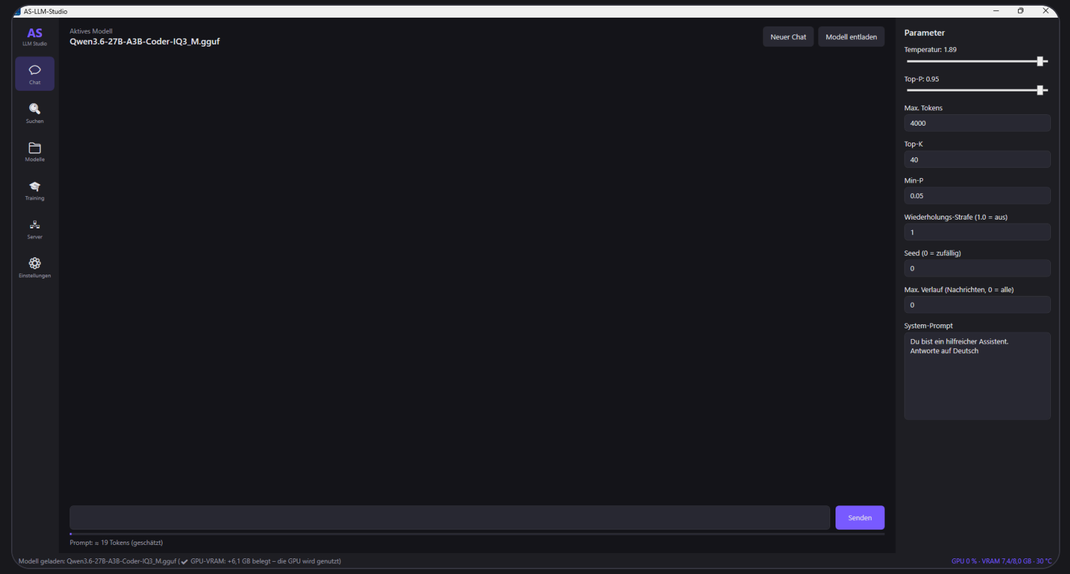


LM-STUDIO-KLON FÜR WINDOWS

AS-LLM-Studio

Lokale Sprachmodelle chatten, trainieren und als API bereitstellen – beschleunigt durch die NVIDIA-GPU.



