What Can You Actually Run on 4GB VRAM?

January 30, 2026 · by Mark Bartlett

Download this guide as PDF

> **Quick Answer:** 4GB VRAM runs 1B-3B models comfortably at 18-55+ tokens per second. Qwen 2.5 3B at Q4 is the sweet spot — it fits with room for context and genuinely handles chat, summarization, and simple coding. You cannot run 7B models at usable quality (Q2 is terrible), and image generation is limited to SD 1.5 at 512x512 with optimization flags. For anything bigger, either go CPU-only with 16-32GB RAM or spend $150-200 on a used RTX 3060 12GB. A 4GB GPU is a starting point, not a destination.

📚 **More on this topic:** Best Models Under 3B Parameters · CPU-Only LLMs · 8GB VRAM Guide

Let's be direct: 4GB of VRAM is not a lot. It was entry-level five years ago, and it's the absolute floor for local AI today. But "floor" doesn't mean "useless." If you've got a GTX 1050 Ti sitting in an old PC or a GTX 1650 in a gaming laptop, you can do more than you'd expect — as long as you pick the right models and don't try to punch above your weight class.

This guide covers exactly what fits, what runs well, what barely works, and when you're better off going CPU-only or upgrading. No sugarcoating.

Which GPUs Have 4GB?

GPU	CUDA/Stream Cores	Memory Bandwidth	Used Price	Notes
GTX 1050 Ti	768 CUDA	112 GB/s	$65-80	Most common 4GB card. Pascal architecture.
GTX 1650	896 CUDA	128 GB/s	$65-75	Slightly faster. Turing, but no tensor cores.
RX 570 (4GB)	2048 stream	224 GB/s	$45-65	Higher bandwidth but AMD ROCm dropped Polaris — must use Vulkan.
RX 580 (4GB)	2304 stream	256 GB/s	$50-55	Same ROCm problem. Use llama.cpp with Vulkan backend.

The AMD cards have 2x the memory bandwidth of the NVIDIA cards, which theoretically means faster token generation. But Ollama doesn't support Vulkan, so you'll need to use llama.cpp directly. If that's not your thing, stick with NVIDIA — Ollama just works with CUDA.

What Actually Fits in 4GB

Here's the math. Your 4GB needs to hold model weights + KV cache + ~500MB of runtime overhead. That leaves roughly 3.5GB for the model and its context.

Model	Quant	Weight Size	Fits in 4GB?	Room for Context?
Qwen 2.5 0.5B	Q4_K_M	491 MB	Easily	Plenty (32K+ tokens)
Qwen 2.5 0.5B	Q8_0	676 MB	Easily	Plenty
Llama 3.2 1B	Q4_K_M	~700 MB	Easily	Plenty
Qwen 2.5 1.5B	Q4_K_M	1.12 GB	Yes	Good (16K+ tokens)
Qwen 2.5 1.5B	Q8_0	1.89 GB	Yes	Moderate (4-8K tokens)
Gemma 2 2B	Q4_K_M	1.71 GB	Yes	Moderate
Qwen 2.5 3B	Q4_K_M	2.10 GB	Yes	Limited (2-4K tokens)
Llama 3.2 3B	Q4_K_M	~1.8 GB	Yes	Limited (2-4K tokens)
Phi-3.5 Mini 3.8B	Q4_K_M	~2 GB	Tight	Very limited
Qwen 2.5 3B	Q8_0	3.62 GB	Barely	Almost none
Any 7B	Q2_K	~3.0 GB	Technically	Unusable context + terrible quality
Any 7B	Q4_K_M	~4.0 GB	No	—

The sweet spot for 4GB: Qwen 2.5 3B at Q4_K_M. It's 2.1GB, leaves ~1.4GB for KV cache and overhead, and the model is genuinely capable.

The trap: 7B at Q2. Yes, a Q2_K 7B model fits in ~3GB. But Q2 quality is severely degraded — outputs are incoherent more often than not. Multiple benchmarks show a 3B model at Q4 outperforms a 7B at Q2. Don't bother.

→ Check what fits your hardware with our Planning Tool.

