

# Taller de deep learning

---

Isaac Pérez Borrero

[isaac.perez@alu.uhu.es](mailto:isaac.perez@alu.uhu.es)

# Índice

## 1. Toma de contacto.

# Índice

---

1. Toma de contacto.
2. Mejoras con respecto al *machine learning*.

# Índice

---

1. Toma de contacto.
2. Mejoras con respecto al *machine learning*.
3. Breve historia del *deep learning*.

# Índice

1. Toma de contacto.
2. Mejoras con respecto al *machine learning*.
3. Breve historia del *deep learning*.
4. Estado del arte.

# Índice

1. Toma de contacto.
2. Mejoras con respecto al *machine learning*.
3. Breve historia del *deep learning*.
4. Estado del arte.
5. Ejemplos de aplicación.

# Índice

1. Toma de contacto.
2. Mejoras con respecto al *machine learning*.
3. Breve historia del *deep learning*.
4. Estado del arte.
5. Ejemplos de aplicación.
6. Fundamentos del *deep learning*.

# Índice

1. Toma de contacto.
2. Mejoras con respecto al *machine learning*.
3. Breve historia del *deep learning*.
4. Estado del arte.
5. Ejemplos de aplicación.
6. Fundamentos del *deep learning*.
7. Algoritmo de *deep learning*: *redes neuronales convolucionales*.

# Índice

1. Toma de contacto.
2. Mejoras con respecto al *machine learning*.
3. Breve historia del *deep learning*.
4. Estado del arte.
5. Ejemplos de aplicación.
6. Fundamentos del *deep learning*.
7. Algoritmo de *deep learning*: *redes neuronales convolucionales*.
8. Caso práctico: *LeNet5*.

## **Toma de contacto**

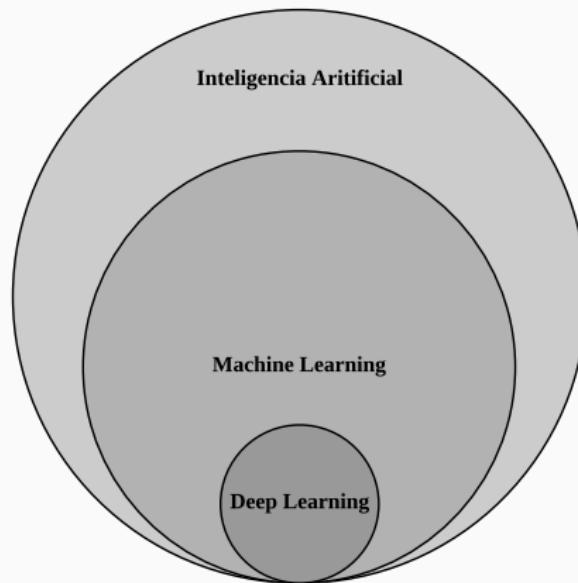
---

## Toma de contacto

- Una primera **definición** de lo que es el *deep learning*:  
*Son un conjunto de algoritmos que se sitúan dentro del machine learning.*

# Toma de contacto

- Una primera **definición** de lo que es el *deep learning*:  
*Son un conjunto de algoritmos que se sitúan dentro del machine learning.*

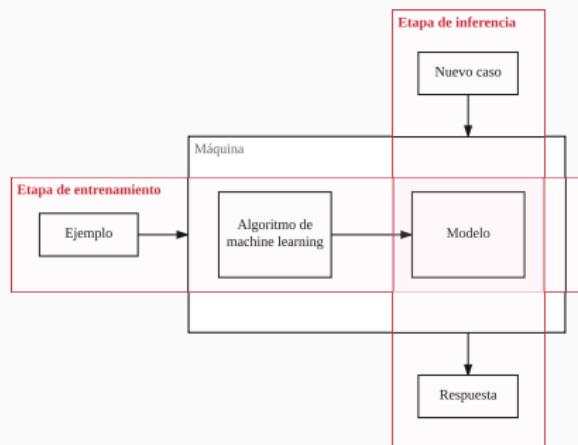


## Toma de contacto

- Similitud con el *machine learning*:
  - Busca el “algoritmo” que resuelva el problema mediante una etapa de aprendizaje con un conjunto de datos.

# Toma de contacto

- Similitud con el *machine learning*:
  - Busca el “algoritmo” que resuelva el problema mediante una etapa de aprendizaje con un conjunto de datos.

