Best Local Models for OpenClaw 2026: Qwen 3.6 + DeepSeek V4

February 3, 2026 · Updated: May 26, 2026

Download this guide as PDF

> **Quick Answer:** OpenClaw is harness-agnostic — it works with any model from any provider, but model choice determines what your agent can actually do. This guide ranks the realistic local picks across VRAM tiers from 8GB to 48GB, plus Mac unified memory, with the recent Qwen 3.6 and DeepSeek V4 releases reframing the local picks. Each tier gets a specific recommendation, the agentic-capability tradeoffs that matter, sampling and quantization rules for tool-call-heavy workflows, and the frontier-adjacent API option for users who don't want local hardware at all. Where local closes the gap and where it doesn't.

📚 **More on this topic:** Qwen Models Family Guide · DeepSeek V4 Flash vs Pro · Best Local Coding Models 2026 · OpenClaw Token Optimization · Anthropic Cut OpenClaw Subscriptions · Local Alternatives to Claude Code · llama.cpp vs Ollama vs vLLM · Best Local LLMs for Mac

OpenClaw doesn't care what model powers it — you can plug in Claude, GPT-5, Gemini, or a local model through Ollama, llama.cpp, or vLLM. But the model choice matters enormously for agent performance. An agent that needs to write code, debug failures, use tools, and recover from errors requires different capabilities than a chatbot.

This guide covers which local models actually work for agent tasks in May 2026, what VRAM you need, and what the recent Qwen 3.6 and DeepSeek V4 releases changed. Updated for the post-Qwen-3.6 landscape.

What changed since the April refresh

Four things, all relevant to agentic work:

- **Qwen 3.6-35B-A3B** (April 16, 2026) — 35B MoE with 3B active params, Apache 2.0, 262K native context. Per Qwen's announcement, 73.4 on SWE-bench Verified.
- **Qwen 3.6-27B dense** (April 22, 2026) — 27B Apache 2.0, 77.2 on SWE-bench Verified, 59.3 on Terminal-Bench 2.0, 83.9 on LiveCodeBench v6 per Qwen's own numbers. Qwen positions it as flagship-level coding in a single dense model.

- **DeepSeek V4-Flash** (April 23, 2026) — 284B/13B active MoE, MIT, 1M context. See our V4 Flash vs Pro guide. Practical replacement for Haiku-class API tool-calling at $0.14/$0.28 per M tokens.
- **Anthropic admitted Claude Code was nerfed.** Per Fortune's April 24 reporting, Anthropic reduced Claude Code's default reasoning effort from "high" to "medium" on March 4 to cut latency, shipped a bug on March 26 that discarded reasoning history mid-session, and added a 25-word response cap on April 16. All three were reverted by April 20. The local-model competitive position improved by default during that window.
- **ik_llama.cpp fused MoE kernels** plus native NVFP4 / MXFP4 support landing in llama.cpp for Blackwell GPUs are real performance improvements for serving MoE agent models on consumer hardware. Up to 25% prompt processing speedup reported on RTX 5000/6000 Blackwell.

Qwen 3.5, GLM-4.6, and Kimi K2.6 are all still in rotation for specific cases — they're demoted, not deleted.

What Agent Tasks Require

Agent work is harder than chat. Here's why:

Capability	Why Agents Need It	What Tests It
Tool use	Agents call APIs, run shell commands, manipulate files	Function calling, structured output
Multi-step reasoning	Tasks span many actions with dependencies	Chain-of-thought, planning
Code generation	Building skills, debugging, automation	SWE-bench, LiveCodeBench
Error recovery	First approach often fails; agent must adapt	Self-correction, alternative solutions
Instruction following	Complex prompts with multiple constraints	Following formats precisely
Long context	Conversation history, file contents, task state	200K+ context utilization

7B models struggle here. They can chat, but they can't reliably orchestrate complex workflows. As of May 2026, the dependable floor for serious agentic work is 24GB VRAM running Qwen 3.6-27B dense, or 16GB with smart RAM offload running Qwen 3.6-35B-A3B MoE.

