# Best Local Models for OpenClaw Agent Tasks

February 3, 2026 · by Mark Bartlett

Download this guide as PDF

> **Quick Answer:** For OpenClaw on 24GB VRAM, Qwen 3.5 27B (Q4_K_M, ~17GB) is the top pick — it matches GPT-5 mini on SWE-bench (72.4), its 262K native context handles long agent conversations without truncation, and tool calling works in Ollama v0.17.6+. The 35B-A3B MoE variant hits 112 tok/s on an RTX 3090 if you want speed over raw capability. On 16GB, use Qwen 3 14B or DeepSeek-R1-Distill-Qwen-14B. On 8GB, Qwen 3.5 9B (~6.6GB Q4) is the recommended backbone — beats GPT-OSS-120B on GPQA Diamond with working tool calling, vision, and 262K context. Quickest setup: `ollama launch openclaw`. Power users like Wes Roth run Claude Opus 4.5 via API for the orchestrator while using local models and Gemini for specialized sub-tasks. For pure local operation: 7B-9B models handle simple tasks, 14B is marginal, 27B-32B is where agents start working reliably. Budget for 24GB VRAM minimum.

📚 **More on this topic:** OpenClaw Setup Guide · OpenClaw Security Guide · Qwen Models Guide · DeepSeek Models Guide · What Can You Run on 24GB VRAM

OpenClaw doesn't care what model powers it — you can plug in Claude, GPT-4, Gemini, or a local model through Ollama. 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, what VRAM you need, and what the power users are running.

---

## 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 |

| Capability | Why Agents Need It | What Tests It |
|---|---|---|
| **Code generation** | Building skills, debugging, automation | Coding benchmarks, real-world bugs |
| **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 | Context utilization at 8K-32K |

The famous restaurant reservation story from OpenClaw's early days captures this: when OpenTable didn't have availability, the agent autonomously downloaded voice software, called the restaurant, and made the reservation over the phone. That required code generation, tool use, error recovery, and multi-step planning — all in one task.

7B models struggle here. They can chat, but they can't reliably orchestrate complex workflows.

---

## What Power Users Actually Run

### The "Society of Minds" Approach

Wes Roth, one of the most active OpenClaw experimenters, doesn't rely on a single model. His setup uses:

- **Claude Opus 4.5 (via API)** — Main orchestrator for complex tasks
- **Gemini 2.0 Pro** — Specialized queries (Google APIs, YouTube optimization)
- **Local models (Ollama)** — Cost-effective sub-tasks, always-on availability

The insight: Claude Opus struggled with YouTube API efficiency. Gemini, being Google-adjacent, suggested using RSS feeds instead of expensive API calls — a solution Claude didn't surface. Different models have different knowledge and strengths.

**The practical lesson:** Pure local-only is a constraint, not a virtue. The most capable setups use the right model for each task.

### What Wes Roth's Agent Actually Does

In his first 24 hours:

- Set up voice communication (Whisper + 11Labs)
- Created YouTube analytics tools (pulling thousands of videos)
- Built thumbnail analysis (5,700 images analyzed)
- Self-replicated to a VPS
- Generated AI videos on demand
- Created WordPress pages autonomously

All of this was done with Claude Opus as the backbone. When he tried to run everything locally, he noted the limitations.

### The Local-First Community

Others in the OpenClaw community run local-only for privacy or cost reasons. Their reports:

- **32B models (Qwen 3, DeepSeek-R1-Distill-32B):** Work reasonably well for most agent tasks
- **14B models:** Marginal — succeed at simpler tasks, fail at complex chains
- **7B models:** Not recommended for serious agent work

---

## Recommended Models by VRAM Tier

### 8GB VRAM (RTX 3060, 4060)

**Honest assessment:** Limited agent capability. 7B models can handle simple, single-step tasks but struggle with complex workflows.

| Model | Size | Context | Agent Suitability |
|---|---|---|---|
| **Qwen 3.5 9B (Q4)** | ~6.6GB | 262K | **Recommended** — beats GPT-OSS-120B on GPQA Diamond. Vision-capable. Tool calling works in Ollama v0.17.6+. |
| Qwen 3 8B (Q4) | ~5GB | 32K | Basic tasks, will fail on complex chains |
| Llama 3.1 8B (Q4) | ~5GB | 128K | Longer context helps, still limited reasoning |
| DeepSeek-R1-Distill-Qwen-7B | ~5GB | 32K | Better reasoning, smaller capability |

**Recommendation:** Qwen 3.5 9B is the backbone model at this tier. Stronger reasoning, 262K context, built-in vision, and function calling that actually works (the Ollama pipeline mismatch was fixed in v0.17.6). Update Ollama before pulling: `curl -fsSL https://ollama.com/install.sh | sh`. Still best to route complex tasks to an API.

