Clase 123 — PyTorch Lightning: Trainer, callbacks, distributed

Parte: 2 — Deep Learning · Fuente: Lightning docs. Duración estimada: 80 min.

Clase 123 — PyTorch Lightning: Trainer, callbacks, distributed

Parte: 2 — Deep Learning · Fuente: Lightning docs. Duración estimada: 80 min.

Objetivo

Aprender PyTorch Lightning — la capa de abstracción que convierte PyTorch puro (mucho boilerplate) en algo tan productivo como Keras pero conservando flexibilidad. Cubrir LightningModule, Trainer, callbacks, logging (W&B/TensorBoard), distributed training con un solo kwarg, mixed precision automática.

Resultados de aprendizaje

Al finalizar, el estudiante podrá:

- Subclassear LightningModule con training_step, validation_step, configure_optimizers.
- Usar Trainer(max_epochs, accelerator='auto', devices='auto', precision='bf16-mixed', logger=...).
- Aplicar callbacks: EarlyStopping, ModelCheckpoint, LearningRateMonitor.
- Activar distributed con strategy='ddp' o 'fsdp' para multi-GPU sin reescribir nada.
- Loggear a W&B / TensorBoard / MLflow vía 1 línea.

Temas

- LightningModule vs nn.Module puro.
- Trainer args: max_epochs, devices, precision, strategy, accumulate_grad_batches.
- Callbacks integrados.
- LightningDataModule para data pipelines.
- Distributed training: ddp, fsdp, deepspeed.
- Logging multi-backend.

Definiciones y características

- LightningModule: extiende nn.Module con hooks (training_step, etc.).
- Trainer: orquesta el entrenamiento. Maneja loops, device, distributed.
- self.log(...): registra métrica al logger; se agrega en epoch/batch.
- strategy: 'auto' | 'ddp' | 'fsdp' | 'deepspeed_stage_2'.
- precision: '32-true' | '16-mixed' | 'bf16-mixed'.

Dataset / recursos

- Fashion-MNIST o cualquier dataset previo.
- Librerías: lightning, torch, torchmetrics.

Ejercicios

1. LightningModule básico: convertir el MLP de 108a a Lightning.
2. Callbacks: agregar EarlyStopping(patience=5) + ModelCheckpoint(save_top_k=3).

3. Mixed precision: Trainer(precision='bf16-mixed'). Comparar tiempo.
4. W&B logging: Trainer(logger=WandbLogger(project='test')). Ver curvas online.
5. DDP: si tenés 2+ GPUs, strategy='ddp', devices=2. Verificar speedup.

Homework verificable

Re-entrenar el modelo Fashion-MNIST en Lightning + W&B/TensorBoard:

1. LightningModule con train/val/test steps.
2. Trainer con EarlyStopping, ModelCheckpoint, mixed precision.
3. Logging a TensorBoard.
4. Reportar accuracy + screenshot del dashboard.

Criterio de aceptación: accuracy ≥ 0.87; dashboard muestra curvas de train/val.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
self.log no aparece en logger	Falta llamar a self.log('val_loss', loss)
Olvido return loss en training_step	Lightning no puede backprop. Fix: devolver
Multi-GPU con DDP rompe en Jupyter	DDP no funciona en notebooks. Fix: usar dd
Trainer(max_epochs=100) con datasets gigan	Sin max_steps. Fix: max_steps=10_000 como
Mixed precision con OOM	A veces empeora. Fix: bajar batch o usar b

Preguntas frecuentes

Lightning o PyTorch puro?

Lightning para producción / experimentos serios — quita boilerplate, agrega features (distributed, callbacks). Puro para tutoriales y casos muy custom.

Lightning vs HuggingFace Trainer?

Trainer (HF) está más integrado con transformers. Lightning es más general. Si trabajás con LLMs/HF, Trainer; sino Lightning.

FSDP cuándo?

Cuando el modelo no entra en una GPU. Lightning lo activa con strategy='fsdp'.

lightning o pytorch-lightning?

Mismo proyecto. lightning es el nuevo nombre desde 2023.

Logger recomendado?

W&B (Weights & Biases) — mejor DX, comparación de experimentos. TensorBoard si tenés que ser local-only.

Referencias

- Lightning docs.

- Falcon et al. (2019), PyTorch Lightning.
- W&B: <https://wandb.ai/>.

Siguiente clase

Clase 124 — tf.data API

Apéndice: notebook (primer bloque)

Lightning organiza tu código en LightningModule + LightningDataModule + Trainer. Soporta DDP, FSDP, mixed precision, checkpoints y logging automático. Fallback completo si lightning/torch no están.

```
USE_PL = False
try:
    import torch
    import torch.nn as nn
    import torch.nn.functional as F
    from torch.utils.data import DataLoader, TensorDataset
    import pytorch_lightning as pl
    USE_PL = True
    print('lightning:', pl.__version__)
except Exception as e:
    print('lightning no disponible. Fallback con sklearn MLP. Motivo:', type(e).__name__)
import numpy as np
np.random.seed(42)
```

Archivos complementarios

- notebook.ipynb