Local AI Troubleshooting Guide: Every Common Problem and Fix

February 3, 2026 · by Mark Bartlett

Download this guide as PDF

Quick Answer: Most local AI problems fall into five categories: (1) Model won't load — usually VRAM too small, reduce context or use a smaller model; (2) Slow generation — GPU not being used, check nvidia-smi and your tool's GPU settings; (3) Garbled output — corrupted download or incompatible quant, re-download the model; (4) Out of memory — reduce num_ctx, use lower quantization, or unload other models; (5) Quality disappointing — you're using the wrong model or quantization for your task. This guide covers diagnosis and fixes for each.

📚 **More on this topic:** Ollama Troubleshooting Guide · Run Your First Local LLM · VRAM Requirements · Quantization Explained · Planning Tool

Running AI locally means you’re your own IT department. When something breaks, there’s no support ticket to file. The good news: most problems have the same handful of causes, and they’re all fixable.

This guide covers the most common issues across all local AI tools — Ollama, LM Studio, llama.cpp, text-generation-webui, ComfyUI, and others. Find your symptom, follow the diagnosis, apply the fix.

Quick Diagnosis: What’s Your Problem?

Symptom	Most Likely Cause	Jump To
Model won’t load at all	Not enough VRAM	Model Won’t Load
Model loads but is painfully slow	Running on CPU instead of GPU	Slow Generation
Output is garbled or nonsense	Corrupted download or bad quant	Garbled Output
“Out of memory” errors mid-generation	Context length too high	Out of Memory
Output quality is disappointing	Wrong model or quantization	Quality Issues
CUDA/GPU errors	Driver or library issues	CUDA and GPU Errors

Symptom	Most Likely Cause	Jump To
Tool won't start	Installation or port conflict	Installation Issues

---

Model Won't Load

The symptom: You try to run a model and get an error about memory, size, or the model simply fails to initialize.

Why It Happens

Every model needs a certain amount of VRAM to load. This includes:

- **Model weights** — the actual neural network parameters
- **KV cache** — memory for the conversation context
- **Overhead** — CUDA/Metal runtime, tool overhead (~500MB-1GB)

If the total exceeds your available VRAM, the model won't load — or it will, but spill to system RAM and run at CPU speeds.

Quick VRAM Reference

Model Size	Q4 Weights	+4K Context	+8K Context	Minimum VRAM
3B	~2 GB	+0.2 GB	+0.4 GB	4 GB
7-8B	~4.5 GB	+0.5 GB	+1 GB	6 GB
13-14B	~8 GB	+0.8 GB	+1.6 GB	10 GB
32-34B	~20 GB	+1.5 GB	+3 GB	24 GB
70B	~40 GB	+3 GB	+6 GB	48 GB

For detailed requirements: VRAM Requirements Guide

Fixes

1. Use a smaller model

The most reliable fix. If you have 8GB VRAM, don't try to run a 14B model. Stick to 7-8B.

```
# Instead of:
ollama run llama3.1:14b

# Try:
ollama run llama3.1:8b
```

2. Use a lower quantization

Q4_K_M uses less VRAM than Q5_K_M or Q6_K. If a Q5 model won't load, try Q4.

In LM Studio: Download a different quantization from the model's HuggingFace page.

In Ollama: Check what quants are available:

```
ollama show llama3.1 --modelfile
```

3. Reduce context length

Context length is the hidden VRAM killer. A 14B model might fit at 4K context but not 16K.

In Ollama:

```
ollama run llama3.1 /set parameter num_ctx 2048
```

In LM Studio: Settings → Context Length → lower to 2048 or 4096

In llama.cpp:

```
./llama-cli -m model.gguf -c 2048
```

4. Enable partial GPU offloading (slower but works)

If your model almost fits, you can run some layers on GPU and the rest on CPU.

In Ollama: This happens automatically when VRAM is tight.

In LM Studio: Set GPU layers to a number less than 100% (try 30-40 for tight fits).

In llama.cpp:

```
./llama-cli -m model.gguf -ngl 30   # 30 layers on GPU, rest on CPU
```

Slow Generation

The symptom: The model loads and runs, but generates only 2-10 tokens per second instead of the expected 30-80+.

