

# Introducción al Deep Learning y análisis de la primera práctica

Curso de Deep Learning y CUDA

Titulaciones Propias. Universidad de Málaga



Manuel Ujaldón

Catedrático de Universidad

Departamento de Arquitectura de Computadores

Universidad de Málaga



DEEP  
LEARNING  
INSTITUTE

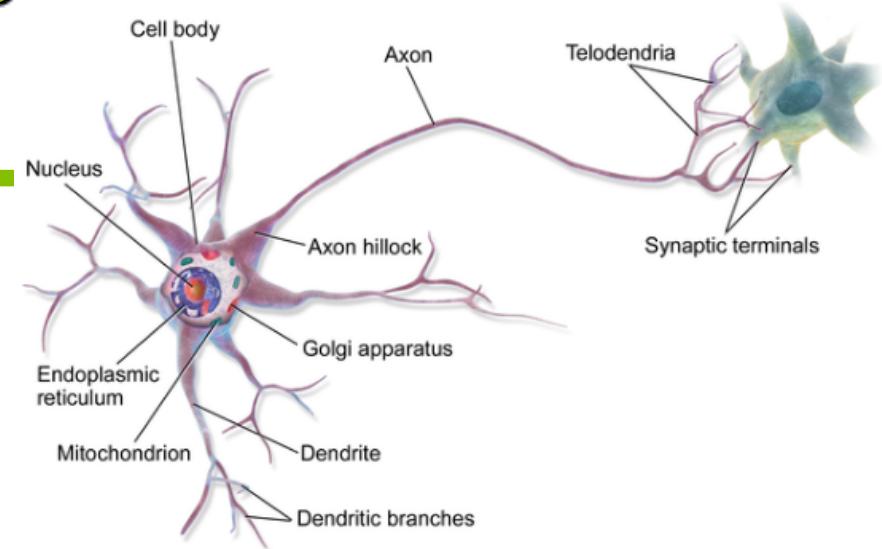
UNIVERSITY  
AMBASSADOR



# I. Conceptos preliminares

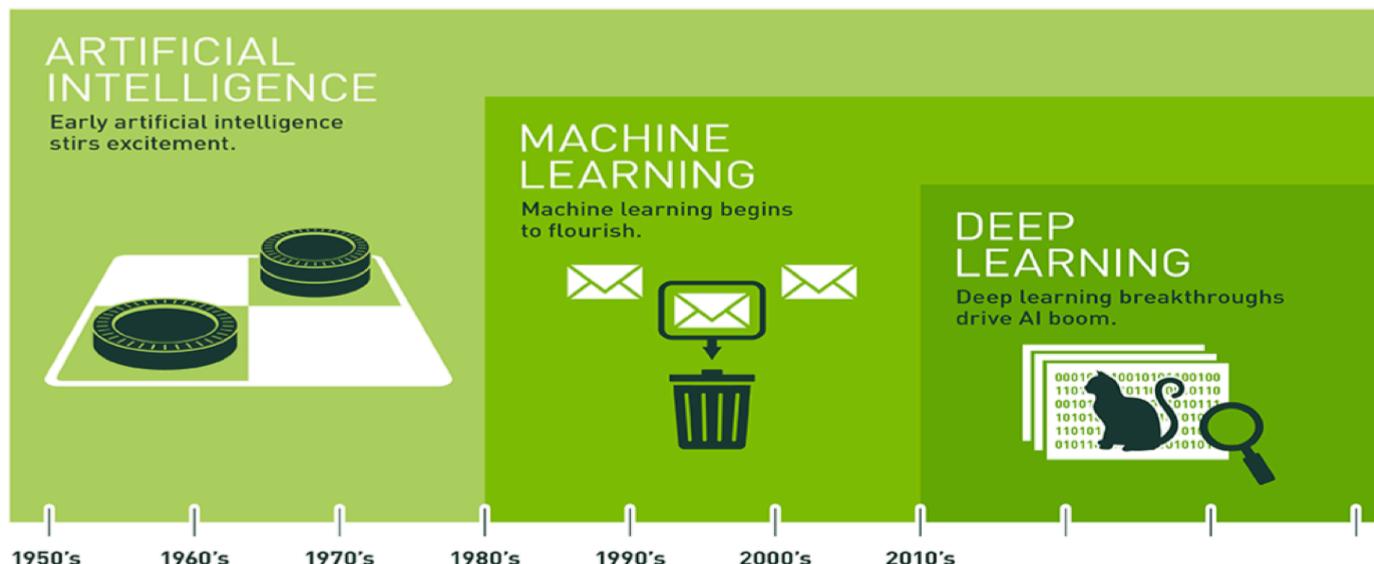
# Humildad: No pretendamos simular el cerebro humano

- El estado del arte del *Deep Learning (DL)* aún está lejos de crear inteligencias similares al cerebro humano.
- Nuestro cerebro tienen “sólo” 100 millones de neuronas. Usando DL puedes crear muchas más, pero esas neuronas de IA se modelan con una ecuación que cabe en un renglón. Una sola neurona real contiene 100 billones de átomos y decenas de miles de proteínas que emiten señales simultáneas y capaces de responder a estímulos. Es “otra cosa”.
- Un microprocesador como la GPU es mejor analogía para una neurona, y es una de las estructuras artificiales más complejas creadas por la ingeniería (54.000 Mt. en Ampere).



# Pilares conceptuales

- Inteligencia es la capacidad de desarrollar tareas complejas. Si utilizamos la GPU y sus herramientas, pasa a ser inteligencia artificial, que NO está relacionado con pensar.
- *Machine Learning* es la capacidad de enseñar al computador sin programarlo explícitamente. Dentro de él, *Deep Learning* emplea algoritmos inspirados en el cerebro.



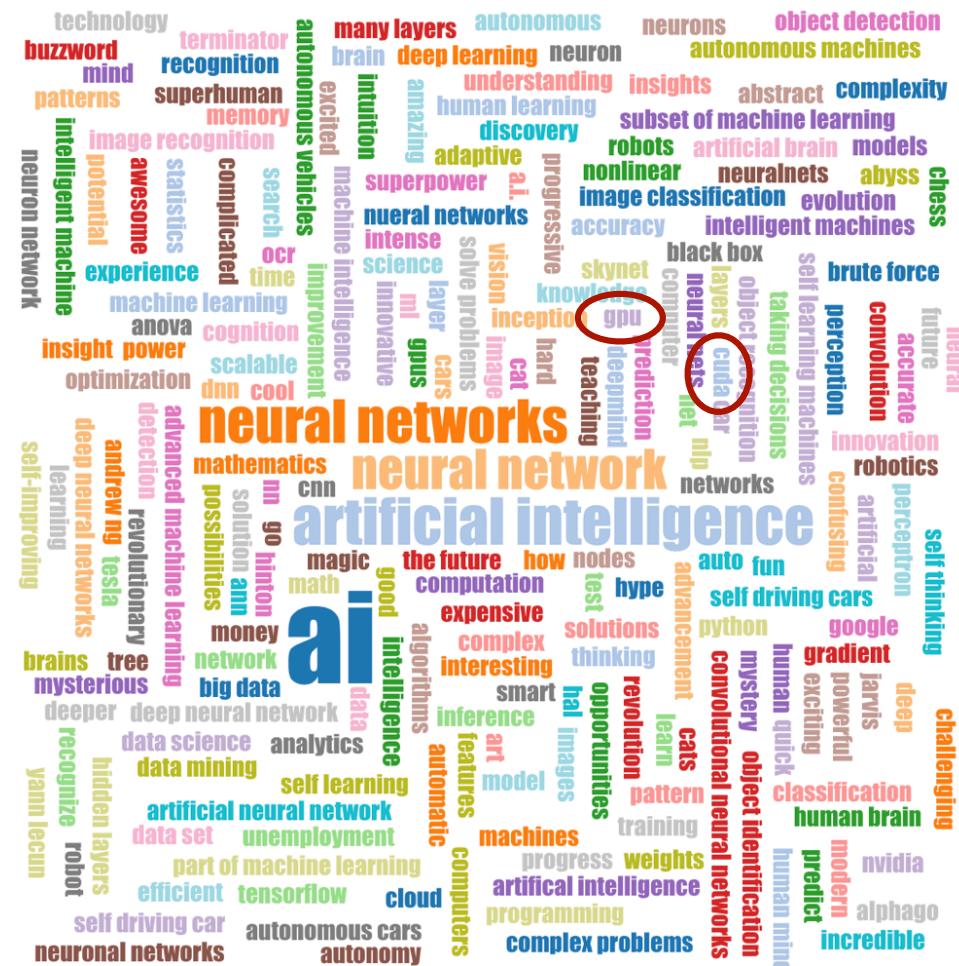
# El mapa de pensamientos iniciales

---

- Escribe la primera palabra que te viene a la mente al escuchar *Deep Learning*.

