Unsloth Studio Setup Guide: Fine-Tune Qwen 3.5 on Your GPU (Step by Step)

March 17, 2026 · by Mark Bartlett

Download this guide as PDF

> **Quick Answer:** Unsloth Studio is a free, open-source web UI that does what no other local AI tool does: inference AND training in one interface. Install with `pip install unsloth && unsloth studio setup`, open localhost:8888, and you can chat with GGUF models, fine-tune them on your data, and export to GGUF for Ollama or LM Studio. Needs an NVIDIA GPU (RTX 30/40/50) for training. Mac works for chat only — MLX training is coming.

📚 **Related:** LoRA Training on Consumer Hardware · Qwen 3.5 Local Guide · Ollama Setup Guide · VRAM Requirements

Every local AI tool makes you choose. LM Studio runs models. Ollama runs models. Neither trains them. If you want to fine-tune, you open a Jupyter notebook, wrestle with Hugging Face configs, and hope your VRAM doesn't run out.

Unsloth Studio is the first tool that puts inference and training in the same window. Load a GGUF, chat with it, drag in a PDF to build a dataset, fine-tune a LoRA, export to GGUF, and run the result — without leaving the browser. It launched today (March 17, 2026) as an open-source beta.

I've been running it for a few hours. Here's how to get set up.

What you need

For chat/inference (any OS):

- Mac, Windows, or Linux
- Python 3.12+
- 8GB+ RAM (more for bigger models)
- CPU-only works for GGUF inference, just slower

For training (NVIDIA only, for now):

- RTX 30, 40, or 50 series GPU
- 8GB VRAM minimum (12-24GB recommended)

- CUDA toolkit installed
- Mac MLX training is listed as “coming soon”

Installation

Three options. Pick whichever matches your setup.

Option 1: pip (fastest)

```
pip install unsloth
unsloth studio setup
unsloth studio -H 0.0.0.0 -p 8888
```

The `studio setup` step compiles llama.cpp binaries for your system. This takes 5-10 minutes the first time. Subsequent launches are fast.

Option 2: From source

```
git clone https://github.com/unslothai/unsloth.git
cd unsloth
pip install -e .
unsloth studio setup
unsloth studio -H 0.0.0.0 -p 8888
```

Use this if you want to track the repo or need the latest fixes before they hit PyPI.

Option 3: Docker

```
docker run -p 8888:8888 --gpus all unsloth/unsloth
```

Cleanest option if you already have Docker with NVIDIA Container Toolkit configured. No Python environment to manage.

Windows note: Conda is the safest path on Windows. Create a fresh environment first:

```
conda create -n unsloth python=3.12
conda activate unsloth
pip install unsloth
unsloth studio setup
unsloth studio -H 0.0.0.0 -p 8888
```

Once it's running, open `http://localhost:8888` in your browser.

Running your first model

The Chat tab is where you start. Click it and you'll see a model search bar.

Picking a Qwen 3.5 model

Type "Qwen 3.5" and you'll see every size and quantization available as GGUF downloads. Here's what fits where:

Model	Quant	RAM + VRAM needed	Good for
Qwen 3.5 0.8B	Q4_K_M	~3.5 GB	Testing, edge devices
Qwen 3.5 2B	Q4_K_M	~3.5 GB	Light tasks, fast responses
Qwen 3.5 9B	Q4_K_M	~6.5 GB	Best balance on 8GB GPUs
Qwen 3.5 27B	Q4_K_M	~17 GB	Full power on 24GB GPUs
Qwen 3.5 35B-A3B	Q4_K_M	~22 GB	MoE — fast inference, fits 12GB+ at lower quant

Pick a model, click download, and it pulls the GGUF. Once loaded, you're chatting.

Thinking mode

Qwen 3.5 supports hybrid reasoning — a thinking mode where the model shows its chain-of-thought before answering. Studio has a toggle for this in the chat interface. For general tasks, leave thinking off. For math, coding, or multi-step reasoning, turn it on.

The recommended settings for thinking mode: temperature 1.0, top_p 0.95, presence_penalty 1.5. For precise coding without thinking: temperature 0.6, top_p 0.95, presence_penalty 0.0.

Other chat features

- **Tool calling:** The model can call tools and self-heal if a call fails. Web search and code execution are built in.
- **File uploads:** Drag in images, PDFs, audio files, code files, or DOCX documents. The model processes them in context.
- **Model Arena:** Load two models side by side and compare their outputs on the same prompt. Useful for testing whether your fine-tune actually improved anything.

Fine-tuning walkthrough

This is where Studio earns its spot. The entire training pipeline runs in the browser.