Alles an einem Ort

💬 Chat

Streaming, Systemprompt, Sampling, Live-Tok/s

🔍 Modelle suchen

GGUF von Hugging Face laden, mit Fortschritt

🖧 Lokaler Server

OpenAI-kompatible API auf localhost

🎓 Training

LoRA-Fine-Tuning mit eigenem Datensatz

⚡ GPU & Leistung

CUDA, Auto-GPU-Layer, KV-Cache, Benchmark

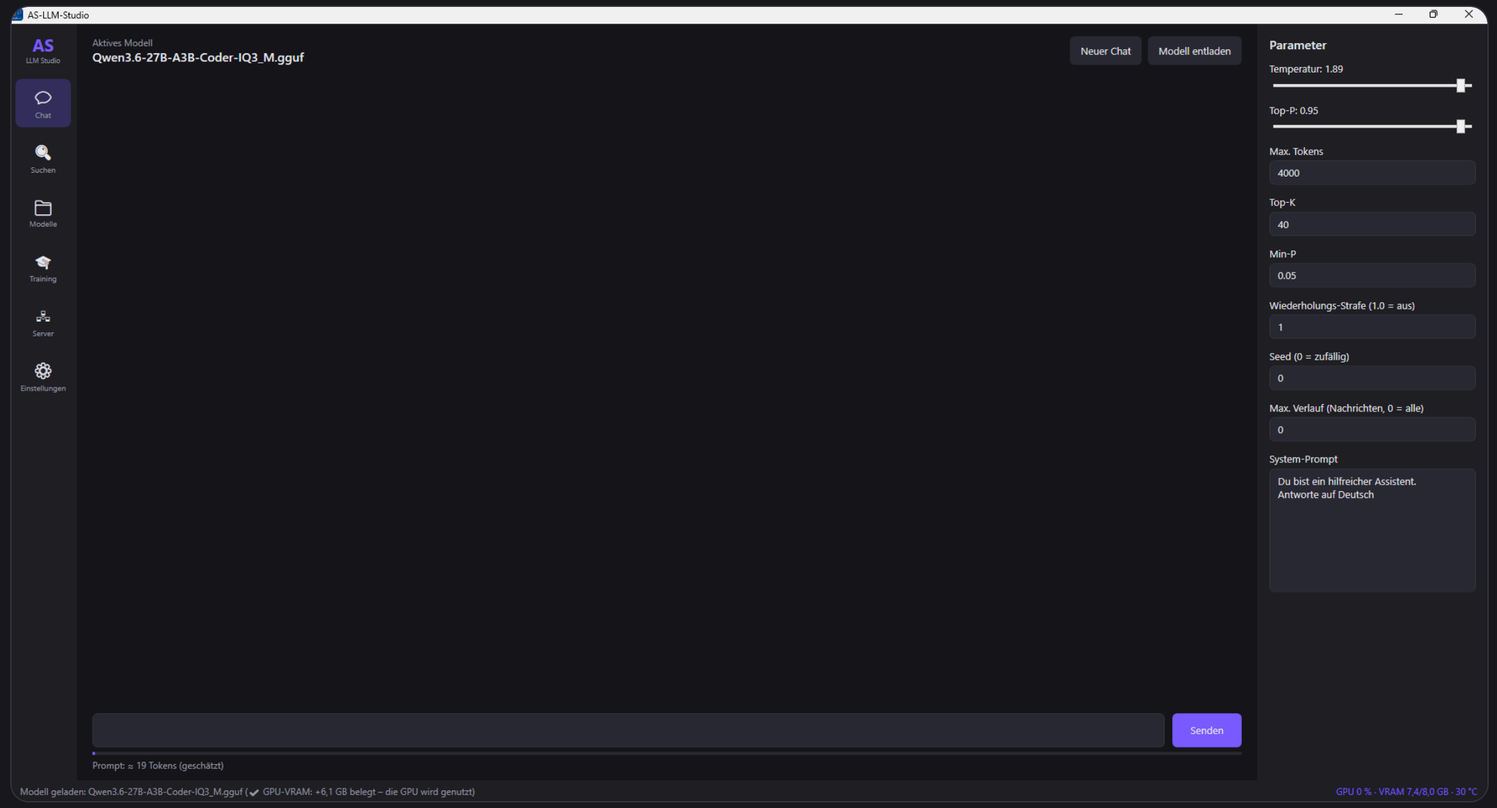
🩺 Diagnose

GPU-Check, Token-Vorschau, Ladebericht

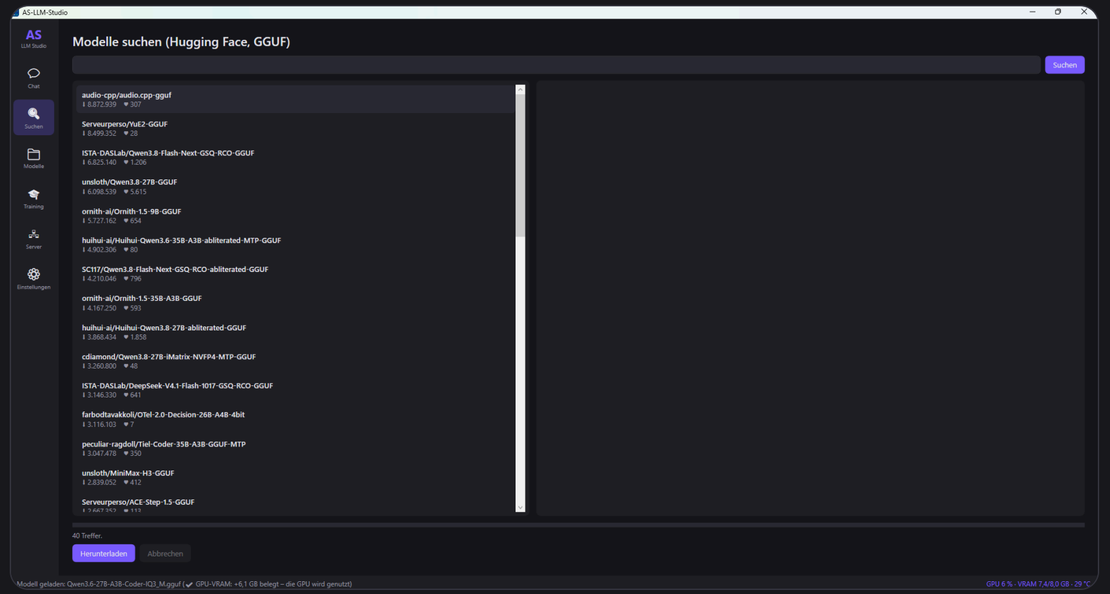
CHAT

Chat mit Live-Messung

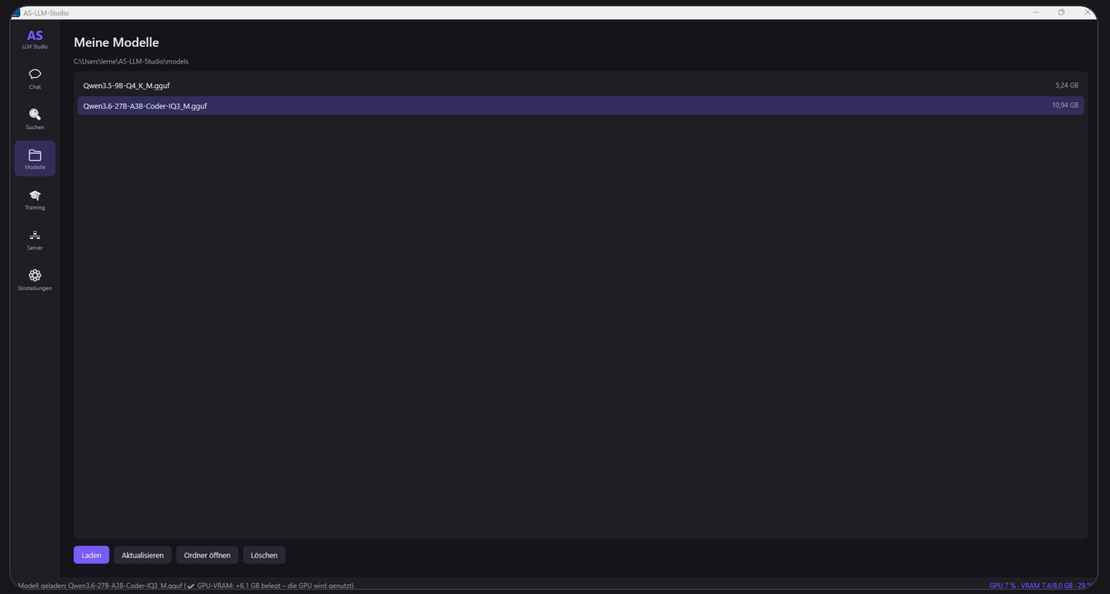