Recommended Models by VRAM Tier

8GB VRAM (RTX 3060, 4060)

Honest assessment: Painful. OpenClaw with 8GB is feasible for very simple, single-step tasks; serious agentic work fails.

Model	Size	Context	Agent Suitability
Qwen 3.5 9B (Q4)	~6.6GB	262K	Best of a tight situation — beats GPT-OSS-120B on GPQA Diamond
Qwen 3.6-4B if/when it ships	~2.5GB	262K	Watch for it on the Qwen org page
Llama 3.1 8B (Q4)	~5GB	128K	Longer context helps, weaker reasoning

Recommendation: Qwen 3.5 9B remains the recommended backbone here. Stronger reasoning than Llama 3.1 8B, native 262K context, working tool calling in Ollama v0.17.6+. For anything beyond simple workflows, route hard tasks to an API.

```
ollama run qwen3.5:9b
```

16GB VRAM (RTX 4060 Ti 16GB, 4070 Ti Super, 4080)

Assessment: This is where Qwen 3.6-35B-A3B MoE changes the math. The model is 35B total but only 3B active per token, so token speed stays high even when partial weights spill to system RAM via llama.cpp's hybrid offload.

Model	Size	Context	Agent Suitability
Qwen 3.6-35B-A3B (UD-Q4_K_XL)	~22GB total	65K+ tested	**Top pick** — community reports ~100 tok/s on RTX 3090 with full GPU; offload to RAM for 16GB
Gemma 4 26B-A4B	~15GB Q4	256K	Alternative MoE, 4B active params, Google ecosystem
Qwen 3 14B Q8	~15GB	32K	Safe dense pick, no offload needed
DeepSeek-R1-Distill-Qwen-14B Q8	~15GB	32K	Reasoning focus, no tool-call training

Recommendation: Qwen 3.6-35B-A3B with Unsloth's UD-Q4_K_XL GGUF via llama.cpp. The MoE architecture is what makes 16GB viable here — only 3B active params hit memory bandwidth per token, so the model feels much smaller than its 22GB footprint suggests. Per a community benchmark on RTX 3090, short-prompt decode hits 101.7 tok/s and 65K-context decode hits 80.9 tok/s. On 16GB you'll need RAM offload — speeds drop, but it remains usable.

```
# llama.cpp via Unsloth GGUF
llama-server -m Qwen3.6-35B-A3B-UD-Q4_K_XL.gguf -c 65536 --n-gpu-layers 99
```

Honest caveat: several r/LocalLLaMA threads report MoE models like 3.6-35B-A3B can degrade instruction-following on very long multi-step agent chains compared to dense models of similar capability. If your agent runs 20+ tool calls in a single session, test with Qwen 3 14B dense as a control.

24GB VRAM (RTX 3090, 4090, 5090)

Assessment: This is where local agentic work gets reliable in 2026. Qwen 3.6-27B dense is the new default.

Model	Size	Context	Agent Suitability
Qwen 3.6-27B (Q4_K_M)	~16.8GB	262K	**Top pick** — 77.2 SWE-bench Verified, 59.3 Terminal-Bench 2.0 per Qwen
Qwen 3.6-35B-A3B (Q6/Q8)	~22-30GB	200K+	Higher quant fits comfortably, faster decode than 27B dense
Qwen 3.5 27B (Q4_K_M)	~17GB	262K	Previous-gen pick, still works
Qwen 3 32B (Q4_K_M)	~20GB	32K	Dense alternative, narrower context
Qwen 2.5 Coder 32B (legacy)	~20GB	32K	Pure-code workload, less general — superseded by Qwen 3.6 for general agent work
DeepSeek-R1-Distill-32B	~20GB	32K	Reasoning focus, no tool-call training

Recommendation: Qwen 3.6-27B dense at Q4_K_M. Per Qwen's announcement, it scores 77.2 on SWE-bench Verified and 59.3 on Terminal-Bench 2.0 — both strong agentic numbers for a 27B dense model. Simon Willison ran the 16.8GB Unsloth Q4_K_M GGUF and clocked 25.57 tok/s. Multiple r/LocalLLaMA threads report pointing Claude Code and OpenCode at a local

Qwen3.6-27B endpoint and getting genuinely useful agentic runs — community phrasing is “vibe codes perfectly fine.”

