

# Entrenamiento de redes neuronales profundas

Diego Andrade Canosa



UNIVERSIDADE DA CORUÑA



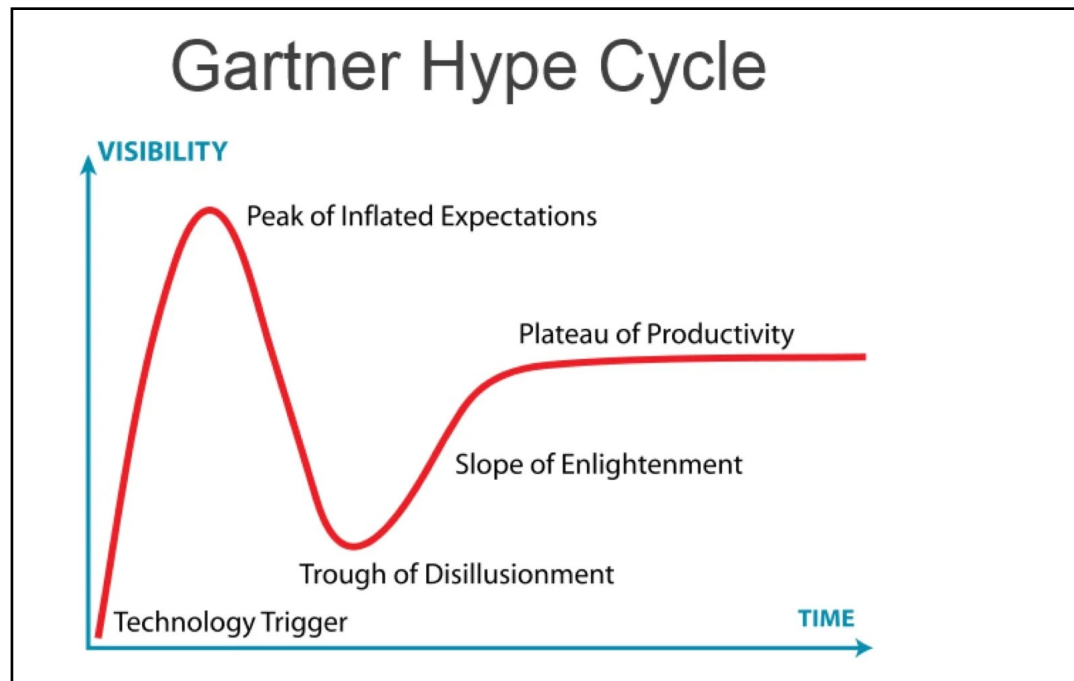
# 1. Introducción -

Diego Andrade Canosa

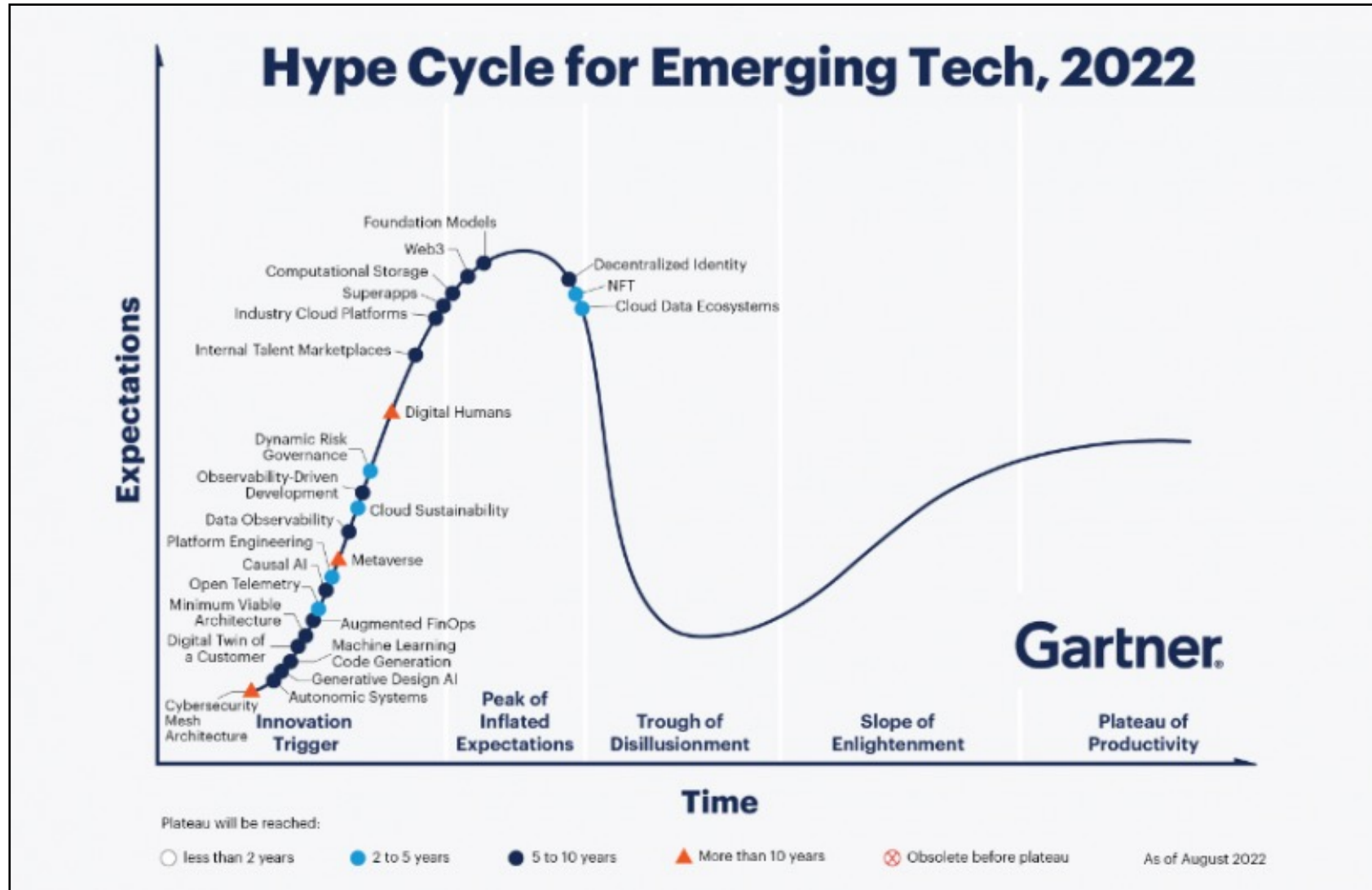
# Introducción al curso: Motivación

[Why AI is the new electricity](#) por Andrew Ng

- **Motivo 1:** La Inteligencia Artificial (IA) juega y jugará un papel fundamental



# Introducción al curso: Motivación

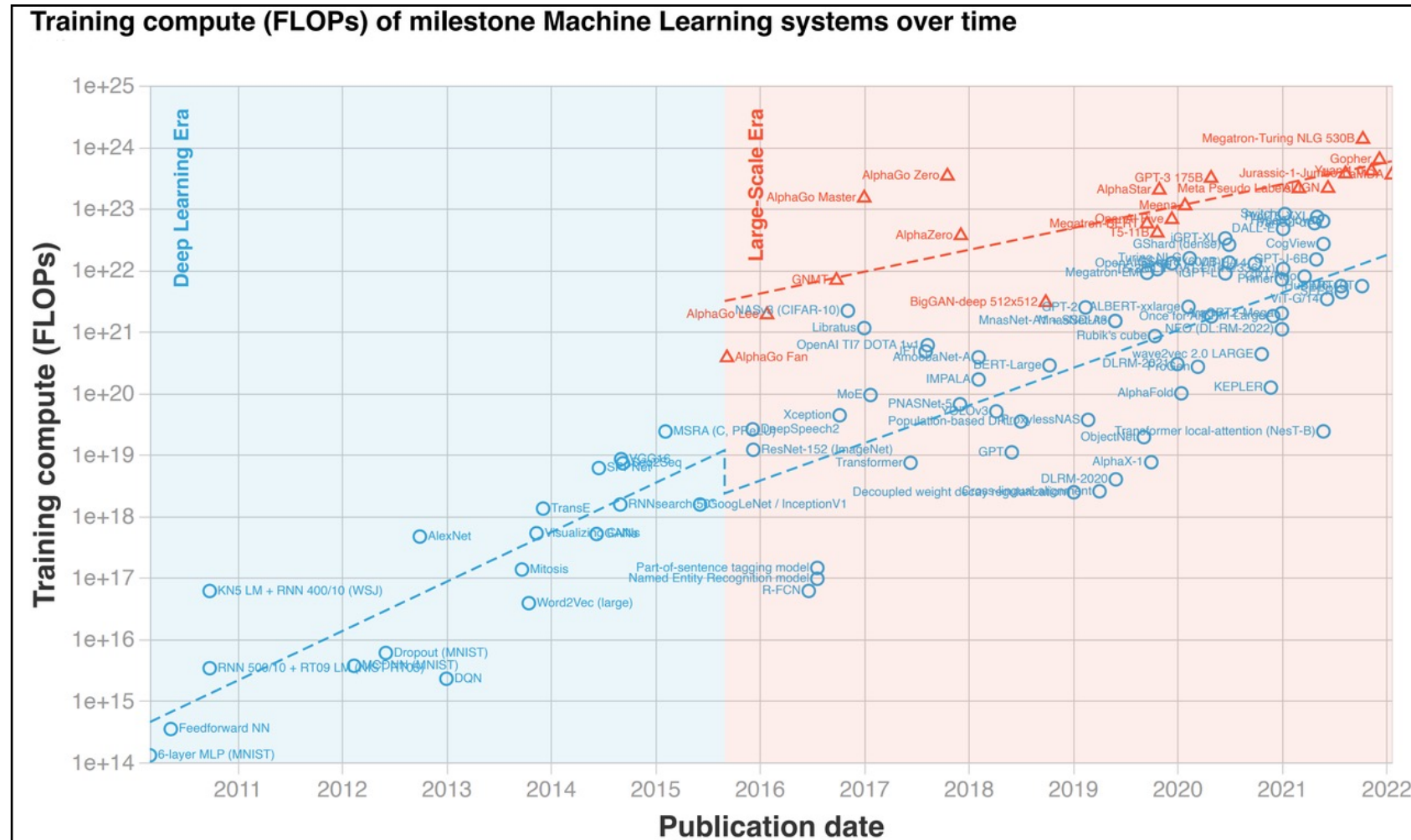


Ciclo del Hype de Gartner

# Introducción al curso: Motivación

- **Motivo 2:** El coste de recursos (computo y memoria) de entrenar nuevos modelos está creciendo aceleradamente