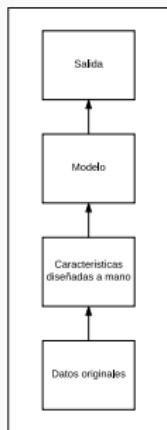


## Toma de contacto

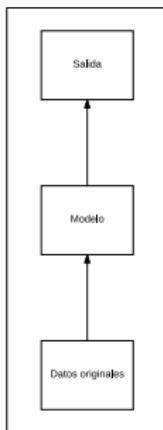
- Diferencia con el *machine learning*:
  - El propio algoritmo es el encargado de **extraer las características** que considere más relevantes, **de los datos en bruto**, para resolver el problema.

# Toma de contacto

- Diferencia con el *machine learning*:
  - El propio algoritmo es el encargado de **extraer las características** que considere más relevantes, **de los datos en bruto**, para resolver el problema.



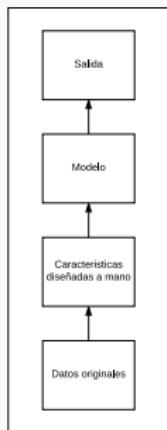
Sistema basado en machine learning



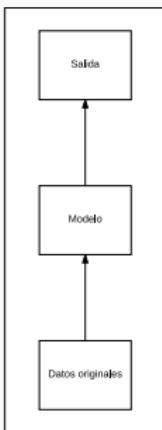
Sistema basado en deep learning

# Toma de contacto

- Diferencia con el *machine learning*:
  - El propio algoritmo es el encargado de **extraer las características** que considere más relevantes, **de los datos en bruto**, para resolver el problema.



Sistema basado en machine learning



Sistema basado en deep learning



## **Mejoras con respecto al machine learning**

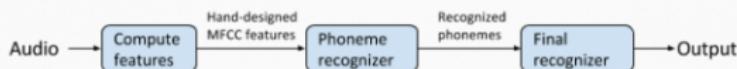
---

# Mejoras con respecto al machine learning

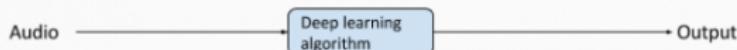
- Facilita el **desarrollo**.

