Quantization Explained: What It Means for Local AI

January 27, 2025 · by Mark Bartlett

Download this guide as PDF

> **Quick Answer:** Quantization compresses AI models by reducing the precision of their numbers—like going from RAW photos to JPEGs. Q4_K_M is the sweet spot for most users: it shrinks a 7B model from 14GB to ~4GB while keeping ~90-95% of the quality. Start there. Only go higher (Q5, Q6, Q8) if you have VRAM to spare, or lower (Q3, Q2) if you absolutely must.

📚 **More on this topic:** VRAM Requirements · GPU Buying Guide · Run Your First Local LLM

You download a model. You see this:

```
llama-3.1-8b-instruct-Q4_K_M.gguf
llama-3.1-8b-instruct-Q5_K_M.gguf
llama-3.1-8b-instruct-Q6_K.gguf
llama-3.1-8b-instruct-Q8_0.gguf
llama-3.1-8b-instruct-F16.gguf
```

And you think: What the hell do these mean? Which one do I pick?

You're not alone. Quantization is one of those topics where everyone assumes you already know what they're talking about. Nobody stops to explain it clearly.

This guide fixes that. By the end, you'll understand what quantization is, why it matters, and exactly which format to choose for your hardware.

The Problem Nobody Explains

Every AI model is, at its core, billions of numbers. A 7B parameter model has 7 billion numerical values that define how it thinks. At full precision (16 bits per number), storing those values takes about 14GB of space.

That's a problem if your GPU only has 8GB or 12GB of VRAM.

Quantization is the solution. It's a way to compress those numbers so the model takes less space—and therefore less VRAM to run. The tradeoff is some loss in precision, which might affect quality. The art is finding the compression level where you save the most space with the least quality loss.

That's what all those letters and numbers (Q4_K_M, Q8_0, etc.) represent: different compression levels with different tradeoffs.

What Quantization Actually Is

The Plain English Version

Imagine you're storing the number 3.14159265359.

At full precision, you keep all those decimal places. Accurate, but takes space.

With quantization, you might round it to 3.14 or even just 3. Less accurate, but way smaller.

Now multiply that by 7 billion numbers, and you see why this matters. Rounding each number a little bit adds up to massive space savings.

The analogy: Quantization is to AI models what JPEG compression is to photos. A high-quality JPEG looks nearly identical to the RAW file but takes a fraction of the space. Push compression too far, and you see artifacts. Quantization works the same way.

Why Models Need So Much Memory

A 7B parameter model at full precision (FP16) needs:

```
7 billion parameters × 2 bytes per parameter = 14 GB
```

That's just to load the model weights. Running inference adds more overhead for the "working memory" (called KV cache). A 7B model at FP16 realistically needs 16-18GB of VRAM to run comfortably.

Most consumer GPUs have 8-12GB. Something has to give.

Quantization is what gives. By reducing precision from 16 bits to 8, 6, 5, or 4 bits, you can shrink that memory requirement dramatically:

Precision	Bits per Weight	7B Model Size	Approximate VRAM
FP16	16 bits	~14 GB	16-18 GB

Precision	Bits per Weight	7B Model Size	Approximate VRAM
Q8_0	8 bits	~8.5 GB	10-12 GB
Q6_K	6.5 bits	~6.6 GB	8-10 GB
Q5_K_M	5.5 bits	~5.7 GB	7-9 GB
Q4_K_M	4.5 bits	~4.9 GB	6-8 GB
Q4_K_S	4 bits	~4.7 GB	6-7 GB

That's why quantization matters: it's the difference between "runs on my hardware" and "doesn't."

The Tradeoff You're Making

What You Gain

Smaller files. A Llama 3.1 8B model at FP16 is around 16GB. At Q4_K_M, it's under 5GB. That's a 70% reduction.

Lower VRAM requirements. The same model that needed 18GB of VRAM now runs on 8GB. Suddenly your RTX 3060 can run models that previously required a 3090.

Faster loading times. Smaller files load faster. A Q4_K_M model loads in seconds versus a minute or more for FP16.

More headroom for context. VRAM not used by model weights can be used for longer conversations (larger KV cache). Quantization indirectly gives you longer context windows.

What You Lose

Some precision. Every quantization level introduces small errors. Those errors compound through the model's calculations.

Potentially worse output. On complex tasks—multi-step reasoning, precise math, nuanced creative writing—highly quantized models may produce slightly worse results.

Diminishing returns at extremes. Q4 to Q3 saves less space than Q8 to Q4, but the quality drop is more noticeable. Below Q3, quality degrades rapidly.

Here's the good news: for most tasks, the quality loss is barely perceptible. You'd need to run careful benchmarks to notice the difference between Q4_K_M and Q8_0 in casual conversation.