Real-World Performance

These speeds come from benchmarks on the Quadro P1000 (4GB, comparable to GTX 1050 Ti class) and bandwidth-based estimates for the GTX 1650:

Model	Quant	GTX 1050 Ti (est.)	GTX 1650 (est.)
Qwen 2.5 0.5B	Q4	~50-55 tok/s	~55-65 tok/s
TinyLlama 1.1B	Q4	~55-62 tok/s	~60-70 tok/s
Llama 3.2 1B	Q4	~25-30 tok/s	~30-40 tok/s
Qwen 2.5 1.5B	Q4	~30-35 tok/s	~35-45 tok/s
Gemma 2 2B	Q4	~18-20 tok/s	~22-28 tok/s
Qwen 2.5 3B	Q4	~17-20 tok/s	~22-30 tok/s
Llama 3.2 3B	Q4	~18-20 tok/s	~25-32 tok/s
Phi-3.5 Mini 3.8B	Q4	~17-19 tok/s	~20-25 tok/s

Everything above 15 tok/s feels responsive for chat. You're fine with any model up to 3B on these cards.

What Works Well on 4GB

These use cases are genuinely practical on a 4GB GPU:

Chat and Q&A (3B models): Qwen 2.5 3B and Llama 3.2 3B handle general conversation, question answering, and brainstorming well. You won't mistake it for GPT-4, but for quick local tasks it's solid.

Simple coding assistance: Qwen 2.5 3B Instruct handles code completion, explaining functions, and writing short scripts. Not full-project coding, but useful for autocomplete and quick snippets.

Text classification and extraction: Small models excel at structured tasks like sentiment analysis, entity extraction, and categorization. A 1.5B model at Q8 runs at 30+ tok/s and handles these well.

Summarization (short docs): Feed in a few paragraphs and get a summary. Keep your context window short — 2-4K tokens is the practical limit on 4GB with a 3B model.

Embeddings for RAG: Embedding models like `nomic-embed-text` (270MB) or `all-minilm` (80MB) fit easily. You can run embeddings on GPU while keeping the LLM on CPU if needed. See our RAG guide for setup.

What Doesn't Work on 4GB

Be honest with yourself about these limitations:

7B+ models at usable quality: You can't run Llama 3.1 8B, Mistral 7B, or Qwen 2.5 7B at Q4 or above — they simply don't fit. Q2/Q3 fits but the quality is so degraded it's not worth using. Go CPU-only instead.

Image generation (mostly): SDXL needs 6GB+ and won't load. Flux is out of the question. SD 1.5 technically works at 512x512 with `--medvram` flags, but you're looking at 30-80 seconds per image on a GTX 1050 Ti. It's possible, but painful. See our Stable Diffusion guide for details.

Long context: With a 3B model at Q4 taking ~2GB of VRAM, you have about 1-1.5GB left for KV cache. That's roughly 2-4K tokens of context with FP16 KV cache. You can stretch to 6-8K tokens with Q8 KV cache quantization, but forget about 16K+ conversations.

Running model + other GPU tasks: If your GPU is also driving your display, doing video decode, or running a game — subtract 200-500MB from your available VRAM. On a 4GB card, that can be the difference between a model loading or not.

GPU Offloading: The Partial Solution

If you want to try a 7B model, you can split it between GPU and CPU. Put some layers in VRAM (fast) and the rest in system RAM (slow). This is called partial offloading.

In Ollama

```
# Set number of layers on GPU (lower = more on CPU)
# A 7B model has ~32-36 layers
OLLAMA_NUM_GPU=10 ollama run llama3.1:8b
```

Or in a Modelfile:

```
FROM llama3.1:8b
PARAMETER num_gpu 10
PARAMETER num_ctx 2048
```

In llama.cpp

```
./llama-cli -m model-Q4_K_M.gguf -ngl 10 -c 2048 -p "Hello"
```

The `-ngl` flag controls how many layers go to the GPU. Start low and increase until you're near the VRAM limit.

Is Partial Offloading Worth It?

Honest answer: **usually not on 4GB.**

Config	7B Q4 Speed
Full GPU (needs 6-8GB VRAM)	30-40 tok/s
Partial offload (~10 layers on 4GB GPU)	4-10 tok/s
CPU-only (modern CPU, 16GB+ RAM)	7-15 tok/s

Partial offloading on a 4GB card is often slower than just running on a decent CPU. The PCIe bus becomes the bottleneck — data constantly shuttles between VRAM and system RAM at 16 GB/s (PCIe 3.0), while your VRAM bandwidth is 112-256 GB/s.