Why It Happens

Almost always because the model is running on CPU instead of GPU. A 7B model on a modern CPU runs at 5-15 tok/s. The same model on an RTX 3060 runs at 35-50 tok/s.

How to Diagnose

Check if your GPU is being used:

NVIDIA:

```
nvidia-smi
# Look at "GPU Memory Usage" — if it's near 0 while generating, GPU isn't being used
```

AMD:

```
rocm-smi
```

Apple Silicon:

```
# In Activity Monitor, check GPU History — it should spike during generation
```

Check your tool's status:

Ollama:

```
ollama ps
# Look at the "Processor" column — should say "GPU" not "CPU"
```

LM Studio: Look at the bottom status bar — it shows GPU vs CPU and current memory usage.

Fixes

1. Install/update GPU drivers

This is the #1 cause of GPU not being detected.

NVIDIA (Linux):

```
# Check current driver
nvidia-smi

# If no output, install drivers:
sudo apt update
sudo apt install nvidia-driver-535   # or latest version
sudo reboot
```

NVIDIA (Windows): Download the latest driver from nvidia.com.

AMD: Install ROCm (Linux) or AMD Adrenalin (Windows).

2. Configure your tool to use GPU

In LM Studio: Settings → GPU Offloading → Enable + set layers to maximum.

In Ollama: GPU is automatic, but check with `ollama ps`. If it shows CPU, see our Ollama Troubleshooting Guide.

In llama.cpp: Use `-ngl 999` to offload all layers to GPU:

```
./llama-cli -m model.gguf -ngl 999
```

In text-generation-webui: Use `--loader exllamav2` for NVIDIA GPUs or check the GPU checkbox.

3. Reduce context length

If your model partially fits in VRAM, it might be running with a CPU/GPU split. This is faster than pure CPU but much slower than pure GPU.

Lower context until the model fits entirely in VRAM:

```
ollama run llama3.1 /set parameter num_ctx 2048
```

4. Unload other models

Some tools keep multiple models loaded. Each consumes VRAM.

Ollama:

```
ollama ps          # See what's loaded
ollama stop llama3.1   # Unload a model
```

LM Studio: Only one model loads at a time by default.

Expected speeds (full GPU):

Model Size	RTX 3060 12GB	RTX 3090 24GB	M2 Pro 16GB
7-8B Q4	35-45 tok/s	80-110 tok/s	25-35 tok/s
13-14B Q4	20-28 tok/s	45-60 tok/s	15-22 tok/s
32B Q4	—	25-35 tok/s	8-12 tok/s

If you're seeing numbers much lower than these on GPU, something's wrong.

Garbled Output

The symptom: The model generates text, but it's nonsense — random characters, repeated tokens, gibberish, or text that makes no grammatical sense.

Common Causes

1. Corrupted model download

The most common cause. Large model files can get corrupted during download, especially over spotty connections.

Fix: Delete and re-download.

Ollama:

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

LM Studio: Delete the model file and re-download from the Discover tab.

Manual: Check the file size against what's listed on HuggingFace. If it's smaller, the download was incomplete.

2. Incompatible or mismatched model files

This happens when you download a model that requires files your tool doesn't have, or when model files get mixed up.

Signs:

- Error messages about missing tokens or vocabulary
- Model loads but immediately produces garbage
- Tool crashes after loading

Fix: Make sure you have the complete model. GGUF models should be a single file. SafeTensors models may need multiple files plus a tokenizer.

3. Wrong model format for your tool

Tool	Supports
Ollama	GGUF only
LM Studio	GGUF
llama.cpp	GGUF
text-generation-webui	GGUF, GPTQ, AWQ, EXL2

Tool	Supports
vLLM	SafeTensors, GPTQ, AWQ

If you downloaded a GPTQ model and tried to load it in Ollama, it won't work.

For format details: Model Formats Explained

4. Extremely low quantization

Q2 and Q3 quantizations sacrifice significant quality. Some models become nearly unusable at these levels.

