

你的代码副驾驶，100% 本地运行

第 13 章——出问题时：诊断与修复 (节选) · v2026.07 · nagellm.com/copilote-local

第 13 章——出问题时：诊断与修复

本地环境搭建的第 1 周，问题几乎从来不是“模型太差”。通常就是五种故障，而且总是那几种。它们会让你以为本地方案不行。实际上，往往只是少了两行配置。本章按**症状**进行诊断：从你看到的现象（错误消息、吞吐量骤降、自动补全卡顿）出发，追溯到原因，再找到修复方法。这是诊断问题时的参考章节。第 4、6 和 12 章一旦出现故障，就会把你引到这里。

工作基准：你运行在自己的 `monprojet` 上（自第 7 章以来一直作为主线的小型 Flask/Python Web API），后端为 Ollama，硬件基准是 RTX 5070 Ti 16 GB。每次我都会给出“针对你的 VRAM”的版本。在 24 GB 上有效的修复方法，在 8 GB 上可能恰恰是错误的做法。

✅ **重点——本地调试的黄金法则。** 第 1 周 90% 的故障都来自一个数字：`num_ctx`。`num_ctx` 就是模型工作内存的大小。你为了装下整个仓库而把它调大，结果把模型挤出了 VRAM。在怀疑其他问题之前，先看看模型实际在哪里运行（`ollama ps`），以及你把上下文扩大了多少。

📄 在完整指南中，本章以一棵决策树开篇（症状 → 小节 → 修复），涵盖第一周最常见的 5 类故障。本免费节选包含故障 1（显存溢出 OOM）和故障 2（GPU 未被使用）的开头部分。

A 节——GPU OOM：“out of memory”或吞吐量骤降

症状

它有两种表现。明显的一种是：出现 `CUDA error: out of memory` 消息（或 ROCm/Metal 的等效消息），模型拒绝响应。更隐蔽、也更常见、更容易误判的一种是：**能响应，但只有几 token/秒**。模型超出了 VRAM。Ollama 把一部分层切换到了 CPU（这就是 offload），而你的 16 GB GPU 只能看着 CPU 艰难运行。

原因

VRAM 预算很简单，你可以自己列出来：

```
VRAM nécessaire ≈ (poids du modèle quantifié)
                  + (cache KV, qui grandit avec num_ctx)
                  + overhead
```

KV 缓存是模型存储已处理上下文的内存。在你的 16 GB 显存上，默认的 agentic 模型——使用 `q4_K_M` 的 Devstral Small 24B（约 14 GB）——**只要你不把 `num_ctx` 调大**，就能装下。典型陷阱是：为了容纳

更多 `monprojet`，你把 `num_ctx` 从 8192 调到 32768。KV 缓存会翻倍或翻三倍，最终超过 16 GB，却不会显示错误消息——只会静默地发生 offload。

一条命令完成诊断

```
ollama ps
```

读取 `PROCESSOR` 列（详细解释见第 4 章）。这是整章的核心诊断工具。下面是用它诊断 OOM 的方法：

<code>ollama ps</code> 显示内容	解释	操作
100% GPU	全部位于 VRAM 中，状态正常	OOM 方面无需操作
58%/42% CPU/GPU（或任何非零的 %CPU）	部分 offload → 吞吐量骤降	减小 <code>num_ctx</code> 或模型大小
100% CPU	完全没有使用 GPU	前往B 节（这不是 OOM）

表 13.1——读取 `ollama ps` 以诊断 OOM。

修复方法（按顺序，从破坏性最小的开始）

1. 先降低 `num_ctx`。这是第一优先级，也是成本最低的调节手段。调回一个能装下的值（8192 是 16 GB 上聊天/agent 的默认值）。用 `ollama ps` 验证，然后逐级调高，直到刚好达到发生 offload 之前。

```
# 直接测试：强制使用适中的上下文，查看是否恢复为 100% GPU
OLLAMA_CONTEXT_LENGTH=8192 ollama run devstral:24b
```

在 Modelfile 中（完整结构见第 11 章），也可以将同样的设置固定下来：

```
FROM devstral:24b
PARAMETER num_ctx 8192
```

2. 降低一个量化级别或模型尺寸。如果在合理的 `num_ctx` 下仍然超出显存，说明你的模型对于这张显卡来说太大了。使用 `q4_K_M`（默认的质量/占用空间折中方案），不要选择更耗资源的版本。温度/top-p/ `num_ctx` 的微调在第 12 章介绍；针对你的 VRAM 选择合适的模型，则见第 5 章。

