Escuela Técnica Superior de Ingeniería Informática

Máster Universitario en Ingeniería Informática

# Tema 4.3
# Aceleración del Entrenamiento con GPUs

Deep Learning

Máster Oficial en Ingeniería Informática

Universidad de Sevilla

# Contenido

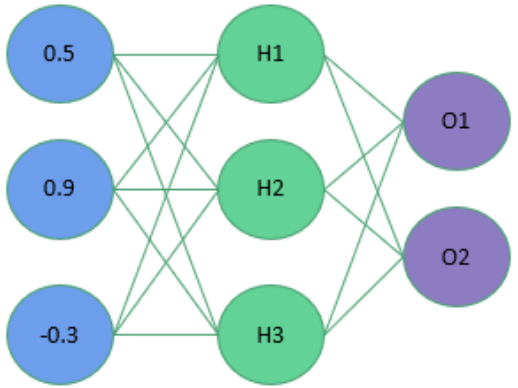
- Necesidad de cómputo paralelo
- Hardware paralelo:
  - CPUs multinúcleo
  - GPUs
  - TPUs
- Plataformas Cloud

# Necesidad de cómputo paralelo

- Hacer **inferencia** (propagación hacia adelante) “no es demasiado” costoso.
- El **entrenamiento** mediante backpropagation es muy **costoso.**
- Una **red pequeña** puede ejecutarse sin problema en una **CPU**.
- Una **red profunda** puede usarse para **inferir** en una **CPU**.
- Sin embargo, cuando necesitamos **entrenar** una **red profunda** en un tiempo razonable (de meses a días), o hacer **inferencia** en **tiempo real** (cámaras de tráfico), necesitamos hardware **paralelo**.

# Necesidad de cómputo paralelo

- Las redes neuronales se implementan como matrices
- **Tensor:** Generalización de matriz (0, 1, 2, 3, … n dimensiones)
- Capas **MLP** →




H1 Weights = (1.0, -2.0, 2.0)
H2 Weights = (2.0, 1.0, -4.0)
H3 Weights = (1.0, -1.0, 0.0)