Fix: Use Q4_K_M or higher. Q2/Q3 should only be used when you literally cannot fit Q4.

5. Temperature too high

Temperature above 1.5-2.0 can produce increasingly random output.

Fix: Use temperature 0.7-1.0 for most tasks, 0.3-0.5 for factual/coding work.

Out of Memory Errors

The symptom: The model loads and runs initially, but crashes with "out of memory" or "CUDA OOM" errors during longer conversations.

Why It Happens

The KV cache grows with conversation length. A model that fits fine with a short prompt might run out of memory after 10+ messages.

KV cache memory by context length (8B model):

Context	KV Cache Size
2K	~0.3 GB
4K	~0.6 GB
8K	~1.2 GB
16K	~2.4 GB
32K	~5 GB

Fixes

1. Reduce context length

The fastest fix. Most conversations don't need more than 4K context.

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

2. Enable KV cache quantization

Cuts KV cache memory in half with minimal quality loss.

Ollama:

```
export OLLAMA_KV_CACHE_TYPE=q8_0
ollama serve
```

llama.cpp:

```
./llama-cli -m model.gguf --cache-type-k q8_0 --cache-type-v q8_0
```

3. Limit parallel requests

If you're running an API server with multiple concurrent requests, each request needs its own KV cache allocation.

Ollama:

```
export OLLAMA_NUM_PARALLEL=1
```

4. Start fresh conversations

Long conversations accumulate context. Starting a new chat clears the KV cache.

Model Quality Disappointing

The symptom: The model runs fine, but the output quality is much worse than expected — wrong answers, poor reasoning, bland writing.

Common Causes

1. Wrong model for the task

Not all models are good at everything. A general chat model might struggle with coding. A coding model might be bad at creative writing.

Task	Best Model Types
Coding	DeepSeek Coder, Qwen Coder, CodeLlama
Math/Reasoning	DeepSeek R1 (distills), Qwen-thinking
Creative Writing	Mistral, Llama 3.1, Qwen
General Chat	Llama 3.1, Qwen 2.5, Mistral

See our use case guides: Coding · Math · Writing

2. Model too small

Smaller models have less knowledge and reasoning capability. If you need complex multi-step reasoning, a 3B model won't cut it.

Model Size	Capability Level
1-3B	Basic tasks, simple Q&A, constrained devices
7-8B	Good general use, coding, most tasks
13-14B	Better reasoning, nuance, longer coherence
32B+	Complex tasks, expert-level output

3. Quantization too aggressive

Going from Q4_K_M to Q2_K saves VRAM but costs quality. The quality loss is more noticeable on reasoning tasks than on simple generation.

Rule of thumb: Use the highest quantization that fits in your VRAM:

- Q6_K or Q8 if VRAM allows

- Q5_K_M for a slight VRAM savings
- Q4_K_M as the go-to default
- Q3 and Q2 only when desperate

For details: Quantization Explained

4. Context length too short

If the model is "forgetting" earlier parts of the conversation, your context window might be too small.

Check your current context:

```
ollama show llama3.1 --modelfile   # Look for num_ctx
```

Increase if needed — but watch VRAM.

5. Temperature and sampling settings

- Temperature too low (0.1-0.3): Repetitive, boring output
- Temperature too high (1.5+): Random, incoherent output
- Top-p too low: Limited vocabulary, same phrases repeated

Good defaults:

- Creative work: temp 0.8-1.0, top_p 0.9
- Coding: temp 0.3-0.6, top_p 0.95
- Factual Q&A: temp 0.3-0.5, top_p 0.9

CUDA and GPU Errors

The symptom: Errors mentioning CUDA, GPU, driver versions, or your tool refuses to detect your graphics card.

Common CUDA Errors and Fixes

"CUDA driver version is insufficient"

Your NVIDIA driver is too old for the CUDA version your tool needs.

Fix: Update to the latest NVIDIA driver.

```
# Check current version
nvidia-smi

# Update (Ubuntu/Debian)
sudo apt update
sudo apt install nvidia-driver-535
sudo reboot
```

“CUDA out of memory”

You’ve run out of VRAM mid-operation.