# Introducción al curso: Motivación



Fuente: <https://www.marktechpost.com/2022/07/13/colossal-ai-a-unified-deep-learning-system-for-big-models-seamlessly-accelerates-large-models-at-low-costs-with-hugging-face/>

# Introducción al curso: Motivación

- **Motivo 3:** El entrenamiento de modelos requiere el uso de recursos computacionales heterogéneos

- Tarjetas gráficas: Graphic Processing Units (GPUs)
  - Conceptos asociados: GPGPU, CUDA
  - Tienen hw especializado para IA: Tensor Cores, Sparse Tensor Cores
    - Explotado por las librerías especializadas de Nvidia ([cudNN](#), cuBLAS, cuSparseLT, cutlass), usadas a su vez por los frameworks más conocidos de IA (TF, Pytorch)
- CPUs multinúcleo: Multicores
  - Más lentas para IA que las GPU
  - Tienen hw especializado para IA
    - Principalmente extensiones del juego de instrucciones vectoriales o multimedia del procesador
      - Explotado por librerías especializadas de los fabricantes ([Intel OneAPI AI Analytics](#))
- Hardware de propósito específico
  - Ejemplo: Tensor Processing Units (TPUs) de Google
- Computadores (o aceleradores) cuánticos (*en desarrollo*)

# Introducción al curso: Contenidos

1. Introducción al curso y al uso del Supercomputador FT3: Recursos disponibles y modos de uso
2. Breve introducción a Pytorch
3. Soporte nativo para Pytorch Distribuido
4. Herramientas avanzadas para Pytorch Distribuido
5. Breve introducción a Keras y Tensorflow
6. Soporte nativo para TF Distribuido



tldr;

*iUsa Pytorch Lightning!*

# Introducción al curso: Metodología

- El curso está dividido en 6 sesiones de 4 horas
  - Intercalaremos presentaciones del profesor con actividades dirigidas más o menos autónomas (y descansos)
- Usaremos los recursos del FT3 para la mayoría de las prácticas
- Los principales entornos de IA que usaremos serán Tensorflow (TF) y Pytorch (Torch)
- Otras herramientas
  - Conda
  - Python
    - Jupyter (y Jupyter-Lab)
    - Diversos módulos de Python o de TF y Pytorch
  - Otros entornos
    - Intel OneAPI
  - Herramientas
    - Ssh (openSSH)
    - VSCode
    - Navegador web

# Índice

- Conceptos básicos de RNP
- Arquitectura y recursos en el supercomputador Finisterrae3
  - Guía básica de uso
- Actividades
  - Configuración del entorno del curso
  - Gestión de paquetes de Python
  - Uso básicos de SLURM

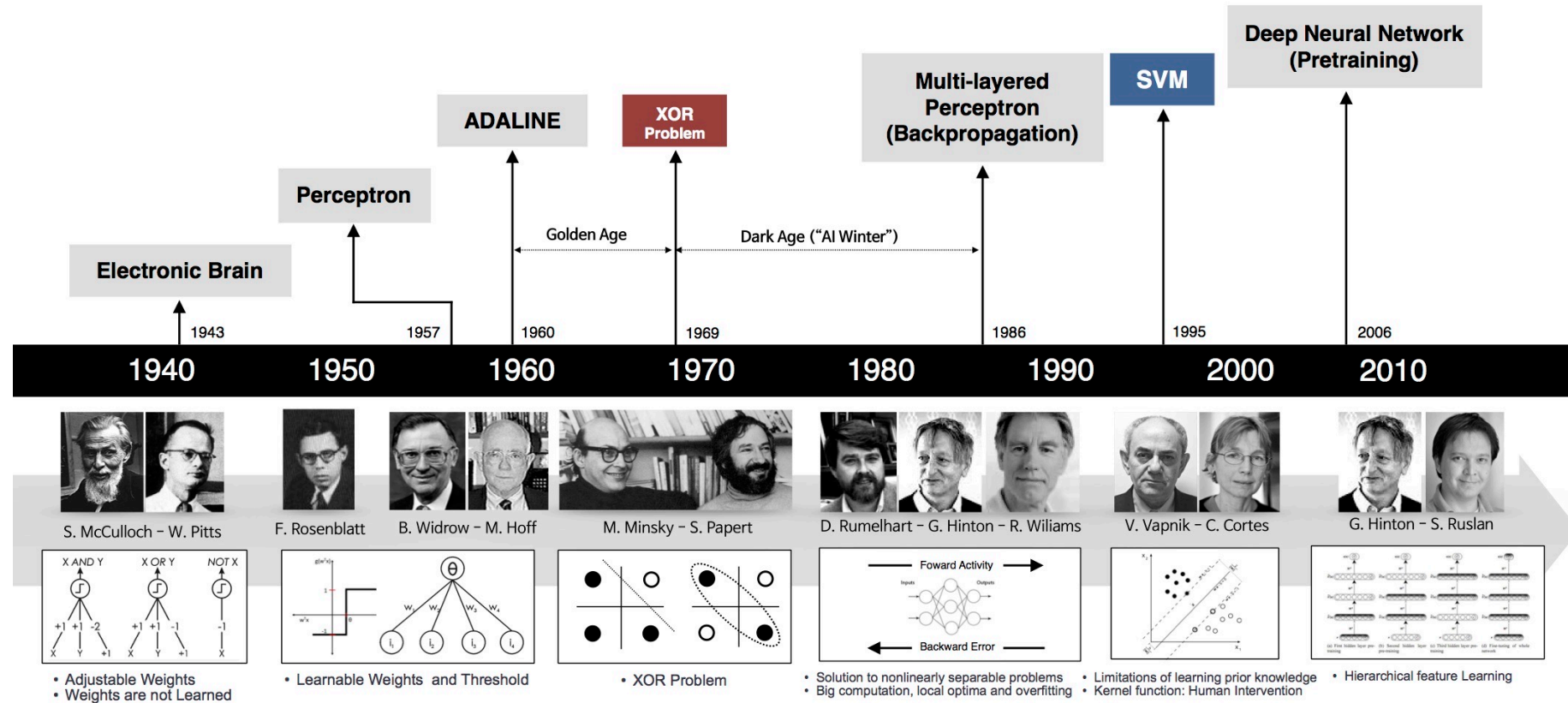
# Introducción

- *Artificial Intelligence* (AI) es cualquier técnica que permita un computador emular el comportamiento humano
- *Machine Learning* (ML) es una rama de la Inteligencia Artificial (IA) que permite aprender a realizar tareas que requieran inteligencia sin una programación explícita
- *Deep Learning* (DL) es un subconjunto de ML que usa redes profundas (con muchas capas ocultas)
  - Permite procesar datos no estructurados
  - Extrae patrones de los datos
  - Basado en *representation learning*
- Tipos de aprendizaje:
  - Aprendizaje supervisado
  - No supervisado
  - Auto (self)-supervisado

# Razones del auge del DL

- El Perceptrón Multicapa (1969) es el concepto fundacional de las RNP
- El auge actual tiene varios hitos clave
  - Año 1990: Se acuña el concepto de DL -> [Artículo: Deep Learning Miraculous Year](#)
  - Año 2009: Artículo de *Imagenet* -> [ImageNet: A large-scale hierarchical image database](#)
  - Año 2017: Artículo *Transformers* -> [Attention is all you need](#)
  - Otros conceptos recientes clave: *Generative Adversarial Networks, Autoencoders, NLP, Diffusion models*
- Razones del auge:
  - Recopilación masiva de datos digitalmente en diversos ámbitos
  - Avances técnicos que permiten procesar datos no estructurados (imágenes, vídeo, audio, texto, voz)
  - HPC Hardware: GPUs, Supercomputers, Clusters, Cloud
  - Software accesible: Google Colab, Kaggle, Tensorflow, Pytorch

# Cronología general del ML



Fuente: [http://beamandrew.github.io//images/deep\\_learning\\_101/nn\\_timeline.jpg](http://beamandrew.github.io//images/deep_learning_101/nn_timeline.jpg)

# Modelos en auge

- [Stable Difussion](#)
- GPT-3 (GPT-4 a punto de publicarse)
  - [ChatGPT](#)
- [Galactica](#)
- [Whisper](#)
- Visita el programa de NeuroIPS  
(<https://nips.cc/virtual/2022/events/highlighted>)
- DramaTron (hace unos pocos días)
- ...

