Finetune Gemma 3n, Qwen3, Llama 4, Phi-4 & Mistral 2x faster with 80% less VRAM!

Finetune for Free

Notebooks are beginner friendly. Read our guide. Add your dataset, click "Run All", and export your finetuned model to GGUF, Ollama, vLLM or Hugging Face.

Unsloth supports	Free Notebooks	Performance	Memory use
Gemma 3n (4B)	Start for free	1.5x faster	50% less
Qwen3 (14B)	Start for free	2x faster	70% less
Qwen3 (4B): GRPO	Start for free	2x faster	80% less
Gemma 3 (4B)	Start for free	1.6x faster	60% less
Llama 3.2 (3B)	Start for free	2x faster	70% less
Phi-4 (14B)	Start for free	2x faster	70% less
Llama 3.2 Vision (11B)	Start for free	2x faster	50% less
Llama 3.1 (8B)	Start for free	2x faster	70% less
Mistral v0.3 (7B)	Start for free	2.2x faster	75% less
Orpheus-TTS (3B)	Start for free	1.5x faster	50% less

- See all our notebooks for: Kaggle, GRPO, **TTS** & Vision
- See all our models and all our notebooks
- See detailed documentation for Unsloth here

Quickstart

- **Install with pip (recommended)** for Linux devices:

```
pip install unsloth
```

For Windows install instructions, see here.

Unsloth.ai News

- **Gemma 3n** by Google: Read Blog. We uploaded GGUFs, 4-bit models.
- **Text-to-Speech (TTS)** is now supported, including `sesame/csm-1b` and STT `openai/whisper-large-v3`.
- **Qwen3** is now supported. Qwen3-30B-A3B fits on 17.5GB VRAM.
- Introducing **Dynamic 2.0** quants that set new benchmarks on 5-shot MMLU & KL Divergence.
- **Llama 4** by Meta, including Scout & Maverick are now supported.

- **EVERYTHING** is now supported - all models (BERT, diffusion, Cohere, Mamba), FFT, etc. MultiGPU coming soon. Enable FFT with `full_finetuning = True`, 8-bit with `load_in_8bit = True`.
- Introducing Long-context Reasoning (GRPO) in Unsloth. Train your own reasoning model with just 5GB VRAM. Transform Llama, Phi, Mistral etc. into reasoning LLMs!
- DeepSeek-R1 - run or fine-tune them with our guide. All model uploads: here.
  Click for more news
- Introducing Unsloth Dynamic 4-bit Quantization! We dynamically opt not to quantize certain parameters and this greatly increases accuracy while only using <10% more VRAM than BnB 4-bit. See our collection on Hugging Face here.
- Phi-4 by Microsoft: We also fixed bugs in Phi-4 and uploaded GGUFs, 4-bit.
- Vision models now supported! Llama 3.2 Vision (11B), Qwen 2.5 VL (7B) and Pixtral (12B) 2409
- Llama 3.3 (70B), Meta's latest model is supported.
- We worked with Apple to add Cut Cross Entropy. Unsloth now supports 89K context for Meta's Llama 3.3 (70B) on a 80GB GPU - 13x longer than HF+FA2. For Llama 3.1 (8B), Unsloth enables 342K context, surpassing its native 128K support.
- We found and helped fix a gradient accumulation bug! Please update Unsloth and transformers.
- We cut memory usage by a further 30% and now support 4x longer context windows!

Links and Resources

Type	Links
Documentation & Wiki	Read Our Docs
Twitter (aka X)	Follow us on X
Installation	Pip install
Our Models	Unsloth Releases
Blog	Read our Blogs
Reddit	Join our Reddit page

Key Features

- Supports **full-finetuning**, pretraining, 4b-bit, 16-bit and **8-bit** training
- Supports **all transformer-style models** including TTS, STT, multimodal, diffusion, BERT and more!
- All kernels written in OpenAI's Triton language. **Manual backprop engine**.

