

L U M B R E



Frameworks: PyTorch y TensorFlow

Grado: 7

Materia: Deep Learning · Inteligencia Artificial

11/10/2026

Antes de empezar

Un framework libera al investigador de escribir el cálculo diferencial a mano: ofrece tensores acelerados, autodiferenciación y bloques reutilizables. Saber definir un modelo, entrenarlo y guardarlo es la albañilería básica del aprendizaje profundo aplicado.

QUÉ APRENDERÁS

- Definir, entrenar y guardar un modelo profundo en un framework moderno, manejando tensores, autodiferenciación, bucles de entrenamiento y serialización de pesos.

Frameworks de deep learning

Declarar, dejar que derive

PyTorch y TensorFlow no inventan las matemáticas: las automatizan. Declaras el modelo y la pérdida, y el framework deriva, mueve tensores a la GPU y guarda el resultado. Dominar esa mecánica es el oficio diario.

VIDEO

PyTorch desde cero

Vídeo en español que construye paso a paso un modelo con tensores y autodiferenciación.



Escanea para verlo en Lumbre

<https://www.youtube.com/watch?v=QetoD5LXlEg>

El tensor como unidad

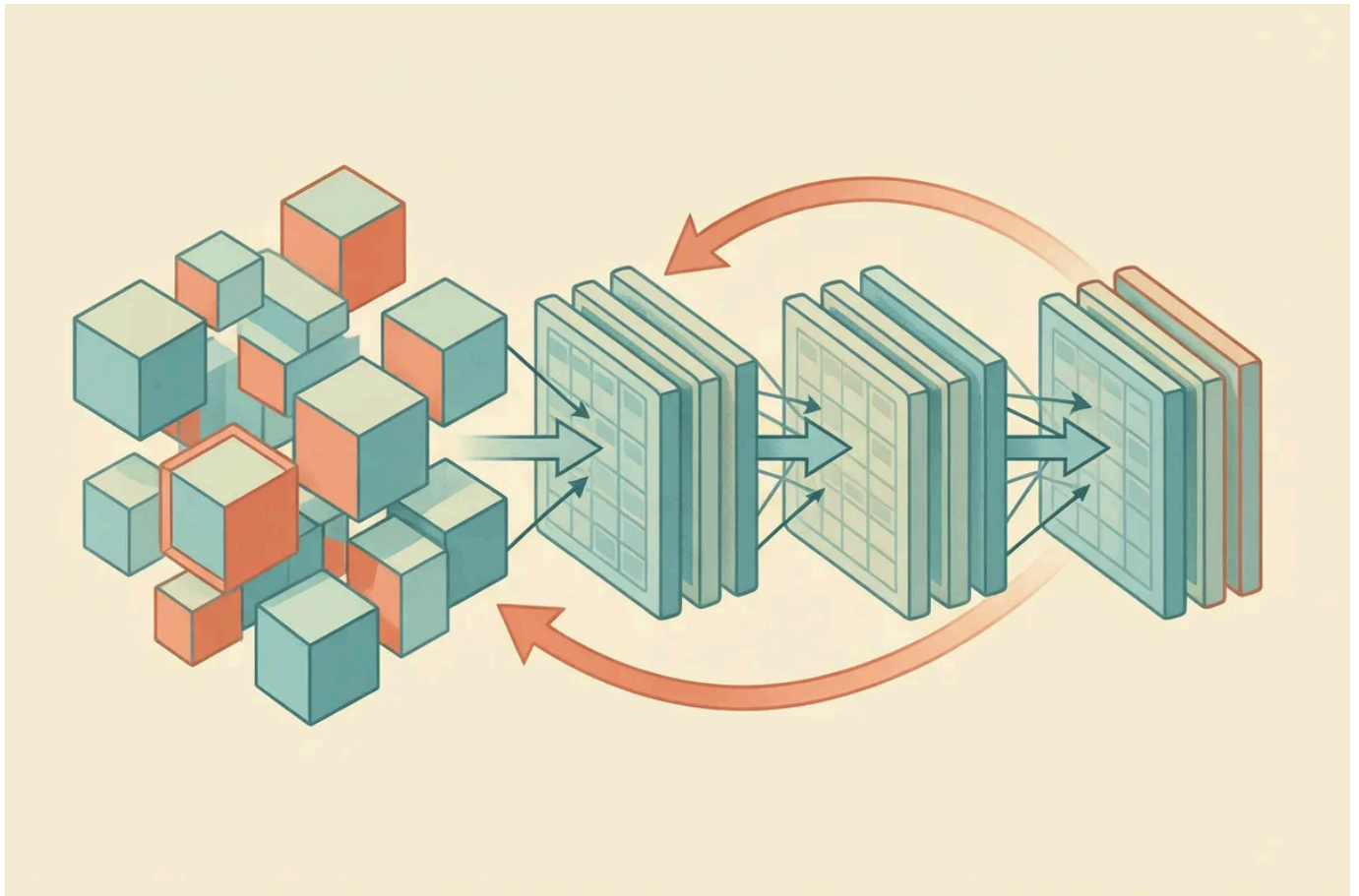
Todo dato y todo parámetro es un tensor: un array multidimensional con un dispositivo asociado. Las operaciones entre tensores se registran en un grafo que hará posible la retropropagación.