## Speech recognition

### Traditional model:

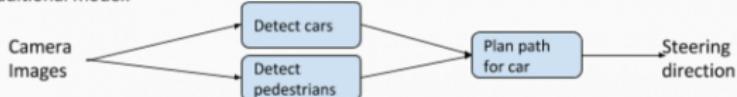


### End-to-end learning:



## Autonomous driving

### Traditional model:

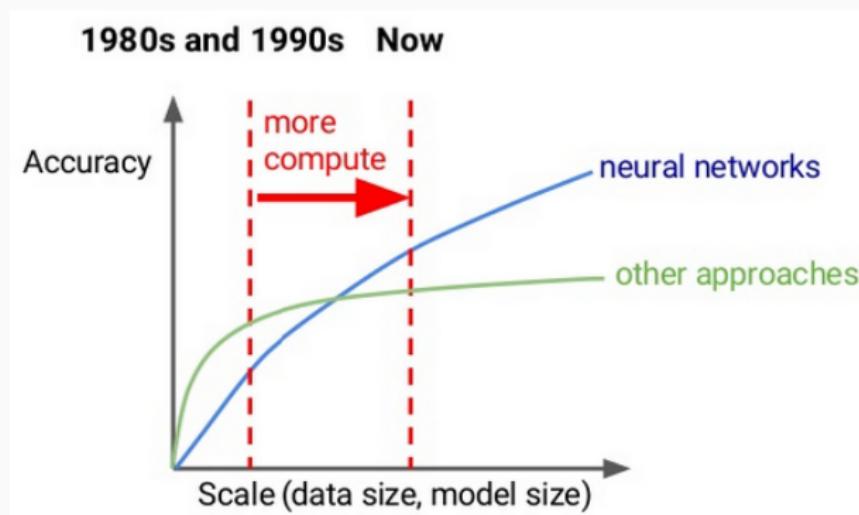


### End-to-end learning:



# Mejoras con respecto al machine learning

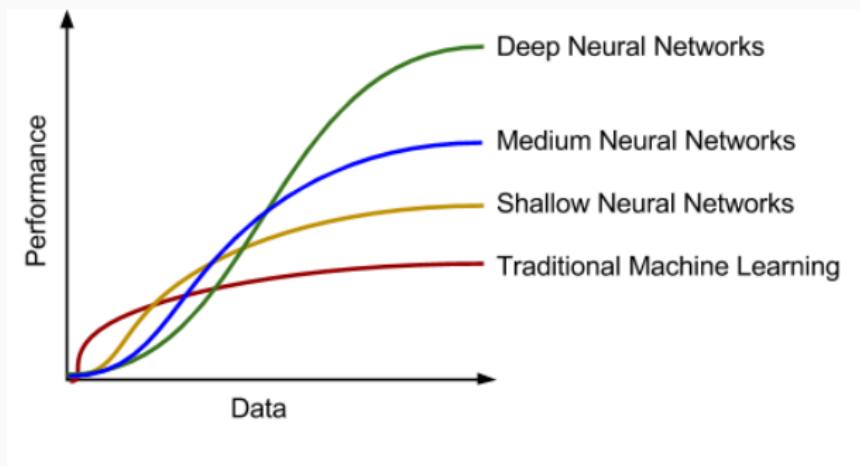
- Mejor rendimiento gracias a la mayor capacidad de almacenamiento y **cómputo**.



# Mejoras con respecto al machine learning

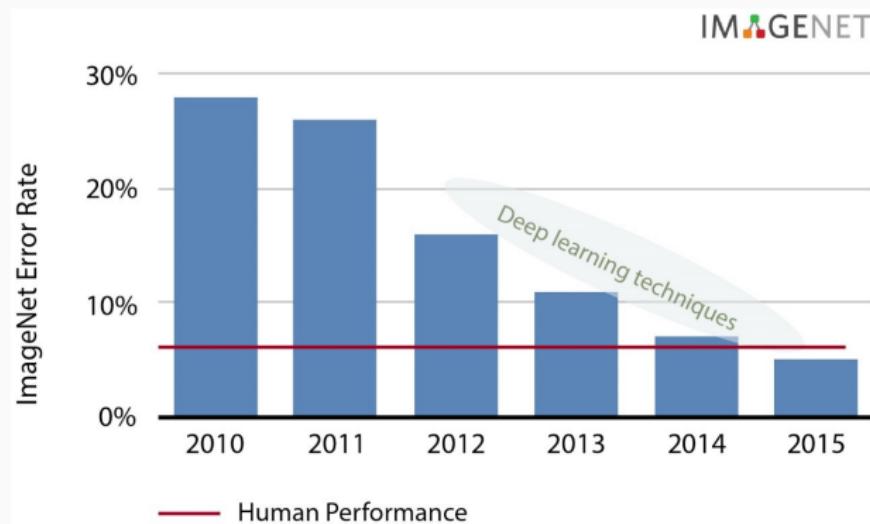
- Mejora el **rendimiento** de las técnicas tradicionales:

$$\uparrow \# \text{Capas} + \uparrow \# \text{Datos} = \uparrow \text{Rendimiento}$$



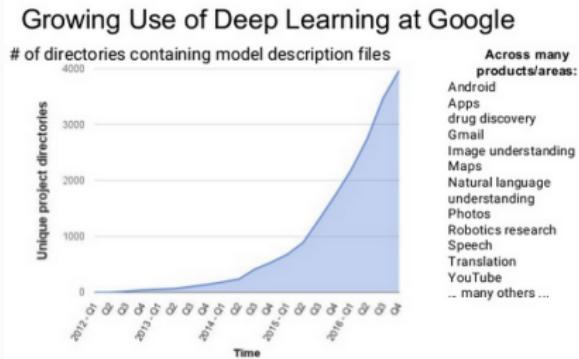
# Mejoras con respecto al machine learning

- Ha demostrado de forma práctica su **potencial**.



# Mejoras con respecto al machine learning

- Creciente **interés** por parte de empresas y usuarios en esta tecnología debido a sus resultados.