# Compañías y sitios web clave

- Todas las compañías *Big Tech*
  - [Meta AI](#)
  - [Google AI](#)
  - [Baidu Research](#)
- Especializadas en AI
  - [DeepMind](#)
  - [Stability AI](#)
- Hubs de recursos
  - [Kaggle](#)
  - [Huggingface](#)
  - [Google Colab](#)



# Potenciadores de la innovación

- Accesibilidad
  - Modelos *open-source*
  - Herramientas *open-source*
  - Acceso gratuito a recursos computacionales básicos
- Documentación
  - Recursos de formación abiertos
  - Artículos de investigación *open-Access*
- Recursos computacionales masivos

***¡La IA ha encontrado su transistor!***

# El proceso de *ML*

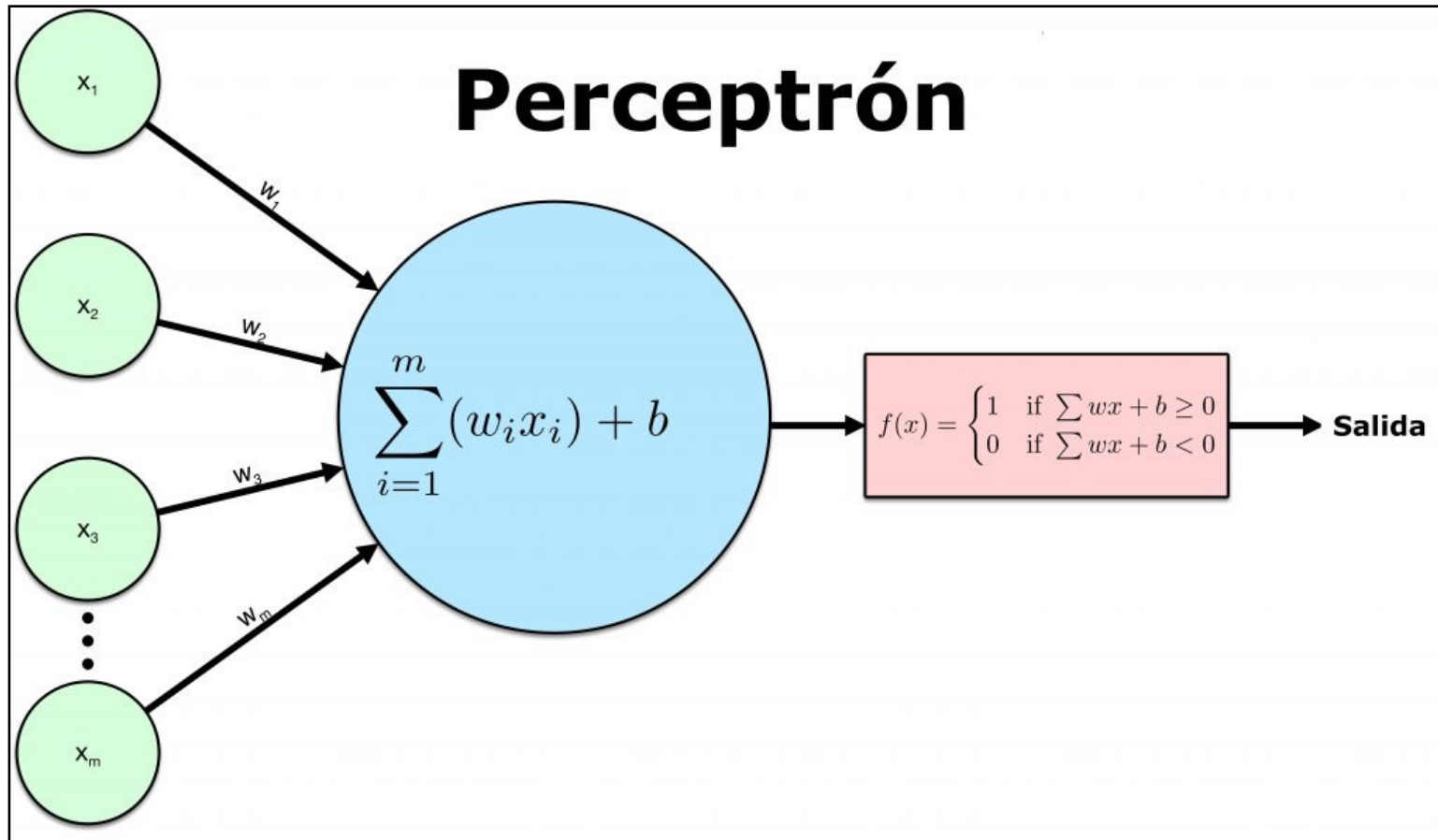
- Debemos entender el problema que queremos resolver
  - Qué salida proporcionará el modelo
  - En base a qué entradas
- Lo anterior nos permitirá definir...
  - Los datos que consideramos relevantes para el modelo
  - La representación más apropiada de estos datos
  - A partir de la anterior definimos el conjunto de datos (en adelante *dataset*), que a su vez dividiremos en:
    - Entrenamiento
    - Validación

# El proceso de *representation learning*

- El aprendizaje de la representación (en adelante *representation learning*) se puede aprender:
  - Las características (e.a. *features*) relevantes de las entradas
  - Cómo relacionar esas características con cada una de las salidas posibles

# El perceptrón

*Forward propagation*

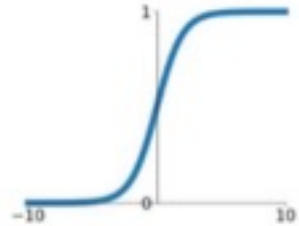


# Funciones de activación

## Activation Functions

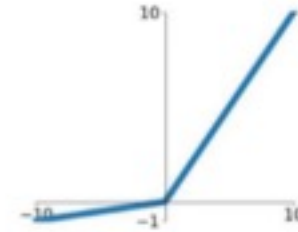
### Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



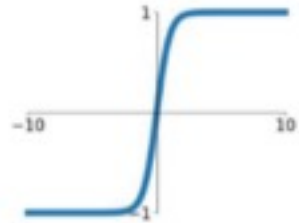
### Leaky ReLU

$$\max(0.1x, x)$$



### tanh

$$\tanh(x)$$

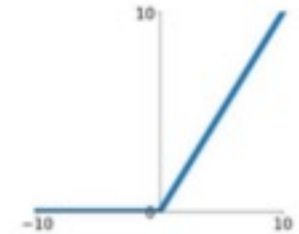


### Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

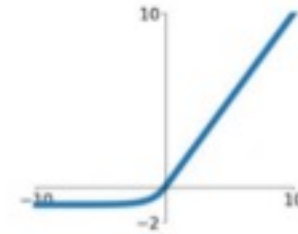
### ReLU

$$\max(0, x)$$

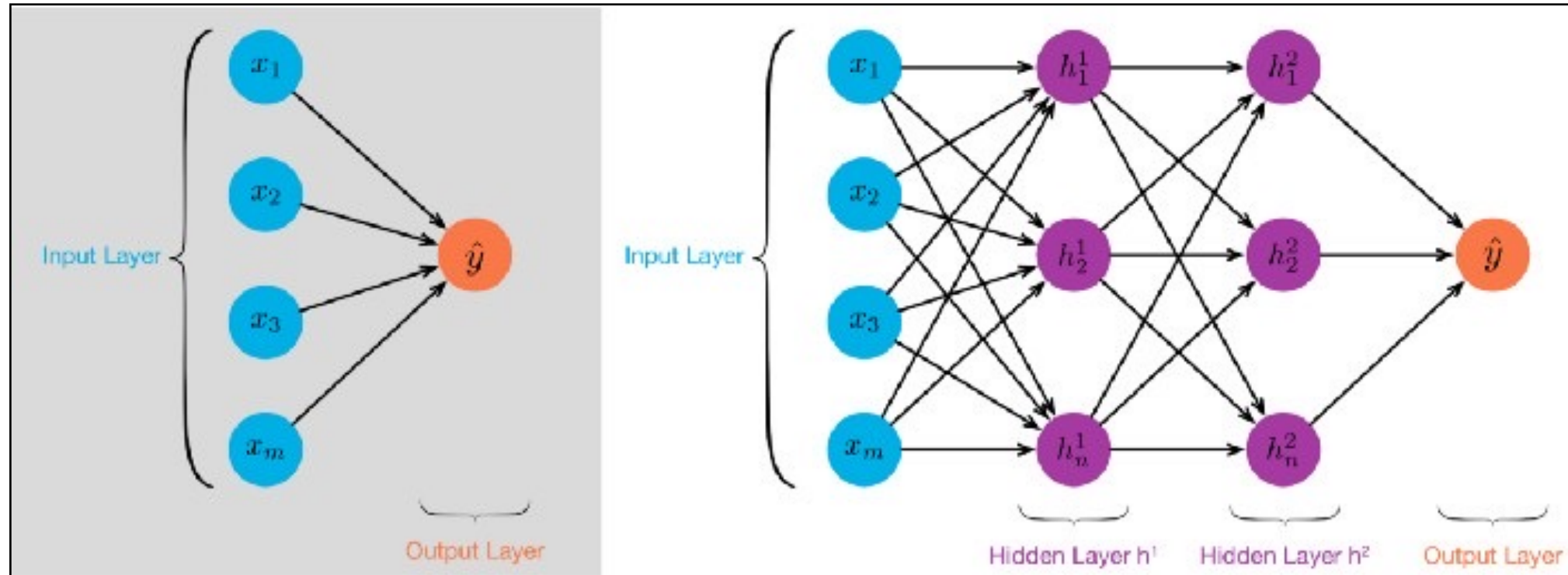


### ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



# Perceptrón multicapa



Fuente: <https://towardsdatascience.com/multi-layer-neural-networks-with-sigmoid-function-deep-learning-for-rookies-2-bf464f09eb7f>

# Componentes básicos de un modelo de DL

- Tres bloques de capas identificables:
  - Capa de entrada
  - Capas ocultas (intermedias). En inglés, *hidden layers*
  - Capa de salida
- Cada capa está compuesta de neuronas, interconectadas entre si, cuya señal se caracteriza por:
  - Peso
  - *Bias*
  - Función de activación

# Conceptos clave: Repaso

- Inferencia, Entrenamiento
- *Dataset*: Conjunto de entrenamiento, conjunto de validación
- Función de activación
- Función de pérdida
  - Optimización de la función de pérdida
    - Algoritmos de *Gradient Descent*
      - SGD, Adam, Adadelta, Adagrad, RMSProp



