Clase 148 — LLMs aplicados: fine-tuning, prompting (+ LoRA / QLoRA, DPO, vLLM)

Parte: 2 — Deep Learning · Fuente: docs HuggingFace PEFT + TRL + vLLM + papers LoRA (Hu et al. 2021), QLoRA (Dettmers et al. 2023), DPO (Rafailov et al. 2023). Duración estimada: 110 min.

Clase 148 — LLMs aplicados: fine-tuning, prompting (+ LoRA / QLoRA, DPO, vLLM)

> *Parte: 2 — Deep Learning · Fuente: docs HuggingFace PEFT + TRL + vLLM + papers LoRA (Hu et al. 2021), QLoRA (Dettmers et al. 2023), DPO (Rafailov et al. 2023). Duración estimada: 110 min.*

Objetivo

Manejar Large Language Models (Llama 3, Mistral, Qwen, etc.) en flujos reales: prompting técnico (zero-shot, few-shot, chain-of-thought), fine-tuning eficiente con LoRA / QLoRA (entrenar solo 0.1-1 % de parámetros), alineamiento con DPO (preferencias, en lugar de RLHF clásico), e inference de producción con vLLM (continuous batching, PagedAttention).

Resultados de aprendizaje

Al finalizar, el estudiante podrá:

- Diseñar prompts con system prompts, few-shot examples y chain-of-thought.
- Aplicar LoRA con peft library: agregar adapters chicos a un LLM grande, entrenar solo ~0.5 % de params.
- Aplicar QLoRA (cuantización 4-bit) para fine-tunear Llama 7B en una sola GPU de 24 GB.
- Alinear con preferencias usando DPO (trl library) — más simple y estable que RLHF.
- Servir un modelo con vLLM y medir throughput.

Temas

- LLM stack en 2026: base → SFT (Supervised Fine-Tuning) → DPO/RLHF → inference optimizada.
- Prompting técnico: structure, few-shot, chain-of-thought, function calling.
- PEFT (Parameter-Efficient Fine-Tuning): LoRA, QLoRA, adapters.
- DPO vs RLHF.
- vLLM: continuous batching, PagedAttention, OpenAI-compatible API.
- Evaluación: MMLU, HumanEval, MT-Bench, LMSys Arena.

Versión profundizada — 2026

El tema moderno que antes vivía como complemento dentro de esta clase ahora tiene su(s) clase(s) propia(s) con patrón completo, ejercicios y homework:

- Clase 128a — LoRA / QLoRA: fine-tuning eficiente de LLMs
- Clase 128b — DPO y RLHF: alineamiento de LLMs
- Clase 128c — vLLM y TGI: serving de LLMs en producción

Definiciones y características

- Prompting: arte de estructurar el input para que el modelo dé buenos outputs.
- Few-shot: incluir 2-5 ejemplos del task en el prompt.

- Chain-of-thought (CoT): pedirle al modelo que "piense paso a paso" — mejora dramática en tareas razonamiento.
- System prompt: instrucciones generales antes del diálogo (rol, restricciones).
- PEFT: técnica de fine-tuning con pocos parámetros.
- LoRA: rank-r decomposition para los deltas.
- QLoRA: LoRA + 4-bit quantization del base.
- DPO: loss directa sobre preferencias, sin reward model.
- vLLM: framework de inference optimizada open-source.
- KV cache: cache de K, V de tokens previos, crítico para inference autoregresiva.

Dataset / recursos

- HuggingFace Hub para modelos.
- Datasets de instrucción: Alpaca, ShareGPT, OpenOrca.
- Datasets de preferencia: Anthropic HH-RLHF, UltraFeedback.
- Librerías: transformers, peft, trl, bitsandbytes, vllm.

Ejercicios

1. Prompt engineering: para una tarea (e.g., clasificación de quejas), iterar 5 versiones de prompt (zero-shot, few-shot, CoT, etc.). Medir accuracy en un set chico.
2. LoRA SFT: fine-tunear un Mistral 7B Instruct con LoRA sobre un dataset propio chico (1k-10k ejemplos).
3. QLoRA: el mismo experimento pero con quantization 4-bit. Comparar memoria y velocidad.
4. DPO: con un dataset de preferencias mínimo (e.g., 500 pairs), aplicar DPO sobre el modelo SFT.
5. vLLM serving: levantar vLLM con el modelo + LoRA adapter, medir throughput vs HF pipeline.

Homework verificable

Fine-tunear un LLM open con LoRA para una tarea propia:

1. Elegir Llama 3 8B Instruct o Mistral 7B Instruct.
2. Dataset propio: 500-2000 pares (instrucción, respuesta).
3. QLoRA config: r=16, target=['q_proj','k_proj','v_proj','o_proj'], 4-bit.
4. Entrenar 3 épocas; evaluar perplexity en val.
5. Servir con vLLM y testear 10 ejemplos.

Criterio de aceptación: perplexity en val mejora vs base; el modelo responde apropiadamente al dominio para los 10 ejemplos test.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
RuntimeError: CUDA out of memory con Llama	Sin quantization. Fix: QLoRA con bnb_4bit.
LoRA con target_modules='all-linear' lento	Demasiados adapters. Fix: solo q/v en atte
Modelo SFT sobre dataset chico pierde capa	Catastrophic forgetting parcial. Fix: subi
DPO sin ref_model actualizado	El ref model debe ser SFT init. Fix: usar
vLLM no carga LoRA adapter	Hay que mergear LoRA + base primero o usar

Preguntas frecuentes

¿Open-source o API (GPT-4, Claude)?

API para prototipos, casos non-críticos. Self-host con open-source cuando importan: privacidad, costo a escala, control total. Ambos coexisten.

¿LoRA o full fine-tune?

LoRA en 99 % de los casos. Full fine-tune solo si tenés cluster grande y querés cambiar significativamente la base.

¿DPO o RLHF?

DPO por default — más simple y casi tan bueno. RLHF para tareas muy específicas donde necesitás un reward model fino.

¿vLLM o TGI?

vLLM ligeramente más rápido en benchmarks. TGI (HuggingFace) más integrado con HF Hub. Ambos buenos.

¿Cuánto cuesta entrenar un LLM desde cero?

Llama 3 70B: estimado $10M-50M en compute. Para custom: forget about it — fine-tunea uno existente.

Referencias

- Hu et al. (2021), LoRA: Low-Rank Adaptation of Large Language Models, ICLR.
- Dettmers et al. (2023), QLoRA: Efficient Finetuning of Quantized LLMs, NeurIPS.
- Rafailov et al. (2023), Direct Preference Optimization, NeurIPS.
- Kwon et al. (2023), Efficient Memory Management for LLM Serving with PagedAttention, SOSP.
- PEFT docs, TRL docs, vLLM docs.

Siguiente clase

Clase 149 — LoRA / QLoRA: fine-tuning eficiente de LLMs

Apéndice: notebook (primer bloque)

Primera celda ejecutable del notebook de la clase.

```
# Imports y configuración inicial
```

Archivos complementarios

- notebook.ipynb