Caveat on Qwen’s claims: Qwen positions Qwen 3.6-27B as tying Claude Sonnet 4.6 on the AA Agentic Index. As of April 25, that’s Qwen’s own claim from their announcement — Qwen 3.6-27B is not yet independently scored on the Artificial Analysis Agentic Index. Treat the comparison as directional, not settled. Test against your actual workload.

```
# Easiest setup
ollama launch openclaw --model qwen3.6:27b

# Or via llama.cpp directly with Unsloth quants
llama-server -m Qwen3.6-27B-Q4_K_M.gguf -c 200000 --n-gpu-layers 99
```

For coding-heavy agents, the 27B dense is the pick. For general agentic work where you want maximum throughput, Qwen 3.6-35B-A3B at Q6 fits comfortably in 24GB and the MoE architecture pushes more tokens per second.

32-48GB VRAM (RTX 6000 Ada, dual 3090 / 4090, A6000)

Assessment: Full capability tier. Multi-model serving becomes realistic. NVFP4 starts mattering on Blackwell hardware.

Model	Size	Context	Agent Suitability
Qwen 3.6-27B (Q6/Q8)	~22-30GB	262K full	Highest-quality 27B locally
Qwen 3.6-35B-A3B (Q8)	~37GB	262K full	Largest MoE at near-full quality
Llama 3.3 70B (Q4_K_M)	~40GB	128K	Older flagship, still capable
GLM-4.6 / Kimi K2.6	varies	200K+	Specialty picks for long context

Recommendation: Qwen 3.6-27B at Q6 with full 262K context loaded as the primary, plus Qwen 3.6-35B-A3B at Q4 ready to swap in for high-throughput sub-tasks. With multi-GPU via vLLM you can serve both concurrently.

If you’re on RTX 5000/6000 Blackwell, the native MXFP4 / NVFP4 support landing in llama.cpp gives up to 25% faster prompt processing — worth tracking the feature PRs as they land in the main branch.

Multi-GPU and team serving

For team OpenClaw deployments, vLLM is the answer:

```
# Serve Qwen 3.6-27B with vLLM, OpenAI-compatible endpoint
vllm serve Qwen/Qwen3.6-27B \
  --tensor-parallel-size 2 \
  --max-model-len 262144 \
  --enable-prefix-caching
```

OpenClaw connects to any OpenAI-compatible endpoint. vLLM gives you batching, prefix caching for shared system prompts, and per-request priority for interactive vs background agents.

Mac

Mac is unified-memory territory and behaves differently from VRAM math above. Quick summary: 32GB+ Apple Silicon runs Qwen 3.6-35B-A3B MoE comfortably via MLX or llama.cpp. 48GB+ runs Qwen 3.6-27B dense at Q6/Q8. See the Mac 2026 guide for the full breakdown.

Best for Claude Code / OpenCode workflows

Pointing Claude Code or OpenCode at a local model is one of the most-asked OpenClaw-adjacent setups in May 2026. Several r/LocalLLaMA threads in the week after release describe this working with Qwen 3.6 — usable for vibe-coding sessions, less reliable for very long autonomous chains.

Setup: both Claude Code and OpenCode accept an OpenAI-compatible base URL. Run an Ollama, llama.cpp, or vLLM endpoint locally, point the tool at it.

```
# Ollama route
ollama launch claude --model qwen3.6:27b
ollama launch opencode --model qwen3.6:27b

# llama.cpp route — start a server, then point tools at http://localhost:8080
llama-server -m Qwen3.6-27B-Q4_K_M.gguf -c 200000 --port 8080
```

Which model:

- **Qwen 3.6-27B dense** for instruction-heavy work and multi-step plans. Better instruction-following on long chains than the MoE.
- **Qwen 3.6-35B-A3B** for vibe-coding sessions where speed matters more than precise instruction-following.