# El proceso de entrenamiento

- Durante el entrenamiento
  - El modelo se entrena con una parte del *dataset: training dataset*
  - Su precisión se calcula con otra parte del *dataset: validation dataset*
- Durante el entrenamiento, el modelo se expone a todo el *dataset* de entrenamiento de forma repetida
  - Cada exposición completa se conoce como *epoch*
- El *dataset* está compuesto de varios (muchos) *samples* formados por una entrada y una salida deseada (aprendizaje supervisado). Para cada sample:
  - Ejecutamos la inferencia (*forward pass*)
    - *Activation*
  - Calculamos cómo de lejos estamos de la salida deseada: *loss function*
  - Usamos esa información para actualizar los pesos de delante hacia atrás (de la salida a la entrada): *back propagation*
    - *Learning rate*

# El proceso de entrenamiento

- Por eficiencia, el procesamiento de los samples del *dataset* se realiza en lotes (*batches*).
- En cada paso (*step*) del entrenamiento se procesan *batch samples* del conjunto de entrenamiento

$$\text{num\_step} = \text{num\_samples} / \text{tamaño\_batch}$$

- Aspectos básicos configurables del entrenamiento:
  - Función de activación
  - *Loss function*
  - *Nº epochs*
  - *Learning rate*
  - *Tamaño\_batch*

# Otros conceptos a revisar relacionados con el entrenamiento

- *Minibatches*
  - Asociado al concepto de *Stochastic Gradient Descent* (SGD)
  - Los gradientes de los pesos se calculan procesando a la vez *minibatches* de elemento del conjunto de datos
  - Permite acelerar el entrenamiento
    - Mediante paralelización
- *Learning rate*
  - Son adaptativos a la pendiente observada
- *Overfitting*
- Regularización
  - Mejora la capacidad de generalización de nuestro modelo
  - Mejora el rendimiento de nuestro modelo en datos no vistos

# Batches y minibatches

- *batch\_size=total\_samples* -> El conjunto de datos se procesa por completo antes de actualizar los pesos
  - Batch Gradient Descent
- *batch\_size=n* -> El conjunto de entrenamiento se procesa en batches de *n samples*. Después de procesar cada batch se actualizan los gradientes
  - Minibatch Gradient Descent
- *batch\_size=1* -> Los gradientes se actualizan después de procesar cada *sample*
  - *Stochastic Gradient Descent*

# Epoch y step

- Cada procesamiento del conjunto de datos completo se conoce como *epoch*
- El procesamiento de los *batch\_size* samples requeridos para actualizar los gradientes, produce que una *epoch* se divida en  $total\_samples / batch\_size$

# Métodos de regularización

- *Dropout*: Técnica que consiste en poner algunos pesos a 0 aleatoriamente
  - Alrededor del 50%
- *Early Stopping*: Parar un entrenamiento antes de que llegue a overfitting

# Clasificación de modelos: Por tipo de aprendizaje

- Aprendizaje supervisado: El sistema se entrena con un conjunto de datos que contiene pares formados por una entrada y su correspondiente salida correcta (*ground-truth*)
- Semi-supervisado: Una parte del conjunto de datos de entrenamiento está etiquetado y otra no
  - La parte no etiquetada se etiqueta usando el modelo ya entrenado con la parte etiquetada
- Aprendizaje no-supervisado: El conjunto de entrenamiento está compuesto de datos no etiquetados
  - Concepto de clustering
- Aprendizaje auto-supervisado: El conjunto de entrenamiento usa datos no etiquetados. Debe usar técnicas creativas para identificar la ground-truth. Ejemplos:
  - Denoising: Se usan imágenes completas, se corrompe una parte de la imagen, y se usa el conocimiento de la imagen original (sin corromper) para entrenar el modelo
  - Autoregresión: Se quiere predecir el siguiente valor de una secuencia, se pueden usar distintas “ventanas” de esa secuencia para predecir el siguiente valor
  - Aprendizaje de embedding: En modelos de lenguaje la representación numérica de los tokens se suele aprender y promueve que los tokens con significados similares tengan valores numéricos parecidos

# Por el tipo de tarea que aprende el modelo

- Modelos de regresión: Aprenden a predecir un valor
- Modelos de clasificación: Aprenden a clasificar la entrada en una de entre varias categorías
  - Binaria: Si las categorías son 2
  - Multiclase: Si las categorías son más que 2
- Modelos de clustering: Aprenden a agrupar los datos de entrada en varias categorías, o clústeres
- Aprendizaje por refuerzo: Aprenden a tomar decisiones que se evalúan en base a una función que calcula la recompensa de la decisión adoptada



# Por tipo de arquitectura de red

- Perceptrón Multicapa
- *Convolutional Neural Networks* (CNN)
- *Recurrent Neural Networks* (RNN)
  - *Long Short-Term Memory* (LSTM)
- *Generative Adversarial Networks* (GAN)
- *Transformers: (encoder + decoder)*

# Clasificaciones de modelos de DL

- Por tipo de aplicación:
  - Modelado de secuencia. Predicciones (RNNs)
    - Procesamiento de lenguaje natural (LSTMs)
      - Traductores
      - Asistentes/conversadores virtuales
      - Reconocimiento de voz
  - Visión por computador (CNNs)
    - Seguimiento de objetos
    - Identificación de imágenes
    - Conducción autónoma (LIDAR)
  - Modelos generativos (Autoencoders)
    - Imágenes. *Prompt models*
    - Vídeos
  - ...

# Modelos escalables vs no escalables

- Hay arquitecturas de red que son difícilmente paralelizables
  - Ejemplo RNNs
    - Al ser recurrentes se elimina la posibilidad de paralelizarlas
      - No son escalables
    - Las redes de atención (*attention networks*) están concebidas para implementar RNNs de forma escalable
      - Permiten procesar diferentes partes de la entrada, en paralelo, con distintas cabezas (*heads*)
      - Cada *head* utiliza una máscara diferente

# Modelos que requieren gran capacidad de cómputo

- Los modelos de Deep Learning (DL) actuales requieren cada vez más recursos:
  - Memoria
  - Cómputo
    - Entrenamiento
      - Inicial + Fine-tuning
    - Inferencia

# Ejemplos de modelos grandes: Bert

	BERT	RoBERTa	DistilBERT	XLNet
Size (millions)	Base: 110 Large: 340	Base: 110 Large: 340	Base: 66	Base: ~110 Large: ~340
Training Time	Base: 8 x V100 x 12 days* Large: 64 TPU Chips x 4 days (or 280 x V100 x 1 days*)	Large: 1024 x V100 x 1 day; 4-5 times more than BERT.	Base: 8 x V100 x 3.5 days; 4 times less than BERT.	Large: 512 TPU Chips x 2.5 days; 5 times more than BERT.
Performance	Outperforms state-of-the-art in Oct 2018	2-20% improvement over BERT	3% degradation from BERT	2-15% improvement over BERT
Data	16 GB BERT data (Books Corpus + Wikipedia). 3.3 Billion words.	160 GB (16 GB BERT data + 144 GB additional)	16 GB BERT data. 3.3 Billion words.	Base: 16 GB BERT data Large: 113 GB (16 GB BERT data + 97 GB additional). 33 Billion words.
Method	BERT (Bidirectional Transformer with MLM and NSP)	BERT without NSP**	BERT Distillation	Bidirectional Transformer with Permutation based modeling

# Ejemplos de de modelos grandes: GPT-3

- Llevaría 355 años entrenar este modelo en una sola Tesla V100
  - 4.6 millones de dólares de coste en un proveedor de cloud [[fuente](#)]

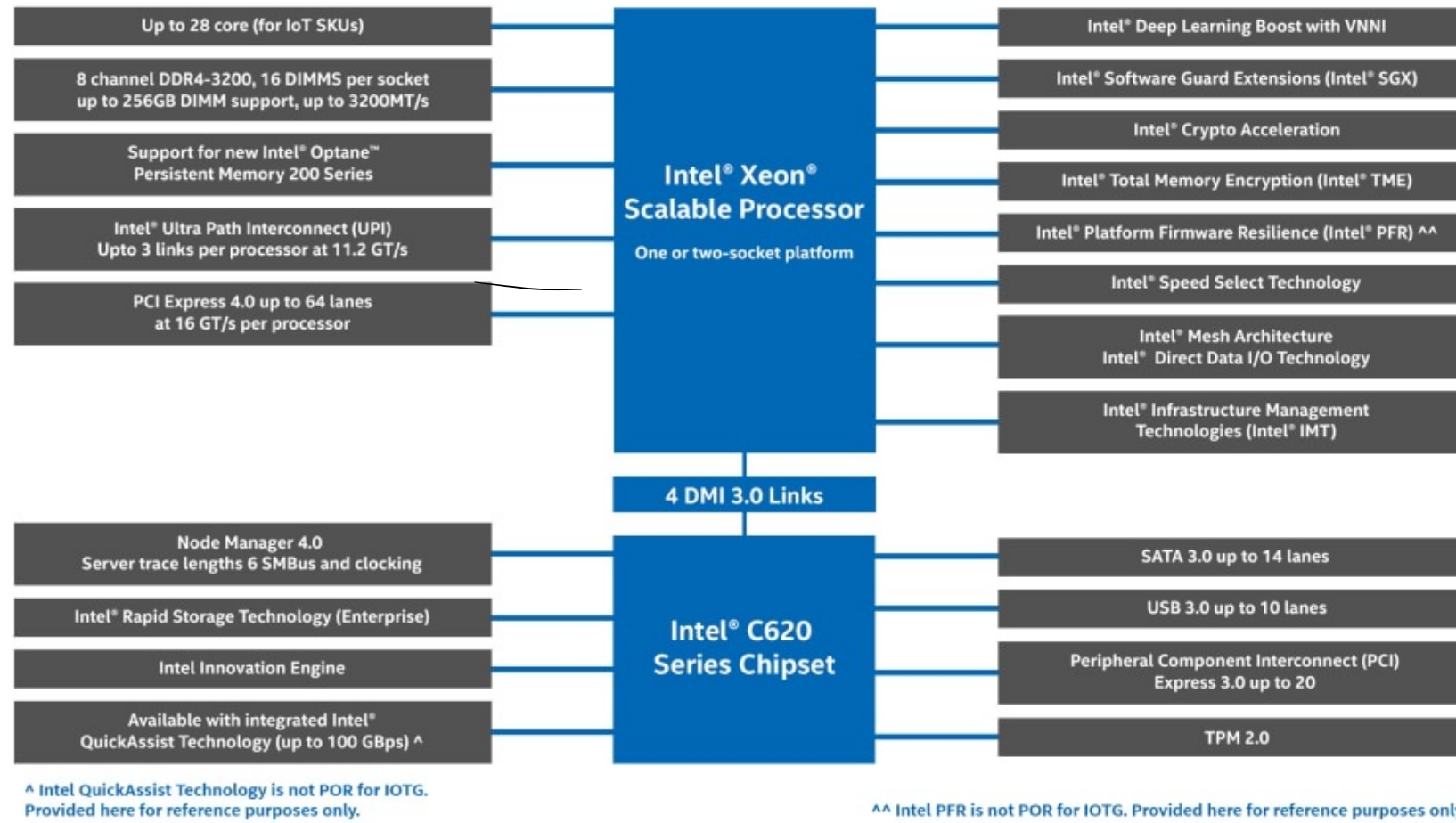
# Descripción avanzada de Finisterrae III (FT3)

