From Pretrained to Personal: Privacy-First Fine-Tuning on AI PCs

Why your next fine-tune might never touch the cloud

Daniel Holanda, Iswarya Alex – ML Engineers @ AMD

AMD
together we advance_

PyTorch Conference Europe 2026 – Paris, France

What would you fine-tune…

if sharing data outside your institution was possible?

(Turns out you don’t need to!)

Reality Check: Why don’t we fine-tune?

Answer: Because privacy is often non-negotiable

Key Sectors

- Healthcare Data
- Government and public-sector workloads
- Financial Services
- Proprietary Enterprise Systems

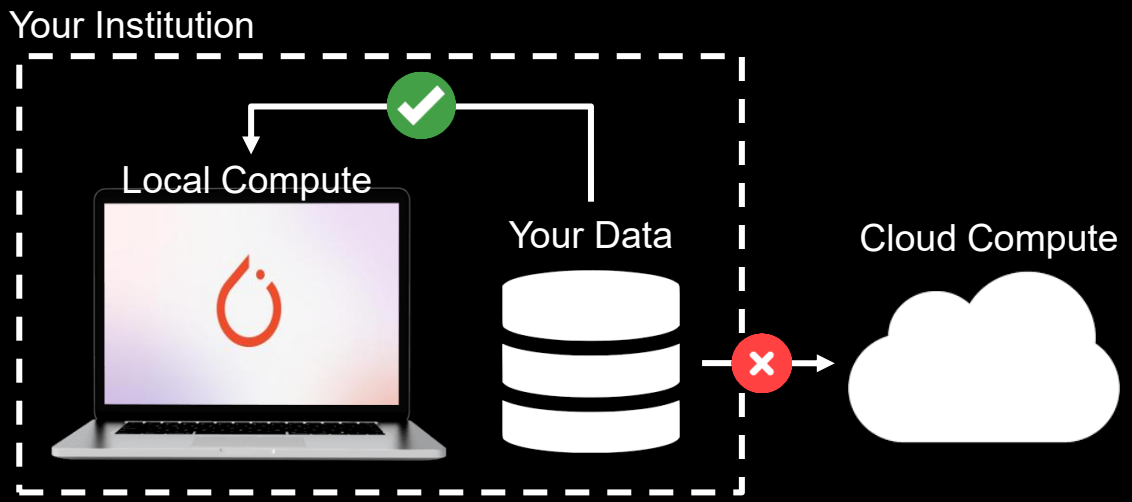
Key Reasons

- External:
  - Regulatory reasons
  - Customer trust with their data
- Internal
  - Trust on cloud providers
  - Security risks and liability

Result: Most companies use OOB models not tuned for their data and use cases

The Shift: Fine-Tuning is Viable Locally

Pytorch on AI PCs crossed a threshold
Local hardware can now support meaningful model fine-tuning, not just inference



What changed? (3 enablers)

- Efficient methods
  - LoRA / QLoRA / Full Fine-Tuning
- Hardware
  - AI PCs (NPUs + modern GPUs)
- Efficient Frameworks
  - Local Pytorch+ROCm training support

This is a capability shift, not incremental improvement

Practical Lessons:
Choosing the ideal tuning method for local fine-tuning

There's no one-size-fits-all in local fine-tuning
The right method depends on what you need to optimize for.

Method	Description	Typical VRAM*	Recommended For
Full Fine-tuning	Updates all model parameters. Maximum quality; highest memory and compute usage.	High	Maximum quality; research; large VRAM
LoRA	Trains small adapter matrices while freezing base model. 3–5x faster; ~95–98% full quality.	Medium	Advanced users; multiple adapters; more VRAM
QLoRA	4-bit quantization + LoRA adapters. Lowest memory use, fastest, small quality trade-off.	Low	Most users; fast experiments; limited VRAM

Careful: Serving full fine-tunes on cloud can be expensive, while LoRA/QLoRA may not cost anything extra

Practical Lessons:
Choosing the ideal adaptation method for local fine-tuning

Terminology

- Model tuning methods (Full FT, LoRA, QLoRA, …) ⟶ (Mostly) which weights are updated
- Adaptation Methods (SFT, RLHF, DPO, …) ⟶ How behavior is aligned

Adaptation Methods

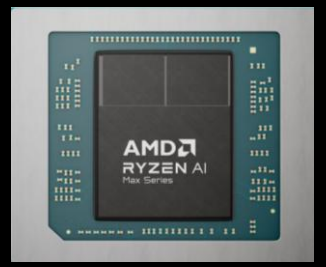
Method	Description	VRAM Required
SFT Supervised Fine-Tuning	Train the model on labeled prompt–response pairs.	Low. Only the model itself is trained.
RLHF Reinforcement Learning from Human Feedback	Use a reward model (trained on human preferences) to guide policy updates via reinforcement learning.	High. Requires policy model + reward model (+ sometimes a reference model) and multiple forward passes.
DPO Direct Preference Optimization	Directly optimize the model on preference pairs (chosen vs rejected responses) without a separate reward model or reinforcement learning.	Medium. Needs a policy model + reference model for comparison, but simpler than RLHF.

Practical Lessons:
Choosing the ideal framework for local fine-tuning

Selecting a fine-tuning tool requires considering factors like hardware requirements, ease of use, and compatibility with your models.