Common Quantization Formats Explained

The GGUF Naming System

When you see a filename like `Q4_K_M`, here's what each part means:

- **Q** = Quantized
- **Number (4, 5, 6, 8)** = Bits per weight (lower = smaller file, more compression)
- **K** = K-quant method (newer, better than legacy methods)
- **S/M/L** = Size variant (Small, Medium, Large—refers to how different layers are quantized)

So **Q4_K_M** means: 4-bit quantization, using the K-quant method, medium variant.

Why K-Quants Are Better

Older quantization methods (Q4_0, Q4_1, Q5_0, etc.) used simple uniform rounding. K-quants are smarter: they use a two-level scheme that preserves more important weights at higher precision while compressing less critical ones more aggressively.

The result: K-quants achieve better quality at the same file size. **Always prefer K-quants over legacy formats.** If you see Q4_0 and Q4_K_M available, pick Q4_K_M.

Format Breakdown

Format	Bits	Relative Size	Quality	Use Case
FP16/BF16	16	Largest (100%)	Perfect baseline	Benchmarking, max quality
Q8_0	8	~50%	Near-lossless	When VRAM isn't tight
Q6_K	6.5	~40%	Excellent	Quality-sensitive tasks
Q5_K_M	5.5	~35%	Very good	Coding, reasoning, writing
Q5_K_S	5.25	~33%	Good	Slight quality trade for size
Q4_K_M	4.5	~30%	Good (sweet spot)	General use, **recommended**
Q4_K_S	4	~28%	Acceptable	Memory-constrained

Format	Bits	Relative Size	Quality	Use Case
Q3_K_M	3.5	~22%	Noticeable loss	Very tight VRAM only
Q2_K	2.5	~18%	Significant loss	Extreme cases only

The Winner: Q4_K_M

For most people, **Q4_K_M is the right choice**. Here's why:

- 70% smaller than FP16
- Runs on 8GB GPUs (for 7B models)
- Retains ~90-95% of original quality
- Fast inference
- Widely available for most models

It's marked as "recommended" by llama.cpp for good reason. Start here unless you have a specific reason not to.

How Much VRAM You Actually Save

Let's look at real file sizes for popular models:

Llama 3.1 8B Instruct

Format	File Size	VRAM Needed (approx)
F16	16.1 GB	18-20 GB
Q8_0	8.5 GB	10-12 GB
Q6_K	6.6 GB	8-10 GB
Q5_K_M	5.7 GB	7-9 GB
Q4_K_M	4.9 GB	6-8 GB
Q4_K_S	4.7 GB	6-7 GB

Quick VRAM Estimation Formula

For a rough estimate of VRAM needed:

```
VRAM ≈ (Parameters × Bits per Weight ÷ 8) + 1-2 GB overhead
```

For a 7B model at Q4 (4 bits):

```
(7B × 4 ÷ 8) + 1.5GB = 3.5GB + 1.5GB = ~5GB VRAM
```

Real-world usage is slightly higher due to KV cache and runtime overhead, but this gets you in the ballpark.

What Different VRAM Amounts Get You

Your VRAM	What You Can Run
6 GB	7B at Q4_K_S, smaller models at higher quants
8 GB	7B at Q4_K_M comfortably, 13B at Q3 (slow)
12 GB	7B at Q6_K, 13B at Q4_K_M, small 30B at Q3
16 GB	7B at Q8, 13B at Q5_K_M, 30B at Q4
24 GB	Almost anything at Q4_K_M or higher

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

Quality Impact: When You Notice, When You Don’t

Tasks Where Quantization Barely Matters

For these use cases, Q4_K_M performs nearly identically to Q8 or FP16:

- **Casual conversation** — Chatting, Q&A, brainstorming
- **Simple coding tasks** — Boilerplate, syntax help, basic debugging
- **Summarization** — Condensing text
- **Translation** — Common language pairs
- **Creative writing** — First drafts, idea generation

If you’re using a local LLM as a general assistant, Q4_K_M is plenty.

Tasks Where Quality Matters More

For these, consider Q5_K_M or Q6_K:

- **Complex reasoning** — Multi-step logic, math problems
- **Precise coding** — Subtle bugs, complex algorithms
- **Instruction following** — Very specific formatting requirements
- **Long-context tasks** — Maintaining coherence over many pages
- **Factual retrieval** — When accuracy of specific details matters

The difference isn't dramatic—maybe 5-10% worse on benchmarks—but if you're doing serious work and have the VRAM, it's worth going higher.

The Perplexity Numbers

Perplexity measures how "surprised" a model is by text—lower is better. Here's how quantization affects it (Llama 2 7B):

Format	Perplexity	Change from FP16
FP16	7.49	baseline
Q8_0	7.49	+0.00 (negligible)
Q6_K	7.53	+0.04
Q5_K_M	7.54	+0.05
Q4_K_M	7.57	+0.08
Q4_K_S	7.61	+0.12
Q3_K_M	7.76	+0.27
Q2_K	8.65	+1.16

Notice how small the differences are until you hit Q3 and below. The jump from Q4_K_M to Q2_K is larger than FP16 to Q4_K_M.

