Model Formats Explained: GGUF vs GPTQ vs AWQ vs EXL2

January 30, 2026 · by Mark Bartlett

Download this guide as PDF

Quick Answer: If you use Ollama or LM Studio, or run on CPU or Apple Silicon: download GGUF (Q4_K_M is the default pick). If you want the fastest NVIDIA GPU inference for personal use: EXL2 via ExLlamaV2. If you're serving models to multiple users with vLLM: GPTQ with Marlin kernels. AWQ offers the best quality preservation at 4-bit but its speed advantage depends on the engine. When in doubt, GGUF works everywhere and is the safest choice.

📚 **More on this topic:** Quantization Explained · llama.cpp vs Ollama vs vLLM · Text Generation WebUI Guide · VRAM Requirements

You’re on HuggingFace looking for a model. There are six uploads of the same thing: GGUF, GPTQ, AWQ, EXL2, SafeTensors, and some older format you’ve never heard of. They’re all the same model at roughly the same size. Which one do you download?

The answer depends on your hardware and which inference tool you use. Each format is optimized for a different setup, and picking the wrong one means slower speeds or outright incompatibility. This guide breaks down what each format is, what runs it, and when to use it.

The Four Formats at a Glance

Format	Runs On	Best Tool	Speed (RTX 3090, 13B 4-bit)	Quality (Perplexity)	Best For
GGUF	CPU + GPU	Ollama, LM Studio, llama.cpp	~31-35 tok/s	4.33	Most people. Works everywhere.
GPTQ	GPU only	vLLM (Marlin), text-gen-webui	~42-64 tok/s	4.34	Production serving, multi-user
AWQ	GPU only	vLLM, TensorRT-LLM	~39 tok/s	4.33	Best quality at 4-bit
EXL2	GPU only (NVIDIA)	ExLlamaV2, TabbyAPI	~52-56 tok/s	4.31	Fastest personal inference

All four formats deliver nearly identical quality at 4-bit. The differences are in speed, hardware compatibility, and tool support.

GGUF: The Universal Format

GGUF (GPT-Generated Unified Format) is the format used by llama.cpp. It's the only format that runs on CPU, and it supports every GPU backend — CUDA, Metal, Vulkan, ROCm. If you use Ollama or LM Studio, you're already using GGUF.

Why GGUF wins for most people:

- **Runs everywhere:** CPU, NVIDIA GPU, Apple Silicon, AMD GPU, Intel Arc, even Raspberry Pi
- **Partial offloading:** Split layers between GPU and CPU with `-ngl`. No other format does this.
- **Single file:** Everything the model needs — weights, tokenizer, architecture — in one portable file
- **Widest quant range:** From IQ1_S (1.69 bits/weight) to Q8_0 (8.5 bits/weight), plus the K-quant series that allocates more bits to important layers

Common GGUF quantization levels:

Quant	Bits/Weight	Size (8B model)	Quality	Use When
Q2_K	3.16	~3.0 GB	Poor	Extreme VRAM limits only
Q3_K_M	4.00	~3.7 GB	Acceptable	4GB VRAM cards
Q4_K_M	4.89	~4.6 GB	Good	**Default choice for most setups**
Q5_K_M	5.70	~5.3 GB	Very good	When you have VRAM headroom
Q6_K	6.56	~6.1 GB	Excellent	Near-lossless
Q8_0	8.50	~8.0 GB	Near-original	When VRAM isn't a concern

Start with Q4_K_M. It's the community standard and the best balance of size, speed, and quality.

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

Tool support: Ollama, LM Studio, llama.cpp, KoboldCpp, text-generation-webui, GPT4All, Open WebUI

GPTQ: The Production GPU Format

GPTQ uses calibration data and second-order optimization to find the best quantized weights layer by layer. It's GPU-only — the dequantization happens in CUDA kernels during inference.

Why use GPTQ:

- **Fastest at scale with Marlin kernels:** vLLM + GPTQ + Marlin hit 712 tok/s on Qwen 2.5 32B — 1.5x faster than FP16. This matters for serving multiple users.
- **Widest GPU tool support:** HuggingFace Transformers, vLLM, text-generation-webui, TensorRT-LLM, ExLlamaV2
- **Mature ecosystem:** Most models on HuggingFace have a GPTQ variant available

Why skip GPTQ:

- GPU-only. No CPU fallback, no partial offloading
- Without Marlin kernels, it's actually slower than FP16 in vLLM (276 vs 461 tok/s in one benchmark)
- Slightly lower quality than AWQ and EXL2 at the same bitrate
- Requires CUDA 12.1+

Best for: Production API servers using vLLM where throughput matters more than anything else. If you're serving a model to multiple users, GPTQ with Marlin is the standard.

Tool support: vLLM, HuggingFace Transformers, AutoGPTQ/GPTQModel, text-generation-webui, ExLlamaV2, TensorRT-LLM

AWQ: Best Quality at 4-Bit

AWQ (Activation-Aware Weight Quantization) takes a different approach. Instead of optimizing weights directly, it identifies the ~1% of weight channels that matter most based on activation patterns, then scales them up before quantizing. The result: better quality retention than GPTQ with no runtime overhead from the scaling.