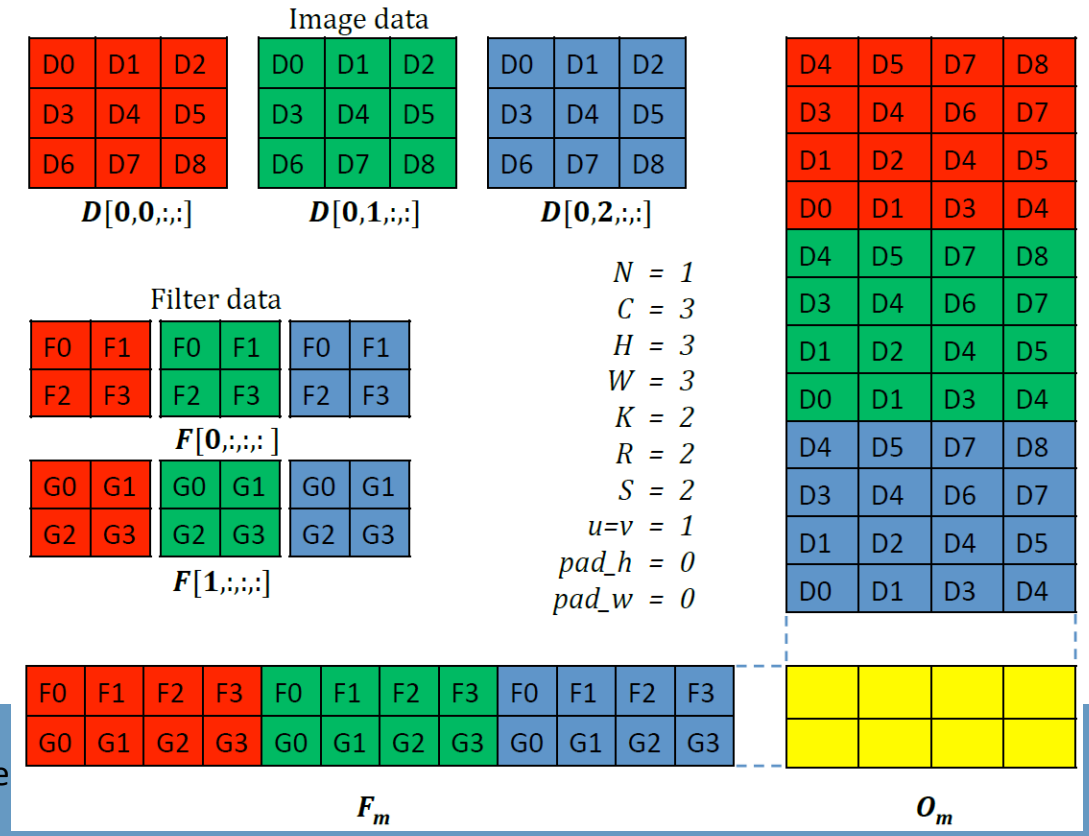
Hidden Layer Weights | Inputs

$$S\left(\begin{bmatrix}1.0 & -2.0 & 2.0\\ 2.0 & 1.0 & -4.0\\ 1.0 & -1.0 & 0.0\end{bmatrix} * \begin{bmatrix}0.5\\ 0.9\\ -0.3\end{bmatrix}\right) = S\left(\begin{bmatrix}-1.9 & 3.1 & -0.4\end{bmatrix}\right) = \begin{bmatrix}0.13 & 0.96 & 0.4\end{bmatrix}$$

Hidden Layer Outputs

# Necesidad de cómputo paralelo

- Las redes neuronales se implementan como matrices
- **Tensor:** Generalización de matriz (0, 1, 2, 3, … n dimensiones)
- Capas **MLP**
- Capas **convolucionales: im2col** →

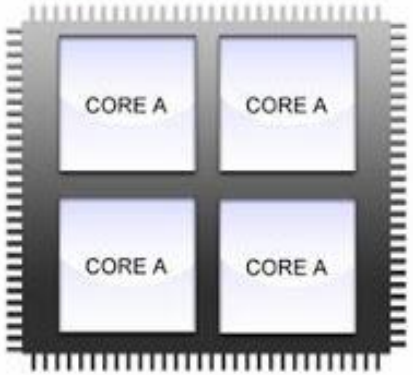


# Necesidad de cómputo paralelo

- Las redes neuronales se implementan como matrices
