	485	0	120	17 · 78	91.2	83.7

Instruct mode (thinking off) on three files.

Lossless to IQ3_M (15.4 GB). We count a file as lossless when every set sits inside run-to-run noise, the spread we measured between two BF16 runs and across the clean files. FP8, GPTQ and every GGUF from F16 down to

IQ3_M pass on all eight sets. The F16 GGUF on llama.cpp scores the same as BF16 on vLLM, so the engine does not move the numbers, and instruct mode at FP8 and Q4_K_M matches BF16 instruct.

At 4 bits the recipe matters more than the bits. openbmb ships three W4A16 files of the same format and size (22.7 GB). Against BF16 on Global-MMLU-Lite, GPTQ costs 0.6 points, plain int4 1.8, AWQ 9.2. The AWQ file leaves 122 answers inside an unclosed <think> block, against 16–26 for BF16. Both community NVFP4 files cost about 2 points.

The order it breaks in. The first slip is at IQ3_XXS (13.6 GB): MGSM −1.4, Global-MMLU −1.6 points. At IQ2_M (11.7 GB) the long work goes first: ledger 9 of 60 and the first tool-call misses (118 of 120), while HumanEval, MBPP and GSM8K still look fine. IQ2_XXS (9.5 GB) is the cliff: GSM8K 400, MGSM −27.0 and Global-MMLU −27.2 points. Most of that is failing to finish, not failing to know: 516 of 2,300 Global-MMLU replies end with the answer still inside an unclosed <think> and 51 run to the 32K cap.

4 · The 1B

set	1B thinking on	1B instruct	Bonsai 2 2B
HumanEval (164)	78 (12 empty)	81	98
MBPP (500)	219 (21 empty)	197	224
GSM8K (500)	194 (267 empty)	408	426
Ledger walk (60)	0 (1 empty)	0	0
Tool-call payload (120)	15	19	82
Self-repair (fixed of failed)	78 of 367	30 of 386	39 of 342
Agentic tier 3 · tier 2	0 · 17	0 · 11	2 · 30
MGSM, %	43.4 (713 empty)	53.2	58.3
Global-MMLU-Lite, %	47.5 (33 empty)	45.6	47.7

Counts over all items; multilingual as % of all items. “Empty” = the reply finished with an empty answer field. Bonsai 2 2B: our 15–16 October runs on llama.cpp (PrismML's fork), thinking on, presence penalty 1.0, on PrismML's pre-release 2B file (the public file has since changed and was not re-measured): context, not a controlled pair.

Thinking mode fails at the last step. With thinking on, the 1B often writes its working and a final answer and then stops without closing <think>. vLLM's reasoning parser then files the whole reply as thinking, and the answer field comes back empty: 267 of 500 on GSM8K, 713 of 2,750 on MGSM. The reasoning itself is fine. Read out of the thinking with the same scoring rule, 236 of the 267 hidden GSM8K answers are right, which would put thinking mode at 430 of 500 (86%), above instruct mode's 408. We score what the server returns, so the table keeps 194. The model card documents only instruct mode for the 1B.

Served as-is, instruct mode is its operating point, and there it trails Bonsai 2 2B on code (HumanEval 81 vs 98) and badly on byte-exact tool calls (19 vs 82 of 120), with multilingual knowledge close (45.6 vs 47.7%). It scores 0 of 34 on the hard agentic tier in both modes.

file	GB	HumanEval	MBPP	GSM8K	payload	tier 3 · 2	MGSM %	GMMLU %
BF16 (vLLM)	3.24	81	197	408	19	0 · 11	53.2	45.6
F16 GGUF	2.02	86	195	405	25	1 · 14	53.5	47.4
Q8_0	1.08	81	195	402	20	0 · 17	52.8	45.7
Q6_K	0.84	83	204	403	25	0 · 9	52.6	47.6
Q4_K_M	0.67	65	160	397	22	0 · 3	48.3	43.4
IQ4_XS	0.64	70	184	406	24	0 · 5	50.2	41.7
IQ3_M	0.56	41	127	358	2	0 · 0	37.9	42.6
Q2_K (imatrix)	0.51	not scored: failed the tool-call probe twice						
IQ2_XXS (imatrix)	0.44	not scored: failed the tool-call probe twice						

The 1B quant ladder, instruct mode, llama.cpp for the GGUFs. The ledger is 0 of 60 on every file (9–27 walks loop to the cap). One run per file.