- **0% loss in accuracy** - no approximation methods - all exact.
- No change of hardware. Supports NVIDIA GPUs since 2018+. Minimum CUDA Capability 7.0 (V100, T4, Titan V, RTX 20, 30, 40x, A100, H100, L40 etc) Check your GPU! GTX 1070, 1080 works, but is slow.
- Works on **Linux** and **Windows**
- If you trained a model with Unsloth, you can use this cool sticker!

Install Unsloth

You can also see our documentation for more detailed installation and updating instructions here.

Pip Installation

Install with pip (recommended) for Linux devices:

```
pip install unsloth
```

To update Unsloth:

```
pip install --upgrade --force-reinstall --no-cache-dir unsloth unsloth_zoo
```

See here for advanced pip install instructions. ### Windows Installation > [!warning] > Python 3.13 does not support Unsloth. Use 3.12, 3.11 or 3.10

1. **Install NVIDIA Video Driver:** You should install the latest version of your GPUs driver. Download drivers here: NVIDIA GPU Drive.
2. **Install Visual Studio C++:** You will need Visual Studio, with C++ installed. By default, C++ is not installed with Visual Studio, so make sure you select all of the C++ options. Also select options for Windows 10/11 SDK. For detailed instructions with options, see here.
3. **Install CUDA Toolkit:** Follow the instructions to install CUDA Toolkit.
4. **Install PyTorch:** You will need the correct version of PyTorch that is compatible with your CUDA drivers, so make sure to select them carefully. Install PyTorch.
5. **Install Unsloth:**

```
pip install unsloth
```

Notes To run Unsloth directly on Windows: - Install Triton from this Windows fork and follow the instructions here (be aware that the Windows fork requires PyTorch >= 2.4 and CUDA 12) - In the SFTTrainer, set `dataset_num_proc=1` to avoid a crashing issue:

```
trainer = SFTTrainer(
    dataset_num_proc=1,
```

```
    ...
)
```

Advanced/Troubleshooting For **advanced installation instructions** or if you see weird errors during installations:

1. Install `torch` and `triton`. Go to https://pytorch.org to install it. For example `pip install torch torchvision torchaudio triton`
2. Confirm if CUDA is installed correctly. Try `nvcc`. If that fails, you need to install `cudatoolkit` or CUDA drivers.
3. Install `xformers` manually. You can try installing `vllm` and seeing if `vllm` succeeds. Check if `xformers` succeeded with `python -m xformers.info` Go to https://github.com/facebookresearch/xformers. Another option is to install `flash-attn` for Ampere GPUs.
4. Double check that your versions of Python, CUDA, CUDNN, `torch`, `triton`, and `xformers` are compatible with one another. The PyTorch Compatibility Matrix may be useful.
5. Finally, install `bitsandbytes` and check it with `python -m bitsandbytes`

Conda Installation (Optional)

`Only use Conda if you have it. If not, use Pip.` Select either `pytorch-cuda=11.8,12.1` for CUDA 11.8 or CUDA 12.1. We support `python=3.10,3.11,3.12`.

```
conda create --name unsloth_env \
    python=3.11 \
    pytorch-cuda=12.1 \
    pytorch cudatoolkit xformers -c pytorch -c nvidia -c xformers \
    -y
conda activate unsloth_env

pip install unsloth
```

If you're looking to install Conda in a Linux environment, read here, or run the below

```
mkdir -p ~/miniconda3
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh -O ~/miniconda3
bash ~/miniconda3/miniconda.sh -b -u -p ~/miniconda3
rm -rf ~/miniconda3/miniconda.sh
~/miniconda3/bin/conda init bash
~/miniconda3/bin/conda init zsh
```

Advanced Pip Installation

`Do **NOT** use this if you have Conda.` Pip is a bit more complex since there are dependency issues. The pip command is different for `torch`

2.2,2.3,2.4,2.5 and CUDA versions.

For other torch versions, we support torch211, torch212, torch220, torch230, torch240 and for CUDA versions, we support cu118 and cu121 and cu124. For Ampere devices (A100, H100, RTX3090) and above, use cu118-ampere or cu121-ampere or cu124-ampere.