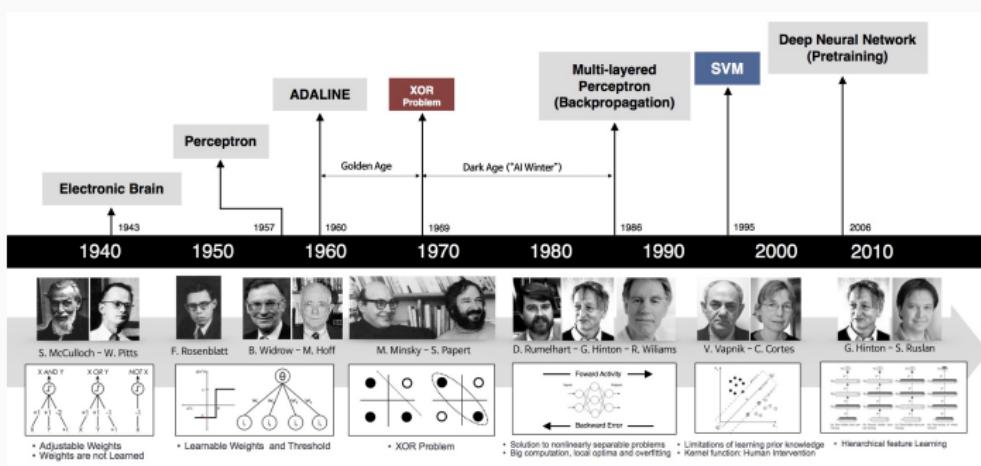


## Breve historia del deep learning

---

# Breve historia del deep learning

- Historia ligada al desarrollo de la **inteligencia artificial** y, en concreto, al desarrollo de las **redes neuronales**.



## Estado del arte

---

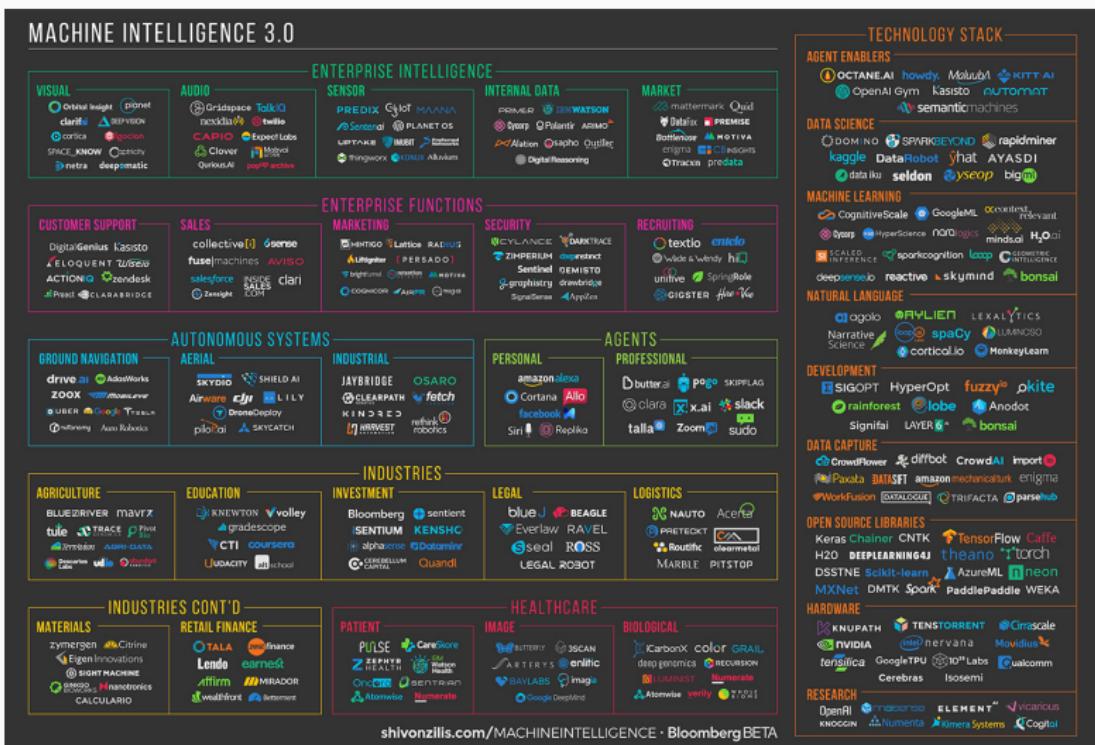
# Estado del arte

- Principales referentes del *deep learning* (de izda. a dcha.):  
*Yann LeCun, Geoffrey Hinton, Yoshua Bengio,  
Andrew Ng.*