Definir el modelo

- Se componen capas o módulos en una clase que describe el pase hacia delante.
- Los parámetros aprendibles quedan registrados automáticamente.
- El mismo esqueleto sirve para CPU, GPU o varias tarjetas.

Autodiferenciación

El framework construye el grafo de operaciones y aplica la regla de la cadena por ti. Solo programas el avance; el gradiente llega con llamar a retroceder, sin derivadas manuales.



El framework registra el grafo de operaciones: programas el pase hacia delante y el gradiente llega solo.

El bucle de entrenamiento

Cada iteración limpia gradientes, avanza el lote, calcula la pérdida, retropropaga y actualiza con el optimizador. Ese ciclo de cinco pasos, repetido por épocas, es el corazón de todo entrenamiento.

Pérdida y optimizador

Se elige una función de pérdida acorde a la tarea y un optimizador que combine el gradiente con su tasa de aprendizaje. La mayoría de decisiones de ajuste viven en estas dos piezas.

Datos y lotes

Un dataset expone muestras y un dataloader las agrupa en lotes, los baraja y los carga en paralelo. Alimentar bien la tubería de datos suele marcar la velocidad tanto como la propia red.

Guardar y cargar

Se serializan los pesos —el estado del modelo— y a veces también la arquitectura. Un checkpoint permite reanudar un entrenamiento interrumpido o exportar el modelo entrenado para servirlo.

Elegir framework

PyTorch domina la investigación por su estilo imperativo y natural; TensorFlow aportó despliegue robusto y herramientas como TFLite. Ambos son maduros; la decisión suele ser de ecosistema y equipo.

Errores frecuentes

- Olvidar poner los gradientes a cero en cada iteración y acumular los del paso anterior, sesgando la actualización.
- Mover el modelo y los datos a dispositivos distintos, mezclando CPU y GPU, y provocar errores o lentitud.
- Confundir guardar la arquitectura con guardar solo los pesos, y no poder recargar el modelo entrenado.

Ejemplo resuelto: el ciclo de cinco pasos

Vas a entrenar tu primera red en PyTorch y la pérdida no baja. Revisas el bucle de entrenamiento paso a paso:

1. Limpiar los gradientes acumulados del paso anterior con `cero_grad`.
2. Avanzar el lote por el modelo para obtener la predicción.
3. Calcular la pérdida comparando predicción y etiquetas.
4. Retropropagar con `backward` para poblar los gradientes.
5. Actualizar los pesos con el optimizador y repetir por épocas.

¿Para qué sirve en la realidad?

Manejar un framework es el oficio diario del deep learning aplicado: casi nadie escribe ya la retropropagación a mano, y dominar tensores, bucle y checkpoints es lo que separa una idea de un modelo entrenado y desplegado.

1. En un framework moderno, ¿qué hace la autodiferenciación por el programador?

- ☐ diseñar la arquitectura de la red
- ☐ calcular automáticamente los gradientes aplicando la regla de la cadena
- ☐ elegir la tasa de aprendizaje óptima
- ☐ limpiar los datos de entrada

2. Guardar el estado de un modelo suele implicar serializar sus parámetros aprendibles en un checkpoint.

- ☐ Verdadero
- ☐ Falso

ACTIVIDAD INTERACTIVA DISPONIBLE EN LUMBRE

3. Frameworks



4. Ordena los pasos del bucle de entrenamiento de un framework:

Elementos:

- dejar a cero los gradientes del paso previo
- pasar el lote por el modelo
- calcular la pérdida contra las etiquetas
- retropropagar para obtener gradientes
- aplicar el optimizador a los pesos

Limpiar	. . .
Avanzar	. . .
Perdida	. . .
Atras	. . .
Actualizar	. . .

PyTorch <https://pytorch.org/> Documentación del framework de tensors y autodiferenciación.

5. Une cada pieza del framework con su función.

5.	array multidimensional con dispositivo asociado	-----
5.	registro de operaciones que permite derivar	-----
5.	borra los gradientes acumulados del paso previo	-----
5.	archivo con los pesos para reanudar o servir	-----

Opciones: tensor · grafo · cero grad · checkpoint

Declarar, dejar que derive

Frameworks de deep learning

Tensor

datos y pesos

en CPU o GPU

Modelo

capas compuestas

parámetros registrados

Bucle

limpiar

avanzar

retroceder

actualizar

Guardar

checkpoint de pesos

reanudar o servir

El mapa mental interactivo es interactivo en la versión online.



TensorFlow <https://www.tensorflow.org/?hl=es> El otro gran framework de aprendizaje profundo automático.

6. Tu entrenamiento va lento aunque la GPU apenas se usa. Mirando el bucle y el dataloader, ¿qué dos cuellos de botella sospechas y cómo los comprobarías?

Referencia curricular

Grado:	7
Materia:	Deep Learning · Inteligencia Artificial

Versión interactiva completa (con actividades, videos y autoevaluación):



<https://univ.lumbre.lat/object/cdia-dep-004-frameworks-pytorch-y-tensorflow>

Material educativo generado desde Lumbre · lumbre.lat