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

## 12GB VRAM (RTX 3060 12GB, 4070)

**Assessment:** Can run 14B models, which handle simple-to-moderate agent tasks.

| Model | Size | Context | Agent Suitability |
|---|---|---|---|
| Qwen 3 14B (Q4) | ~9GB | 32K | Decent all-rounder |
| DeepSeek-R1-Distill-Qwen-14B | ~9GB | 32K | Strong reasoning, good for planning |
| Mistral Nemo 12B (Q4) | ~8GB | 128K | Long context, moderate capability |

**Recommendation:** DeepSeek-R1-Distill-Qwen-14B for reasoning-heavy workflows. Qwen 3 14B for general use.

```
# Best for 12GB — reasoning focus
ollama run deepseek-r1:14b

# Alternative — general purpose
ollama run qwen3:14b
```

## 16GB VRAM (RTX 4060 Ti 16GB, 4080)

**Assessment:** Sweet spot starts here. Can run larger 14B models at higher quantization or squeeze in smaller 30B+ models.

| Model | Size | Context | Agent Suitability |
|---|---|---|---|
| Qwen 3 14B (Q8) | ~15GB | 32K | Higher quality 14B |
| DeepSeek-R1-Distill-Qwen-14B (Q8) | ~15GB | 32K | Best reasoning at this tier |
| Qwen 3 32B (Q4, low context) | ~18GB | 8K | Possible with aggressive settings |

**Recommendation:** DeepSeek-R1-Distill-Qwen-14B at Q8 for best reasoning. The Q8 quantization matters for agent work — fewer errors on complex instructions.

```
# Best for 16GB
ollama run deepseek-r1:14b-q8_0

# Pushing it — needs tuning
ollama run qwen3:32b --ctx 8192
```

## 24GB VRAM (RTX 3090, 4090)

**Assessment:** This is where local agents get practical. 32B models handle most agent tasks reliably.

| Model | Size | Context | Agent Suitability |
|---|---|---|---|
| **Qwen 3.5 27B (Q4_K_M)** | ~17GB | 262K | **Top pick** — SWE-bench 72.4, GPQA Diamond 85.5, native vision. Tool calling works in Ollama v0.17.6+. |
| **Qwen 3.5 35B-A3B (MoE)** | ~17GB | 262K | 112 tok/s on RTX 3090 — faster than most dense 7B models. 3B active params per token. |
| Qwen 3 32B (Q4_K_M) | ~20GB | 32K | Solid all-rounder, 32K context limit |
| DeepSeek-R1-Distill-Qwen-32B | ~20GB | 32K | Excellent reasoning, thinking mode |
| Qwen 2.5 Coder 32B | ~20GB | 32K | Best for code-heavy skills |
| Llama 3.3 70B (Q4, partial offload) | ~40GB | Variable | Possible with CPU offload |

**Recommendation:** Qwen 3.5 27B. SWE-bench 72.4, fits at ~17GB Q4, function calling works properly since Ollama v0.17.6. If you want speed over raw capability, the 35B-A3B MoE variant runs at 112 tok/s on an RTX 3090 — faster than most dense 7B models because only 3B parameters are active per token. The KV cache stays smaller at long contexts thanks to the linear attention layers.

```
# Top pick — tool calling fixed in Ollama v0.17.6+
ollama run qwen3.5:27b

# Speed option — 112 tok/s on 3090, 3B active params
ollama run qwen3.5:35b-a3b
```

```
# For complex reasoning / planning
ollama run deepseek-r1:32b

# For skill creation / coding
ollama run qwen2.5-coder:32b
```

### 48GB+ VRAM (Dual 3090, A6000, etc.)

**Assessment:** Full capability. Can run 70B models that approach API quality.

| Model | Size | Context | Agent Suitability |
|---|---|---|---|
| Llama 3.3 70B (Q4_K_M) | ~40GB | 128K | Flagship open model |
| Qwen 3 72B (Q4) | ~42GB | 32K | Strong alternative |
| DeepSeek-R1-Distill-Llama-70B | ~40GB | 32K | Best open reasoning model |

**Recommendation:** Llama 3.3 70B for general agent work. DeepSeek-R1-Distill-Llama-70B when you need maximum reasoning capability.

```
# Best overall at 48GB
ollama run llama3.3:70b

# Maximum reasoning
ollama run deepseek-r1:70b
```

→ Use our Planning Tool to check exact VRAM for your setup.

