Mac Runs 70B Models That Need Multi-GPU on PC — Here's How

February 3, 2026 · by Mark Bartlett

Download this guide as PDF

> **Quick Answer:** Every M-series Mac can run local LLMs — unified memory means your entire RAM is available for model loading. An M1 MacBook Air with 8GB runs 3B-7B models well. An M4 Pro Mac Mini with 24-48GB handles 14B-32B. An M4 Max Mac Studio with 128GB runs 70B+ models that would need multiple GPUs on PC. Both Ollama (v0.17+) and LM Studio (v0.3.38+) now use MLX natively on Mac for maximum speed. Expect 30-50% slower token generation than an RTX 3090, but you can load much larger models.

📚 **More on this topic:** Mac vs PC for Local AI · Ollama vs LM Studio · Run Your First Local LLM · VRAM Requirements

Apple Silicon Macs have a superpower for local AI: unified memory. Unlike a PC where GPU VRAM is separate (and limited to 24GB on consumer cards), your Mac's entire RAM pool is available to both CPU and GPU. An M4 Max with 128GB can load models that would require a $3,000+ multi-GPU PC setup.

The tradeoff is speed — an RTX 3090 generates tokens faster for models that fit in its 24GB. But for models that don't fit in 24GB, Mac wins by running them at all.

This guide covers what you can actually run on each M-series chip, which tools to use, realistic performance expectations, and how to set up a Mac Mini as an always-on AI server.

M-Series Chips at a Glance

Chip	Memory Options	Memory Bandwidth	GPU Cores	Best For
M1	8-16 GB	68.25 GB/s	7-8	3B-7B models, light use
M1 Pro	16-32 GB	200 GB/s	14-16	8B-14B models
M1 Max	32-64 GB	400 GB/s	24-32	14B-32B models
M1 Ultra	64-128 GB	800 GB/s	48-64	32B-70B models

Chip	Memory Options	Memory Bandwidth	GPU Cores	Best For
M2	8-24 GB	100 GB/s	8-10	7B-8B models
M2 Pro	16-32 GB	200 GB/s	16-19	8B-14B models
M2 Max	32-96 GB	400 GB/s	30-38	14B-32B models
M2 Ultra	64-192 GB	800 GB/s	60-76	32B-70B+ models
M3	8-24 GB	100 GB/s	8-10	7B-8B models
M3 Pro	18-36 GB	150 GB/s	11-14	8B-14B models
M3 Max	36-128 GB	300-400 GB/s	30-40	14B-70B models
M4	16-32 GB	120 GB/s	10	7B-14B models
M4 Pro	24-64 GB	273 GB/s	16-20	14B-32B models
M4 Max	36-128 GB	546 GB/s	40	32B-70B+ models

Key insight: Memory bandwidth determines how fast tokens generate. The M4 Max at 546 GB/s generates tokens roughly 5x faster than the base M4 at 120 GB/s for the same model.

What Can You Run by Memory Tier

8GB Unified Memory

Models that fit: 3B-7B at Q4 quantization

Model	Size	Performance
Llama 3.2 3B	~2 GB	25-35 tok/s (M1), 30-45 tok/s (M4)
Phi-4 Mini	~2.3 GB	25-40 tok/s
Qwen 3.5 4B Q4	~2.5 GB	25-40 tok/s
Mistral 7B Q4	~4.5 GB	12-18 tok/s (tight fit)
Llama 3.1 8B Q3	~4 GB	10-15 tok/s (quality tradeoff)

With 8GB, you're in small model territory. The system needs 2-3GB for macOS itself, leaving 5-6GB for models. 7B models at Q4 (~4.5GB) fit but leave little room for context. Stick to 3B models for comfortable use, or use aggressive quantization (Q3/Q2) for 7B.

Realistic experience: 8GB Macs work for casual use with small models. Don't expect to run coding assistants or models that need long context.

16GB Unified Memory

Models that fit: 7B-8B comfortably, 13B-14B at Q4 (tight)

Model	Size	Performance
Qwen 3.5 9B Q4	~5.5 GB	25-40 tok/s
Llama 3.1 8B Q4	~4.5 GB	25-40 tok/s
Mistral 7B Q6	~5.5 GB	25-40 tok/s
DeepSeek R1 Distill 8B	~4.5 GB	25-40 tok/s
Llama 3.1 14B Q4	~8 GB	15-25 tok/s (needs reduced context)

16GB is the sweet spot for 8B-9B models. You have room for the model plus healthy context (8K-16K tokens). 14B models fit but require reduced context length (4K or less) or lower quantization.