# Estado del arte

- Principales empresas relacionadas con el *deep learning*:



# Estado del arte

- Principales *frameworks* de desarrollo de *deep learning*:



Framework Comparison: Basic information\*

Viewpoint	Torch.n*	Theano***	Caffe	autograd (NumPy, Torch)	Chainer	MXNet	Tensor- Flow
GitHub stars	4,719	3,457	9,590	N: 654 T: 554	1,295	3,316	20,981
Started from	2002	2008	2013	2015	2015	2015	2015
Open issues/PRs	97/26	525/105	407/204	N: 9/0 T: 3/1	95/25	271/18	330/33
Main developers	Facebook, Twitter, Google, etc.	Université de Montréal (U.C. Berkeley)	BVLC (U.C. Berkeley)	N: HIPS (Apple Unix) T: Twitter	Preferred Networks	DMLC	Google
Core languages	C/Lua	C/Python	C++	Python/Lua	Python	C++	C++/Python
Supported languages	Lua	Python	C++/Python MATLAB	Python/Lua	Python	C++/Python R/Julia/Go etc.	C++/Python

\* Data was taken on Apr. 12, 2016

\*\* Includes statistics of Torch7

Top libraries by GitHub issues opened	Top libraries by GitHub stars
#1: 2988   <a href="#">DL4J</a> /deeplearning4j	#1: 29467   <a href="#">tensorflow/tensorflow</a>
#2: 2458   <a href="#">keras</a> /keras	#2: 29456   <a href="#">pytorch/pytorch</a>
#3: 2458   <a href="#">tenserflow/tensorflow</a>	#3: 7595   <a href="#">chainer/chainer</a>
#4: 1867   <a href="#">dl4j</a> /dl4j	#4: 5885   <a href="#">Microsoft/CNTK</a>
#5: 1855   <a href="#">Theano</a> /theano	#5: 5883   <a href="#">keras/keras</a>
#6: 1857   <a href="#">deeplearning4j/deeplearning4j</a>	#6: 5598   <a href="#">mxnet/mxnet</a>
#7: 1855   <a href="#">mxnet/mxnet</a>	#7: 5161   <a href="#">matconvnet/MatConvNet</a>
#8: 1495   <a href="#">MILAN/MLblocks</a>	#8: 4510   <a href="#">Theano/Theano</a>
#9: 458   <a href="#">pfrer/chainer</a>	#9: 3723   <a href="#">deeplearning4j/deeplearning4j</a>
#10: 394   <a href="#">pythontensor/tensorflow</a>	#10: 3676   <a href="#">keras/keras</a>
#11: 394   <a href="#">Lasagne/Lasagne</a>	#11: 3382   <a href="#">amplab/amazon-dsone</a>
#12: 242   <a href="#">tflearn/tflearn</a>	#12: 2372   <a href="#">Lasagne/Lasagne</a>
#13: 234   <a href="#">DeepLearningJL/DeepLearningJL</a>	#13: 2369   <a href="#">DeepLearningJL/DeepLearningJL</a>
#14: 206   <a href="#">tflearn/tflearn</a>	#14: 1577   <a href="#">pfrer/chainer</a>
#15: 82   <a href="#">ID3D4/transformer</a>	#15: 1271   <a href="#">Microsoft/CNTK</a>
#16: 75   <a href="#">Lasagne/Lasagne</a>	#16: 1262   <a href="#">mxnet/mxnet</a>
#17: 39   <a href="#">amplab/amazon-dsone</a>	#17: 879   <a href="#">mlie/mlie/mlie</a>
#18: 27   <a href="#">torcheet/torcheet</a>	#18: 787   <a href="#">torcheet/torcheet</a>