	Thin nodes	GPU nodes	Vis nodes	Fat nodes	Transfer nodes	QLM node
Cantidad	256	64	16	16	2	Simulador de circuitos cuánticos Atos 30 qbits
Proc.	2x Intel Xeon Ice Lake 8352Y 32 cores 2.2 GHz					
Cores	64					
Memoria	256 GB			2 TB	256 GB	
Disco local	960 GB SSD NVMe			1920 GB SSD NVMe	960 GB SSD NVMe	
GPU	2x Nvidia A100		Nvidia T4			

Fuente: [CESGA - Portal de usuarios](#)

# Descripción avanzada de Finisterrae III (FT3)

Cada nodo tiene 2 procesadores Intel Xeon 8352Y de 32 cores cada uno -> 64 cores por nodo

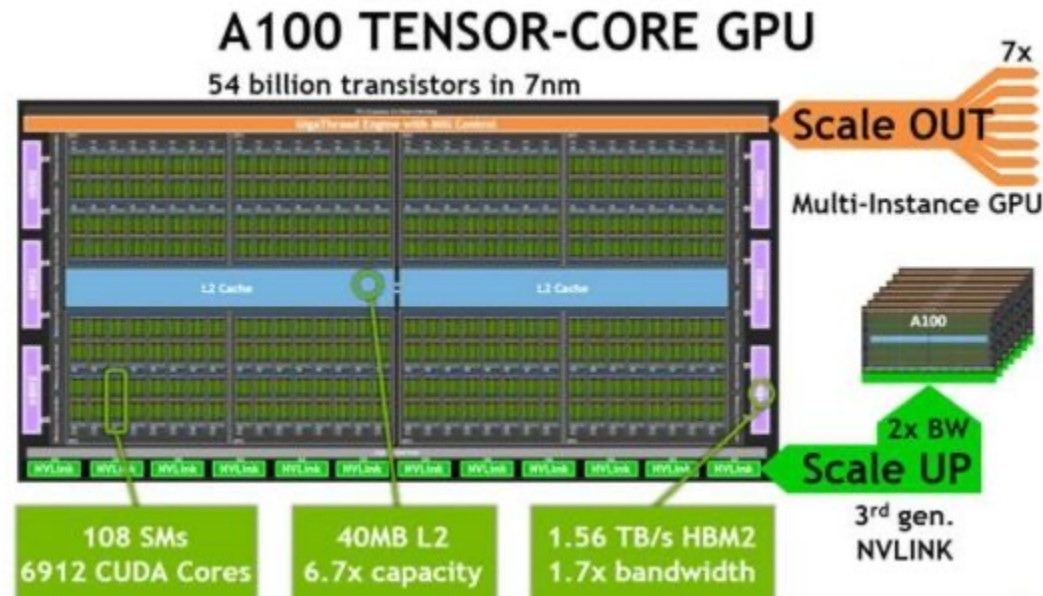


Fuente: [Procesador Intel® Xeon® Platinum 8352Y](#) y [Xeon Platinum - Intel - WikiChip](#)



# Descripción avanzada de Finisterrae III (FT3)

- Varios modelos de GPU:
  - Tesla T4. Arquitectura Turing.
  - Tesla A100. Arquitectura A100.



# Descripción avanzada de Finisterrae III (FT3)

- Tesla A100. Arquitectura A100.



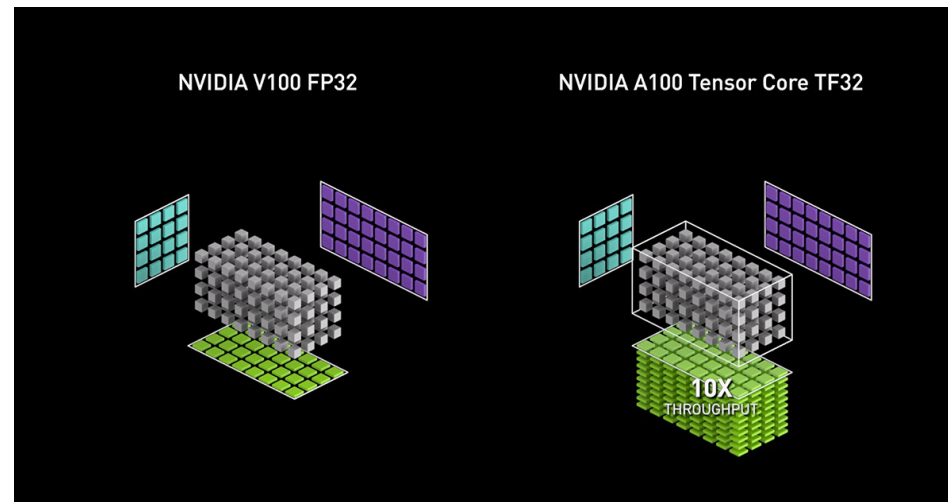
Fuente: [NVIDIA A100](#) | [NVIDIA](#)

# Descripción avanzada de Finisterrae III (FT3)

Nvidia Tensor Cores

TENSOR CORE 4X4X4 MATRIX-MULTIPLY ACC

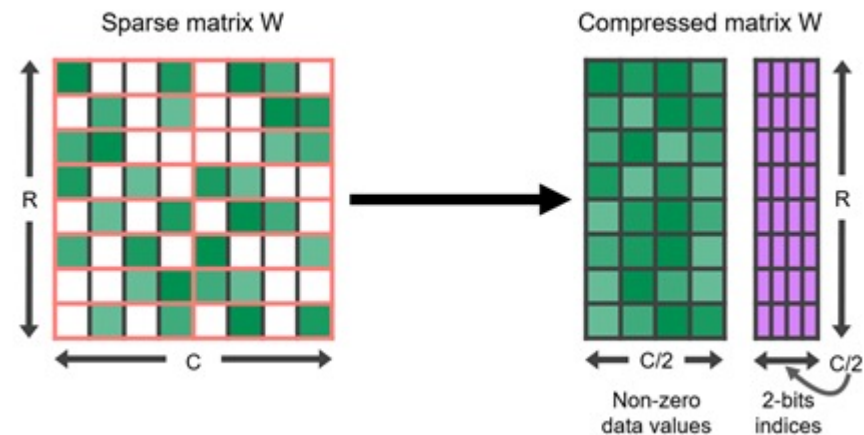
$$D = \begin{matrix} \text{FP16 or FP32} & \begin{pmatrix} A_{0,0} & A_{0,1} & A_{0,2} & A_{0,3} \\ A_{1,0} & A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,0} & A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,0} & A_{3,1} & A_{3,2} & A_{3,3} \end{pmatrix} & \text{FP16} & \begin{pmatrix} B_{0,0} & B_{0,1} & B_{0,2} & B_{0,3} \\ B_{1,0} & B_{1,1} & B_{1,2} & B_{1,3} \\ B_{2,0} & B_{2,1} & B_{2,2} & B_{2,3} \\ B_{3,0} & B_{3,1} & B_{3,2} & B_{3,3} \end{pmatrix} & + & \begin{pmatrix} C_{0,0} & C_{0,1} & C_{0,2} & C_{0,3} \\ C_{1,0} & C_{1,1} & C_{1,2} & C_{1,3} \\ C_{2,0} & C_{2,1} & C_{2,2} & C_{2,3} \\ C_{3,0} & C_{3,1} & C_{3,2} & C_{3,3} \end{pmatrix} & \text{FP16 or FP32} \end{matrix}$$



Fuente: [Tensor Cores: versatilidad para HPC e IA | NVIDIA](#)