For example, if you have torch 2.4 and CUDA 12.1, use:

```
pip install --upgrade pip
pip install "unsloth[cu121-torch240] @ git+https://github.com/unslothai/unsloth.git"
```

Another example, if you have torch 2.5 and CUDA 12.4, use:

```
pip install --upgrade pip
pip install "unsloth[cu124-torch250] @ git+https://github.com/unslothai/unsloth.git"
```

And other examples:

```
pip install "unsloth[cu121-ampere-torch240] @ git+https://github.com/unslothai/unsloth.git"
pip install "unsloth[cu118-ampere-torch240] @ git+https://github.com/unslothai/unsloth.git"
pip install "unsloth[cu121-torch240] @ git+https://github.com/unslothai/unsloth.git"
pip install "unsloth[cu118-torch240] @ git+https://github.com/unslothai/unsloth.git"

pip install "unsloth[cu121-torch230] @ git+https://github.com/unslothai/unsloth.git"
pip install "unsloth[cu121-ampere-torch230] @ git+https://github.com/unslothai/unsloth.git"

pip install "unsloth[cu121-torch250] @ git+https://github.com/unslothai/unsloth.git"
pip install "unsloth[cu124-ampere-torch250] @ git+https://github.com/unslothai/unsloth.git"
```

Or, run the below in a terminal to get the **optimal** pip installation command:

```
wget -qO- https://raw.githubusercontent.com/unslothai/unsloth/main/unsloth/_auto_install.py
```

Or, run the below manually in a Python REPL:

```
try: import torch
except: raise ImportError('Install torch via `pip install torch`')
from packaging.version import Version as V
v = V(torch.__version__)
cuda = str(torch.version.cuda)
is_ampere = torch.cuda.get_device_capability()[0] >= 8
if cuda != "12.1" and cuda != "11.8" and cuda != "12.4": raise RuntimeError(f"CUDA = {cuda}
if   v <= V('2.1.0'): raise RuntimeError(f"Torch = {v} too old!")
elif v <= V('2.1.1'): x = 'cu{}{}-torch211'
elif v <= V('2.1.2'): x = 'cu{}{}-torch212'
elif v  < V('2.3.0'): x = 'cu{}{}-torch220'
elif v  < V('2.4.0'): x = 'cu{}{}-torch230'
elif v  < V('2.5.0'): x = 'cu{}{}-torch240'
elif v  < V('2.6.0'): x = 'cu{}{}-torch250'
else: raise RuntimeError(f"Torch = {v} too new!")
```

```
x = x.format(cuda.replace(".", ""), "-ampere" if is_ampere else "")
print(f'pip install --upgrade pip && pip install "unsloth[{x}] @ git+https://github.com/unsl
```

Documentation

- Go to our official Documentation for saving to GGUF, checkpointing, evaluation and more!
- We support Huggingface's TRL, Trainer, Seq2SeqTrainer or even Pytorch code!
- We're in Hugging Face's official docs! Check out the SFT docs and DPO docs!
- If you want to download models from the ModelScope community, please use an environment variable: `UNSLOTH_USE_MODELSCOPE=1`, and install the modelscope library by: `pip install modelscope -U`.