- **Tensor:** Generalización de matriz (0, 1, 2, 3, … n dimensiones)
- Capas **MLP**
- Capas **convolucionales**
- Capas **Self-Attention** (Transformers, LLMs):
  - $Softmax\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) \times V$

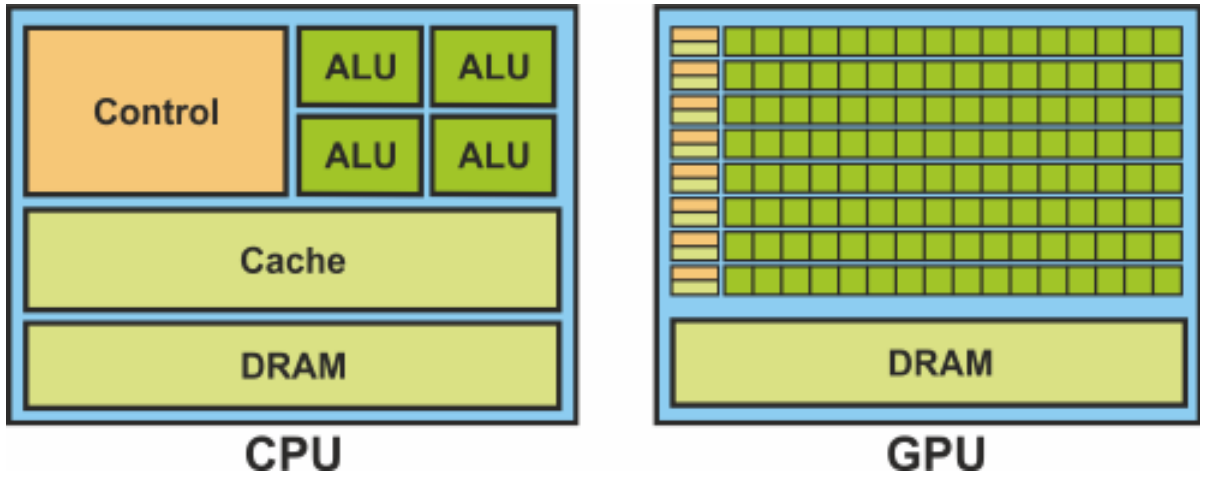
# Hardware paralelo: CPUs multinúcleo

- Las CPUs actuales son multiprocesadores con varios núcleos (del orden de 4 a varias decenas).
  - La librería por excelencia para programar procesadores multinúcleo (multicore) es OpenMP.
- Las librerías actuales utilizan:
  - **MKLDNN** para Intel
  - **AOCL** para AMD
  - **CoreML** para Apple