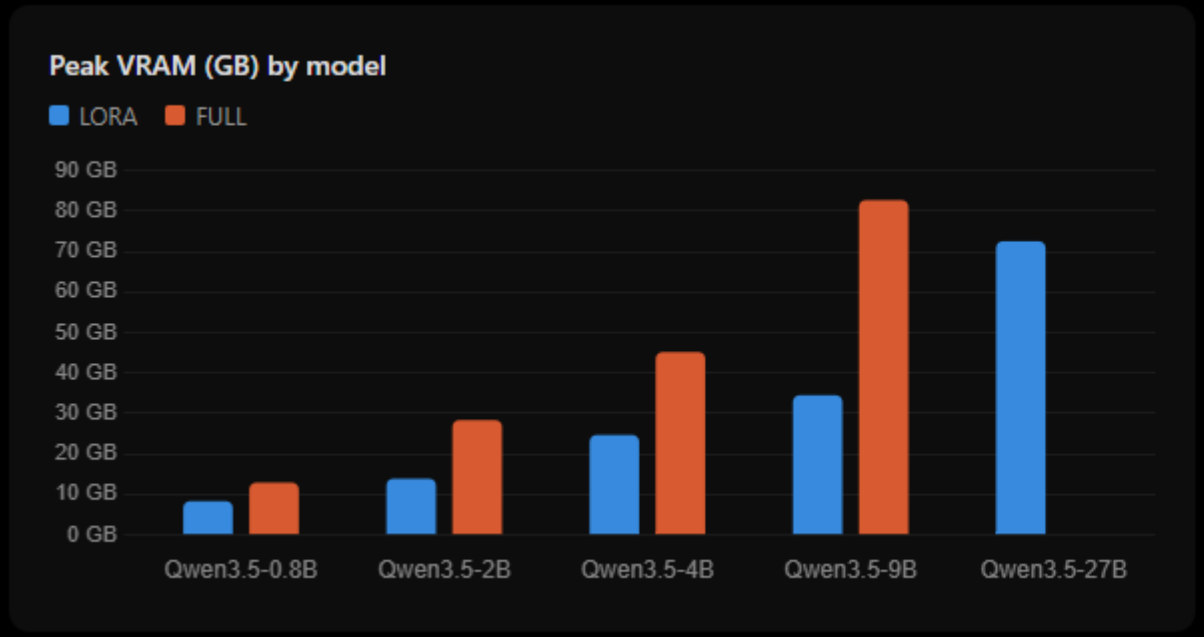
Feature	Pytorch Direct	Unsloth	LlamaFactory
Ease of Use	Low – Full Code Required	Medium – Minimal Scripts	High – UI + CLI
Speed / Memory	Baseline	memory-efficient	memory-efficient
Models Supported	Any HF/PyTorch	Most HF open-source	Most HF open-source
Fine-tuning Methods	Full, LoRA, QLoRA	Full, LoRA, QLoRA	Full, LoRA, QLoRA
Best For	Full control & customization	Fast experiments on various hardware	Easy large-scale fine-tuning & team use

Case Study:
Fine-Tuning on AMD Ryzen™ AI Max+ 395



SETUP USED
AMD Ryzen™ AI Max+ 395
128GB unified memory
45-120W TDP
Pytorch 2.10.0+ROCm7.12

Training Time and VRAM usage

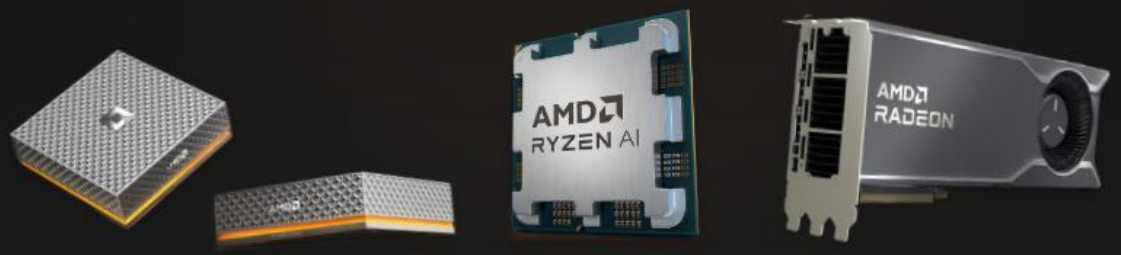


* Training time measured over 100 optimizer steps (batch size 2, gradient accumulation 4, effective batch size 8, max sequence length 512) on the databricks-dolly-15k dataset (200 samples: 160 train / 40 eval). Python 3.12.13, PyTorch 2.10.0+ROCm 7.12.0 (HIP 7.2.0), Transformers 5.4.0, PEFT 0.18.1, TRL 0.29.1, Datasets 4.2.0. Date collected on 3/30/2026.

Start your AI journey with AMD Developer Playbooks

Step-by-step playbooks and powerful AI models ready to run. Choose your device to get started.

AMD Ryzen™ AI Halo | Ryzen™ AI APUs | Radeon™ GPUs | All



PLAYBOOKS

Learn. Build. Deploy.

Step-by-step guides to help you master AI development on AMD hardware. From fine-tuning to deployment, we've got you covered.

Platform: All | Windows | Linux | fine-tuning

3 playbooks matching "fine-tuning"

Fine-tune LLMs with PyTorch and ROCm
Fine-tune large language models using PyTorch and ROCm on STX Halo™
45 MIN intermediate

LLM Fine-tuning with llama factory
Fine-tune large language models using Llama Factory and LoRA techniques on your STX Halo™
1 HR intermediate

Optimized Fine-tuning with Unsloth QLoRA
Use Unsloth for memory-efficient QLoRA fine-tuning on STX Halo™
1h 30m advanced

Launching later this month
amd.com/playbooks

Available Today
amd.com/developer

AMD
together we advance_

What would you fine-tune…

if your data never had to leave your laptop?


AMD