Realistic experience: This is the minimum for serious local LLM use. Qwen 3.5 9B is the strongest pick here right now — better instruction-following and reasoning than Qwen 2.5 7B, with a `/think` mode for chain-of-thought when you need it. Llama 3.1 8B and DeepSeek R1 Distill 8B are solid alternatives.

24GB Unified Memory

Models that fit: 8B at high quality, 14B comfortably, 32B at Q3-Q4 (tight)

Model	Size	Performance
Qwen 3.5 9B Q6	~7.5 GB	25-45 tok/s
Llama 3.1 8B Q8	~8.5 GB	25-45 tok/s
Mistral Nemo 12B Q4	~7.5 GB	20-35 tok/s
Qwen 2.5 14B Q4	~8.5 GB	18-30 tok/s
DeepSeek R1 Distill 14B Q4	~8.5 GB	18-30 tok/s

24GB opens up the 14B tier properly. You can run DeepSeek R1 Distill 14B and Qwen 2.5 14B with room for 8K+ context. Or run Qwen 3.5 9B at Q6 for higher quality than Q4 with headroom to spare.

Realistic experience: This is the Mac Mini M4 Pro base config and a great entry point. 14B models offer noticeably better reasoning than 8B.

36-48GB Unified Memory

Models that fit: 14B at high quality, 32B comfortably, 70B at Q2-Q3 (tight)

Model	Size	Performance
Qwen 2.5 14B Q8	~15 GB	18-35 tok/s
Qwen 2.5 32B Q4	~20 GB	12-22 tok/s
Llama 3.3 70B Q2	~30 GB	5-10 tok/s (quality tradeoff)
DeepSeek R1 Distill 32B Q4	~20 GB	12-22 tok/s
Mixtral 8x7B Q4	~26 GB	15-25 tok/s

The 32B tier becomes practical. Models like Qwen 2.5 32B and DeepSeek R1 Distill 32B run with good context windows. 70B is technically possible at Q2 but the quality loss is significant.

Realistic experience: 32B models are a major step up in capability. This is where you start getting expert-level responses on complex topics.

64-96GB Unified Memory

Models that fit: 32B at high quality, 70B at Q4 comfortably

Model	Size	Performance
Qwen 2.5 32B Q6	~26 GB	12-25 tok/s
Llama 3.1 70B Q4	~40 GB	8-15 tok/s
Qwen 2.5 72B Q4	~42 GB	8-14 tok/s
DeepSeek R1 Distill 70B Q4	~40 GB	8-14 tok/s
Mixtral 8x22B Q4	~80 GB	5-10 tok/s

This is the sweet spot for 70B models. You have room for 40GB of model weights plus generous context (32K+ tokens). The M3 Max 96GB and M4 Max 64GB configurations hit this tier.

Realistic experience: 70B models are genuinely impressive — they match or exceed GPT-3.5 on most tasks. 8-15 tok/s is slower than reading speed but perfectly usable for interactive chat.

128GB+ Unified Memory

Models that fit: 70B at high quality, 100B+

Model	Size	Performance
Llama 3.1 70B Q6	~55 GB	8-15 tok/s
Qwen 2.5 72B Q8	~75 GB	8-12 tok/s
Qwen 3 235B Q3	~88 GB	3-5 tok/s
Llama 3.1 405B Q2	~150 GB	Not practical (too slow)

At 128GB, you're in territory no single consumer GPU can reach. The M4 Max 128GB configuration costs ~$3,500 and runs 70B models that would require $1,600+ in dual GPUs on PC. For models above 100B parameters, you're looking at M3 Ultra with 192GB+ ($5,500+).

Realistic experience: This is the "money is no object, I want the biggest models" tier. 70B at Q6/Q8 is noticeably better than Q4. Models above 100B are possible but slow.

→ Check what fits your hardware with our Planning Tool.

Qwen 3.5 on Mac: The New Default

The Qwen 3.5 family fills out the Mac model lineup better than anything else right now. Every Qwen 3.5 model is natively multimodal (text + images + video), has 262K context, and uses a Gated DeltaNet architecture that keeps KV-cache lean at long contexts. Both Ollama (0.17.4+) and MLX support them.