Honest caveat: Claude Code has its own system prompt and tool-use conventions. Local models without Claude-specific training will still fail on edge cases the official Claude doesn't. The April 2026 Anthropic admission about reduced reasoning effort somewhat closes the gap — per the Register's coverage, Claude Code's quality regression spanned March 4 through April 20, and local models with full reasoning effort during that window were genuinely competitive on harder tasks.

Frontier-adjacent option: DeepSeek V4-Flash

DeepSeek V4-Flash dropped April 23, 2026: 284B total / 13B active MoE, MIT licensed, 1M context. Per DeepSeek's API pricing page: $0.14 per M input tokens (cache hit and cache miss alike — the prior cache discount was removed in May 2026), $0.28 per M output. Effectively Haiku-tier pricing on a frontier-adjacent model.

For OpenClaw via the DeepSeek API: OpenAI-compatible endpoint, drop-in for any tool that accepts a base URL and key. Best low-cost option for tool-calling-heavy pipelines as of May 2026.

```
# Set OpenClaw to use DeepSeek's endpoint
export OPENAI_BASE_URL=https://api.deepseek.com/v1
export OPENAI_API_KEY=sk-your-key
# Model: deepseek-v4-flash
```

For self-hosting V4-Flash: weights are ~140GB at FP4. Not consumer single-GPU territory. You need a serious homelab — dual RTX 6000 Ada at 96GB combined, or H100 80GB + DDR5 offload, or an Apple Silicon Mac Studio M3 Ultra 192GB. **No independent benchmarks for OpenClaw integration with self-hosted V4-Flash yet as of May 26, 2026.** vLLM has day-one V4 support; GGUF builds from Unsloth and bartowski are pending at time of writing.

Verdict: test V4-Flash via DeepSeek's API first. Local self-hosting only makes sense if your hardware budget already supports 1T-class models or if API dependence is a hard constraint.

Sampling and quantization for agentic work

A few things matter more for agents than for chat:

Sampling parameters. Per Qwen's Hugging Face pages, recommended sampling is **temperature 1.0, top-p 1.0** for both Qwen 3.6 variants. Lower temperature for tool-call-heavy workflows is fine but don't push below 0.6 — the model starts producing degenerate completions on edge cases.

Quantization. Don't go below Q4_K_M for serious agentic work. Q3 quants drop instruction-following accuracy enough to break multi-step pipelines that worked at Q4. Community consensus across r/LocalLLaMA in 2026 is that Q4_K_M is the floor, Q5_K_M is comfortable, Q6_K is the sweet spot when you have the VRAM, Q8_0 is overkill for most cases.

KV cache quantization. Q8_0 KV cache is safe across the board. Q4 KV cache works for Qwen 3.6-35B-A3B at short-to-medium contexts but quality degrades on long agent chains — the dropped precision compounds across many sequential tool calls. If you're running 100K+ context agents, keep KV at Q8.

Thinking modes. Qwen 3.6 inherits the `/think` and `/no_think` modes. For agentic work, leave thinking on for planning steps, off for tool-call execution where you want minimal output tokens. OpenClaw skills can route per-step.

Whitespace gotcha when running llama-server directly. If you're starting `llama-server` yourself (not via `ollama launch openclaw`), Qwen 3.6 silently rejects `--chat-template-kwargs '{"enable_thinking": false}'` when there's whitespace around the colon — tool calls route to `reasoning_content` instead of `content`. Write `{"enable_thinking":false}` with no space. Full detail in Best Local Models for PI Agent.