  unsloth_cli.py also supports `UNSLOTH_USE_MODELSCOPE=1` to download models and datasets. please remember to use the model and dataset id in the ModelScope community.

```
from unsloth import FastLanguageModel, FastModel
import torch
from trl import SFTTrainer, SFTConfig
from datasets import load_dataset
max_seq_length = 2048 # Supports RoPE Scaling internally, so choose any!
# Get LAION dataset
url = "https://huggingface.co/datasets/laion/OIG/resolve/main/unified_chip2.jsonl"
dataset = load_dataset("json", data_files = {"train" : url}, split = "train")

# 4bit pre quantized models we support for 4x faster downloading + no OOMs.
fourbit_models = [
    "unsloth/Meta-Llama-3.1-8B-bnb-4bit",      # Llama-3.1 2x faster
    "unsloth/Meta-Llama-3.1-8B-Instruct-bnb-4bit",
    "unsloth/Meta-Llama-3.1-70B-bnb-4bit",
    "unsloth/Meta-Llama-3.1-405B-bnb-4bit",    # 4bit for 405b!
    "unsloth/Mistral-Small-Instruct-2409",     # Mistral 22b 2x faster!
    "unsloth/mistral-7b-instruct-v0.3-bnb-4bit",
    "unsloth/Phi-3.5-mini-instruct",           # Phi-3.5 2x faster!
    "unsloth/Phi-3-medium-4k-instruct",
    "unsloth/gemma-2-9b-bnb-4bit",
    "unsloth/gemma-2-27b-bnb-4bit",            # Gemma 2x faster!

    "unsloth/Llama-3.2-1B-bnb-4bit",           # NEW! Llama 3.2 models
    "unsloth/Llama-3.2-1B-Instruct-bnb-4bit",
    "unsloth/Llama-3.2-3B-bnb-4bit",
    "unsloth/Llama-3.2-3B-Instruct-bnb-4bit",

    "unsloth/Llama-3.3-70B-Instruct-bnb-4bit" # NEW! Llama 3.3 70B!
```

```
] # More models at https://huggingface.co/unsloth

model, tokenizer = FastModel.from_pretrained(
    model_name = "unsloth/gemma-3-4B-it",
    max_seq_length = 2048, # Choose any for long context!
    load_in_4bit = True,  # 4 bit quantization to reduce memory
    load_in_8bit = False, # [NEW!] A bit more accurate, uses 2x memory
    full_finetuning = False, # [NEW!] We have full finetuning now!
    # token = "hf_...", # use one if using gated models
)

# Do model patching and add fast LoRA weights
model = FastLanguageModel.get_peft_model(
    model,
    r = 16,
    target_modules = ["q_proj", "k_proj", "v_proj", "o_proj",
                      "gate_proj", "up_proj", "down_proj",],
    lora_alpha = 16,
    lora_dropout = 0, # Supports any, but = 0 is optimized
    bias = "none",    # Supports any, but = "none" is optimized
    # [NEW] "unsloth" uses 30% less VRAM, fits 2x larger batch sizes!
    use_gradient_checkpointing = "unsloth", # True or "unsloth" for very long context
    random_state = 3407,
    max_seq_length = max_seq_length,
    use_rslora = False,  # We support rank stabilized LoRA
    loftq_config = None, # And LoftQ
)

trainer = SFTTrainer(
    model = model,
    train_dataset = dataset,
    tokenizer = tokenizer,
    args = SFTConfig(
        max_seq_length = max_seq_length,
        per_device_train_batch_size = 2,
        gradient_accumulation_steps = 4,
        warmup_steps = 10,
        max_steps = 60,
        logging_steps = 1,
        output_dir = "outputs",
        optim = "adamw_8bit",
        seed = 3407,
    ),
)
trainer.train()
```

```
# Go to https://github.com/unslothai/unsloth/wiki for advanced tips like
# (1) Saving to GGUF / merging to 16bit for vLLM
# (2) Continued training from a saved LoRA adapter
# (3) Adding an evaluation loop / OOMs
# (4) Customized chat templates
```

Reinforcement Learning RL including DPO, GRPO, PPO, Reward Modelling, Online DPO all work with Unsloth. We're in Hugging Face's official docs! We're on the GRPO docs and the DPO docs! List of RL notebooks:

- Advanced Qwen3 GRPO notebook: Link
- ORPO notebook: Link
- DPO Zephyr notebook: Link
- KTO notebook: Link
- SimPO notebook: Link

