Qwen 3.5: What Local AI Builders Need to Know

February 20, 2026 · by Mark Bartlett

Download this guide as PDF

> **Quick Answer:** Qwen3.5-397B-A17B is a Mixture-of-Experts model with 397B total parameters but only 17B active per forward pass. Competitive with GPT-5.2 and Claude 4.5 Opus on benchmarks. Apache 2.0 license. The catch: you still need to load all 397B params into memory. Minimum viable local config: 192GB Mac at 3-bit quant, or 256GB Mac at 4-bit. On a 24GB GPU with RAM offloading, Unsloth reports ~25 tok/s. If you have less than 192GB total memory, wait for smaller variants.

📚 **More on this topic:** Qwen Models Guide · Qwen 3 Complete Guide · VRAM Requirements · Running LLMs on Mac

Alibaba released Qwen3.5-397B-A17B on February 16, 2026. It's a frontier-class model under Apache 2.0 — open weights, no restrictions. Benchmarks put it competitive with GPT-5.2 and Claude 4.5 Opus. Native multimodal (text and images trained together, not bolted on). 256K context window.

The first question our readers ask: can I run this?

The honest answer: maybe. It depends on how much memory you have, and "memory" means something specific with this architecture.

The MoE Trade-Off

Qwen3.5 uses Mixture-of-Experts (MoE). The model has **397 billion total parameters** across 512 experts, but only **17 billion activate** per forward pass (10 routed experts + 1 shared expert per token). This means inference is fast — you're only computing through 17B parameters per token, roughly equivalent to a dense 17B model in speed.

The catch: all 397B parameters must be loaded into memory. Every expert sits in RAM or VRAM waiting to potentially activate. MoE gives you fast inference at the cost of a massive memory footprint.

This is why "17B active" doesn't mean "17B VRAM." You're paying 397B prices for storage and 17B prices for compute.

Architecture: What's New

Qwen3.5 isn't just a bigger MoE. It introduces a hybrid attention system called **Gated Delta Networks** — a mix of linear attention and standard attention in a 3:1 ratio across 60 layers.

In plain English: 75% of the layers use a faster, more memory-efficient attention mechanism (Gated DeltaNet), while 25% use traditional full attention for accuracy on harder patterns. The result is **8.6x to 19x faster decoding** than Qwen3-Max (their previous flagship) with much lower KV cache memory.

It also uses multi-token prediction and a 248K vocabulary covering 201 languages, which reduces token counts by 10-60% depending on the language. Fewer tokens = less context consumed = more room in the 256K window.

Hardware Requirements

Here's what actually matters. Every number below is total system memory — VRAM plus RAM if you're offloading.

Quantization	Model Size	Minimum Memory	Can Run On
FP16	~800 GB	~800+ GB	Multi-GPU cluster only
FP8	~400 GB	~512 GB	512GB Mac Pro or 8x GPUs
Q6_K	~300 GB	~384 GB	512GB Mac or 4x 80GB GPUs
Q4_K_XL	**~214 GB**	**~256 GB**	**256GB Mac Studio**
Q3_K_XL	~176 GB	~192 GB	192GB Mac Studio/Pro
Q2_K_XL	~130 GB	~150 GB	192GB Mac (with headroom)

The Mac Sweet Spot

If you're going to run this locally, Apple Silicon with high unified memory is the most accessible path:

- **M3/M4 Ultra 256GB** — runs Q4_K_XL (the recommended quant). This is the sweet spot.
- **M3/M4 Ultra 192GB** — runs Q3_K_XL. Quality drop is noticeable vs Q4 but still very usable.

- **M4 Max 128GB** — too small for any reasonable quant of this model. Wait for smaller variants.

Expected performance: ~5-8 tok/s generation based on DGX Spark benchmarks and Apple Silicon memory bandwidth (~800 GB/s on M4 Ultra). Usable for interactive chat, not fast for batch processing.

The GPU Path

- **1x RTX 4090 24GB + 64GB+ system RAM** — possible with MoE offloading (expert weights stay in RAM, activated experts get moved to GPU). Unsloth reports **~25 tok/s** with this setup using UD-Q4_K_XL. Prompt processing will be slow, but generation is surprisingly decent because only 17B params activate.
- **2x RTX 3090/4090** — same approach, slightly better prompt processing.
- **4x 80GB GPUs (A100/H100)** — runs Q6_K or FP8 entirely in VRAM. Production-grade but not consumer hardware.

Who Should Skip It

If you have less than 192GB total memory (VRAM + RAM), this model isn't for you yet. A 14B dense model on your 24GB GPU will give you better quality per dollar than a heavily quantized 397B squeezed into insufficient memory. Wait for distilled smaller variants — Alibaba has a history of releasing the flagship first, then smaller versions within weeks.

→ Check what fits your hardware with our Planning Tool.

How to Run It Locally

llama.cpp (GGUF)

Unsloth quants are available now: unsloth/Qwen3.5-397B-A17B-GGUF

```
# Download the recommended Q4 quant (~214GB)
huggingface-cli download unsloth/Qwen3.5-397B-A17B-GGUF \
  --local-dir Qwen3.5-GGUF \
  --include "*UD-Q4_K_XL*"

# Run with llama.cpp (Mac or GPU offload)
```

```
llama-cli -m Qwen3.5-GGUF/Qwen3.5-397B-A17B-UD-Q4_K_XL.gguf \
  -c 8192 -ngl 999
```

LM Studio community quants are also available on HuggingFace.