# El mapa de pensamientos iniciales

- Escribe la primera palabra que te viene a la mente al escuchar *Deep Learning*.



4094 words submitted in total.

Your words were:

- gpu 1%

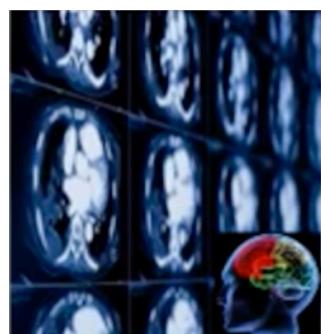
# Dónde es frecuente encontrar *Deep Learning*

## Servicios de internet



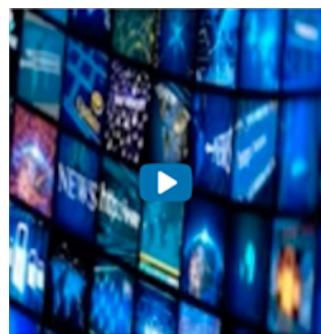
- ✓ Clasificación de imágenes y vídeo.
- ✓ Reconocimiento del habla.
- ✓ Procesamiento del lenguaje natural.

## Medicina



- ✓ Detección de células cancerígenas.
- ✓ Cuantificación diabética.
- ✓ Descubrimiento de fármacos.

## Industria del ocio



- ✓ Caracterización de video.
- ✓ Búsqueda basada en contenido.
- ✓ Traducción en tiempo real.

## Seguridad y defensa



- ✓ Reconocimiento de caras.
- ✓ Video vigilancia.
- ✓ Ciber seguridad.

## Conducción autónoma



- ✓ Detección de peatones.
- ✓ Seguimiento de carriles.
- ✓ Reconocimiento de la señalización del tráfico.

La mayoría de estas aplicaciones pertenecen al área de Computer Vision objeto de este curso

# Qué tareas se adaptan mejor a *Deep Learning*

---

- Un ejemplo: Los sistemas que pueden entendernos, como Siri, Google Assistant o Alexa. Cualquiera de ellos convierte:
  - 1. Sonidos a texto.
  - 2. Ese texto a su significado.
  - 3. Ese significado a una orden.
- ¿Cuál de estas tareas hace más uso del *Deep Learning*?

# Qué tareas se adaptan mejor a *Deep Learning*

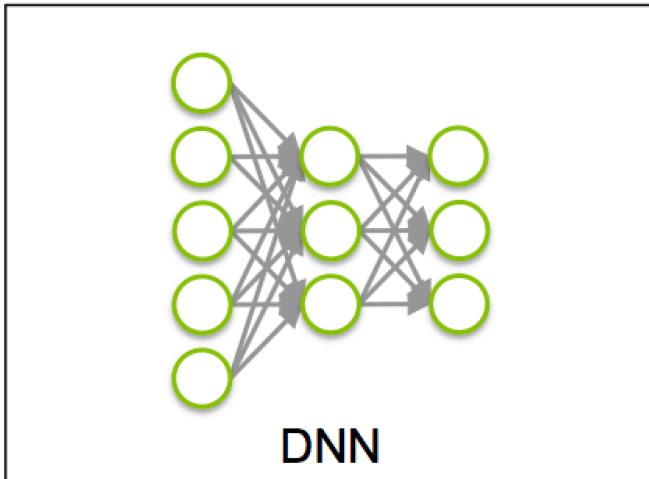
---

- Un ejemplo: Los sistemas que pueden entendernos, como Siri, Google Assistant o Alexa. Cualquiera de ellos convierte:
  - 1. Sonidos a texto.
  - 2. Ese texto a su significado.
  - 3. Ese significado a una orden.
- ¿Cuál de estas tareas hace más uso del *Deep Learning*?
  - La tarea 3 es la típica de la vieja escuela: Se basa en instrucciones o comandos. No necesita del *Deep Learning*.
  - La tarea 2 es la más compleja, puesto que se presta a multitud de interpretaciones. Aquí necesitamos mucha “inteligencia” y “aprendizaje”.
  - La tarea 1 tiene menos entradas (los fonemas están acotados), y su traducción a texto es tarea más sencilla de automatizar. No requiere tanto *Deep Learning*, se sitúa en un punto intermedio entre las 2 anteriores.

# El Big Bang en Machine Learning

---

- Las tres grandes novedades que han provocado la transición de *Machine Learning* a *Deep Learning*:



- Serán los grandes protagonistas de nuestro taller.



## II. Deep Neural Networks (*DNNs*)

# Diferencias con el aprendizaje tradicional

---

● La computación tiene una limitación: Sigue las instrucciones de un programa, y para construir éste tenemos que:

1. Saber cómo se resuelve el problema.
2. Describirlo mediante un algoritmo.
3. Expresarlo en un lenguaje de programación.

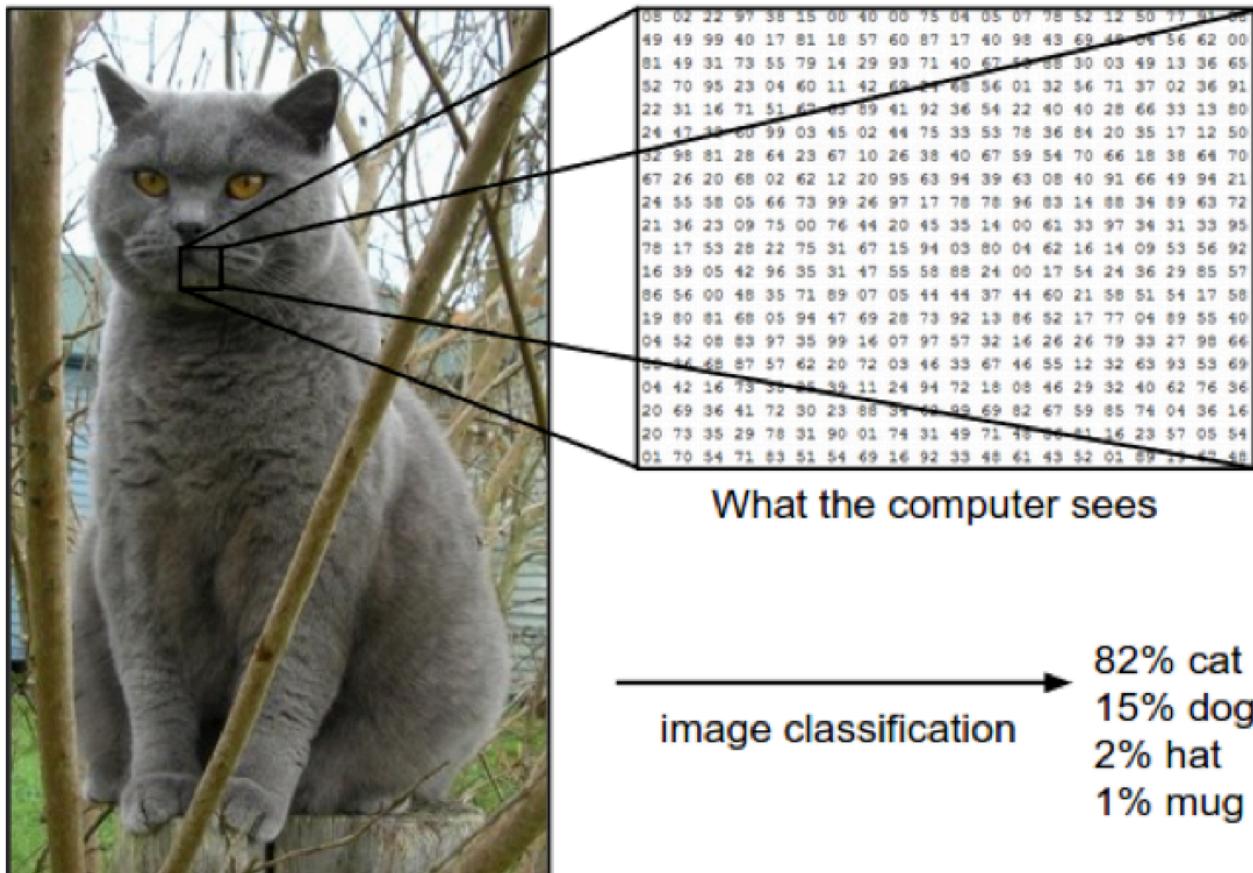
● DL permite al computador aprender a partir de ejemplos. Para ello, tenemos que:

4. Encontrar buenos ejemplos que definan la entrada y la salida.
5. Colocar una red neuronal intermedia para relacionar ambas.
6. Enseñar a la red para que establezca su mapeo correcto.

● Una DNN convierte un bloque de datos o **tensor** en otro a través de un modelo que empareja ambos. Lo que esos tensores representan es muy flexible, de ahí su polivalencia para la resolución de problemas.

# Un ejemplo clásico: Clasificación de imágenes

- Tensor de entrada: Píxeles de la imagen (512 x 256 x 3)
- Tensor de salida: Vector de 4 probabilidades (82, 15, 2, 1)



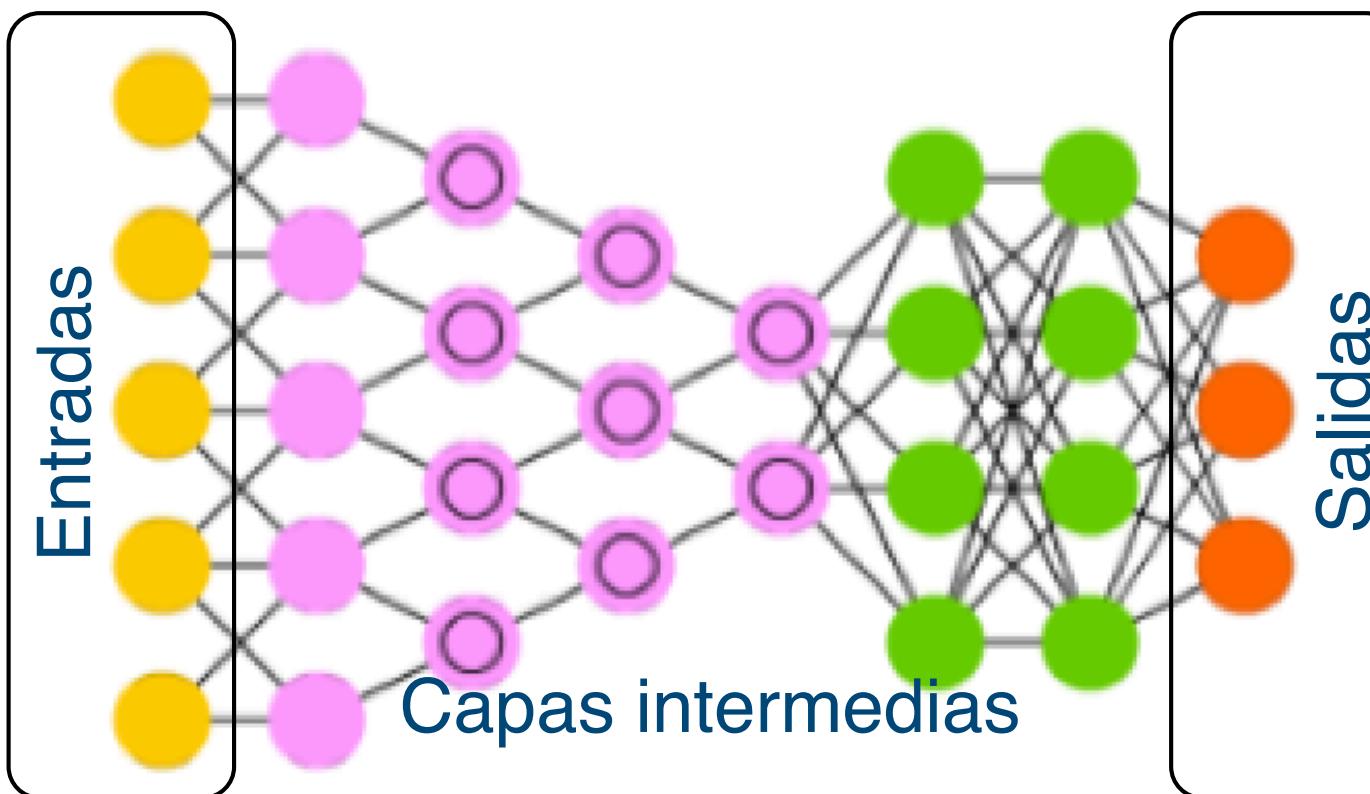
# Algunos mapeos entre tensores de entrada y salida para aplicaciones populares

Flujo de trabajo	Tensor de entrada	Tensor de salida
Clasificación de imágenes	Matriz de píxeles	Un vector con tantas componentes como clases queramos clasificar, en el que cada valor representa la probabilidad de que la imagen pertenezca a esa clase
Detección de objetos	Matriz de píxeles	Un vector con pares de coordenadas (X, Y) para las esquinas superior izquierda e inferior derecha de cada objeto localizado en la imagen
Segmentación de imágenes	Matriz de píxeles	Una superposición de la imagen para cada clase que está siendo segmentada, donde cada valor indica la probabilidad de que el píxel pertenezca a cada clase
Generación de texto	Un vector exclusivo para cada token (palabra, letra, ...)	Un vector representando al token que tenga mayor probabilidad de sucederle en la secuencia
Renderizado de imágenes	Matriz de píxeles de una imagen granulada	Matriz de píxeles de una imagen diáfana

# Otro ejemplo más sofisticado: Conducción autónoma



- Entradas: Miles de señales procedentes de sensores y cámaras.
- Capas intermedias: Cientos o incluso miles.
- Salidas: Volante, freno y acelerador.



EL GRUPO AVANZA LIDERÁ LA PRIMERA GRAN EXPERIENCIA DE UN AUTOBÚS ELÉCTRICO SIN CONDUCTOR DE 12 METROS Y HASTA 60 PASAJEROS QUE SE ESTRENARÁ EN LA CAPITAL DE LA COSTA DEL SOL.

## Málaga pone a prueba el autobús urbano autónomo

Málaga será el laboratorio para el primer autobús sin conductor en mayo

En mayo, un autobús eléctrico y sin conductor recorrerá 7 kilómetros de la capital desde la Estación Marítima hasta el Paseo del Parque

Ana I. Montañez | 24.01.2020 | 14:11

En los próximos meses, parte del centro histórico de Málaga será escenario de un novedoso autobús eléctrico de 12 metros de longitud que circulará junto al tráfico rodado de la capital de forma autónoma, esto es, sin conductor, aunque sí que contará con un operario que vigilará el comportamiento del vehículo durante el trayecto.