# Hardware paralelo: GPUs

- **GPU** = Graphics Processing Unit (núcleo tarjeta gráfica).
- Con el tiempo este procesador ha evolucionado y hoy en día se puede usar para cómputo paralelo.
- Una GPU actual incluye del orden de cientos a miles de núcleos.
- Los núcleos son más básicos que los de una CPU, pero son muchos más!



# Hardware paralelo: GPUs

- Tecnología clave para Deep Learning.

**Big Data Availability**

facebook 350 millions images uploaded per day

Walmart 2.5 Petabytes of customer data hourly

YouTube 300 hours of video uploaded every minute

**New ML Techniques**

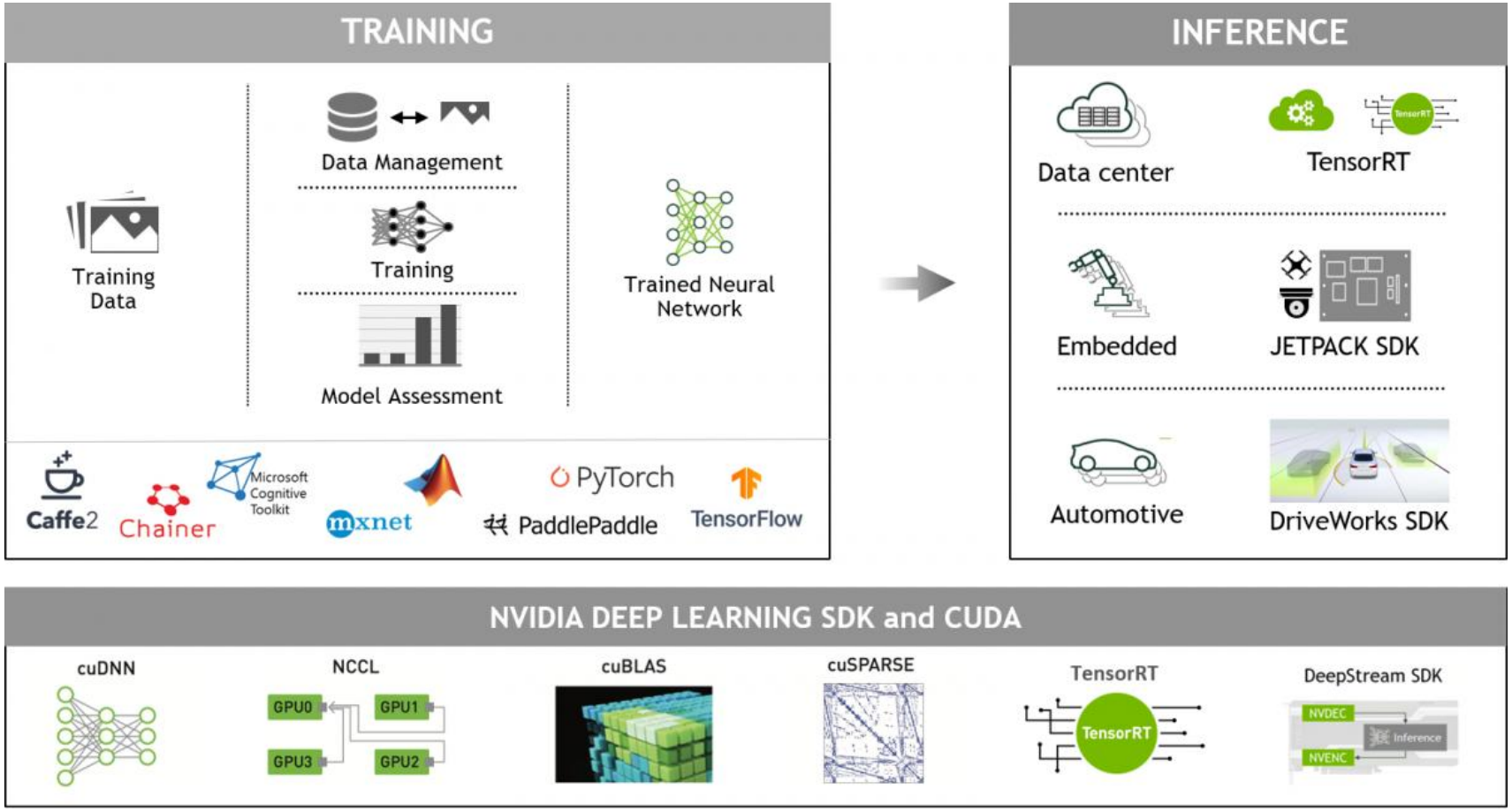

**GPU Acceleration**

# Hardware paralelo: GPUs


- Librerías para programar GPUs y DL:
  - **CUDA** → NVIDIA
  - **ROCm** → AMD
  - **MPS** → Apple
  - OpenCL → NVIDIA, AMD, Intel…
- **NVIDIA** invirtió en Deep Learning desde el principio, y ahora ofrece un **mayor soporte**:
  - Todos los frameworks soportan de forma nativa GPUs mediante CUDA.
  - Sus GPUs dan el mejor rendimiento para entrenamiento a gran escala.