Important: Make sure you're on the latest llama.cpp build. A CUDA bug (resolved in #19683) caused degenerate output on NVIDIA GPUs when using `-DGGML_CUDA_FORCE_CUBLAS=ON`. If your build uses that flag, rebuild without it.

Ollama

Cloud-only right now. Ollama has `qwen3.5` tags, but they route to Alibaba's hosted API — not local GGUF inference. Local Ollama support is pending while llama.cpp backend bugs for this architecture get resolved.

If you need Ollama specifically, you can create a custom Modelfile pointing to a downloaded GGUF, but expect rough edges until the `qwen35moe` architecture type stabilizes in llama.cpp.

vLLM

For multi-GPU setups, vLLM supports Qwen3.5 with tensor parallelism. If you have the GPU memory to run it in FP8 or FP16 across multiple cards, vLLM will give you the best throughput for serving.

Benchmarks: Where It Wins

Benchmark	Qwen3.5-397B	Category
MMLU-Redux	94.9	Language understanding
GPQA Diamond	88.4	Graduate-level reasoning
AIME26	91.3	Math competition
LiveCodeBench v6	83.6	Coding
SWE-bench Verified	76.4	Software engineering
MMMU	85.0	Multimodal understanding
OmniDocBench v1.5	90.8	Document analysis

Alibaba claims it outperforms GPT-5.2, Claude 4.5 Opus, and Gemini 3 Pro on 80% of evaluated categories. Whether those claims hold across real-world usage is a different question — benchmarks and practical performance don't always align. But the numbers are strong enough that this is genuinely frontier-class.

The native multimodal capability is worth highlighting. Qwen3.5 was trained on text and images as a unified stream from the start — not a language model with a vision encoder bolted on afterward. UI screenshot understanding, document analysis, and visual reasoning should all be stronger than add-on approaches.

Known Issues (Early Days)

This model is a week old. Expect early bugs.

Forced prompt reprocessing (#19690): The multimodal model assertion prevents KV cache reuse, forcing full prompt reprocessing on every generation. This kills performance in multi-turn conversations. A fix PR exists but hasn't merged. Workaround: don't pass `--mmproj` if you're only doing text.

Multi-GPU crashes (#19676): Long prompts (~20K+ tokens) with `--op-offload` on multi-GPU setups cause segfaults. Still open.

Thinking mode default: Qwen3.5 generates chain-of-thought reasoning tokens by default. If you don't want the thinking overhead, disable it explicitly in your API calls. Otherwise, responses include hidden reasoning tokens that consume context and slow generation.

Repetition: Some users report the model falling into loops. Workaround: set `--presence-penalty` up to 2.0 or `--repeat-penalty 1.15`.

The Coding Angle: Qwen3-Coder-Next

If you want Qwen's coding power on consumer hardware, the model to look at is Qwen3-Coder-Next — released separately in early February.

- 80B total / 3B active (same MoE approach, much smaller)
- 256K context, Apache 2.0
- 70%+ on SWE-Bench Verified
- Runs on Ollama: `ollama run qwen3-coder-next`

On a 12GB GPU with MXFP4 quant, people report 23 tok/s at 131K context. On 8GB VRAM, it technically runs but at ~1.2 tok/s — not practical. The sweet spot is 24GB VRAM with system RAM offloading.

For most local AI builders, Qwen3-Coder-Next is the Qwen model to run today. Qwen3.5-397B is the model to aspire to.

Who Should Care

256GB Mac Studio/Pro owners: This is your moment. A frontier-class model that actually fits on your hardware at Q4. Nothing else in this weight class runs on a single machine this accessible.

Multi-GPU server operators: If you're running 2-4 high-VRAM GPUs for local inference, Qwen3.5 at Q4-Q6 gives you a model that competes with the best commercial APIs.

Anyone already running Qwen3-235B: Qwen3.5 is the upgrade. Better benchmarks, faster decoding (8-19x), better multimodal, same open license.

Everyone else: Watch this space. Alibaba will likely release smaller Qwen3.5 variants. In the meantime, Qwen3 8B/14B/32B remain the best local models for consumer hardware, and Qwen3-Coder-Next handles coding on modest GPUs.

Bottom Line

Qwen3.5-397B-A17B is the most capable open-weights model released to date. Apache 2.0. Native multimodal. 256K context. Competitive with the best commercial models on benchmarks.

The MoE architecture makes it surprisingly fast once loaded — 17B active means generation speed is closer to a 14B dense model than a 397B one. But "once loaded" requires 192-256GB of memory. That's a Mac Studio, a multi-GPU server, or a single GPU with aggressive RAM offloading.

If you have the hardware, this is the first time a truly frontier model has been available for local inference with no API key and no license restrictions. If you don't have the hardware, Qwen3 32B at Q4 on a 24GB GPU is still the practical sweet spot — and it's excellent.

The future of local AI isn't just small models on laptops. It's frontier models on machines you own. Qwen3.5 is the first real proof of that.

Source: https://insiderllm.com/guides/qwen-3-5-local-guide/

Free guides for running AI locally