Click for DPO code

```
import os
os.environ["CUDA_VISIBLE_DEVICES"] = "0" # Optional set GPU device ID

from unsloth import FastLanguageModel
import torch
from trl import DPOTrainer, DPOConfig
max_seq_length = 2048

model, tokenizer = FastLanguageModel.from_pretrained(
    model_name = "unsloth/zephyr-sft-bnb-4bit",
    max_seq_length = max_seq_length,
    load_in_4bit = True,
)

# Do model patching and add fast LoRA weights
model = FastLanguageModel.get_peft_model(
    model,
    r = 64,
    target_modules = ["q_proj", "k_proj", "v_proj", "o_proj",
                      "gate_proj", "up_proj", "down_proj",],
    lora_alpha = 64,
    lora_dropout = 0, # Supports any, but = 0 is optimized
    bias = "none",    # Supports any, but = "none" is optimized
    # [NEW] "unsloth" uses 30% less VRAM, fits 2x larger batch sizes!
    use_gradient_checkpointing = "unsloth", # True or "unsloth" for very long context
    random_state = 3407,
    max_seq_length = max_seq_length,
)

dpo_trainer = DPOTrainer(
```

```
    model = model,
    ref_model = None,
    train_dataset = YOUR_DATASET_HERE,
    # eval_dataset = YOUR_DATASET_HERE,
    tokenizer = tokenizer,
    args = DPOConfig(
        per_device_train_batch_size = 4,
        gradient_accumulation_steps = 8,
        warmup_ratio = 0.1,
        num_train_epochs = 3,
        logging_steps = 1,
        optim = "adamw_8bit",
        seed = 42,
        output_dir = "outputs",
        max_length = 1024,
        max_prompt_length = 512,
        beta = 0.1,
    ),
)
dpo_trainer.train()
```

Performance Benchmarking

- For our most detailed benchmarks, read our Llama 3.3 Blog.
- Benchmarking of Unsloth was also conducted by Hugging Face.

We tested using the Alpaca Dataset, a batch size of 2, gradient accumulation steps of 4, rank = 32, and applied QLoRA on all linear layers (q, k, v, o, gate, up, down):

Model	VRAM	Unsloth speed	VRAM reduction	Longer context	Hugging Face + FA2
Llama 3.3 (70B)	80GB	2x	>75%	13x longer	1x
Llama 3.1 (8B)	80GB	2x	>70%	12x longer	1x

Context length benchmarks

Llama 3.1 (8B) max. context length We tested Llama 3.1 (8B) Instruct and did 4bit QLoRA on all linear layers (Q, K, V, O, gate, up and down) with rank = 32 with a batch size of 1. We padded all sequences to a certain maximum sequence length to mimic long context finetuning workloads. | GPU VRAM | Unsloth context length | Hugging Face + FA2 | |———-|———————|————-| | 8 GB | 2,972 | OOM | | 12 GB | 21,848 | 932 | | 16 GB | 40,724 | 2,551 | | 24

GB | 78,475 | 5,789 | | 40 GB | 153,977 | 12,264 | | 48 GB | 191,728 | 15,502 | | 80 GB | 342,733 | 28,454 |

Llama 3.3 (70B) max. context length We tested Llama 3.3 (70B) Instruct on a 80GB A100 and did 4bit QLoRA on all linear layers (Q, K, V, O, gate, up and down) with rank = 32 with a batch size of 1. We padded all sequences to a certain maximum sequence length to mimic long context finetuning workloads.

GPU VRAM	Unsloth context length	Hugging Face + FA2
48 GB	12,106	OOM
80 GB	89,389	6,916

Citation

You can cite the Unsloth repo as follows:

```
@software{unsloth,
  author = {Daniel Han, Michael Han and Unsloth team},
  title = {Unsloth},
  url = {http://github.com/unslothai/unsloth},
  year = {2023}
}
```

Thank You to

- The llama.cpp library that lets users save models with Unsloth
- The Hugging Face team and their TRL library
- Erik for his help adding Apple's ML Cross Entropy in Unsloth
- Etherl for adding support for TTS, diffusion and BERT models
- And of course for every single person who has contributed or has used Unsloth!