Important caveat: Perplexity doesn't tell the whole story. Some quantized models score worse on perplexity but perform similarly (or even better) on specific benchmarks. Always test on your actual use case.

How to Choose the Right Quant for Your Hardware

Match Your VRAM

Your VRAM	Recommended Quant for 7-8B	Recommended Quant for 13B
6 GB	Q4_K_S (tight fit)	Too big
8 GB	Q4_K_M	Q3_K_M (slow)
12 GB	Q6_K or Q5_K_M	Q4_K_M
16 GB	Q8_0	Q5_K_M or Q6_K
24 GB	FP16 (why not?)	Q8_0

The Decision Flowchart

1. **Does the model fit at Q4_K_M?** Start there. It's the sweet spot.
2. **Want better quality?** Try Q5_K_M or Q6_K. Worth it for coding and reasoning.
3. **Still too big?** Drop to Q4_K_S or Q3_K_M. Expect some quality loss.
4. **Have VRAM to spare?** Go Q8_0 or higher. Diminishing returns, but why not.
5. **Q3 still too big?** You need a smaller model, not more compression.

The Bigger Model Rule

Here's a key insight: **a larger model at lower quantization often beats a smaller model at higher quantization**.

Example: A 13B model at Q4_K_M typically outperforms a 7B model at Q8_0—even though the 7B has higher precision. Model capability matters more than quantization level.

If you're choosing between:

- Llama 3.1 8B at Q8_0 (~8.5 GB)
- Llama 3.1 70B at Q4_K_M (~40 GB)

And both fit in your VRAM? Take the 70B. It's not even close.

Where to Find Quantized Models

Hugging Face

The main source. Look for uploaders like:

- **bartowski** — Reliable, consistent, well-documented
- **TheBloke** — Huge library (mostly older models now)
- **QuantFactory** — Good selection of newer models

Search for your model name + “GGUF” and you’ll find options.

Ollama

Pre-quantized and ready to run. When you do `ollama pull llama3.1:8b`, you’re getting a quantized version (typically Q4_K_M equivalent). No decisions needed. If you’re new to Ollama, our beginner’s guide walks through the full setup.

```
ollama pull llama3.1:8b      # Default quantization
ollama pull llama3.1:8b-q8   # Higher quality, more VRAM
```

LM Studio

Built-in model browser with Hugging Face integration. Filter by quantization level, see file sizes, one-click download. Good for exploring options visually.

Quick Reference Table

Format	File Size (8B)	VRAM (8B)	Quality	Speed	Best For
FP16	16 GB	18-20 GB	100%	Baseline	Benchmarking
Q8_0	8.5 GB	10-12 GB	~99%	Fast	When VRAM allows
Q6_K	6.6 GB	8-10 GB	~97%	Fast	Quality-sensitive work
Q5_K_M	5.7 GB	7-9 GB	~95%	Fast	Coding, reasoning
Q4_K_M	**4.9 GB**	**6-8 GB**	**~92%**	**Fast**	**General use (recommended)**

Format	File Size (8B)	VRAM (8B)	Quality	Speed	Best For
Q4_K_S	4.7 GB	6-7 GB	~90%	Fastest	Memory-constrained
Q3_K_M	3.8 GB	5-6 GB	~85%	Faster	Very tight VRAM
Q2_K	3.0 GB	4-5 GB	~70%	Fastest	Last resort

The Bottom Line

Quantization lets you run AI models that wouldn't otherwise fit on your hardware. It's not magic—you're trading some precision for smaller size—but the tradeoff is usually worth it.

The practical advice:

1. **Start with Q4_K_M.** It's the default for a reason. Good quality, runs on most hardware, widely available.
2. **Go higher if you can.** Have 12GB+ VRAM? Try Q5_K_M or Q6_K. The quality bump is noticeable for coding and reasoning tasks.
3. **Go lower only if you must.** Q3 and Q2 exist for extreme cases. Expect quality loss.
4. **Model size > quantization level.** A bigger model at Q4 beats a smaller model at Q8. Always.
5. **Test on your actual tasks.** Benchmarks are useful, but your experience is what matters. If Q4_K_M works for what you do, that's your answer.

Stop overthinking it. Download Q4_K_M, start using the model, and only revisit the decision if you hit actual limitations.

Related Guides

- How Much VRAM Do You Need for Local LLMs?
- GPU Buying Guide for Local AI
- Run Your First Local LLM in 15 Minutes

Source: https://insiderllm.com/guides/llm-quantization-explained/

Free guides for running AI locally