"En los niveles a los que nos referimos [nivel 3 de automatización], es la primera ciudad de Europa y no tenemos noticia de ninguna otra del mundo hasta ahora de que lo vaya a tener en los próximos meses", ha destacado el regidor, Francisco de la Torre.

Se trata del proyecto I+D+I AutoMost, impulsado por la compañía Avanza junto a la colaboración de un consorcio de siete empresas y el apoyo del Ayuntamiento de Málaga. Con un presupuesto de 9 millones de euros, financiado por el Centro para el Desarrollo Tecnológico Industrial (CDTI), el vehículo, desarrollado por la firma Irizar, replicará el recorrido de la línea 90 (actualmente operada por EMT y Avanza), recogerá a los cruceristas que lleguen a la ciudad y circulará durante unos 7 kilómetros hasta una parada ubicada en el Paseo del Parque frente al Ayuntamiento.



Presentación del proyecto AutoMost, un autobús eléctrico con conducción autónoma. A.I.M.

### La UMA estudiará la percepción de la conducción autónoma

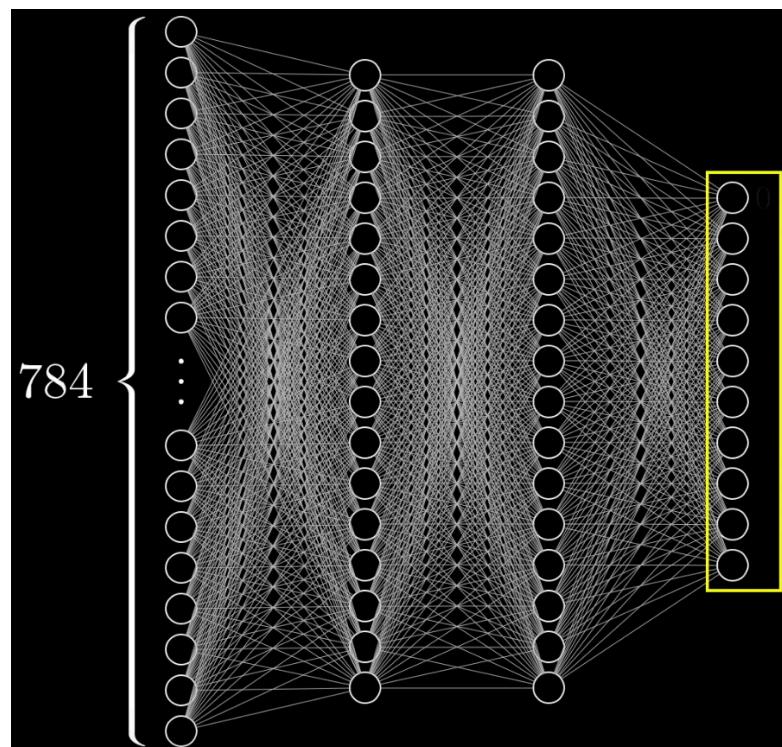
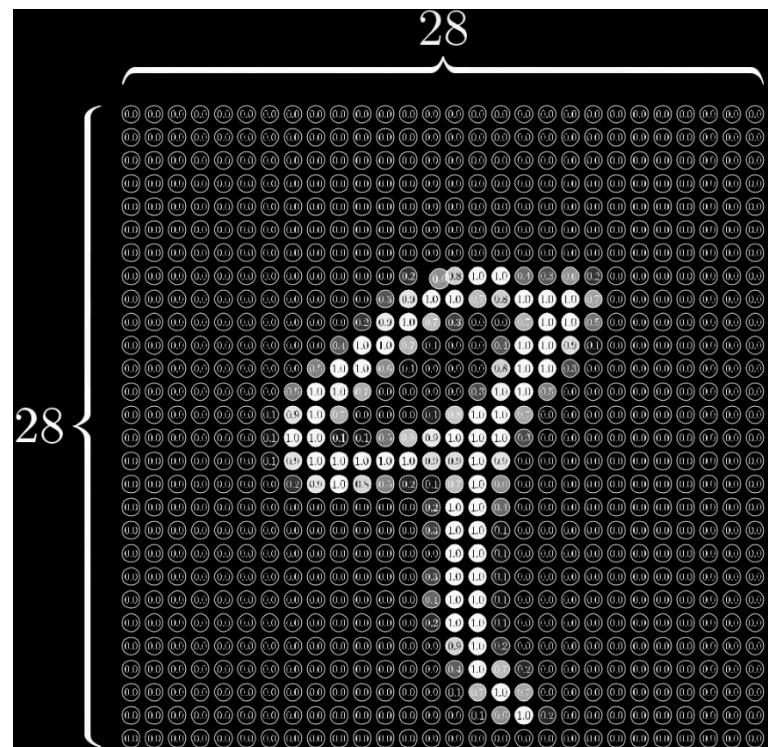
La excepcionalidad de traer la conducción autónoma al transporte público convierte al proyecto de AutoMost en una actuación sin precedentes y con pocas investigaciones al respecto. La posibilidad de que desplazarse en un autobús sin conductor se convierta en una cotidianidad en el futuro ha llevado a la Universidad de Málaga, a través de la Cátedra de Gestión del Transporte, a emprender una serie de estudios científicos para conocer el grado de receptividad que tienen los ciudadanos sobre los vehículos de transporte público con conducción automática. La Universidad de Málaga realizará una serie de encuestas a usuarios del autobús para conocer qué opinan sobre descartar la presencia de un conductor en

# Diferencias entre el método de aprendizaje humano y por computador

- ¿Cómo darías instrucciones para diferenciar el dígito escrito a mano de la derecha?

• A un niño de 2 años.

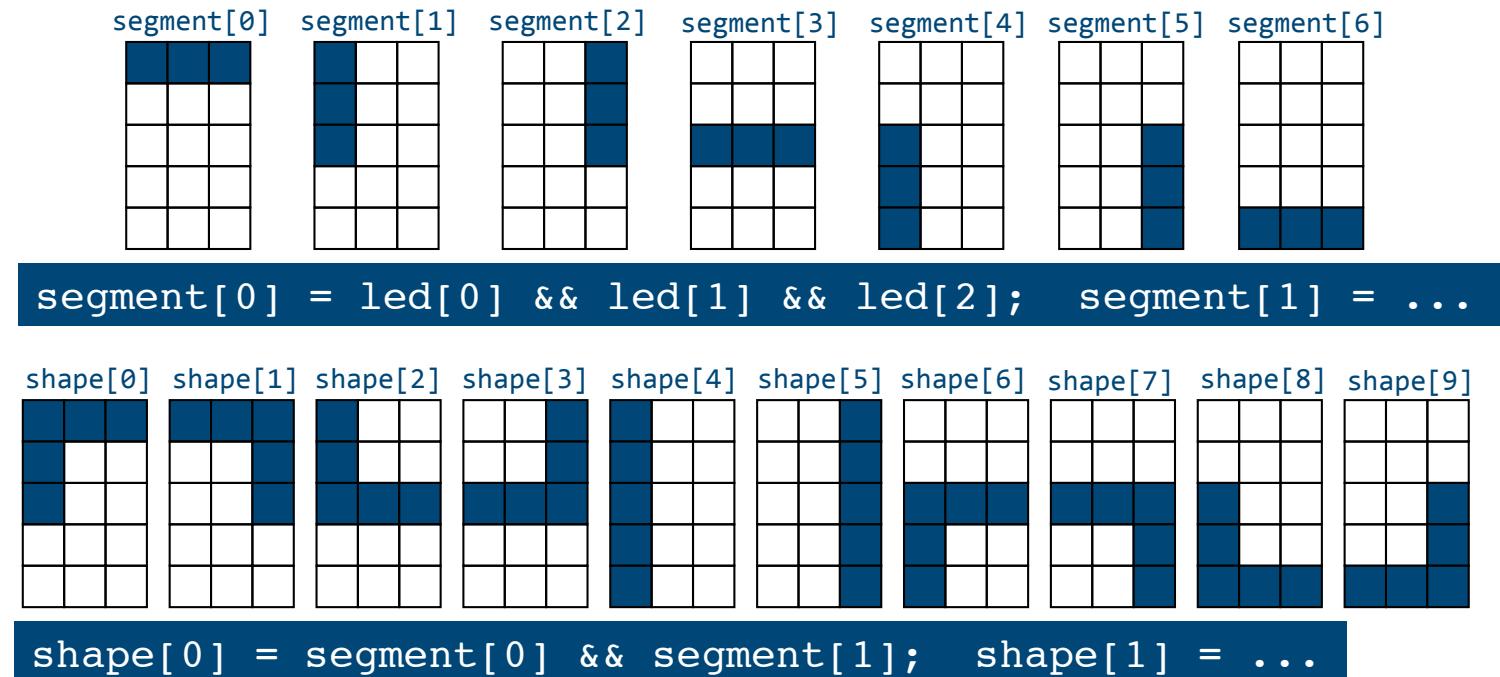
• A un PC mediante un lenguaje de programación (o pseudo-código).