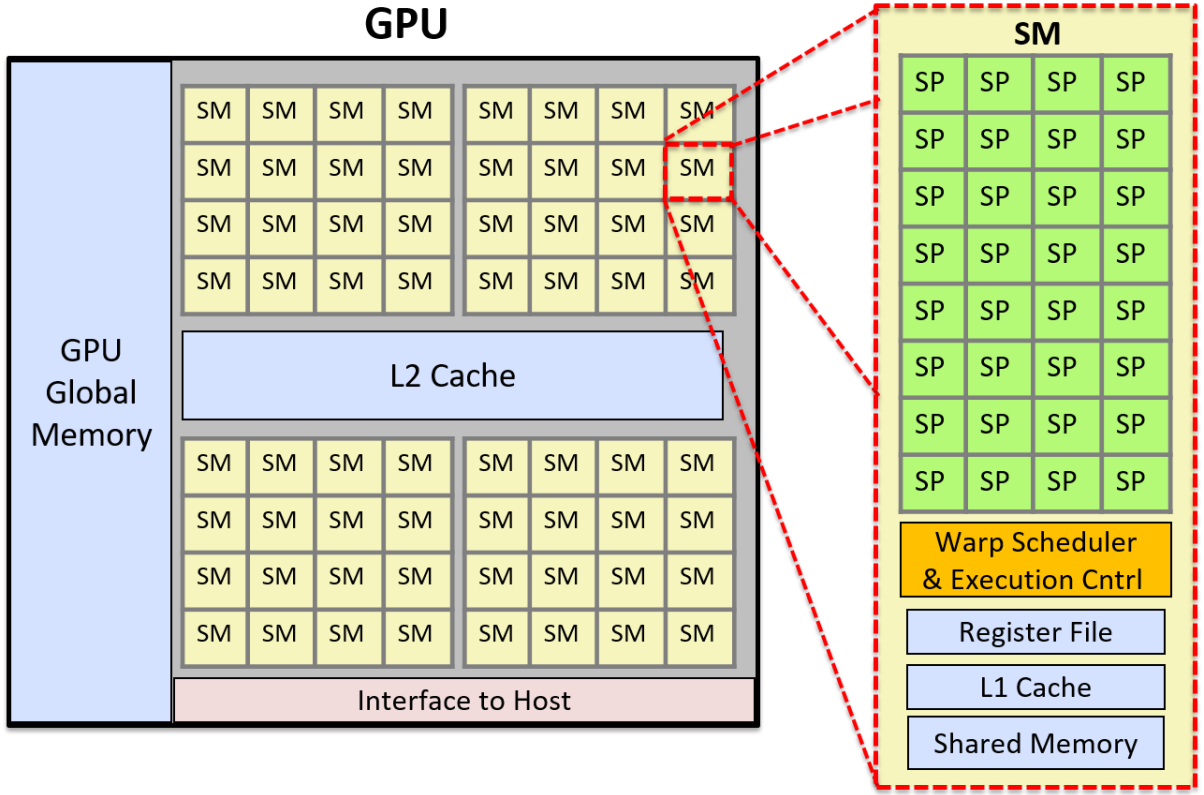
# Hardware paralelo: GPUs

- Ecosistema de NVIDIA



# Hardware paralelo: GPUs

- **Procesador**:
  - Multiprocesadores (SM)
  - Procesadores (SP) / CUDA cores / Shaders
  - Ejecutan la misma instrucción
- **Memoria del dispositivo:**
  - Global Memory ← **Copiar aquí los datos previamente!**
  - Temporal/Rápida:
    - Shared Memory
    - Cachés: L1 y L2
    - Registros

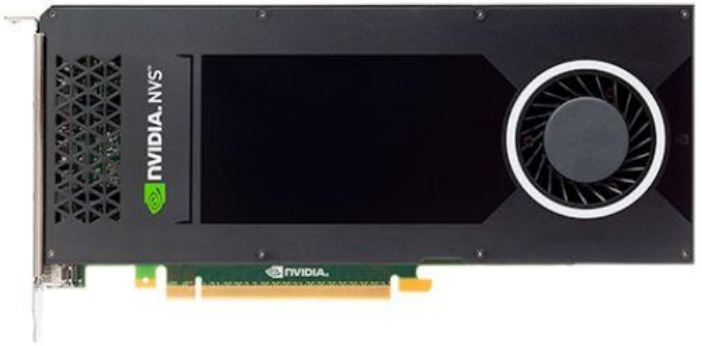


# Hardware paralelo: GPUs

- ¿Qué hay que tener en cuenta al comprar una?
  1. **La cantidad de memoria** en la GPU: cuanto más GigaBytes, mayores modelos se podrán entrenar, y se podrán enviar batches de ejemplos más grandes.
  2. Una **memoria** de la tarjeta **rápida**: el acceso a los datos por los núcleos debe ser eficiente (GDDR7 vs HBM3).
  3. El **número** de **núcleos**: cuantos más, mayor paralelización.
  4. La **tasa de transferencia** de datos a la tarjeta: los ejemplos fluirán continuamente de la CPU a la GPU, y por tanto necesitamos un bus rápido (PCI-e 4.0  vs NVLink).
- Instalación: **CUDA** Toolkit, CUDA **driver** y **CuDNN**