# Descripción avanzada de Finisterrae III (FT3)

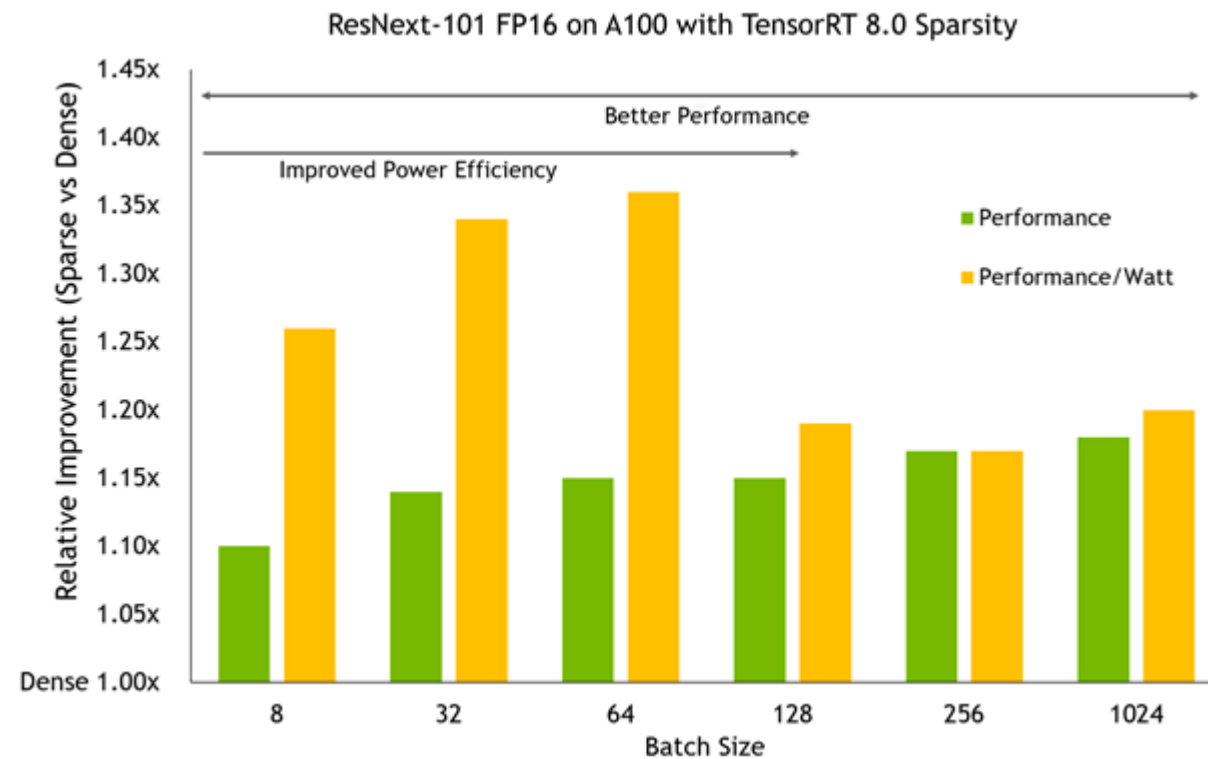
## Nvidia **Sparse** Tensor Cores



Fuente: [Working with sparse tensors | TensorFlow Core](#) y [Accelerating Inference with Sparsity Using the NVIDIA Ampere Architecture and NVIDIA TensorRT | NVIDIA Technical Blog](#)

# Descripción avanzada de Finisterrae III (FT3)

## Nvidia **Sparse** Tensor Cores



Fuente: [Working with sparse tensors | TensorFlow Core](#) y [Accelerating Inference with Sparsity Using the NVIDIA Ampere Architecture and NVIDIA TensorRT | NVIDIA Technical Blog](#)

# Descripción avanzada de Finisterrae III (FT3)

- Almacenamiento compartido: Home y Store
- Accesible desde todas las infraestructuras de CESGA
- Localización cableada en las variables de entorno \$HOME y \$STORE
- Límites:
  - HOME: 10 GB y 100.000 ficheros
  - STORE: 500 GB y 300.000 ficheros
- Sólo se hace backup de Home

Directory	Use	User limits	Backup
\$HOME	Store code files Low speed access	10GB 100.000 files	Yes
\$STORE	Store simulations final results Low speed access	500GB 300.000 files	No
\$LUSTRE	Simulation runs High speed access	3TB 200.000 files	No

```
[ulcesdac@login211-1 dac]$ cd $LUSTRE
[ulcesdac@login211-1 dac]$ pwd
/mnt/lustre/scratch/nlsas/home/ulc/es/dac
[ulcesdac@login211-1 dac]$ cd $HOME
[ulcesdac@login211-1 ~]$ pwd
/home/ulc/es/dac
[ulcesdac@login211-1 ~]$ cd $STORE
[ulcesdac@login211-1 dac]$ pwd
/mnt/netapp2/Store_uni/home/ulc/es/dac
[ulcesdac@login211-1 dac]$ |
```

Fuente: [CESGA - Portal de usuarios](#)

# Descripción avanzada de Finisterrae III (FT3)

- Sistema paralelo de almacenamiento compartido basado en Lustre
- Dividido en 2 pools: NVMe y NLSAS
- Accesible a través de
  - \$LUSTRE: Permanente. Pool NLSAS
  - \$LUSTRE\_SCRATCH: Disponible durante la ejecución de los trabajos. Pool NVMe.
- Límites: Hasta 3 TB por usuario y 200.000 ficheros

```
[ulcesdac@login211-1 dac]$ cd $LUSTRE
[ulcesdac@login211-1 dac]$ pwd
/mnt/lustre/scratch/nlsas/home/ulc/es/dac
[ulcesdac@login211-1 dac]$ cd $HOME
[ulcesdac@login211-1 ~]$ pwd
/home/ulc/es/dac
[ulcesdac@login211-1 ~]$ cd $STORE
[ulcesdac@login211-1 dac]$ pwd
/mnt/netapp2/Store_uni/home/ulc/es/dac
[ulcesdac@login211-1 dac]$ |
```

Fuente: [CESGA - Portal de usuarios](#)

# Descripción avanzada de Finisterrae III (FT3)

- Usamos el comando myquota para conocer el espacio disponible

```
[ulcesdac@login211-1 ~]$ myquota

- HOME and Store filesystems:
-----
Filesystem      space  quota  limit  files  quota  limit
10.117.49.201:/Home_FT2  7966M  10005M  10240M  48959  100k   101k
10.117.49.201:/Store_uni 96361M  499G   500G   280k   300k   301k
10.117.49.101:/Home_BD   3008M  800G   1024G   60     4295m  4295m

- LUSTRE filesystem:
-----
Filesystem      used  quota  limit  grace  files  quota  limit  grace
/mnt/lustre/scratch
                12.6G   3T    3.5T    -      86665  200000  240000  -
```

Fuente: [CESGA - Portal de usuarios](#)



# Actividad 1.1: Configuración y conexión básicas

- Software instalado en nuestro equipo:
  - Obligatorio:
    - Cliente ssh o terminal
      - Linux/Mac (ya disponible de forma nativa)
      - Windows (<https://learn.microsoft.com/en-us/windows/terminal/tutorials/ssh>). Existen otras opciones como:
        - WSL: <https://ubuntu.com/desktop/wsl>
        - OpenSSH: <https://www.openssh.com>
    - Cliente VPN (Endpoint Security VPN) ([https://cesga-docs.gitlab.io/ft3-user-guide/how\\_to\\_connect.html](https://cesga-docs.gitlab.io/ft3-user-guide/how_to_connect.html) )
  - Recomendable: Configurar acceso a FT3 *passwordless*.  
Guía: <https://www.hostinger.es/tutoriales/configurar-ssh-sin-contrasena-linux>
  - Opcional: VSCode: Proporciona:
    - Ayudas en la escritura de código (numerosas extensiones (*plugins*))
    - Control de versiones fácil de utilizar (Git)
    - Acceso remoto transparente (navegación)
    - Terminal integrado

# Actividad 1.1: Configuración y conexión básicas

- Acciones a realizar:

- Conectarse al FT3 por ssh

Desde un terminal local ssh [miusuario@ft3.cesga.es](mailto:miusuario@ft3.cesga.es)

- Comprobar el almacenamiento que tenemos disponible

En FT3: quota -s

- Solicitar un nodo de cómputo con una GPU y comprobar las características de la GPU asignada en el nodo

En FT3:

compute -gpu

En el nodo de cómputo:

nvidia-smi

# Actividad 1.2: Clonación del repositorio del curso

- Conectarse al FT3
- Clonar el repositorio de código del curso en \$STORE  
Cd \$STORE  
git clone <https://github.com/diegoandradecanosa/CFR24>
- Ejecutar el script de comparación de rendimientos de particiones  
cd \$STORE/CFR24/scripts  
./scriptDisks.sh

Examinar e interpretar las salidas

# Módulos de FT3

- El FT3 es un sistema Linux que usa la distribución Rocky Linux (8.4)
- Lógicamente no tenemos permisos de sudo en el sistema, como usuarios convencionales
- Existen mecanismos para cargar nuevas funcionalidades
  - A través del sistema de módulos
    - Basado en TCL modules
    - Accesibles a través del cliente Lua (lmod) -> COMANDO `module`
  - En Python
    - Instalando paquetes localmente
    - A través de la creación y configuración de entornos conda

# Módulos de FT3

- Sistema de módulos de FT3
  - Los módulos están instalados en el S.O. a través de easy\_build
  - El cliente con el que interactuamos con ellos es un cliente Lua (<https://lmod.readthedocs.io/en/latest/> )
- Comandos principales
  - Consulta
    - `module list` Proporciona una lista de módulos cargados
    - `module avail` Proporciona una lista de módulos disponibles
    - `module keyword término1` Proporciona una lista de módulo que contienen un término de búsqueda
    - `module spider módulo1` Proporciona información sobre un módulo determinado
    - `module show módulo1` Muestra información detallada sobre un módulo

# Módulos de FT3

- Carga de módulos

- `module purge`

Descarga todos los módulos cargados

- `module reset`

Vuelve a carga solo los módulos por defecto

- `module load módulo1`

Carga un módulo

- A veces puede haber dependencias (módulos a cargar previamente)

- Averiguables con el comando spider

- `module unload módulo1`

Descarga un módulo

- `module swap módulo1 módulo2`

Descarga el módulo1 y carga el módulo2

# Módulos de FT3

- Entornos de usuario

- `module save nombre1`

Guarda la lista de módulos cargados actualmente como una colección de usuario de nombre *nombre1*

- `module restore nombre1`

Restaura la colección de usuarios *nombre1*

- `module savelist`

Lista de colecciones disponibles

- `module describe nombre1`

Contenido de una colección

# Módulos de FT3

- Existen más comandos y parámetros (--parameter) modificadores a explorar
- Más información sobre Imod:
  - Comando `module --help`
  - Documentación: [https://lmod.readthedocs.io/en/latest/010\\_user.html](https://lmod.readthedocs.io/en/latest/010_user.html)