---

## Model Comparison for Agent Tasks

### Coding & Skill Creation

When your agent needs to write its own tools:

| Model | Skill Building | Debugging | Self-Improvement |
|---|---|---|---|
| Qwen 3.5 27B | Excellent | Excellent | Very Good |

| Model | Skill Building | Debugging | Self-Improvement |
|---|---|---|---|
| Qwen 3.5 35B-A3B | Very Good | Very Good | Good — 112 tok/s on 3090, trades some quality for speed |
| Qwen 2.5 Coder 32B | Excellent | Excellent | Good |
| Qwen 3 32B | Very Good | Very Good | Very Good |
| DeepSeek-R1-Distill-32B | Good | Very Good | Excellent |
| Llama 3.3 70B | Very Good | Very Good | Good |

**Winner:** Qwen 3.5 27B scores 72.4 on SWE-bench Verified and 80.7 on LiveCodeBench v6 — first consumer-GPU model to match GPT-5 mini on SWE-bench. The 35B-A3B MoE variant trades some quality for speed (112 tok/s on 3090). Qwen 2.5 Coder 32B remains strong for pure code tasks.

## Reasoning & Planning

For multi-step tasks with complex dependencies:

| Model | Planning | Error Recovery | Chain-of-Thought |
|---|---|---|---|
| Qwen 3.5 27B (/think mode) | Excellent | Excellent | Excellent |
| DeepSeek-R1-Distill-32B | Excellent | Excellent | Excellent |
| Qwen 3 32B (/think mode) | Excellent | Very Good | Excellent |
| Llama 3.3 70B | Very Good | Good | Good |
| Qwen 2.5 Coder 32B | Good | Good | Fair |

**Winner:** Qwen 3.5 27B scores 85.5 on GPQA Diamond and 92.0 on HMMT — both top marks at this parameter class. DeepSeek-R1-Distill-32B remains excellent for reasoning-heavy chains.

## Tool Use & Function Calling

For structured output and API calls:

| Model | Function Calling | JSON Output | API Integration |
|---|---|---|---|
| Qwen 3.5 27B | Excellent (native) | Excellent | Excellent (fixed in Ollama v0.17.6+) |

| Model | Function Calling | JSON Output | API Integration |
|---|---|---|---|
| Qwen 3 32B | Excellent | Excellent | Excellent |
| Llama 3.3 70B | Very Good | Very Good | Very Good |
| DeepSeek-R1-Distill-32B | Good | Good | Good |
| Mistral Nemo 12B | Good | Good | Fair |

**Winner:** Qwen 3.5 27B. Native tool-calling support, trained on it, and the Ollama pipeline mismatch was fixed in v0.17.6. Update Ollama if you're on an older version.

---

## Configuring Ollama for OpenClaw

### Basic Setup

```
# Install or update Ollama (v0.17.6+ required for Qwen 3.5 tool calling)
curl -fsSL https://ollama.com/install.sh | sh

# Quickest setup — auto-configures Ollama + OpenClaw connection
ollama launch openclaw

# Or pull manually — Qwen 3.5 27B is the top pick
ollama pull qwen3.5:27b

# Speed option — 112 tok/s on RTX 3090
ollama pull qwen3.5:35b-a3b

# Verify it's working
ollama run qwen3.5:27b "Hello, can you confirm you're working?"
```

### Exposing Ollama to OpenClaw

OpenClaw connects to Ollama's API. By default, Ollama only listens on localhost:

```
# Check Ollama is running
curl http://localhost:11434/api/tags
```

If OpenClaw is on a different machine or in Docker, configure Ollama to listen on all interfaces:

```
# Set environment variable (add to ~/.bashrc or systemd service)
OLLAMA_HOST=0.0.0.0:11434
```

### Optimizing for Agent Workloads

Agents benefit from:

1. **Higher context length** — Conversation history accumulates fast
2. **Consistent output** — Lower temperature for reliable tool calls
3. **Longer timeouts** — Complex tasks take time

In your Ollama modelfile or OpenClaw config:

```
# Example modelfile customization
FROM qwen3.5:27b

PARAMETER temperature 0.7
PARAMETER num_ctx 16384
PARAMETER num_predict 4096
```

---

## Hybrid Approaches

### Local + API (Best of Both)

The pattern power users follow:

1. **Use local for:** Always-on availability, simple tasks, privacy-sensitive operations
2. **Use API for:** Complex reasoning, long chains, tasks requiring maximum capability

Configure OpenClaw to route based on task complexity (requires custom skill development).