Mac Config	Best Qwen 3.5 Model	Performance	Notes
Any M-series, 8GB	9B Q4 (~5 GB)	20-40 tok/s	The 8GB default. Leaves room for macOS
M4, 16GB	9B Q6 (~7.5 GB)	25-45 tok/s	Higher quality than Q4
M4 Pro, 24GB+	35B-A3B Q4 (~20 GB)	~15 tok/s	MoE, 3B active. Fast for its quality
M4 Pro, 48GB	27B Q4 (~16 GB)	18-30 tok/s	Dense, room for long context

Mac Config	Best Qwen 3.5 Model	Performance	Notes
M4/M5 Max, 128GB	122B-A10B Q4 (~70 GB)	8-12 tok/s	Best open model for tool use
M3 Ultra, 192GB+	397B-A17B Q4 (~214 GB)	3-5 tok/s	Flagship, tight fit

The 9B runs on any M-series Mac. At Q4 it needs about 5GB, and even the base M1 with 8GB has enough headroom after macOS takes its share. This is the model to start with if you haven't tried Qwen 3.5 yet.

The 35B-A3B is the Mac sweet spot for 24GB+. Despite holding 35B total parameters, it only activates 3B per token, so generation is fast. On a MacBook Air M4 with 24GB unified memory, expect roughly 15 tok/s. On an M4 Pro with 48GB, you can run it at Q6 for better quality.

The 397B-A17B fits on M3 Ultra at Q4. At ~214GB, it needs the 192GB M3 Ultra (or theoretical future M5 Ultra). Only 17B parameters fire per token, so it's not as slow as you'd expect for its size, but it's still 3-5 tok/s. This is "I want to run the frontier model on my desk" territory.

Ollama version matters. Qwen 3.5 uses the Gated DeltaNet architecture, which requires Ollama 0.17.4 or later. Check with `ollama --version` and update if needed. MLX support came through the `mlx-community` HuggingFace models.

```
# Verify Ollama version
ollama --version

# Pull the best model for your Mac
ollama run qwen3.5:9b       # 8-16GB unified memory
ollama run qwen3.5:35b-a3b  # 24GB+ unified memory
ollama run qwen3.5:27b      # 48GB+ unified memory (dense alternative)
```

Which Tool Should You Use?

Ollama: Simplicity

Ollama is the easiest way to run local LLMs on Mac. One install, one command, you're running. As of v0.17, Ollama includes a built-in MLX engine on Mac, so it's no longer slower than running MLX directly.

```
# Install
curl -fsSL https://ollama.com/install.sh | sh

# Run a model
ollama run qwen3.5:9b
```

Pros:

- Dead simple to use
- MLX engine built in (v0.17+) — same speed as native MLX
- Handles model downloads and updates
- API server for integrations

Cons:

- Less control over advanced settings
- Model library limited to what's on ollama.com

Use Ollama when: You want to run models with minimal setup, or you need an API server for other apps.

LM Studio: Visual Interface

LM Studio (v0.3.38+) gives you a ChatGPT-style interface with full control over settings. Recent versions ship with a native MLX engine on Mac, so it's no longer just a llama.cpp wrapper.

Pros:

- Nice GUI with conversation management
- Direct HuggingFace model downloads
- Native MLX engine on Apple Silicon — same speed as raw MLX
- Fine control over temperature, context, sampling
- Built-in server mode

Cons:

- Heavier than command-line tools

Use LM Studio when: You prefer a visual interface, want to browse and download models directly, or need precise parameter control.

MLX: Maximum Speed (Now Built Into the Others)

MLX is Apple's machine learning framework built specifically for unified memory. It's 20-50% faster than llama.cpp on Apple Silicon — and as of early 2026, both Ollama and LM Studio use it as their default Mac engine.

You can still run MLX directly if you want full control:

```
# Install
pip install mlx-lm

# Download and run
mlx_lm.generate --model mlx-community/Qwen3.5-9B-4bit --prompt "Hello"
```

Pros:

- Fastest inference on Apple Silicon
- Zero-copy unified memory access — no CPU/GPU transfer overhead
- Growing model library (mlx-community on HuggingFace)

Cons:

- Requires Python knowledge
- Less plug-and-play than Ollama

Use MLX directly when: You're building applications, need fine-grained control over inference, or want to run models not yet on Ollama's registry.