# Actividad 1.3: Manipulación de módulos

- Conéctate al FT3 (no te olvides de activar el VPN)
- Realizar las siguientes acciones
  - Obtén una lista de módulos cargados: `module list`
  - Obtén una lista de módulos disponibles: `module avail`
  - Busca módulos de pytorch: `module keyword pytorch`
    - Consulta información detallada sobre uno de esos módulos:  
`module spider pytorch`  
`module spider pytorch/2.0.0-cuda`
    - Carga ese módulo:  
`module load cesga/system pytorch/2.0.0-cuda`
  - Crea una colección propia para ese módulo  
`module save torch2cuda`
  - Comprueba que existe la colección  
`module savelist`
  - Purga todos los módulos  
`module purge`
  - Carga la colección  
`module restore torch2cuda`

# Módulos de FT3

Cargando módulos en bash

Añade los siguiente al fichero .bashrc de tu \$HOME

```
if [ -z "$BASHRC_READ" ]; then
    export BASHRC_READ=1
    # Place any module commands here
    # module load git
fi
```

# Módulos relevantes FT3

- Módulos relevantes:
  - Módulos disponibles consultables con el comando: *module avail*
  - Cargables con el comando: *module load nombre\_del\_modulo*
  - Módulos relevantes para IA en FT3:
    - **intel:** Diversas herramientas de intel, algunas relacionadas con IA
    - **tensorflow:** Framework de google para IA
    - **transformers:** Varias herramientas de IA relacionadas con NLP (procesamiento del lenguaje natural), incluido TensorFlow y Pytorch
    - **pytorch:** Popular framework de IA
    - **torchvision:** Conjuntos de datos, y modelos para vision por computador
    - **scikit-learn:** Librería que implementa diversos algoritmos de *Machine Learning*, implementados con numpy, scipy y matplotlib
    - **r-keras:** API en R para TensorFlow keras

# Entornos Python en FT3

- El FT3 tiene varias versiones de Python y módulos instalador y accesibles a través del sistema de módulos
  - cesga/2020
    - Python 3.7.8 (la última es 3.11)
    - tensorflow/2.4.1-cuda-system
  - tensorflow/2.5.0-cuda-system (la última es 2.15)
  - Usando la versión de Python instalada en el FT3 con el módulo cesga/2020 (3.7.8) podemos
    - Instalar nuevos paquetes localmente con `pip install módulo`
      - Los paquetes se instalan en el directorio `.local` de nuestro `$HOME`
    - Ver la lista de paquetes instalados y sus versiones `pip list`
    - O comprobar un paquete concreto con `pip list | grep nombre`
    - Actualizar a una versión más reciente cualquier paquete (incluido TF)
      - Asegúrate de no tener ningún módulo de tensorflow cargado

```
python: python/2.7.18, python/3.6.12, python/3.9.9, python/3.10.10
Python is a programming language that lets you work more quickly and
integrate your systems more effectively. -- cesga/2020 python/2.7.18:
Compiler: Requires gcccore/system
```

# Entornos Python en FT3: Entorno local

- Buscar módulos: <https://pypi.org/search/>
- Problema: La instalación local se almacena en \$HOME (solo 10GB de capacidad)
  - Cambiarlo con variantes de entorno
    - PYTHONPATH, etc...
- Opción en general engorrosa porque dependiendo del uso que le demos a Python, podemos mezclar paquetes y versiones
  - Mejor tener entornos aislados por tipo de aplicación

# Entornos Python en FT3: Virtual Env

- Alternativa Virtualenv:
  - <https://docs.python.org/3/library/venv.html>
  - <https://pypi.org/project/virtualenv/>
- Ejemplo de creación de entorno

```
python3 -m venv $STORE/mypython
cd $STORE/mypython/bin
source ./activate
```

#A partir de aquí estoy dentro del entorno
- Forma de activación alternativa. Poner en la cabecera de cada script Python el path a la copia local del intérprete del entorno venv

```
#!/<path-to-venv>/bin/python
```
- El entorno se almacena de forma local

# Entornos de Python en FT3: Conda

- Conda es una herramienta para la gestión de paquetes y entornos de software (Linux, MacOS, Windows)
- En nuestro caso, es especialmente útil para generar entornos Conda controlados
- Accesible con módulo miniconda3  
module load miniconda3

- Para evitar problemas de cuota

```
conda config --add envs_dirs $LUSTRE/conda/envs
```

```
conda config --add pkgs_dirs $LUSTRE/conda/pkgs
```

O...

```
export CONDA_ENVS_PATH=$LUSTRE/conda/envs
```

```
export CONDA_PKGS_DIRS=$LUSTRE/conda/pkgs
```

# Entorno de Python en FT3: Conda

- Creación de entornos:

- De forma iterativa

- ```
conda create -n nombrentorno python=3.9
```

- ```
conda activate nombrentorno
```

- ```
# Instalar paquetes con conda install y pip
```

- A partir de un fichero yml

- ```
conda env create -n nombrentorno --file fichero.yml
```

- Exportar entorno actual a fichero yml

- ```
conda env export --name nombrentorno > fichero.yml
```



# Entorno de Python en FT3: Conda

- Búsqueda de paquetes
  - Los paquetes están disponibles a través de canales
  - El comando `conda search termino1`
    - Busca en los canales predefinidos
  - También podemos buscar en:
    - <https://anaconda.org/anaconda/repo>
  - El comando `conda info`, me da información sobre conda, incluida una lista de canales por defecto

# Entorno de Python en FT3: Conda

- Puede haber paquetes de Python instalables por
  - conda install
  - pip install
- La diferencia es la misma que cuando instalamos el paquete por pip o por apt en una instalación de Ubuntu
- Conda CheatSheet ->  
<https://docs.conda.io/projects/conda/en/4.6.0/downloads/52a95608c49671267e40c689e0bc00ca/conda-cheatsheet.pdf>

# Consideraciones generales sobre entornos Python

- Las librerías Python, y especialmente las de ML, tienen mal soporte entre versiones
  - Cambios continuos en el API
  - En la localización de los módulos
  - Códigos dejan de funcionar entre versiones
- Modelos de DL de código abierto suelen venir con definiciones de entornos predefinidas
  - Incluyen las versiones necesarias de cada paquete
- Los entornos propios de Python suelen ocupar mucho espacio
  - Problemas de cuota
    - Capacidad
    - Número de ficheros
  - `pip cache remove`

## Especificación de número de versión de paquetes python

| Constraint type          | Specification                      | Result                               |
|--------------------------|------------------------------------|--------------------------------------|
| Fuzzy                    | <code>numpy=1.11</code>            | 1.11.0, 1.11.1, 1.11.2, 1.11.18 etc. |
| Exact                    | <code>numpy==1.11</code>           | 1.11.0                               |
| Greater than or equal to | <code>"numpy&gt;=1.11"</code>      | 1.11.0 or higher                     |
| OR                       | <code>"numpy=1.11.1 1.11.3"</code> | 1.11.1, 1.11.3                       |
| AND                      | <code>"numpy&gt;=1.8,&lt;2"</code> | 1.8, 1.9, not 2.0                    |

# Actividad 1.4: Actualización de TF

- Conectarse a FT3 (recuerda activar VPN)
- Instalar una versión reciente de TF en FT3 siguiendo las 3 vías estudiadas:
  - Entorno local
  - Entorno venv
  - ~~Entorno conda~~ (lleva mucho tiempo)
- Borraremos todas las instalaciones locales al final para evitar problemas de quota en el futuro
  - Entorno local (borrar el contenido de la carpeta local \$HOME/.local)
  - Entorno venv (borrar toda la carpeta)
  - ~~Entorno conda~~ `conda remove nombre del entorno`

# Actividad 1.4: Actualización de TF

- Instrucciones detalladas venv

```
module load python cesga/system python/3.10.10
python3 -m venv $STORE/mypython
cd $STORE/mypython/bin
source ./activate
pip install tensorflow[and-cuda]
```

## Comprobamos la instalación

```
compute -gpu
cd $STORE/mypython/bin
source ./activate
python3 -c "import tensorflow as tf; print(tf.config.list_physical_devices('GPU'))"
```

# Actividad 1.4: Actualización de Pytorch

- Instrucciones detalladas venv  
pip install torch[and-cuda]

Comprobamos la instalación

~~compute --gpu~~

~~cd \$STORE/mypython/bin~~

~~source ./activate~~

python3 -c "import torch; print(torch.cuda.device\_count())"

# Sistema de colas de FT3

- El acceso al FT3 se realiza a través de un sistema de colas basado en SLURM
  - <https://slurm.schedmd.com>
  - <http://portalusuarios.cesga.es>
- El portal de usuarios del CESGA nos permite acceder a
  - Alguna información del sistema de colas
    - Secciones trabajos e información
  - También podemos hacer peticiones especiales de recursos (Sección solicitudes)
    - Almacenamiento y recursos
  - Soporte
    - Tickets (jira)
    - Consultas
    - Teléfono

# Sistema de colas de FT3

- El FT3 está estructurado en
  - Nodos de front-end, a los que nos conectamos en primera instancia
    - No aptos para computación. Tampoco para compilación
  - Nodos de computación, a los que enviamos los trabajos computacionales
  - El almacenamiento, que puede ser local o distribuido
- Manejamos varias abstracciones
  - Nodos
    - Estructurados en particiones (conjuntos de nodos)
    - Trabajos (Jobs)
  - Recursos a solicitar
    - CPU: Sockets y cores. En número
    - **Memoria principal/RAM. En capacidad (GB)**
    - GPUs
    - **Tiempo máximo de computación**
    - ...