- Alle Sampling-Parameter im Panel
- Token-Vorschau unter dem Eingabefeld
- GPU-Live-Anzeige in der Statusleiste



Modelle suchen und verwalten

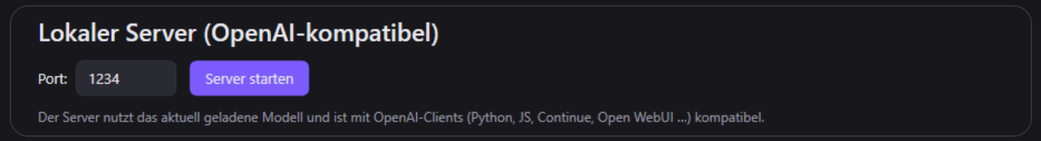


Hugging-Face-Suche für GGUF-Modelle



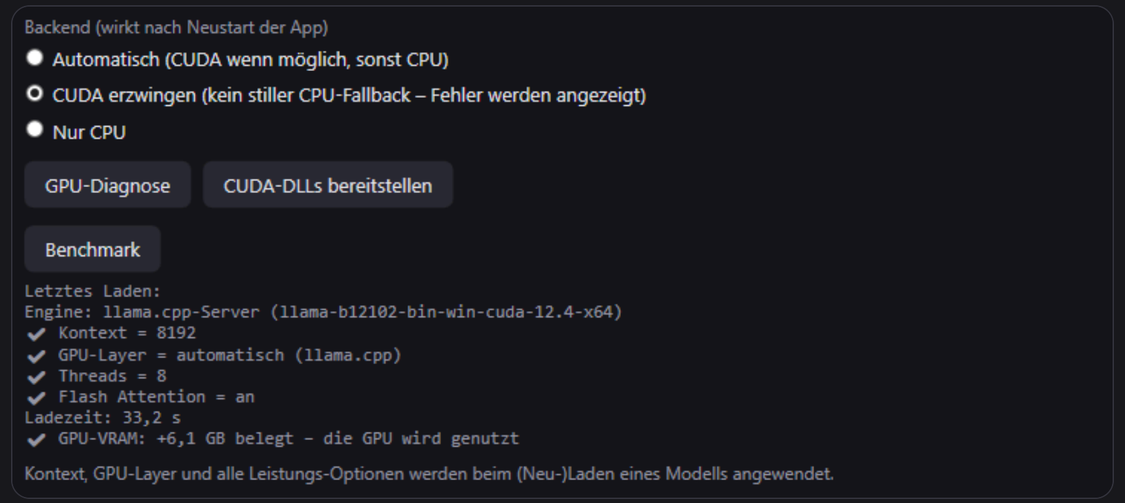
Meine Modelle: laden, öffnen, löschen

Lokaler Server und GPU



OpenAI-kompatibel – läuft mit vorhandenen Clients

```
POST http://localhost:1234
/v1/chat/completions
stream: true · usage · timings
```