Step 1: Prepare your data

Go to the Data tab. You have two paths:

Quick path: Upload a PDF, CSV, DOCX, or TXT file. Studio's Data Recipes auto-generate a training dataset from your document. It uses NVIDIA DataDesigner under the hood to create question-answer pairs, instruction-following examples, or whatever format the base model expects.

Manual path: Upload a JSON or CSV in the standard format (instruction/input/output columns or messages format). If you already have a dataset from Hugging Face, this works too.

Step 2: Pick a base model

For fine-tuning, you want a safetensors model (not GGUF — GGUFs are for inference). Studio handles the download.

Recommendations by GPU:

GPU VRAM	Base model	Training method
8GB	Qwen 3.5 4B	QLoRA (4-bit base)
12GB	Qwen 3.5 9B	QLoRA (4-bit base)
24GB	Qwen 3.5 9B	LoRA (16-bit base)
24GB	Qwen 3.5 27B	QLoRA (4-bit base, tight)

Unsloth's custom kernels cut VRAM usage by up to 70% compared to standard Hugging Face training. A QLoRA fine-tune that normally needs 16GB fits in 8GB. Training runs about 2x faster than stock PyTorch on the same hardware.

Step 3: Configure and train

Set your hyperparameters (or use the defaults — they're reasonable):

- **LoRA rank:** 16 (default, good for most tasks)
- **Learning rate:** 2e-4
- **Epochs:** 3-5 for small datasets (200-500 examples)
- **Batch size:** 2 with gradient accumulation 8

Click train. Studio shows real-time loss curves, gradient norms, and GPU utilization. You can watch training progress from another device on your network.

For a 9B model with 500 examples on an RTX 3060 12GB, expect 1-2 hours. On a 3090 or 4090, roughly half that.

Step 4: Test and export

Once training finishes, chat with the fine-tuned model directly in Studio. Compare it against the base model in Arena mode to see if the training worked.

When you're satisfied, export:

- **GGUF** — for use with Ollama, LM Studio, llama.cpp, or vLLM
- **Safetensors** — for use with Hugging Face, vLLM, or further training

The GGUF export is the practical one. Once exported, you can `ollama create` from the GGUF and run your fine-tuned model like any other Ollama model.

How it compares

Studio occupies a new category — inference + training in one UI — so direct comparisons are slightly unfair. But people will ask.

vs LM Studio: LM Studio is more polished for pure inference. Better model discovery, smoother UI, more mature. But it doesn't train models. If you only need to chat with models, LM Studio is

still the better experience today. If you want to train, Studio is the only GUI option that doesn't involve Jupyter notebooks.

vs Ollama: Different tools for different jobs. Ollama is a CLI and API server — headless, scriptable, great for integrations. Studio is a full GUI with training built in. They complement each other: train in Studio, export to GGUF, run in Ollama.

Current limitations

Studio launched today as a beta. Be honest about what that means:

- **Mac is chat-only.** No training on Apple Silicon yet. MLX support is listed as "coming soon" but no date. If you have a Mac and want to train, you still need a cloud GPU or Colab.
- **First install is slow.** The `studio setup` step compiles llama.cpp from source. 5-10 minutes on a fast machine, longer on older hardware. Precompiled binaries are planned.
- **Multi-GPU works but it's rough.** Training across multiple GPUs is supported, but Unsloth says a major multi-GPU upgrade is coming.
- **NVIDIA only for training.** AMD and Intel GPU support is listed as coming. For now, training requires CUDA.
- **Beta edges.** Expect UI quirks, occasional crashes, and features that don't work exactly as documented. This is a 1.0 release, not a mature product.

The bottom line

The "train AND run in one UI" angle is genuinely new. Before Studio, fine-tuning meant Jupyter notebooks with Unsloth scripts, Axolotl YAML configs, or cloud platforms. Studio wraps all of that in a browser tab. Upload a PDF, click train, get a GGUF. That workflow didn't exist in a local GUI before today.

If you have an NVIDIA GPU and want to try fine-tuning without writing Python, this is the easiest path that exists right now:

```
pip install unsloth
unsloth studio setup
unsloth studio -H 0.0.0.0 -p 8888
```

Pull Qwen 3.5 9B, chat with it, throw your data at it, train a LoRA, and export. The whole loop in one window.

Related guides

- LoRA Training on Consumer Hardware
- Qwen 3.5 Local Guide
- Ollama vs LM Studio
- What Can You Run on 8GB VRAM?
- What Can You Run on 24GB VRAM?

Sources: Unsloth Studio docs, Unsloth GitHub (54.3K stars), Qwen 3.5 model guide. Studio is Apache 2.0 (core) / AGPL-3.0 (UI). No telemetry collected.

Get notified when we publish new guides.

Subscribe — free, no spam