When partial offload helps: If your CPU is old or slow (pre-Ryzen, low RAM bandwidth), even 10 layers on GPU can give a meaningful speedup. But if you have a modern Ryzen or Intel with DDR5, CPU-only is often faster for 7B models.

Should You Upgrade or Go CPU-Only?

This is the real question. Here are your options:

Option 1: Go CPU-Only (Free)

If you have 16-32GB of RAM, CPU-only inference is a legitimate option:

- 7B models at Q4: 7-15 tok/s on a modern CPU
- 13B models at Q4: 3-7 tok/s with 32GB RAM
- No VRAM limitations — context length only limited by RAM
- DDR5 systems get noticeably better speeds than DDR4

Choose CPU-only if: You want to run 7B+ models and don't want to spend money. Your 4GB GPU is still faster for 1-3B models, so use GPU for small models and CPU for larger ones.

Option 2: Upgrade the GPU ($120-200)

The used GPU market has great options that make 4GB look like a different world:

GPU	Used Price	VRAM	What It Unlocks
RTX 2060	~$120	6GB	7B models at Q4 fully on GPU
RX 6600	~$150	8GB	8B models comfortably, some 13B
RTX 3060 12GB	~$200	12GB	13-14B models, serious local AI

The RTX 3060 12GB at ~$200 is the most recommended budget LLM card in the community. It's $16.67 per GB of VRAM and opens up models that are genuinely competitive with cloud AI for everyday tasks.

Choose upgrading if: You want 7B+ models at full speed, plan to do image generation, or want headroom to grow.

The Honest Take

4GB VRAM is the floor, not a foundation. It's enough to experiment with small models, learn the tools, and figure out if local AI is for you. But if you get hooked — and you probably will — you'll want more VRAM within a week. Budget $150-200 for a used GPU upgrade and consider it an investment. The jump from 4GB to 12GB is the biggest quality-of-life improvement in local AI.

Getting the Most Out of 4GB

If you're sticking with 4GB for now, these tips help:

Set context length explicitly. Ollama and llama.cpp default to large context windows that will OOM on 4GB. Always set `num_ctx` to 2048 or 4096 max.

```
# Ollama
OLLAMA_NUM_CTX=2048 ollama run qwen2.5:3b

# llama.cpp
./llama-cli -m model.gguf -ngl 99 -c 2048
```

Enable KV cache quantization. This halves the memory used by context:

```
# Ollama
export OLLAMA_KV_CACHE_TYPE=q8_0
ollama serve

# llama.cpp
./llama-cli -m model.gguf -ngl 99 -c 4096 -ctk q8_0 -ctv q8_0
```

With Q8 KV cache, a 3B model at Q4 can handle 4-6K tokens of context instead of 2-3K. No meaningful quality loss.

Try Flash Attention. On the GTX 1650 (Turing), Flash Attention gives a modest ~10% speedup on prompt processing and reduces VRAM usage. On the GTX 1050 Ti (Pascal), it saves VRAM but may not improve speed. Worth enabling either way:

```
# llama.cpp
./llama-cli -m model.gguf -ngl 99 --flash-attn -c 2048
```

Pick models with small KV caches. Qwen 2.5 uses only 2 KV heads, giving it dramatically smaller KV cache than Llama 3.2 or Phi-3.5 at the same parameter count. At 4K context:

Model	KV Cache (FP16)	KV Cache (Q8)
Qwen 2.5 3B	144 MB	~72 MB
Llama 3.2 3B	448 MB	~224 MB
Phi-3.5 Mini 3.8B	384 MB	~192 MB

Qwen 2.5 3B uses 3x less KV cache than Llama 3.2 3B. On a 4GB card, that's the difference between 4K and 2K context. **Qwen 2.5 is the best model family for 4GB VRAM.**

Close everything else. Your browser, Discord, even your desktop compositor uses VRAM. On a 4GB card, 200MB matters. Close what you can before loading a model.

The Bottom Line

4GB VRAM is enough to get started with local AI. Run Qwen 2.5 3B at Q4, learn Ollama, try RAG with small models, experiment with what's possible. You'll get responsive chat, decent summarization, and basic coding help.

But don't fight the hardware. If you need 7B+ models, go CPU-only or buy a used GPU. The RTX 3060 12GB at $200 transforms the experience from "making it work" to "this is actually good." The used RTX 3090 at ~$750 opens up 24GB of possibilities that make 4GB feel like a different era.

Start small, learn the tools, upgrade when you're ready.