What still works from the previous generation

Qwen 3.5 isn't dead. If you're already running Qwen 3.5 27B in production, there's no urgent reason to migrate — the agentic capability improvement to 3.6-27B is meaningful but not transformative for non-coding workloads. Migrate when:

- You need the SWE-bench Verified jump (72.4 → 77.2)
- You need Terminal-Bench 2.0 capability (3.6-27B is significantly better)
- Your workload hits Qwen 3.5's instruction-following ceiling on long chains

Other previous-gen picks still in rotation:

- **DeepSeek-R1-Distill-Qwen-32B** — best pure-reasoning local model under 24GB
- **Qwen 2.5 Coder 32B** — still strong for code-only workflows
- **Llama 3.3 70B** — flagship dense, best for 48GB+ tiers
- **GLM-4.6** — long-context specialist, used in some research workflows
- **Kimi K2.6** — 1T-class MoE, server-tier hardware required

---

Realistic Expectations

What local models handle well in 2026

- Routine code generation and review (Qwen 3.6-27B)
- Multi-step automation up to 10-15 tool calls
- Structured data extraction and JSON output
- File management and shell automation
- Privacy-sensitive document processing

Where they still struggle

- Very long autonomous chains (30+ tool calls without supervision)
- Novel problem-solving requiring broad world knowledge
- Self-improvement and capability expansion
- Tasks that benefit from frontier-model intuition on ambiguous instructions

The hardware reality

Local OpenClaw at the capability of Anthropic's intended Claude Code (full reasoning effort, no harness bugs) is genuinely possible on 24GB VRAM with Qwen 3.6-27B in May 2026. The competitive position improved measurably during March/April when Claude Code itself was running degraded — and Anthropic has now fixed those bugs, so the bar moves back up. Local models are closing the gap, just not at the rate hype suggests.

For pure local operation, budget 24GB VRAM minimum. For a hybrid setup (local for routine, API for hard tasks), 16GB plus a DeepSeek V4-Flash API key gets you most of the way for a few dollars a month.

Bottom Line

If you have 24GB+ VRAM: Qwen 3.6-27B (Q4_K_M, ~16.8GB) is the new default. SWE-bench Verified 77.2 per Qwen, Terminal-Bench 2.0 59.3, native 262K context. Tested working with Claude Code and OpenCode via local OpenAI-compatible endpoint.

If you have 16GB VRAM: Qwen 3.6-35B-A3B MoE with llama.cpp + RAM offload. The 3B-active design is what makes 16GB viable on a 35B model. Watch for instruction-following degradation on long agent chains.

If you have 8-12GB VRAM: Qwen 3.5 9B remains the recommended backbone. Honest take: serious agentic work wants more VRAM. Hybrid local-plus-API is the realistic path.

If you don't want local hardware at all: DeepSeek V4-Flash via DeepSeek's API at $0.14/$0.28 per M tokens. Frontier-adjacent capability, MIT-licensed weights as a hedge, OpenAI-compatible drop-in.

```
# Quickest setup — Ollama with Qwen 3.6
ollama launch openclaw --model qwen3.6:27b

# Or pull manually
ollama pull qwen3.6:27b
ollama pull qwen3.6:35b-a3b
```

Local agents are real and useful in May 2026. They're not magic — the model determines what they can do, and bigger models do more. Qwen 3.6 is a real step up for coding-heavy agentic work, ik_llama.cpp's MoE optimizations and Blackwell NVFP4 support are pushing inference speed forward, and the brief Claude Code regression made the local-model competitive position look better than usual. That last effect just reverted; treat the rest as durable.

Updated May 26, 2026 for current DeepSeek V4-Flash pricing (cache-hit discount removed), Qwen 3.6 whitespace template gotcha when using llama-server directly, and cross-link refresh. The April 25 refresh added Qwen 3.6-27B dense, Qwen 3.6-35B-A3B MoE, DeepSeek V4-Flash, and the Anthropic Claude Code regression admission.

A weekly email with every new guide and measured benchmark.

Subscribe — free, no spam

Source: https://insiderllm.com/guides/best-local-models-openclaw/

Free guides for running AI locally