

# LLAMA-FACTORY

## What is llama factory?

LLaMA Factory is an all-in-one open source fine-tuning project(framework) that makes LLM training and inference very simple. It works with hundreds of models and datasets.

LLaMA Factory is a UI and CLI based framework that lets you fine-tune LLMs without writing code.

UI (LlamaBoard / WebUI): A beginner-friendly web interface where you can select models, upload datasets, set parameters, and run training, export, or chat — all through a simple GUI.

CLI: If you prefer scripting, automation, or advanced control, the command-line interface lets you run training, export models, start chat, or serve APIs directly from the terminal.

LLAMA Factory is built on top of Hugging Face libraries, It internally wraps Transformers, PEFT, BitsAndBytes, and TRL to provide a “one-click” fine-tuning workflow.

but it also contains a lot of custom code written by the developers for training, UI, templates, dataset handling, and LoRA/QLoRA workflows.

## Training types supported by llama-factory

Method	What it does	Availability
SFT	Instruction Finetuning	Available
LoRA / QLoRA	Low-VRAM fine-tuning	Available
DPO	Direct Preference Optimization	Available
RLHF (PPO + Reward Model)	Reinforcement learning using reward model scoring	Available
RLLAIF	AI-feedback RL	Available
Full Fine-Tune	All weights train	Available

## Model supported by llama-factory

- LLaMA family (Meta)

Including all LLaMA, LLaMA-2, LLaMA-3 variants

- Mistral family

Including Mixtral MoE models

- Qwen family

Qwen-1, Qwen-2, Qwen-VL, Qwen-Coder, etc.

- Gemma models (Google)

- Yi model family (01.AI — China)

- Phi model family (Microsoft)

- Baichuan (Baichuan Inc — China)

- other popular community models

Examples: ChatGLM(Tsinghua University + Zhipu AI), DeepSeek (DeepSeek AI), OpenBuddy, etc

## Dataset format supported by llama-factory

### Alpaca Format (Instruction Finetuning)

```
[  
  {  
    "instruction": "What is preference alignment?",  
    "input": "",  
    "output": "Alignment is making LLM follow human preferences..."  
  }  
]
```

Columns:

- instruction
- input
- output

### ShareGPT Format (Chat finetuning)

Used for role-based chat data.

```
[  
  {  
    "conversations": [  
      {"from": "user", "value": "Hello!"},  
      {"from": "assistant", "value": "Hi, how can I help you?"}  
    ]  
  }  
]
```

### DPO Format (Preference Training)

```
[  
  {  
    "prompt": "Explain RLHF.",  
    "chosen": "RLHF aligns model with human feedback.",  
    "rejected": "I don't know about RLHF."  
  }  
]
```

## Step-by-Step Finetuning (Web UI)

Base Model Select

Dataset Load

Training Configuration

GPU Optimization

Start Training

Evaluation

Chat With Your Model

Export Model

### CLI Training (YAML-Based)

Command	Purpose / Description
<code>llamafactory-cli train config.yaml</code>	Run a training/fine-tuning session (LoRA, QLoRA, Full FT)
<code>llamafactory-cli chat config.yaml</code>	Launch CLI chat after training/export
<code>llamafactory-cli eval config.yaml</code>	Evaluate model (perplexity/custom eval)
<code>llamafactory-cli export config.yaml</code>	Merge adapters and export final model
<code>llamafactory-cli api config_api.yaml</code>	Start an API server endpoint (OpenAI-style)
<code>llamafactory-cli webui</code>	Launch graphical interface (training, eval, chat, export)
<code>llamafactory-cli webchat</code>	Launch web-based chat UI
<code>llamafactory-cli version</code>	Show installed version