Speed Comparison

Tool	8B Q4 on M4 Max	Notes
MLX (direct)	~95-110 tok/s	Baseline
Ollama 0.17+ (MLX engine)	~90-105 tok/s	Minimal overhead vs raw MLX
LM Studio 0.3.38+ (MLX engine)	~90-105 tok/s	Same MLX engine, GUI wrapper

The old 20-30% gap between MLX and Ollama/LM Studio is mostly gone. All three tools now use MLX on Mac, so pick whichever interface you prefer.

Metal Acceleration: What You Need to Know

Metal is Apple's GPU framework, equivalent to NVIDIA's CUDA. Every M-series Mac supports Metal, and every local LLM tool uses it automatically.

You don't need to do anything to enable it. When you run Ollama, LM Studio, or MLX, Metal acceleration is on by default.

Verifying Metal Is Working

In Ollama:

```
ollama ps
# Should show "GPU" in the Processor column, not "CPU"
```

In LM Studio: Check the bottom status bar — it shows GPU usage.

In Activity Monitor: Open GPU History (Window → GPU History). You should see activity when generating.

When Metal Doesn't Work

Metal issues are rare, but can happen:

- **macOS version too old:** Metal for LLMs requires macOS 12.6+ (M1) or 13.3+ (M2/M3/M4). Update if you're behind.
- **Tool version too old:** Update Ollama/LM Studio to the latest version.
- **Memory pressure:** If macOS is swapping heavily, performance collapses. Close other apps.

Realistic Performance Expectations

Let's be honest about what you're getting.

Speed vs. NVIDIA

Model	M4 Max 40c	RTX 3090	Difference
8B Q4	~83 tok/s	~100 tok/s	NVIDIA 20% faster

Model	M4 Max 40c	RTX 3090	Difference
14B Q4	~38 tok/s	~55 tok/s	NVIDIA 45% faster
32B Q4	~20 tok/s	~40 tok/s	NVIDIA 100% faster
70B Q4	~10 tok/s	~3 tok/s (offload)	Mac 3x faster

For models up to 32B, an RTX 3090 is faster. At 70B+, Mac wins because NVIDIA has to offload to system RAM over PCIe, killing performance.

What Speeds Feel Like

Speed	Experience
80+ tok/s	Instant — faster than you can read
40-80 tok/s	Very responsive — slight typing delay
20-40 tok/s	Comfortable — noticeable but not annoying
10-20 tok/s	Acceptable — clear delay but usable
5-10 tok/s	Slow — patience required
<5 tok/s	Painful — only for batch jobs

Most Mac users land in the 15-50 tok/s range depending on model size. That's perfectly usable for interactive chat.

Prompt Processing (Prefill)

Prompt processing — how fast the model reads your input — is where Mac struggles most. NVIDIA's compute advantage is 5-10x here.

For short prompts, you won't notice. For RAG with long documents or code analysis with large files, prefill times can be frustrating on Mac.

Prompt Length	M4 Max	RTX 3090
500 tokens	1-2 sec	<1 sec
2,000 tokens	3-5 sec	<1 sec
8,000 tokens	15-30 sec	2-4 sec

Mac Mini as an Always-On AI Server

The Mac Mini is quietly one of the best local AI servers you can buy:

- **Small and silent:** No fan noise at idle, quiet under load
- **Low power:** 5-15W idle, 30-60W under AI load
- **Unified memory:** Run larger models than any GPU-based server
- **macOS reliability:** Set it up and forget it

Recommended Configuration

Use Case	Config	Price	What It Runs
Budget AI server	Mac Mini M4 16GB	$599	7B-8B models
Balanced	Mac Mini M4 Pro 24GB	$1,399	8B-14B models
Power user	Mac Mini M4 Pro 48GB	$1,799	14B-32B models
Maximum	Mac Mini M4 Pro 64GB	$1,999	32B models, 70B tight

For most home server use, the Mac Mini M4 Pro 48GB at $1,799 hits the sweet spot — enough memory for 32B models and quiet enough to sit in your living room.