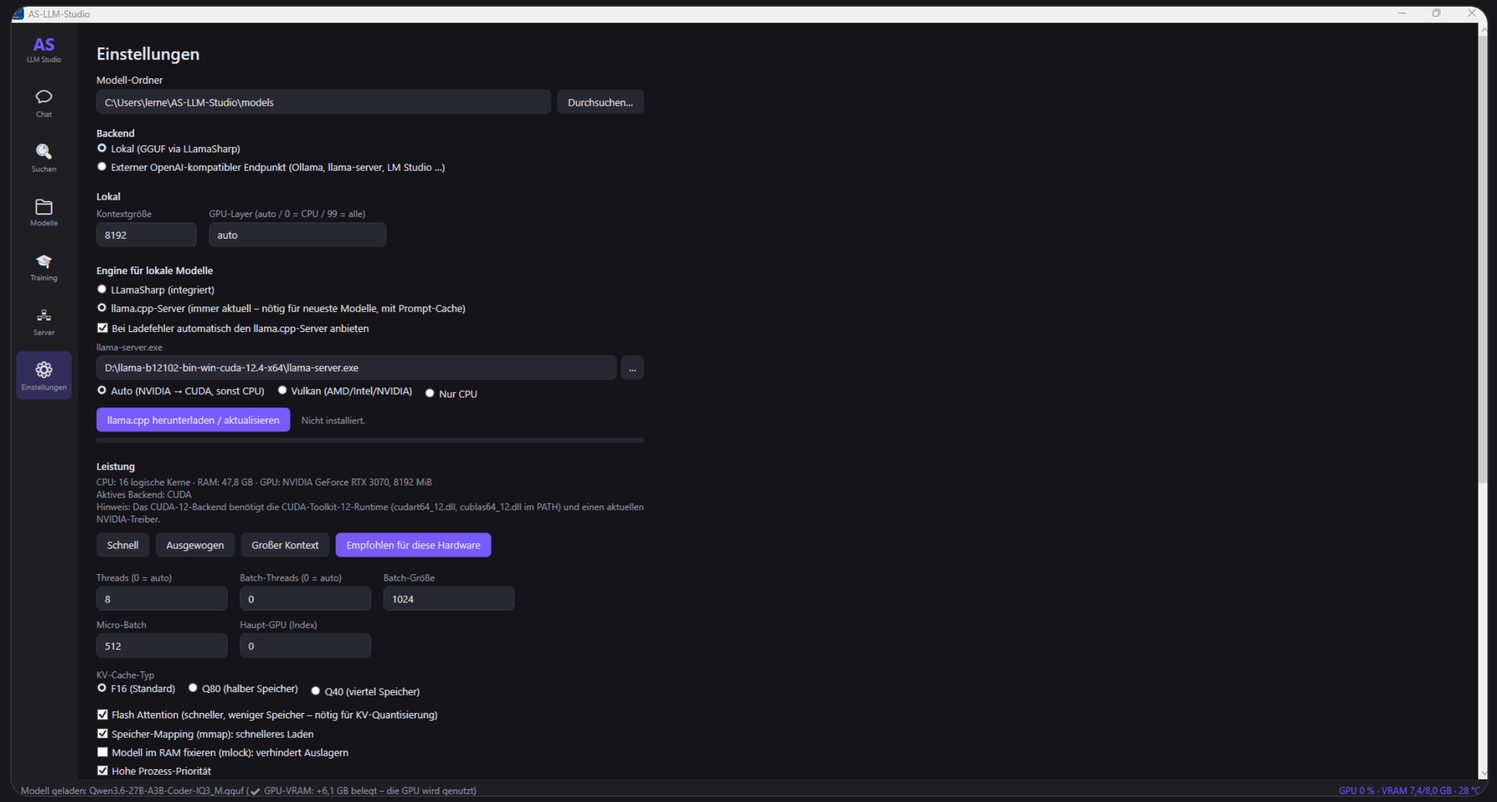


Ladebericht: „GPU-VRAM +6,1 GB belegt – die GPU wird genutzt“

EINSTELLUNGEN

Leistung und Engine

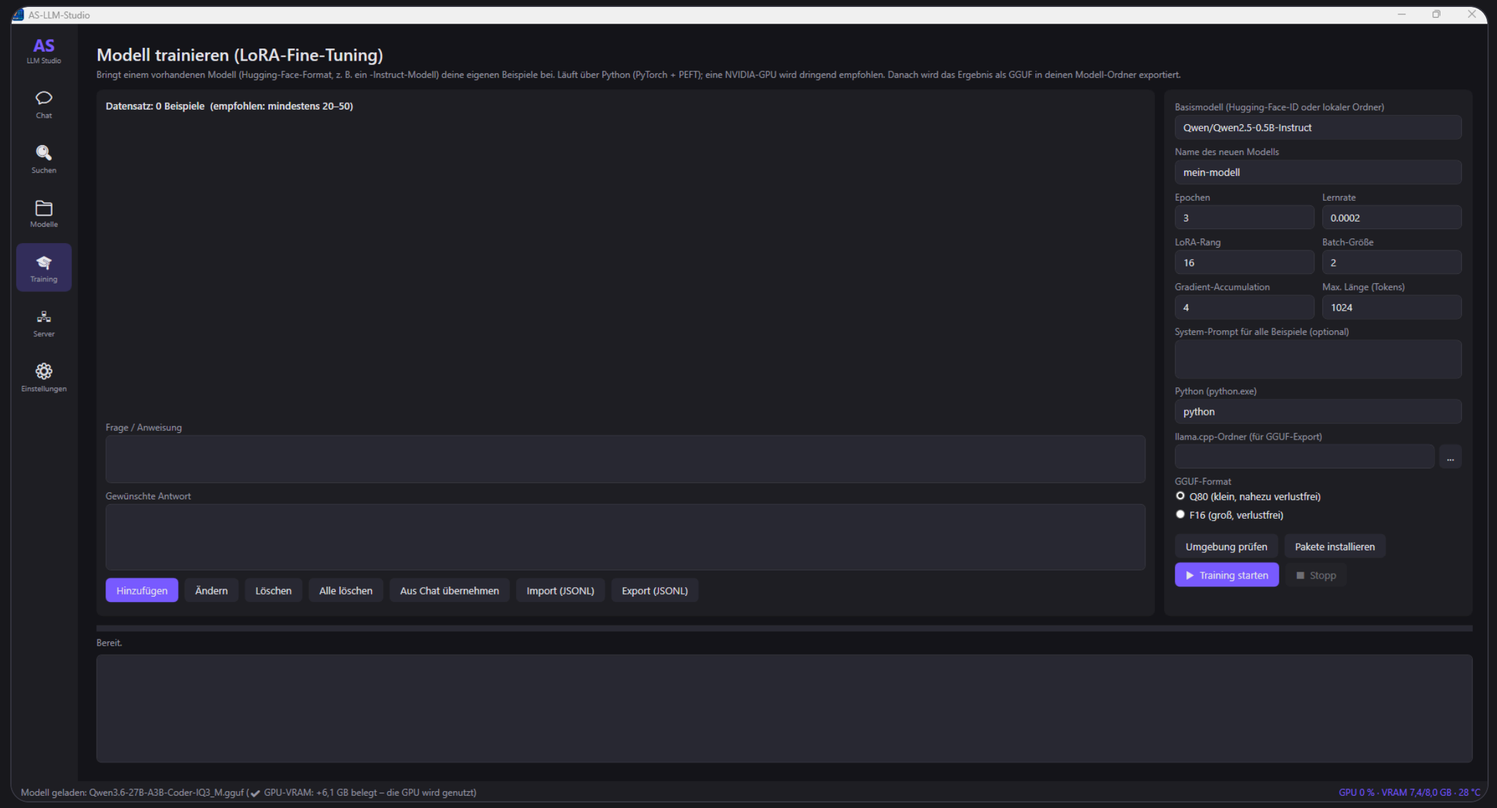
- Engine: LLamaSharp oder llama.cpp-Server mit Prompt-Cache
- Backend: Auto, Vulkan oder nur CPU
- Presets, Threads, KV-Cache, Flash Attention



TRAINING

Eigenes Modell trainieren

- Datensatz anlegen oder importieren
- LoRA-Training mit Live-Loss
- Export als GGUF
- Direkt im Chat testen



LOS GEHT’S

In vier Schritten startklar

1. Projekt in Visual Studio öffnen, F5
2. Modell suchen und laden
3. GPU-Layer auf „auto“ lassen
4. Chatten – oder Server starten

Voraussetzungen

- Visual Studio 2022 mit .NET 8
- NVIDIA-Treiber + CUDA-12-Laufzeit
- Python + PyTorch (nur fürs Training)