

FINETUNING WITH UNSLOTH

What is Unsloth?

Unsloth is an open-source project(framework) for LLM fine-tuning. It helps you to run the end-to-end pipeline. Unsloth allows you to:

1. Load the model
2. Apply quantization (4-bit, 8-bit, or 16-bit)
3. Train the model
4. Run the inference
5. Evaluate performance
6. Save the trained model
7. Export the model

All of this 2x faster and with up to 70% less GPU memory usage.

Inference Support

Models trained with Unsloth can be directly used with popular inference engines means we can export = saving the trained model in a format so that other tools can use

Unsloth lets you export your model to:

- llama.cpp(gguf)
- Ollama
- vLLM
- SGLang
- Hugging Face Hub
- Open WebUI

These are NOT training tools.

These are inference / deployment tools.

Model Support in Unslloth

Unslloth supports almost all model types that work with Hugging Face Transformers:

- Text-to-Text (text generation, chat models)
- Multimodal models (text + image + audio)
- Image-to-Text (OCR, vision-language models)
- Text-to-Speech (TTS)
- Speech-to-Text (STT)
- Vision-language models
- Classical language models like BERT

Training types supported

Full Fine-Tuning

- LoRA
- QLoRA (4-bit and 8-bit)
- 16-bit LoRA
- FP8 Training (supported on compatible hardware)
- Reinforcement Learning (RL) Support

Unslloth has industry-leading reinforcement learning support, including:

- Reward Modeling
- KTO
- PPO
- GRPO
- GSPO
- DPO
- ORPO

Why Unslloth is Special for RL

- Unslloth can use up to 80% less GPU memory for RL workloads
- This makes advanced reasoning and alignment training possible on much smaller GPUs
- Many RL setups that normally require large servers can run on single GPUs or Colab

Goal of Unsloth or Unsloth's Main Claim?

- 2x to 3x faster training
- 50% to 80% less GPU memory (VRAM) usage
- Ability to train large models even on free GPUs like Colab or Kaggle

What Does This Mean in Practice?

If a task normally takes:

- 10 hours and 40 GB VRAM using standard Hugging Face training

With Unsloth, the same task can take:

- ~5 hours and 12–16 GB VRAM

Example: LLaMA 3.1 (8B)

- If Hugging Face needs 24 GB VRAM, Unsloth needs only ~7–8 GB
- If Hugging Face takes 2 hours, Unsloth finishes in ~1 hour

Why is Unsloth so fast and memory efficient?

Unsloth is fast and memory-efficient because of deep internal optimizations, not just configuration tricks.

What Unsloth does internally:

- Custom CUDA & Triton kernels
- Fused attention and MLP operations
- Optimized forward and backward pass
- Smart gradient checkpointing
- Flash-Attention compatibility
- Manual backpropagation engine
- Optimized Hugging Face + PEFT wrapper
- Automatic sequence packing
- Exact math (no approximation methods)

What is the result of these optimizations?

- Accuracy loss

Practically zero or negligible (no approximation-based degradation)

- Training speed

2x to 3x faster compared to standard Hugging Face training

- GPU memory usage

50%–80% less VRAM

TOP KEY FEATURES TO HIGHLIGHT

1. Long Context Training

This is Unsloth's biggest shock factor

20K–300K+ tokens training

Same model HF me OOM / 28K limit

Example: LLaMA 3.1 (8B)

Approximate maximum context lengths with Unsloth:

GPU Memory Max Context (Approx.)

8 GB ~3,000 tokens

12 GB ~21,000 tokens

16 GB ~40,000 tokens

24 GB ~78,000 tokens

80 GB Up to ~340,000 tokens

Possible on smaller GPUs

These numbers are possible due to Unsloth's memory-efficient kernels, smart checkpointing, and RoPE scaling.

Comparison with Standard Hugging Face Training

Using the same LLaMA 3.1 (8B) model:

- On 12 GB GPU → Out of Memory (OOM)

On 80 GB GPU → Only ~28,000 tokens supported

2. Massive VRAM Reduction (50–80%)

3. Reinforcement Learning (GRPO, GSPO) at Low VRAM

4. Custom Triton + CUDA Kernels

5. End-to-End Pipeline + Export Freedom

6. Exact Math (No Approximation)

7. Works on Free GPUs (Colab / Kaggle)

8. GPU Support

9. Operating System Support

What solutions can we build in the future using Unsloth?

Text-to-Text: Mistral, deepseek, qwen, gemma, phi, llama

Multimodal:

Text-to-Speech (TTS): whisper, orpheus

Speech-to-Text (SST) : whisper large

Vision- qwen3vl

GRPO (Reasoning RL)

DPO / ORPO (Alignment)