Q6_K (0.84 GB) is the smallest 1B file that holds: F16 GGUF, Q8_0 and Q6_K sit inside the noise band on every set. **Q4_K_M is lossy for the 1B**, unlike the 35B: HumanEval −9.8, MBPP −7.4, MGSM −5.0 points, tier 2 11 → 3. IQ3_M falls off (HumanEval −24.4 points, 2 of 120 tool calls). Both 2-bit files failed the pre-registered tool-call probe twice (one answered in prose, one opened a <think> block with thinking off), so by our rule they were not scored. openbmb's 1B AWQ file did not load in this vLLM build.

5 · Speed on seven cards

card	experts on CPU	decode tok/s	at 8K	prefill tok/s	real use	DSpark 15	DSpark 7
RTX PRO 6000	none	367	330	12,019	346	251 · 0.73×	361 · 1.04×
RTX 5090	none	339	313	10,505	307	208 · 0.68×	306 · 1.00×
RTX PRO 4500	none	200	186	4,822	189	143 · 0.76×	195 · 1.03×
RTX 3090	none	185	179	4,107	174	138 · 0.79×	174 · 1.00×
RTX 5080	17 of 40	107	106	1,229	104	59 · 0.56×	86 · 0.83×
DGX Spark (GB10)	none	84	79	2,788	78	69 · 0.88×	92 · 1.18×
RTX 5070	29 of 40	40	39	420	39	29 · 0.74×	43 · 1.08×

MiniCPM-V 4.7 35B-A3B, openbmb's Q4_K_M GGUF (21.2 GB), llama.cpp b11696, one card. Decode = llama-bench, 128 tokens, mean of 5, empty context and at 8K; prefill = 512-token prompt. Real use = llama-server, one stream, 24 prompts × 1,024 tokens at the model's sampling with thinking on; DSpark columns give decode and the factor against no draft. The file does not fit 16 GB or 12 GB: on the RTX 5080 the experts of 17 of 40 layers run on the host CPU (AMD EPYC 7763 vCPUs in a virtual machine), on the RTX 5070 29 of 40 (AMD Ryzen 9 9950X3D). Clocks: the RTX PRO 6000 runs a permanent core and memory overclock (GPC +350, MEM +6000), the RTX 5090 a

memory overclock (MEM +6000); the others are stock, the RTX PRO 4500 at its 165 W limit. The RTX 5070 is a desktop display card, screen on. The DGX Spark has unified memory.

Context costs this model little. Thirty of its 40 layers are linear attention, so decode at 8K of context is only 1–10% below empty context. Every card with 24 GB or more holds the whole file and decodes 185–367 tok/s; with experts spilled to the CPU a 16 GB card still runs it at 107 and a 12 GB card at 40.

The DSpark draft model, as its card runs it, makes real use slower on all seven cards. At the model's own sampling with thinking on it keeps 18–20% of 15 drafted tokens, and checking 15 costs more than that saves. At 7 drafted it keeps 35–38% and roughly breaks even on the large cards (1.00–1.04×); it gains only where each token is slowest (DGX Spark 1.18×, RTX 5070 1.08×) and loses on the RTX 5080 (0.83×), whose experts run on the CPU.