### Multi-Model Local Setup

Run different models for different purposes:

```
# Have multiple models available
ollama pull qwen3.5:27b        # Primary — strongest benchmarks, 262K context, tool calling worl
ollama pull qwen3.5:35b-a3b   # Speed option — 112 tok/s on 3090
ollama pull deepseek-r1:32b    # Complex reasoning
ollama pull qwen2.5-coder:32b  # Skill development
```

OpenClaw skills can specify which model to use. A coding skill might route to Qwen Coder while a planning skill routes to DeepSeek-R1.

### The "Society of Minds" Pattern

Wes Roth's approach — multiple models collaborating:

1. **Orchestrator (Claude/GPT-4/Qwen 3.5 27B):** Manages overall task flow
2. **Specialists (Gemini, Coder, etc.):** Handle domain-specific queries
3. **Workers (smaller models):** Execute simple sub-tasks cheaply

This requires custom skill development but produces better results than any single model.

---

## Realistic Expectations

### What Local Models Handle Well

- Simple automation (file management, scheduling)
- Straightforward coding tasks
- Single-step API calls
- Structured data extraction
- Routine inbox triage

### What Local Models Struggle With

- Novel problem-solving (the restaurant phone call story)
- Very long task chains (10+ step workflows)
- Ambiguous instructions requiring inference
- Tasks requiring broad world knowledge
- Self-improvement and capability expansion

### The Hardware Reality

The power users getting impressive results mostly run:

- Claude Opus 4.5 ($15/M input, $75/M output) for complex tasks
- Local models for cost optimization and always-on availability
- Multiple API backends for specialized capabilities

Pure local-only is possible but requires:

- 24GB+ VRAM minimum for reliable agent work
- Acceptance of capability limitations vs frontier APIs
- Willingness to retry failed tasks

---

## Bottom Line

**If you have 24GB+ VRAM:** Qwen 3.5 27B (~17GB Q4) is the top pick — SWE-bench 72.4, 262K native context, tool calling fixed in Ollama v0.17.6+. For speed, the 35B-A3B MoE variant hits 112 tok/s on an RTX 3090. DeepSeek-R1-Distill-32B for complex planning tasks.

**If you have 12-16GB VRAM:** Run DeepSeek-R1-Distill-Qwen-14B or Qwen 3 14B. Expect limitations on complex multi-step tasks. Consider hybrid local + API.

**If you have 8GB VRAM:** Qwen 3.5 9B (~6.6GB Q4) is the recommended backbone — beats GPT-OSS-120B on GPQA Diamond, tool calling works in Ollama v0.17.6+. Still limited for complex agent chains. Route hard tasks to an API.

**The honest take:** The most impressive OpenClaw demos run on Claude Opus via API, not local models. If you want that level of capability locally, budget for serious hardware (48GB+ VRAM) and accept you're still behind the frontier. If you want cost optimization and privacy, local models work for routine agent tasks on 24GB+ VRAM.

```
# Quickest setup — auto-configures everything
ollama launch openclaw

# Or manual setup
ollama pull qwen3.5:27b
ollama run qwen3.5:27b
```

```
# Configure OpenClaw to use it
# In OpenClaw setup: select Ollama, model: qwen3.5:27b
```

Local agents are real and useful. They're not magic — the model determines what they can do, and bigger models do more. Qwen 3.5 is a real step up (262K context, vision, stronger benchmarks at smaller sizes), and the Ollama tool-calling pipeline works correctly as of v0.17.6. Update Ollama before you start: `curl -fsSL https://ollama.com/install.sh | sh`.

Updated March 2026 for Ollama v0.17.7, Qwen 3.5 tool calling fix, and OpenClaw v2026.3.2.

Get notified when we publish new guides.

Subscribe — free, no spam

---