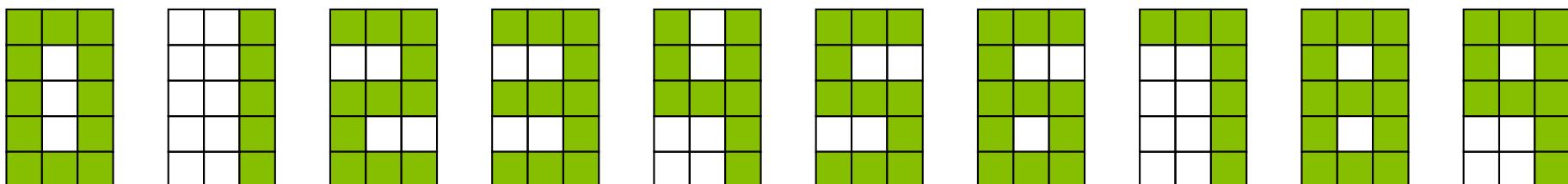
# ¿Podemos reformular el problema para que pueda resolverlo la programación tradicional?

LED 0	LED 1	LED 2
LED 3	X	LED 4
LED 5	LED 6	LED 7
LED 8	X	LED 9
LED10	LED11	LED12

Marcador digital



```
if (segment[2] && segment[5]) && (!segment[<other>]) then printf("One");
if shape[5] && (!shape[<other>]) then printf("One");
if (led[2] && led[4] && led[7] && led[9] && led[12]) && (!led[<other>]) then printf("One");
```



# Así se resolvería la idea anterior con DNNs

## Red sin capas:

entradas:

LED 0

LED 1

LED 2

LED 3

LED 4

LED 5

LED 6

LED 7

LED 8

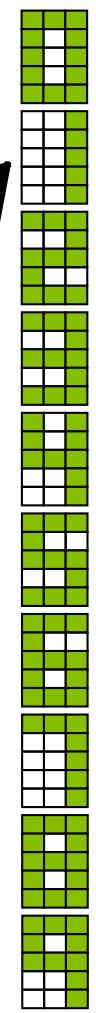
LED 9

LED 10

LED 11

LED 12

salidas:



## Red con 1 capa:

capa 1: segmentos

LED 0

LED 1

LED 2

LED 3

LED 4

LED 5

LED 6

LED 7

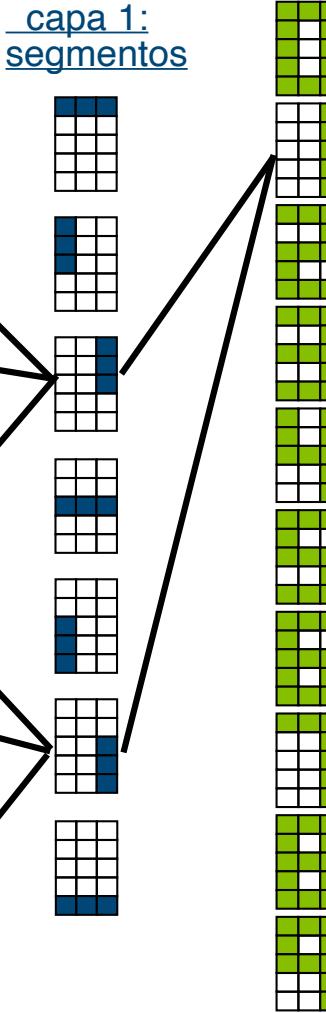
LED 8

LED 9

LED 10

LED 11

LED 12



## Red con 2 capas:

capa 2: formas

capa 1: segmentos

LED 0

LED 1

LED 2

LED 3

LED 4

LED 5

LED 6

LED 7

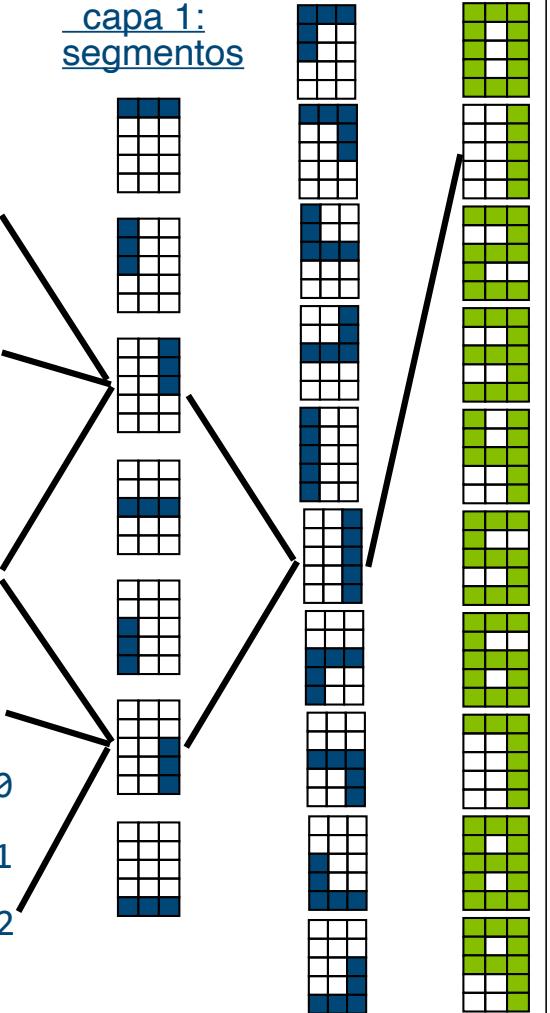
LED 8

LED 9

LED 10

LED 11

LED 12



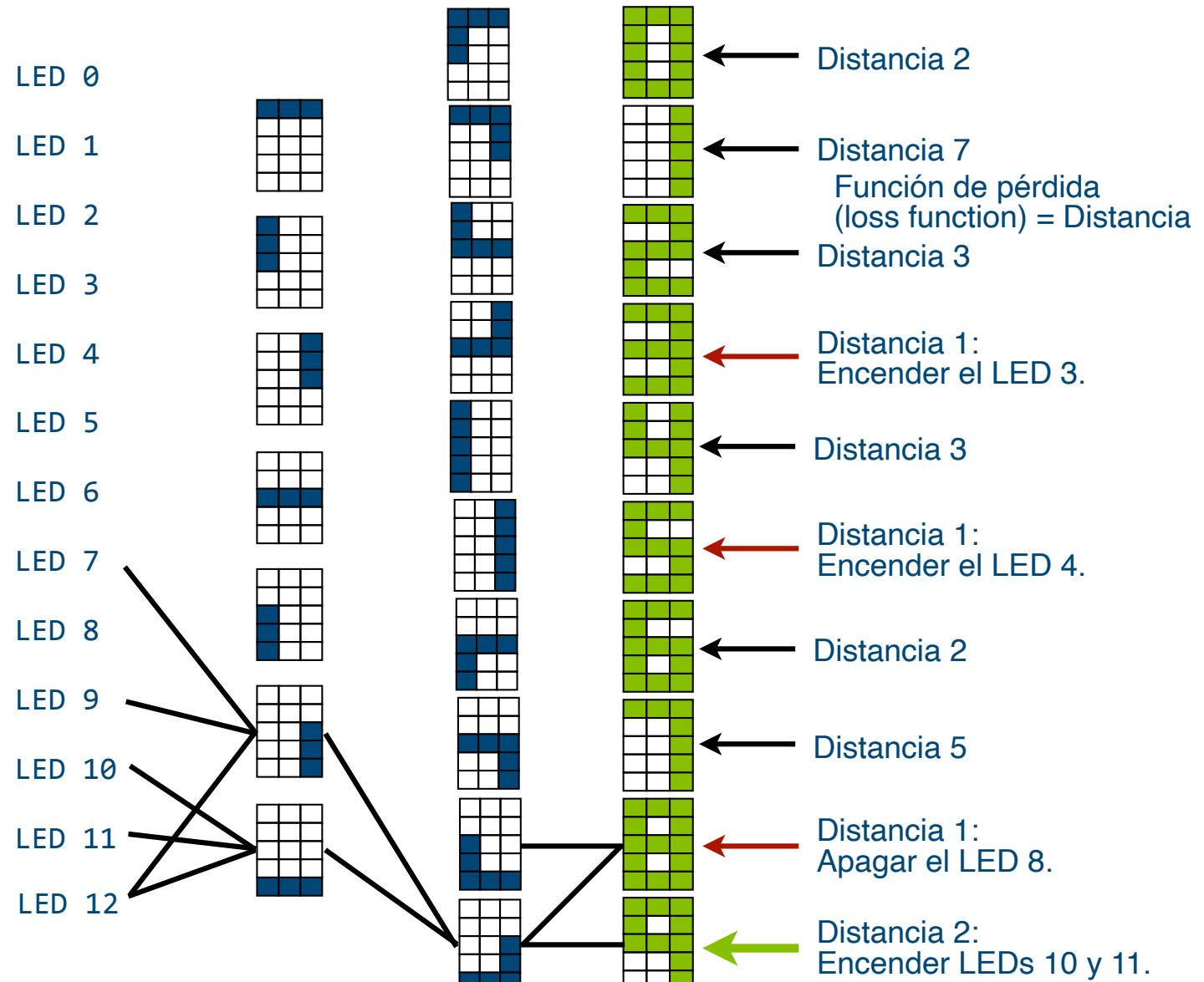
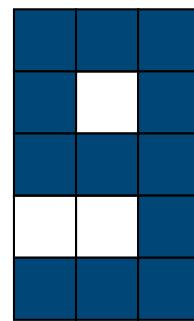
# Analizando la implementación tradicional

---

- ➊ **Fuerza bruta sin inteligencia.** El concepto de segmento simplifica la implementación, pero el de forma no.
  - ➊ Sin embargo, es una idea más inteligente.
- ➋ **Rigidez.** Cualquier otro estado para los 13 LEDs adicional a los 10 ya considerados requiere incorporar código.
  - ➊ El algoritmo ni se adapta ni aprende de lo ya procesado.
- ➌ **Limitaciones.** Cuando redefinimos el problema, pasando del marcador digital a la escritura manual, nada de lo ya programado puede aprovecharse.
  - ➊ En cambio, el planteamiento de la DNN sigue vigente.

# ¿Cómo aprende de los errores la DNN?

LED 0	LED 1	LED 2
LED 3	X	LED 4
LED 5	LED 6	LED 7
LED 8	X	LED 9
LED10	LED11	LED12



# Traslación del problema a *Computer Vision*

---

- ➊ Reconocer dígitos escritos a lápiz sobre un retículo de 28x28 píxeles requiere destreza para detectar formas a partir de segmentos y componer los dígitos a partir de éstas.
- ➋ Cada píxel tendrá su tono de gris, que podemos discretizar a 256 niveles y acotar al rango [0,1] de las entradas de la DNN.
- ➌ Necesitaremos un preprocessamiento **antes** de usar la red y **después** de que nos haya dado los resultados, para:
  - ➍ Delimitar la entrada a la DNN: 28x28 píxeles de 256 tonos.
  - ➎ Diseñar la arquitectura de la red. Podríamos empezar con 2 capas (segmentos y formas), pero ¿cuántas neuronas colocamos de cada tipo?
  - ➏ Establecer la salida (resultados) que esperamos de la DNN: Un vector de 10 probabilidades, una para cada dígito a reconocer.

# Los elementos de nuestra red neuronal

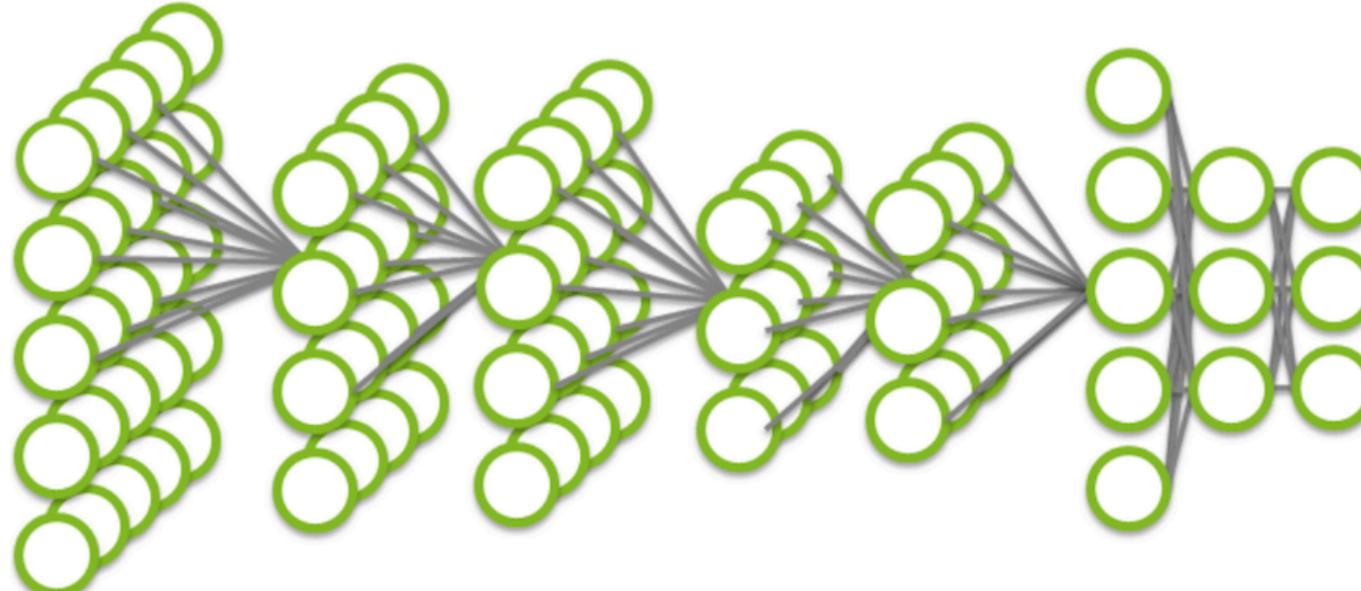
---

- Entradas (o **features**): Determinadas por el problema. Condicionan el preprocesamiento necesario.
- Arquitectura de la red. Hay de diversos tipos:
  - Totalmente conectadas (FCN): Correspondencia total entre capas.
  - Convolucionales (CNN): Relacionan espacialmente las neuronas.
  - Profundas (DNN): Habilitan multitud de capas.
  - Híbridas: Como AlexNet, que es una DNN con 5 capas “C” y 3 capas “F”.
- El modelo de la red se compone de la arquitectura y los pesos (**weights**), que quedan definidos tras el entrenamiento. Para éste, dividimos el conjunto de datos en dos partes que usaremos durante entrenamiento y validación. El modelo se usa para la inferencia de la red, esto es, su aplicación real.
- La salida, que para un clasificador, se compone de un vector con las probabilidades de pertenencia a cada clase.