card	decode tok/s	at 8K	prefill tok/s	real use	DSpark 15	DSpark 7
RTX PRO 6000	921	832	56,338	559	332 · 0.59×	349 · 0.62×
RTX 5090	824	760	46,854	415	251 · 0.61×	272 · 0.65×
RTX PRO 4500	517	467	31,302	319	179 · 0.56×	218 · 0.68×
RTX 3090	476	448	22,609	302	189 · 0.62×	226 · 0.75×
RTX 5080	640	572	38,783	363	220 · 0.61×	247 · 0.68×
DGX Spark (GB10)	281	252	18,223	205	128 · 0.62×	156 · 0.76×
RTX 5070	318	290	19,926	247	130 · 0.53×	166 · 0.67×

MiniCPM-V 4.7 1B, openbmb's Q4_K_M GGUF (0.67 GB), llama.cpp b11696, same cards, clocks and method; real use here with thinking off at the card's instruct sampling. Q4_K_M is the lossy file for the 1B (section 4); its speed is shown for comparison with the 35B.

For the 1B the draft model is slower everywhere (0.53–0.76×): it keeps 16–27% of drafted tokens, and verifying them costs more than a step of a model this small. Real use runs at 50–78% of the engine benchmark, because per-token overhead outside the GPU is large next to a 1B's step.

6 · vLLM in BF16 on one RTX PRO 6000

prompt tokens	no draft tok/s	prefill tok/s	first token	DSpark 15 tok/s	accepted per step
1,075	218	10.8K	0.10 s	506	6.6
4,057	217	37.7K	0.11 s	501	7.3
8,071	215	39.8K	0.20 s	281	4.4
12,566	212	37.2K	0.34 s	217	3.8
32,077	203	31.6K	1.01 s	79.5	2.0
64,120	188	25.2K	2.55 s	33.3	1.3
120,121	171	18.6K	6.46 s	17.7	1.1

MiniCPM-V 4.7 35B-A3B BF16, vLLM nightly 0.32.0rc1.dev190, one RTX PRO 6000 (GPC +350, MEM +6000), 262,144-token context, 64 sequence slots, thinking on, greedy. Decode = 256 tokens, median of 5; prefill and first token from the first, uncached request. Accepted per step = tokens the target keeps per verification step with the DSpark draft model (15 drafted).

Without a draft, long context barely slows it: 218 tok/s at 1K, 171 at 120K, the first token of a 120K prompt in 6.5 s. **With DSpark it is 2.3× faster on short prompts and 10× slower on long ones**: 506 against 218 tok/s at 1K, but the draft's guesses stop being accepted as context grows (6.6 tokens per step at 1K, 1.1 at 120K), it trails no draft from 16,082 tokens, and at 120K it decodes 17.7 tok/s.

users	1	4	8	16	24	32	48	64
no draft, total tok/s	214	514	788	1,189	1,490	1,756	2,177	2,566
no draft, per user	214	130	101	78	65	58	49	43
no draft, first token	0.03 s	0.03 s	0.06 s	0.07 s	0.07 s	0.07 s	0.09 s	4.14 s
DSpark 15, total tok/s	177	385	597	851	604	629	–	–
DSpark 15, first token	0.03 s	0.09 s	0.09 s	0.12 s	5.54 s	7.34 s	–	–

Load: about 10,000-token prompts, 250 tokens each, median of 3. First token = a probe request sent while the others run. The DSpark load test stopped at 32 users.

One card keeps scaling to its 64-sequence limit: 2,566 tok/s in total, 43 per user, with a new request's first token under 0.1 s up to 56 users (4.1 s at 64, when every slot is full). The curve had not bent at the server's cap, so the true peak is higher and we have not measured it. With DSpark the total peaks at 851 tok/s at 16 users and then drops, and from 24 users a new request waits 5.5–7.3 s for its first token.

arm	short JSON tok/s	short prose tok/s	accepted per step
no draft	221	221	1.0
DSpark, 7 drafted	585	261	5.91 · 2.49
DSpark, 15 drafted (card)	636	218	8.35 · 2.65
built-in MTP head (1, 2, 3)	refused to start		

The drafter sweep: short prompts, 384 tokens, median of 5, greedy. The draft model wins this sweep, which is why short-prompt benchmarks favour it.