Fix: Reduce model size, lower context, or close other GPU applications (including browsers with hardware acceleration).

“no CUDA-capable device is detected”

CUDA can’t see your GPU at all.

Fixes:

1. Make sure you have an NVIDIA GPU (AMD uses ROCm, not CUDA)
2. Install NVIDIA drivers
3. Reboot after driver installation
4. On Linux, add your user to the video group: `sudo usermod -aG video $USER`

“CUDA initialization failed”

The driver is installed but CUDA can’t initialize.

Fixes:

1. Try a different driver version (some versions have bugs)
2. Reinstall the driver
3. Check `nvidia-smi` — if it fails, the driver isn’t working

GPU not detected after sleep/resume (Linux)

The GPU can disappear after waking from sleep.

Fix:

```
sudo rmmod nvidia_uvm && sudo modprobe nvidia_uvm
```

Or restart the Ollama service:

```
sudo systemctl restart ollama
```

AMD ROCm Issues

"HSA_STATUS_ERROR_OUT_OF_RESOURCES"

Usually means the GPU architecture isn't recognized.

Fix: Set the GFX version override:

```
# For RX 6600/6700 series:
HSA_OVERRIDE_GFX_VERSION=10.3.0 ollama serve

# For RX 7600/7700 series:
HSA_OVERRIDE_GFX_VERSION=11.0.0 ollama serve
```

For more AMD fixes: AMD vs NVIDIA for Local AI

Installation and Startup Issues

Tool Won't Start

"Address already in use" / port conflict

Another process is using the port (usually 11434 for Ollama, 1234 for LM Studio).

Fix: Find and kill the conflicting process:

```
# Linux/Mac
sudo lsof -i :11434
```

```
kill -9 <PID>

# Windows
netstat -aon | findstr :11434
taskkill /PID <PID> /F
```

“Command not found”

The tool isn’t installed or isn’t in your PATH.

Ollama (Linux):

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

LM Studio: Download from lmstudio.ai and run the installer.

Permission denied (Linux)

Your user doesn’t have access to GPU devices or installation directories.

Fix:

```
sudo usermod -aG video,render $USER
# Log out and back in
```

Model Download Fails

Stuck downloads or network errors

Fixes:

1. Check your internet connection
2. If behind a proxy: `HTTPS_PROXY=https://your-proxy ollama pull model`
3. Clear partial downloads and retry

Ollama:

```
rm -rf ~/.ollama/cache/*
ollama pull llama3.1
```

Not enough disk space

Models are large (2-45GB). Check your free space:

```
df -h
```

Move models to a larger drive if needed:

```
export OLLAMA_MODELS=/mnt/large-drive/ollama
```

When to Restart vs Reinstall

Restart the tool/service first:

- After changing settings
- After system sleep/resume
- When things just stop working for no clear reason

```
# Ollama
sudo systemctl restart ollama

# Kill and restart LM Studio
pkill -f "LM Studio"
```

Reinstall when:

- Restart doesn't fix it
- You get errors about corrupted files
- Major version upgrade isn't working

- You want a clean slate

Clean reinstall (Ollama):

```
sudo systemctl stop ollama
sudo rm /usr/local/bin/ollama
rm -rf ~/.ollama    # This deletes all your models
curl -fsSL https://ollama.com/install.sh | sh
```

The Bottom Line

Most local AI problems come down to:

1. **Not enough VRAM** → Use a smaller model or lower context
2. **GPU not being used** → Check drivers, check tool settings
3. **Corrupted download** → Delete and re-download the model
4. **Wrong settings** → Temperature too high, context too long
5. **Wrong model for the task** → Match model to use case

When in doubt:

- Check `nvidia-smi` to see if your GPU is being used
- Try a known-working model first (Llama 3.1 8B, Mistral 7B)
- Enable debug/verbose logging in your tool
- Search the error message — someone else has probably hit it

For Ollama-specific issues, see our detailed Ollama Troubleshooting Guide.

Get notified when we publish new guides.

Subscribe — free, no spam

Source: https://insiderllm.com/guides/local-ai-troubleshooting-guide/

Free guides for running AI locally