# Propagación hacia adelante y hacia atrás (forward and backward propagation)

La propagación hacia adelante infiere una etiqueta para cada imagen del entrenamiento

Repite el proceso con otra muestra del *dataset* hasta utilizar todas las muestras



La función de pérdida (o coste) se utiliza para calcular la “diferencia” entre la etiqueta predicha y la etiqueta inferida para cada imagen

Se ajustan los pesos durante la propagación hacia atrás

# Etapas de validación e inferencia

---

- Guardamos aprox. el 25% de las muestras para validar el modelo una vez entrenado, antes de usarlo en el mundo real.
- Durante **validación** sólo se realiza propagación hacia adelante. La función de pérdida se promedia para todas las muestras de validación para informar de la eficiencia de nuestro modelo, y se mide el rendimiento para conocer las restricciones de tiempo real.
- Si la eficiencia o el rendimiento son deficientes, aún podemos ajustar los **hiperparámetros** del modelo para cumplir con las exigencias del problema.
- Una vez satisfechas las expectativas, el modelo pasa al modo de **inferencia**, en el que se despliega dentro de una aplicación, aceptando entradas del mundo real y devolviendo salidas útiles. Aquí puede usarse un tercer conjunto de muestras denominado **test**.

# Aprendizaje por lotes (batches): Iteraciones más rápidas a cambio de converger algo menos

0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1
2 2 2 2	2 2 2 2	2 2 2 2	2 2 2 2
3 3 3 3	3 3 3 3	3 3 3 3	3 3 3 3
4 4 4 4	4 4 4 4	4 4 4 4	4 4 4 4
5 5 5 5	5 5 5 5	5 5 5 5	5 5 5 5
6 6 6 6	6 6 6 6	6 6 6 6	6 6 6 6
7 7 7 7	7 7 7 7	7 7 7 7	7 7 7 7
8 8 8 8	8 8 8 8	8 8 8 8	8 8 8 8
9 9 9 9	9 9 9 9	9 9 9 9	9 9 9 9

# Recopilando los 13 conceptos de las 3 etapas de entrenamiento (para una pasada o epoch)

---

- Cada muestra<sup>1</sup> pasa por el modelo<sup>2</sup> DNN (forward process)<sup>1</sup> para obtener las activaciones<sup>3</sup> de cada capa hasta la salida<sup>4</sup>.
- Una función<sup>5</sup> de pérdida (o función<sup>6</sup> de coste) específica de cada problema calcula el error<sup>7</sup> entre la salida y la etiqueta<sup>8</sup> para esa muestra, que es retropropagada<sup>2</sup>, para obtener las derivadas<sup>9</sup> (cambios) que trasladar a pesos<sup>10</sup> (weights) y bias<sup>11</sup>.
- SGD<sup>12</sup> (*Stochastic Gradient Descent*) usa el vector gradiente negativo de la función de coste para computar las derivadas. La función de coste puede promediarse para todas las muestras del lote<sup>13</sup>, retropropagando sólo una vez por lote.
- Repetir para un segundo lote hasta procesarlos todos.
- Las activaciones y las derivadas son lo más costoso.

# Recursos en la Web

- Base de datos MNIST de números escritos a mano:

  - <http://yann.lecun.com/exdb/mnist/>

- ¡Juega y diviértete! Traza un número con el ratón y mira cómo lo analiza por capas y neuronas, así como la respuesta que produce la red:

  - <https://cs.ryerson.ca/~aharley/vis/conv>

- Cómo aprende una red a reconocer los números escritos a mano. Todos los conceptos importantes en:

  - Videos en YouTube: 3Blue1Brown.

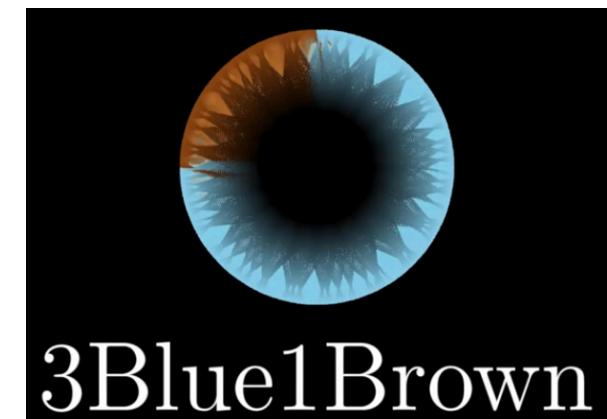
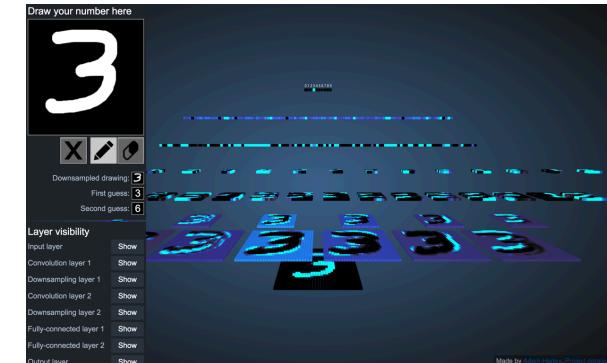
## THE MNIST DATABASE