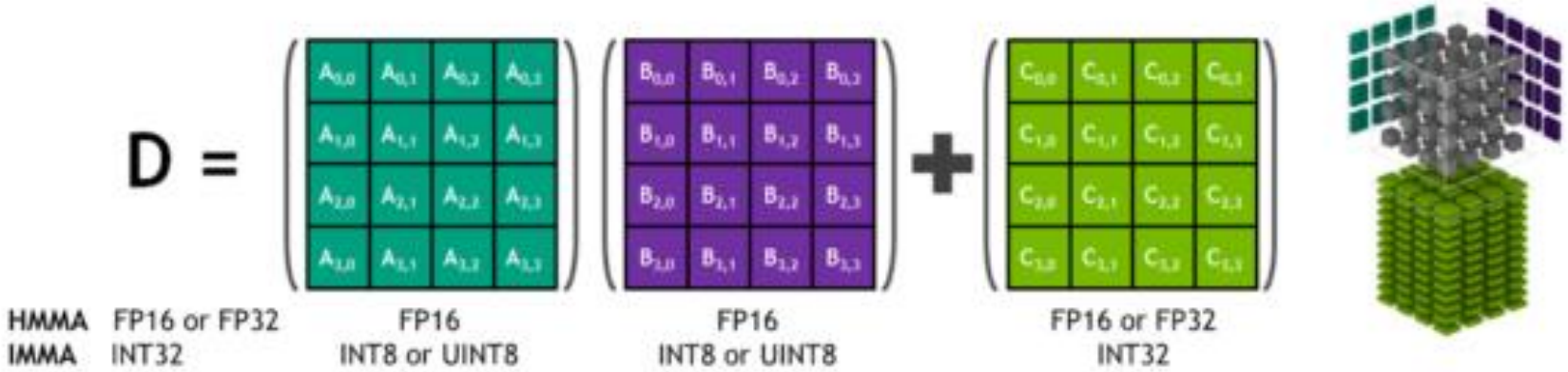
# Hardware paralelo: GPUs

- **NVIDIA** nombra cada **generación** de arquitecturas con un científico:
  - ~~Fermi, Kepler~~, Maxwell, Pascal, Volta, Turing, Ampere, Hopper, Blackwell, *Vera, Feynman*
- Las distintas **categorías** de tarjetas:
  - *Tesla* (cálculo científico, muy caro): A100, H100, B200…
  - *GeForce* (videojuegos, asequible): RTX4090, RTX5080…
  - *Quadro* (gráficos profesionales, caro): RTX4000, …
  - *Jetson* (robótica): Thor, Orin…
- Es posible tener entornos **MultiGPU** (con varias GPUs).



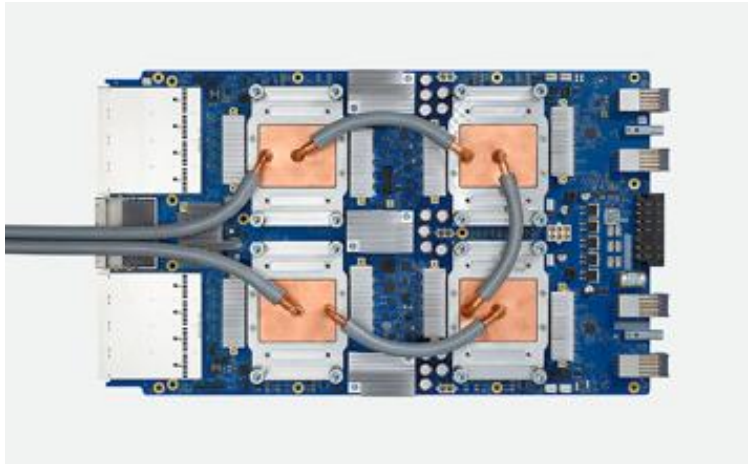
# Hardware paralelo: GPUs

- En **2017**, en la arquitectura **Volta**, NVIDIA introdujo unos núcleos nuevos dentro de sus tarjetas (a parte de los dedicados para CUDA) para solo cálculo con tensores.
- **Tensor Cores**: procesadores para cálculo con tensores.
- **Baja precisión**: float16 → float8 → float6 → float4 (blackwell).
- Operación: Matrix Multiplication and Addition.



# Hardware paralelo: TPUs

- **TPU** = Tensor Processing Unit
- **Chip** introducido por Google en 2016 para Deep Learning.
- **Especializados** para acelerar cálculo tensorial (algebra lineal).
- Uso desde **JAX, TensorFlow** y **PyTorch**
- Incluye:
  - De Gigas a Tera Bytes de memoria.
  - De cientos a miles de núcleos.
- Solo disponible en Cloud



# Plataformas cloud

- Diversas empresas ofrecen servicios donde entrenar nuestros modelos, **evitándonos** tener que tener **hardware** para ello.
- Tecnologías basadas en **contenedores** (Dockers) o **máquinas virtuales**.
- Los tres principales proveedores:
  - Google Cloud
  - Amazon AWS EC2
  - Microsoft Azure

# Plataformas cloud

- **Google Colaboratory**:
  - Basado en Jupyter notebooks, requiere cuenta en Google Drive.
  - Gratuito:
    - Ofrece GPUs (T4) y TPUs (de forma limitada).
    - El trabajo se pierde cuando se cierra sesión: guardar el modelo y datos en Google Drive.
  - De pago: Google Colab Pro. Compra de créditos para acceso a mejores GPUs y acceso menos limitado.
- Más opciones:
  - Paperspace DigitalOcean
  - Lambda.ai
  - Vast.ai
  - IBM supercharge

# Recapitulación

- Para poder entrenar modelos grandes sobre conjuntos de datos grandes necesitaremos de **potencia computacional.**
- Las **GPUs** proveen una solución hardware eficiente respecto a lo que cuestan. Las TPUs están en la nube de Google. Apple ofrece chip bajo consumo.
- Si no se dispone capacidad de compra de GPUs, mejor usar un servicio **en la nube** (p.ej. Google Colab).