The model also ships one built-in multi-token-prediction layer, but this vLLM build refuses its speculative method for this model type at startup (“Unsupported speculative method: 'mtp'”); using it would need a patch to vLLM.

7 · Long context and tool calls

retrieval	8–32K	64K	100K	120K
single passkey	15/15	5/5	5/5	5/5
three-hop and aggregate	18/18	6/6	6/6	6/6
among dense decoys	9/9	2/3	0/3	0/3
recall 15 facts	9/9	2/3	0/3	0/3
long decoyed chain	4/9	0/3	0/3	0/3

vLLM BF16, no draft, thinking on, greedy; passes over repeats (5 per cell for the passkey, 3 for the rest).

Simple retrieval is perfect to 120K tokens. Retrieval among decoys and recall of many facts break past 64K, the same weakness as the ledger walk. Our tool-call check (22 cases × 2 runs) is clean except an ambiguous request (it made two calls with invalid arguments) and recovering from a tool error (it did not retry), both failing in both runs.

8 · What to run

if you want	run
chat, code, extraction, multilingual	the 35B in instruct mode (temperature 0.7, top-k 100, repetition penalty 1.03)
MBPP-style coding, or state kept through a long document	turn thinking on, and expect it to trail a dense 27B on long state
the 35B file	Q4_K_M (21.2 GB) or IQ3_M (15.4 GB) GGUF, or GPTQ (22.7 GB) on vLLM. Not openbmb's AWQ. IQ2_M (11.7 GB) only for short tasks; nothing smaller
a card	any card with 24 GB or more holds Q4_K_M whole (185–367 tok/s); 16 GB runs it at about 107 with experts on the CPU, 12 GB at about 40
long documents, thinking-on chat, many users	no draft model
one user, short structured output, greedy	DSpark at 15 drafted on vLLM; on llama.cpp, if at all, 7 drafted
a 1B-class model	the 1B in instruct mode as Q6_K (0.84 GB) or larger; for tool calls, Bonsai 2 2B measured stronger
not this model	autonomous agent loops on hard tasks, or retrieval among decoys past 64K

9 · Caveats

- One run per cell, sampled decoding for the scores. Two BF16 runs of the 35B agree within 0–3 items on every set except the ledger walk, which spans 24–35 walks across the clean files: read only ledger gaps larger than about 10. Per-language multilingual scores (100 or 250 items) carry about ±9 points.
- The Qwen3.8-27B and Bonsai 2 2B rows come from a different run (llama.cpp, presence penalty 1.0, Qwen at medium effort): context, not a controlled comparison.
- The vLLM speed runs are greedy; the seven-card real-use runs sample at the model's settings. Draft acceptance differs between the two by design.
- The RTX PRO 6000 and RTX 5090 speed rows are overclocked (labelled in each table); on the RTX 5080 and RTX 5070 part of the model runs on the host CPU.
- Not measured: vision, EXL3 quants, IQ1_M and Q2_K of the 35B, the concurrency peak above 64 users, MTP on a patched vLLM.

Every figure on this page was computed from the raw rows of runs made on 15–21 October 2026 on our own machines. Run directories: `bench/runs/20261016-minicpm47` (scores, pre-registration, the 35B ladder), `bench/runs/20261021-minicpm47-1b-ladder`, `bench/runs/20261019-minicpm47-speed` (seven cards), `bench/runs/20261019-minicpm47-qual` and `-nospec` (vLLM), with reference rows from `bench/runs/20261015-bonsai2-family` and `bench/runs/20261015-multilingual`. The rows are kept;

to see them, or to send a question you want measured, write to hello@netsphere.com.ai. Online: netsphere.com.ai/reports/minicpm-v-4-7/