### of handwritten digits

Yann LeCun, Courant Institute, NYU

Corinna Cortes, Google Labs, New York

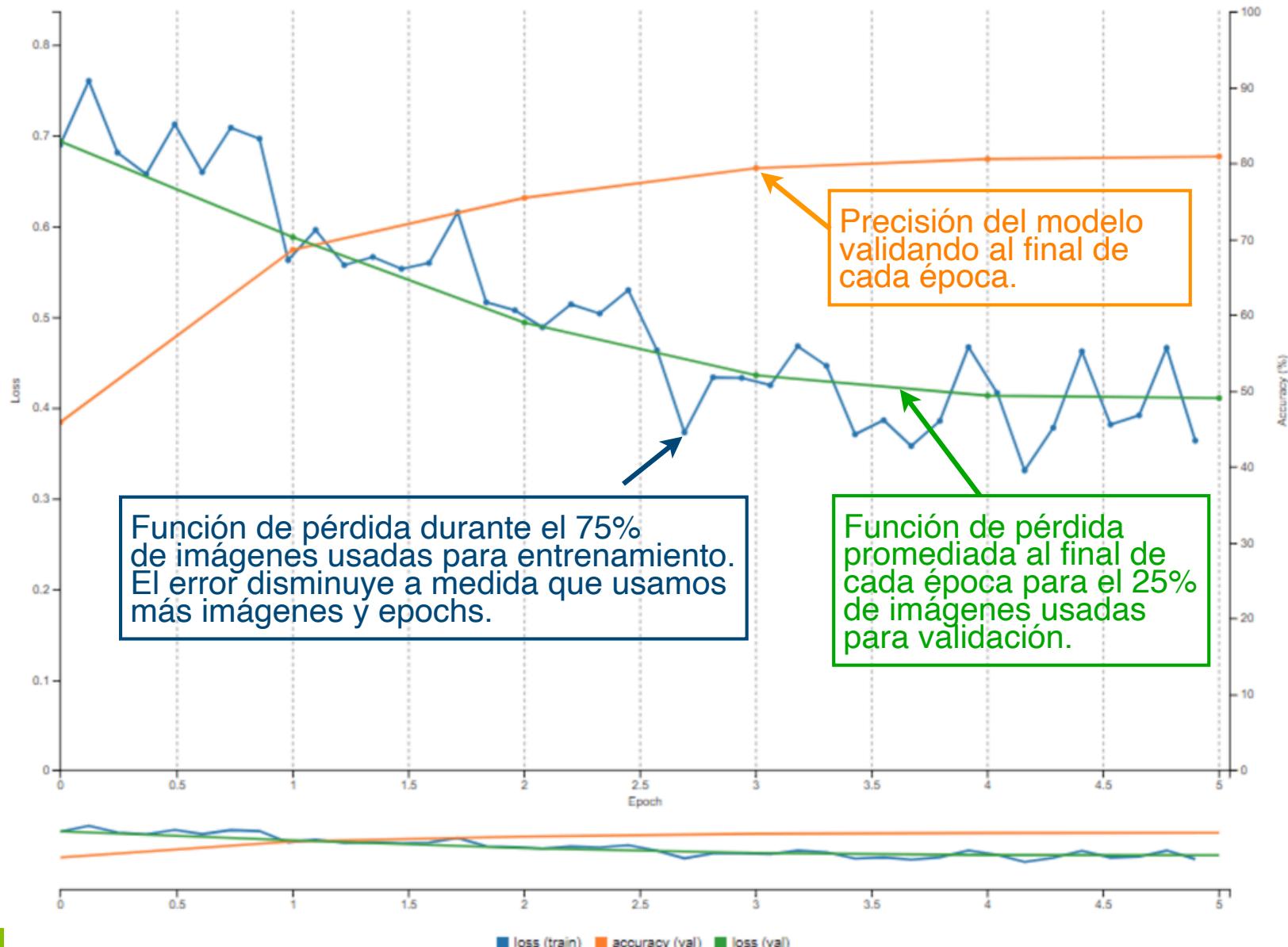
Christopher J.C. Burges, Microsoft Research, Redmond



Realización del  
primer proyecto  
práctico del DLI:  
**MNIST**

# Primer proyecto práctico: MNIST.

## Análisis gráfico de los resultados de Keras



# Aprendizaje supervisado vs. no supervisado

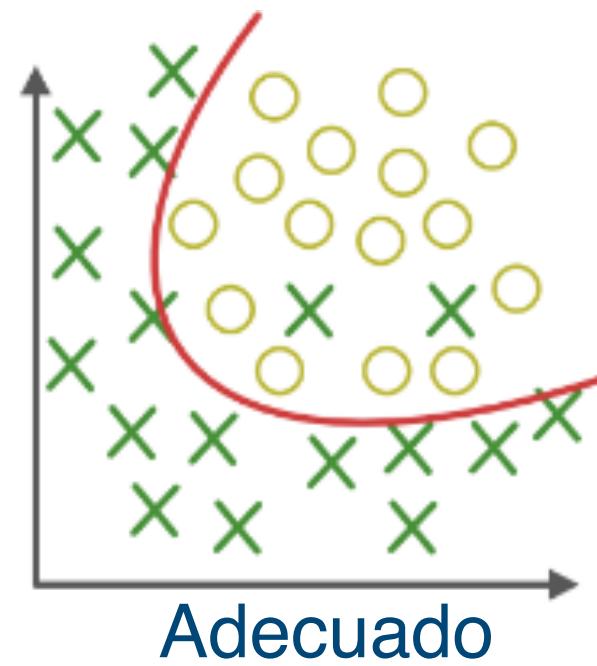
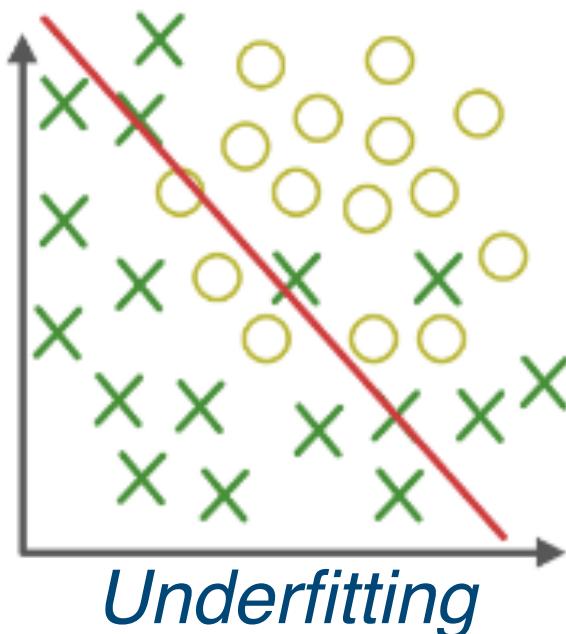
---

- El aprendizaje supervisado necesita entrenamiento y contar con los patrones de entrada y salida al sistema.
- El aprendizaje no supervisado descubre por sí mismo los rasgos de la población de entrada. No dispone de muestras ni establece un conjunto de clases a priori.

	Supervisado	No supervisado
Objetivo	Etiquetar correctamente nuevos datos	Agrupar las muestras en categorías
¿Muestras etiquetadas?	Sí	No
Complejidad computacional	Alta	Baja
Análisis	Fuera de línea	En tiempo real
Precisión	Alta	Baja
Subdominios	Clasificación y regresión	<i>Clustering y association rule mining</i>
Ejemplos	Redes neuronales	<i>K-means, PCA</i>

# *Underfitting vs. overfitting* (infra-entrenamiento y sobre-entrenamiento)

- Problemas que surgen al entrenar y testear *datasets*:
  - Underfitting*: Demasiado simple para explicar la varianza.
  - Overfitting*: Demasiado bueno como para que sea cierto. El sistema clasifica cada vez mejor las muestras de entrenamiento, pero apenas distingue las muestras de validación. Memoriza en lugar de aprender.



# Resumen intuitivo de 10 conceptos clave

Concepto	Analogía en nuestra vida
<i>Epoch</i> (pasada)	Perseverancia del humano a la hora de aprender
<i>Sample</i> (muestra)	Un “átomo” de experiencia
<i>Training</i> (entrenamiento)	Experiencia previa acumulada
Forward propagation	Metodología de aprendizaje, configurable con los hiperparámetros
Backward propagation	Aprender de los errores y rectificar
Entradas de la red	Nuestros cinco sentidos
<i>Batch</i> (lote)	Aprendizaje por el método de “divide y vencerás”
Aprendizaje supervisado	Aprender de la mano de un instructor que te guía ( <i>instructor-led labs</i> )
Aprendizaje no supervisado	Aprender a tu aire ( <i>self-paced labs</i> en el DLI)
<i>Overfitting</i> (sobreentrenam.)	Estudio memorizando y me especializo en pasar el examen, pero en realidad aprendo bien poco (y más viciado cuantas más <i>epochs</i> )

# Mejoras del proceso en los últimos años

---

## ● Propagación hacia adelante:

- Utilizar ReLU en las funciones de activación de las neuronas.
  - Aprende más rápido y tiene mayor estabilidad numérica que Sigmoid.
  - Tanh es otra alternativa que está perdiendo vigencia.
- Normalizar las variables de entrada (media 0, varianza 1).
- Utilizar *dropout* en *epochs* para prevenir *overfitting*.
- Planificar las *epochs* para ir atenuando el *learning rate*.

## ● Propagación hacia atrás:

- Dividir en lotes a los que aplicar SGD, barajando entre *epochs*.

## ● Arquitectura de la red:

- RNNs (Recurrent Neural Networks): Las salidas de las capas realimentan a las entradas para que la información persista entre *epochs* a través de una serie de pasos temporales.