Setup as a Headless Server

1. **Enable Remote Login:** System Settings → General → Sharing → Remote Login
2. **Install Ollama:**

```
curl -fsSL https://ollama.com/install.sh | sh
```

3. **Enable network access:**

```
launchctl setenv OLLAMA_HOST "0.0.0.0:11434"
```

4. **Set to always-on:** System Settings → Energy → Prevent automatic sleeping

Now you can access your Mac Mini from any device on your network:

```
curl http://mac-mini-ip:11434/api/generate -d '{"model": "llama3.1", "prompt": "Hello"}'
```

Power Consumption

State	Power Draw	Annual Cost (US avg $0.12/kWh)
Idle	5-7W	~$6/year
Light AI load	15-25W	~$20/year
Heavy AI load	30-60W	~$35/year

Compare to a PC with an RTX 3090: 150-350W under AI load, ~$200-400/year. The Mac Mini is dramatically cheaper to run 24/7.

Tips for Better Performance

1. Close Memory-Hungry Apps

Safari, Chrome, and Electron apps (Slack, Discord, VS Code) consume significant memory. Each GB they use is a GB you can't use for models.

Check memory pressure in Activity Monitor. If it's yellow or red, close applications.

2. Use Appropriate Quantization

Don't run Q8 models if Q4 fits your needs. The quality difference is subtle; the memory savings are substantial.

Quantization	Quality	Memory vs FP16
Q8_0	~99%	~50%
Q6_K	~98%	~42%
Q5_K_M	~96%	~35%
Q4_K_M	~94%	~28%
Q3_K_M	~88%	~22%

For most tasks, Q4_K_M is the sweet spot. Use Q5 or Q6 for coding or reasoning-heavy tasks where precision matters.

3. Adjust Context Length

Longer context = more memory. If you're hitting limits, reduce context:

```
ollama run llama3.1 /set parameter num_ctx 4096
```

4K context is plenty for most conversations. Only use 8K+ when you actually need it.

4. Use MLX for Speed-Critical Workflows

If you're running the same model repeatedly and every millisecond counts, MLX's 20-30% speed advantage adds up.

5. Keep macOS Updated

Apple regularly improves Metal performance. macOS updates have delivered meaningful speed improvements for AI workloads.

Troubleshooting

Model Won't Load (Out of Memory)

Your model is too large for available memory.

Fixes:

1. Close other applications
2. Use a smaller quantization (Q4 instead of Q6)
3. Use a smaller model
4. Reduce context length

Check available memory: Activity Monitor → Memory → Memory Pressure should be green.

Painfully Slow Generation

Check that Metal is being used (see verification section above). If it's on CPU, something is wrong.

Common causes:

- macOS too old — update
- Tool misconfigured — reinstall
- Heavy memory pressure — close apps

Garbled or Wrong Output

Usually a corrupted model download.

Fix:

```
ollama rm llama3.1
ollama pull llama3.1
```

For more issues, see our Local AI Troubleshooting Guide.

Which Mac Should You Buy for Local AI?

Budget	Recommendation	What You Get
$599	Mac Mini M4 16GB	7B-8B models, basic use
$1,399	Mac Mini M4 Pro 24GB	8B-14B models, good balance
$1,799	Mac Mini M4 Pro 48GB	14B-32B models, best value for AI
$2,700	Mac Studio M4 Max 64GB	32B-70B models
$3,500	Mac Studio M4 Max 128GB	70B+ models, no compromises
$4,000+	MacBook Pro M4 Max 48-128GB	Portable large model inference

The sweet spot: Mac Mini M4 Pro 48GB at $1,799. It runs 32B models comfortably, is silent, uses minimal power, and costs less than a used RTX 3090 PC.

The Bottom Line

Apple Silicon Macs are genuinely good for local LLMs. The unified memory architecture lets you load models that would require expensive multi-GPU setups on PC.

The honest tradeoffs:

- Mac is slower than NVIDIA for models that fit in 24GB VRAM
- Mac wins for models larger than 24GB (70B+)
- Mac is simpler — no driver issues, no CUDA version conflicts
- Mac is quieter and more power-efficient

For most Mac users:

- 8GB: Stick to 3B models
- 16GB: 8B-9B models work great (Qwen 3.5 9B is the current standout)
- 24GB+: 14B models are practical
- 48GB+: 32B models shine
- 64GB+: 70B models become accessible
- 128GB: No practical model limits

Install Ollama, download Qwen 3.5 9B or Llama 3.1, and start chatting. Your Mac is already an AI workstation.