3. 明确限制 GPU 层数（极端情况）。你想主动控制分配，而不是被动接受自动 offload？`num_gpu` 用于设置发送到 GPU 的层数：

```
FROM devstral:24b
PARAMETER num_ctx 8192
PARAMETER num_gpu 99 # “尽可能多”；如果想主动将一部分负载转移到 CPU，就降低这个数字
```

⚠ 警告—— `num_gpu` 不是魔法公式。降低它并不会“释放”性能。恰恰相反：它会把更多层发送到 CPU，因此会变慢。它适用于你明知模型过大，却仍想运行，并且愿意接受速度变慢的情况。这不是日常调节项。正常使用时，让 Ollama 负责管理。优先调整 `num_ctx` 和模型大小。

针对你的 VRAM：

VRAM	OOM 的主要应对方式
8 GB	密集型 24B agentic 层级超出能力范围；选择轻量模型（Qwen2.5-Coder 7B），并将 <code>num_ctx</code> ≤ 8192 。任何上下文扩展都会立刻付出代价。
12 GB (RTX 5070、RTX 4070)	Devstral Small 24B 装不下：选择 Qwen2.5-Coder 14B（约 9 GB），或在 <code>q4_K_M</code> 、 <code>num_ctx</code> 4096 下使用 DeepSeek-Coder-V2-Lite 16B（约 10 GB）。超过这个范围后，要留意 <code>ollama ps</code> 。
16 GB (基准——RTX 5070 Ti / 4080 / 5080 / RX 9070 XT, Mac M4 24 GB)	使用 <code>q4_K_M</code> 的 Devstral Small 24B（约 14 GB）+ <code>num_ctx</code> 8192 = agentic 甜蜜点。使用 Qwen2.5-Coder 14B 的 Q8 也有余量。
24 GB (RTX 3090/4090、RX 7900 XTX、Mac M4 Pro 48 GB)	你可以使用 Qwen2.5-Coder 32B <code>q4_K_M</code> （约 19 GB）、Q5/Q6 量化的 Devstral，或在不发生 offload 的情况下使用长上下文；OOM 会变得少见。
Apple Silicon (Metal)	VRAM 与系统 RAM 共享（统一内存）。不会出现突然的 OOM，但会出现导致交换的内存压力：监控系统内存压力，在大型会话前关闭占用较大的进程。

表 13.2——各 VRAM 档位的 OOM 应对方式。

B 节——完全没有使用 GPU

症状

`ollama ps` 显示 100% CPU，GPU 风扇毫无动静，而且生成速度处处都很慢——不像 offload 那样速度骤降，而是从第一个 token 开始就一直慢。这是最让人恼火的情况：硬件明明在，却没有被利用。

原因

Ollama 没有识别到你的加速器。常见原因有三个：- GPU runtime 没有被检测到（CUDA / ROCm 驱动缺失或版本过旧）。- 在 Linux 下，Ollama 运行在一个无法访问 GPU 设备的环境中（容器、systemd 服务）。- 在 Windows/WSL 下，GPU passthrough 尚未配置。

诊断

```
# NVIDIA (你的 5070 Ti) —生成期间，GPU 应该出现，并列出一个 ollama 进程
nvidia-smi

# AMD ROCm
rocm-smi

# 尤其要注意：读取 Ollama 启动时的日志，它会说明检测到了什么
```

在 Linux 下，服务日志位于：

```
journalctl -u ollama --no-pager | grep -i -E "gpu|cuda|rocm|library"
```

你要找的是这样一行：Ollama 表示找到了 GPU 库（CUDA/ROCm）。如果它声明要切换到 CPU，原因就是找到了。（根据操作系统正确安装 runtime 的方法见**第 4 章**——如果你遇到的是本节问题，通常是跳过了第 4 章中的 GPU 步骤。）

节选到此结束。B 小节的后续内容，以及 C（上下文溢出）、D（标签错误）、E（FIM 自动补全过慢）各小节和“症状 → 原因 → 修复”对照表，都在完整指南中。

后续内容请看完整指南

25 章（约 175 页），外加可直接粘贴使用的配置包：调好的 Modelfile、Cline、Aider 和 Tabby、正确的 FIM token、显存 → 模型 → 命令速查表。一次性付款 24 €，终身更新。

<https://nagellm.com/copilote-local/>