# Particiones

- Son conjuntos de nodos similares a los que se les aplica las mismas restricciones
  - Restricciones: recursos del trabajo, duración máxima
  - Un nodo puede estar en varias particiones

```
[ulcesdac@login210-19 ~]$ scontrol show partition short
PartitionName=short
  AllowGroups=ALL AllowAccounts=ALL AllowQos=short,clk_short,class_a,class_b,class_c,special
  AllocNodes=ALL Default=YES QoS=N/A
  DefaultTime=NONE DisableRootJobs=YES ExclusiveUser=NO GraceTime=0 Hidden=NO
  MaxNodes=UNLIMITED MaxTime=06:00:00 MinNodes=0 LLN=NO MaxCPUsPerNode=UNLIMITED
  Nodes=ilk-[1-256],a100-[1-66],smp-[1-16],optane,clk-[2-39,42-77,101-120]
  PriorityJobFactor=100 PriorityTier=100 RootOnly=NO ReqResv=NO OverSubscribe=NO
  OverTimeLimit=NONE PreemptMode=OFF
  State=UP TotalCPUs=26208 TotalNodes=433 SelectTypeParameters=NONE
  JobDefaults=(null)
  DefMemPerNode=UNLIMITED MaxMemPerNode=UNLIMITED
```

# Sistema de colas del FT3

- Principales particiones:
  - La mayoría de los (336) nodos está en short, medium, long, ondemand y requeue
    - ilk: 256 nodos con 64 cores y 246GB de RAM dispoble
    - clk: 74 nodos (del anterior FT2) con 48 cores y 180GB de memoria
    - smp: 16 nodos con 64 cores y 2011GB de RAM disponible
    - a100: 64 nodos con 64 cores, 246GB de RAM disponible y 2 GPU A100
  - Partición viz con 10 nodos con 64 cores y 246GB de RAM disponible y 1 GPU T4 destinados a usos interactivos (comando compute)
  - 6 nodos para escritorios remotos

# Sistema de colas del FT3

- Particiones short, medium, long, ondemand se diferencian en:
  - Prioridad: Los trabajos cortos tienen prioridad sobre los más largos
  - Duración máxima de un trabajo:
    - Short: 06:00:00
    - Medium: 3-00:00:00
    - Long: 7-00:00:00
    - Ondemand: 42-00:00:00 (solo disponible por solicitud)
- Si no se piden GPUs los trabajos se asignan por defecto a los nodos ilk

# Sistema de colas del FT3

- Por defecto, los nodos asignados suelen ser compartidos con otros usuarios
- Se pueden pedir nodos en exclusiva con el flag `-exclusive`
- La QoS (quality of service) impone restricciones adicionales a las de la partición
  - Consecuencia práctica: Cuanto más ajustemos nuestra solicitud de hardware, más prioridad tendrán nuestros trabajos
  - Normalmente la qos se asigna automáticamente en base a los recursos solicitados
  - Se puede solicitar una diferente usando `-qos nombreqos`
    - Siempre que la petición se ajuste a los límites del qos solicitado
  - El comando `batchlim` proporciona mucha información práctica

- ✓ **GrpTRES:** Es el número máximo de recursos que pueden usar por todos los trabajos que usen esa QoS.
- ✓ **MaxTRES:** Es el número máximo de recursos que pueden pedir al enviar un trabajo.
- ✓ **MaxWall:** Es el tiempo máximo que se puede pedir a enviar un trabajo.
- ✓ **MaxJobsPU:** Número máximo de trabajos en ejecución por un usuario de forma simultánea.
- ✓ **MaxTRESPU:** Máximo número de recursos utilizados de forma simultánea por un usuario entre todos sus trabajos en ejecución.
- ✓ **MaxSubmit:** Número máximo de trabajos en cola por un usuario de forma simultánea.

| Name     | Priority | GrpTRES  | MaxTRES   | MaxWall    | MaxJobsPU | MaxTRESPU             | MaxSubmit |                     |
|----------|----------|----------|-----------|------------|-----------|-----------------------|-----------|---------------------|
| short    | 50       |          | cpu=2048  |            | 30        | cpu=2048              | 100       |                     |
| medium   | 40       |          | cpu=2048  |            | 30        | cpu=2048              | 100       |                     |
| long     | 30       | cpu=4096 | cpu=2048  |            | 5         | cpu=2048              | 10        |                     |
| requeue  | 20       |          | cpu=2048  |            | 5         | cpu=2048              | 10        |                     |
| viz      | 50       |          |           |            | 2         | cpu=64,<br>gres/gpu=1 | 2         |                     |
| ondemand | 10       | cpu=2048 | cpu=1024  |            | 2         | cpu=1024              | 10        | Recursos especiales |
| special  | 30       |          | cpu=16384 |            | 2         | cpu=16384             | 10        |                     |
| data     | 50       |          |           |            | 2         | cpu=64                | 2         |                     |
| class_a  | 200      |          | cpu=4096  | 3-00:00:00 | 40        | cpu=8192              | 50        | RES                 |
| class_b  | 40       |          | cpu=4096  | 2-00:00:00 | 3         | cpu=4096              | 15        |                     |
| class_c  | 10       |          | cpu=4096  | 1-00:00:00 | 1         | cpu=4096              | 5         |                     |

## Sistema de colas del FT3: QoS

# Sistema de colas del FT3: QoS

```
ulcesdac@login210-19 ~]$ sacctmgr show qos short
```

| Name      | Priority | GraceTime | Preempt | PreemptExemptTime | PreemptMode    | Flags       | UsageThres | UsageFactor | GrpTRES   | GrpTRESMins | Grp     |           |             |
|-----------|----------|-----------|---------|-------------------|----------------|-------------|------------|-------------|-----------|-------------|---------|-----------|-------------|
| RESRunMin | GrpJobs  | GrpSubmit | GrpWall | MaxTRES           | MaxTRESPerNode | MaxTRESMins | MaxWall    | MaxTRESPU   | MaxJobsPU | MaxSubmitPU | MaxTRES | MaxJobsPA | MaxSubmitPA |
| MinTRES   |          |           |         |                   |                |             |            |             |           |             |         |           |             |
| short     | 50       | 00:00:00  |         |                   | cluster        |             |            |             |           |             |         |           |             |
|           |          |           |         | cpu=2048          |                |             |            | DenyOnLimit |           | 1.000000    |         |           |             |
|           |          |           |         |                   |                |             | cpu=2048   | 50          | 100       |             |         |           |             |

# Sistema de colas del FT3: Comandos

- sbatch: Se utiliza para enviar trabajos por lotes. El comando queda encolado hasta que se ejecuta.
- salloc: Es una versión online de sbatch, es decir, el comando se queda esperando hasta que el trabajo se encola, se ejecuta y termina
  - Ejecutado en cualquier contexto produce una reserva de recursos nuevas
- srun: Similar a salloc pero:
  - Cuando se ejecuta dentro de un sbatch hereda los recursos pedidos por sbatch, con lo que se tiene que limitar a esa reserva. No genera una reserva nueva de recursos
  - Si se ejecuta sin que haya una reserva de recursos previa, la genera por él mismo

# Sistema de colas del FT3: Comandos

- `sinfo`: Proporciona información sobre el estado de ocupación de las particiones
- `squeue -u miusuario`: Nos dice cuál es el estado de los trabajos de un usuario
- `scancel jobid`: Cancela un trabajo individual
- `scancel -u miusuario`: Cancela todos los trabajos de un usuario



# Sistema de colas del FT3: Información complementaria

- Slurm cheat sheet: <https://slurm.schedmd.com/pdfs/summary.pdf>
- Taller avanzado del FT3 (parte 1) en portalusuarios.cesga.es (slides 35-final (72))  
[https://portalusuarios.cesga.es/layout/download/taller\\_ft3\\_avanzado\\_sesion1.pdf](https://portalusuarios.cesga.es/layout/download/taller_ft3_avanzado_sesion1.pdf)
- Directorio /opt/cesga/job-scripts-examples

```
[ulcesdac@login210-19 ~]$ ls /opt/cesga/job-scripts-examples
clDeviceQuery.cpp      dot.py                  matmat.F
cluster.json           fftw_test.F            matplotlib
compileCUDA.sh         g16                    matrix_inv
compile_deviceQuery.sh GNUParallel.sh         mem-dot
compileMPIOpenMP.sh    help_fortran_find_core_id.c monitoring
compileMPI.sh          hybrid.c                MPIOpenMP_Job_on_exclusive_nodes.sh
compileOpenMP.sh        hybrid.f90              MPIOpenMP_Job_on_shared_nodes.sh
dask-jobqueue.py       Job_Array.sh            multiprog.config
dask-mpi.py            job_signal_timeout.sh   Multiprog_Job.sh
deviceQuery.cpp         JugandoConRedesNeuronales.tar mysecond.c
NVIDIAK80_Job.sh      old                     omphello.c
omphello.c             omphello.f             openfoam
pi3f90.f90             R                       runGP.sh
run_jupyter_notebook.sh run-notebook-cesga2020.sh
run_snakemake.sh        saga_example_FT2.py    sagemath
Simple_MPI_Job.sh      simple_multiprog.config Simple_Multiprog_Job.sh
Simple_OpenMP_Job.sh   singularity            slurm
snakemake_job.sh       task.sh                TensorFlow
test.ipynb             testlapack.cpp         test.snakefile
torun_under_saga.sh    Trivial_Serial_Job.sh  udocker
wine
```

# Actividad 1.5: Demo Slurm

- Conéctate al FT3 (recuerda activar previamente el VPN si es necesario)
- En el Front-end: ejecuta el comando `sinfo` para ver el estado de las particiones
- Usa el comando `compute --gpu` para pedir un nodo interactivo con una GPU (T4)
  - Dentro del nodo ejecuta el comando `hostname`
  - Ejecutar el comando `nvidia-smi` para ver las características de la GPU asignada
  - Salgo de la sesión interactiva con el comando `exit`

# Actividad 1.5: Demo Slurm

- De nuevo desde el front-end ejecuta el comando
  - `srun -N 1 --time=00:01:00 --gres=gpu:a100 -c 32 --mem=4G nvidia-smi`
- De nuevo desde el front-end ejecutar el comando
  - `srun -N 1 --time=00:01:00 --gres=gpu:a100 -c 32 --mem=4G --pty bash`
  - Dentro del nodo puedes ejecutar el comando `nvidia-smi`
- Ahora debes hacer lo mismo pero con `sbatch`