Why use AWQ:

- **Best quality at 4-bit:** ~95% quality retention vs ~90% for GPTQ. Perplexity differences are small (4.33 vs 4.34 on Llama 2 13B) but AWQ consistently edges out GPTQ and holds up better on downstream tasks

- **Better generalization:** AWQ doesn't overfit its calibration data, so it works well across different domains
- **TensorRT-LLM native:** If you're using NVIDIA's production stack, AWQ is a first-class citizen

Why skip AWQ:

- GPU-only, same as GPTQ
- Speed depends heavily on the inference engine — not always faster than GPTQ
- Fewer quantization options than GGUF (typically just 4-bit)

Best for: When quality preservation is the priority. Creative writing, coding, and tasks where subtle accuracy differences matter. Also a strong choice for NVIDIA TensorRT-LLM deployments.

Tool support: AutoAWQ, vLLM, HuggingFace Transformers, text-generation-webui, TensorRT-LLM

EXL2: Fastest Personal Inference

EXL2 is ExLlamaV2's format, and its key feature is **variable bits per weight**. Instead of quantizing every layer to the same bitrate, EXL2 measures which layers are sensitive and allocates more bits to them. You can target any average bitrate — 3.5, 4.65, 5.0, whatever fits your VRAM.

Why use EXL2:

- **Fastest single-user inference on NVIDIA:** 52-56 tok/s on RTX 3090 for 13B at ~4.65 bpw — roughly 1.5x faster than GGUF on the same hardware
- **Best quality at any bitrate:** Variable precision means EXL2 at 4.65 bpw beats uniform 4-bit methods in perplexity (4.32 vs 4.33-4.34)
- **Arbitrary bitrate targets:** Squeeze a model into exactly the VRAM you have, not just the nearest round number

Why skip EXL2:

- **NVIDIA only.** No CPU, no Apple Silicon, no AMD
- **Smaller ecosystem.** ExLlamaV2, TabbyAPI, and text-generation-webui — that's basically it
- **No Ollama/LM Studio support.** If that's your workflow, EXL2 isn't an option
- **Slightly higher VRAM during inference** than GPTQ at comparable bitrate

Best for: Power users on NVIDIA GPUs who want the fastest token generation and are comfortable with text-generation-webui or TabbyAPI. Especially good when you need to fit a model into a specific VRAM budget.

Tool support: ExLlamaV2, TabbyAPI, text-generation-webui

Note: EXL3 (ExLlamaV3) is emerging with trellis-based quantization that's coherent down to 1.6 bpw. Worth watching but still early.

Which Format Should You Use?

Follow this decision tree:

Do you use Ollama or LM Studio? → GGUF. No other format is supported.

Are you on CPU or Apple Silicon? → GGUF. It's the only option.

Are you on AMD (ROCm)? → GGUF. The others have limited or no AMD support.

Are you on NVIDIA and want max personal speed? → EXL2 via ExLlamaV2 or TabbyAPI.

Are you serving to multiple users? → GPTQ with Marlin kernels via vLLM.

Do you need the best quality at 4-bit? → AWQ, or EXL2 at a higher bpw target.

Not sure? → GGUF Q4_K_M. It works everywhere, the quality is competitive, and you can always switch later.

What About Other Formats?

GGML: Deprecated in August 2023, replaced by GGUF. No modern tool supports it. If you find a GGML model, look for a GGUF version instead.

SafeTensors: Not a quantization format — it's HuggingFace's secure storage format for unquantized (FP16/BF16) model weights. SafeTensors files are the starting point from which all quantized versions are created. GPTQ, AWQ, and EXL2 models actually store their quantized weights inside SafeTensors files — only GGUF uses its own container.

BitsAndBytes (load_in_4bit/8bit): On-the-fly quantization in HuggingFace Transformers. Convenient but slow — 23 tok/s vs 31-64 tok/s for pre-quantized formats on the same hardware. Use it for quick testing, not production.

.bin (PyTorch pickle): Legacy model format. Being replaced by SafeTensors for security reasons (pickle files can execute arbitrary code when loaded).

Converting Between Formats

You generally don't need to convert — HuggingFace has pre-quantized versions of popular models in every format. But if you need to:

FP16 SafeTensors → GGUF:

```
# Clone llama.cpp and use convert script
python convert_hf_to_gguf.py /path/to/model --outtype f16
./llama-quantize model-f16.gguf model-Q4_K_M.gguf Q4_K_M
```

FP16 SafeTensors → GPTQ:

```
pip install auto-gptq
# Use AutoGPTQ Python API with calibration dataset
```

FP16 SafeTensors → AWQ:

```
pip install autoawq
# Use AutoAWQ Python API
```

FP16 SafeTensors → EXL2:

```
# Use ExLlamaV2's convert.py
python convert.py -i /path/to/model -o /output -cf /output/4.65bpw -b 4.65
```

Most users never need to do this. The community (particularly TheBloke and bartowski on HuggingFace) provides pre-quantized versions in all major formats.

Get notified when we publish new guides.

Subscribe — free, no spam

Source: https://insiderllm.com/guides/model-formats-explained-gguf-gptq-awq-exl2/