Top libraries by GitHub contributors	Top libraries by GitHub forks
#1: 13   <a href="#">dl4j</a> /deeplearning4j	#1: 1209   <a href="#">tensorflow/tensorflow</a>
#2: 244   <a href="#">Theano/Theano</a>	#2: 7394   <a href="#">DL4J/dl4j</a>
#3: 234   <a href="#">chainer/chainer</a>	#3: 2275   <a href="#">chainer/chainer</a>
#4: 234   <a href="#">DeepLearningJL/DeepLearningJL</a>	#4: 2266   <a href="#">keras/keras</a>
#5: 169   <a href="#">dl4j/dl4j</a>	#5: 1548   <a href="#">Theano/Theano</a>
#6: 162   <a href="#">torcheet/torcheet</a>	#6: 1484   <a href="#">keras/keras</a>
#7: 142   <a href="#">deeplearning4j/deeplearning4j</a>	#7: 1363   <a href="#">Microsoft/CNTK</a>
#8: 73   <a href="#">Microsoft/CNTK</a>	#8: 1264   <a href="#">deeplearning4j/deeplearning4j</a>
#9: 50   <a href="#">pfrer/chainer</a>	#9: 1081   <a href="#">keras/keras</a>
#10: 38   <a href="#">Lasagne/Lasagne</a>	#10: 1043   <a href="#">keras/keras</a>
#11: 40   <a href="#">mlie/mlie/mlie</a>	#11: 482   <a href="#">amplab/amazon-dsone</a>
#12: 38   <a href="#">DeepLearningJL/DeepLearningJL</a>	#12: 479   <a href="#">keras/keras</a>
#13: 38   <a href="#">tflearn/tflearn</a>	#13: 412   <a href="#">Microsoft/CNTK</a>
#14: 28   <a href="#">ID3D4/transformer</a>	#14: 377   <a href="#">pfrer/chainer</a>
#15: 26   <a href="#">amplab/amazon-dsone</a>	#15: 359   <a href="#">keras/keras</a>
#16: 15   <a href="#">ID3D4/transformer</a>	#16: 287   <a href="#">mlie/mlie/mlie</a>
#17: 14   <a href="#">keras/keras</a>	#17: 185   <a href="#">torcheet/torcheet</a>
#18: 14   <a href="#">torcheet/torcheet</a>	#18: 144   <a href="#">ID3D4/transformer</a>

# Estado del arte

- Principales **avances e investigaciones** en *deep learning*:
  - Desarrollo del **coche autónomo**.

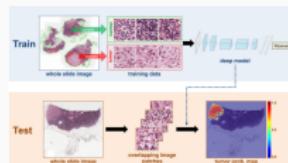


# Estado del arte

- Principales **avances e investigaciones** en *deep learning*:
  - Desarrollo del **coche autónomo**.



- **Análisis médico** de imágenes.

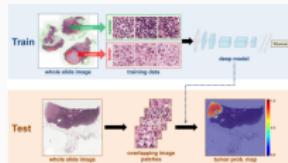


# Estado del arte

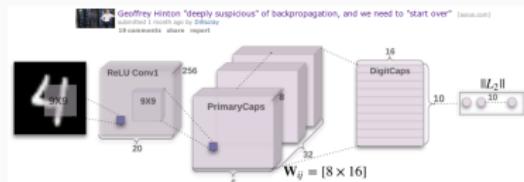
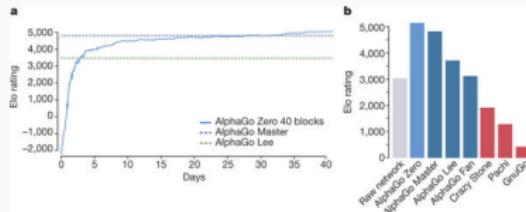
- Principales avances e investigaciones en *deep learning*:
  - Desarrollo del *coche autónomo*.



- Análisis médico de imágenes.



- Optar por aprendizaje no supervisado y por refuerzo y mejora de los algoritmos actuales.



## Ejemplos de aplicación

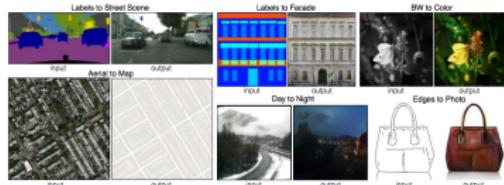
---

# Ejemplos de aplicación

- Nos centraremos en el **procesamiento de imágenes**, aunque cuenta con **multitud de campos de aplicación**:

General use case	Industry
<b>Sound</b>	
Voice recognition	UX/UI, Automotive, Security, IoT
Voice search	Handset maker, Telecoms
Sentiment analysis	CRM
Flaw detection (engine noise)	Automotive, Aviation
Fraud detection (latent audio artifacts)	Finance, Credit Cards
<b>Time Series</b>	
Log analysis/Risk detection	Data centers, Security, Finance
Enterprise resource planning	Manufacturing, Auto., Supply chain
Predictive analysis using sensor data	IoT, Smart home, Hardware manufact.
Business and Economic analytics	Finance, Accounting, Government
Recommendation engine	E-commerce, Media, Social Networks
<b>Text</b>	
Sentiment Analysis	CRM, Social media, Reputation mgt.
Augmented search, Theme detection	Finance
Threat detection	Social media, Govt.
Fraud detection	Insurance, Finance
<b>Image</b>	
Facial recognition	
Image search	Social media
Machine vision	Automotive, aviation
Photo clustering	Telecom, Handset makers
<b>Video</b>	
Motion detection	Gaming, UX, UI
Real-time threat detection	Security, Airports

# Ejemplos de aplicación



Human captions from the training set



Automatically captioned



This bird has a yellow belly and tarsus, grey back, wings, and brown throat, nape with a black face



This bird is white with some black on its head and wings, and has a long orange beak



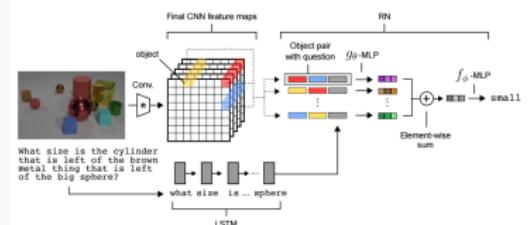
This flower has overlapping pink pointed petals surrounding a ring of short yellow filaments



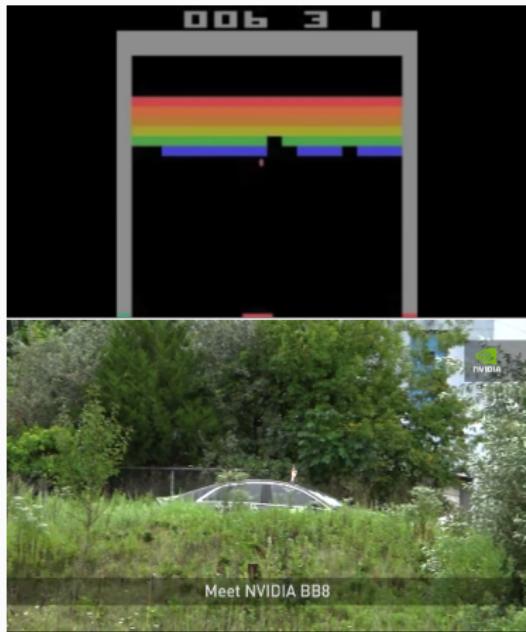
(a) Stage-I images

(b)

Stage-II images



# Ejemplos de aplicación



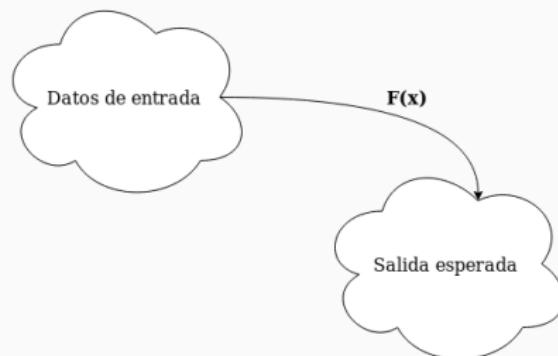
# Fundamentos del deep learning

---

# Fundamentos del deep learning

- Principal objetivo:

*Encontrar la transformación que hay que aplicarle a los datos de entrada para obtener la salida esperada.*



# Fundamentos del deep learning

- Transformación que hay que aplicar para obtener el **doble de un número**:

# Fundamentos del deep learning

- Transformación que hay que aplicar para obtener el **doble de un número**:

$$F(x) = 2 \cdot x$$

# Fundamentos del deep learning

- Transformación que hay que aplicar para obtener el **doble de un número**:

$$F(x) = 2 \cdot x$$

- Transformación que hay que aplicar para obtener el **cuadrado de un número**:

# Fundamentos del deep learning

- Transformación que hay que aplicar para obtener el **doble de un número**:

$$F(x) = 2 \cdot x$$

- Transformación que hay que aplicar para obtener el **cuadrado de un número**:

$$F(x) = x \cdot x$$

# Fundamentos del deep learning

- Transformación que hay que aplicar para obtener el **doble de un número**:

$$F(x) = 2 \cdot x$$

- Transformación que hay que aplicar para obtener el **cuadrado de un número**:

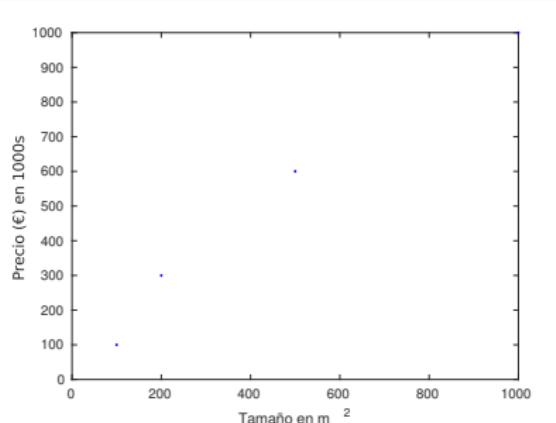
$$F(x) = x \cdot x$$

- Transformación que hay que aplicar para obtener si **hay un gato (1)** o **no (0)** en una imagen:

$$F(\text{gato}) = ??$$

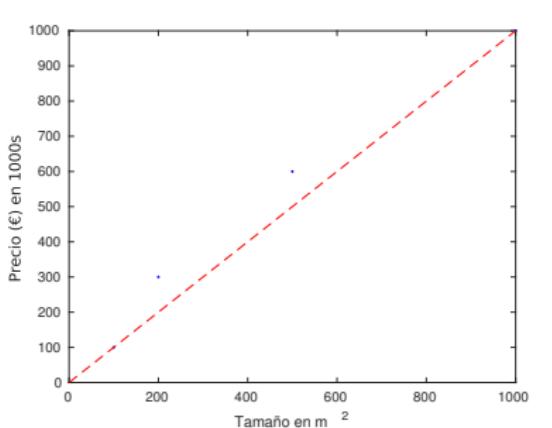
# Fundamentos del deep learning

- Un problema **más sencillo** de abordar:



# Fundamentos del deep learning

- Un problema **más sencillo** de abordar:

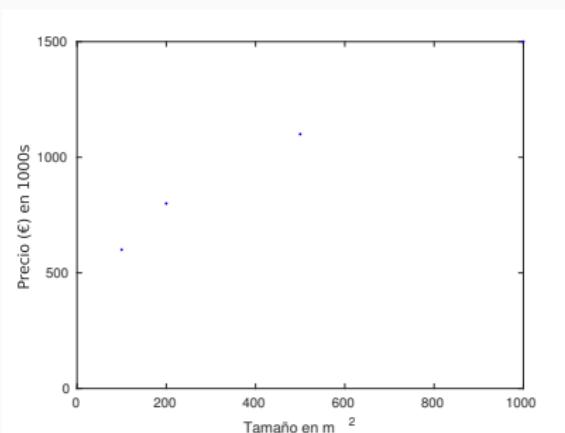


- Podemos usar **la función más básica** para resolver el problema:

$$F(x) = a \cdot x \quad (\text{p. ej. } a = 1)$$

# Fundamentos del deep learning

- Un problema **más sencillo** de abordar:



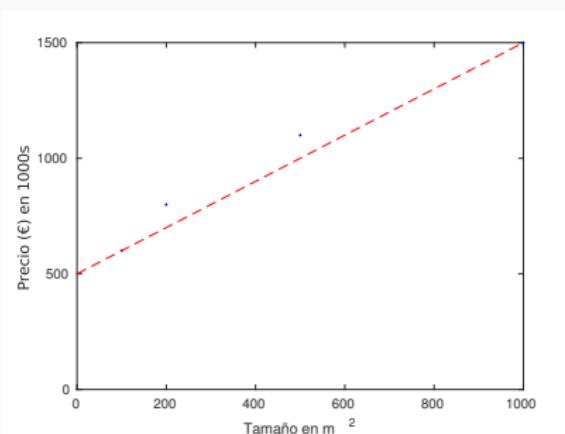
- Podemos usar **la función más básica** para resolver el problema:

$$F(x) = a \cdot x \quad (\text{p. ej. } a = 1)$$

- **Hay un problema:** debemos pasar por el origen.

# Fundamentos del deep learning

- Un problema **más sencillo** de abordar:



- Podemos usar **la función más básica** para resolver el problema:

$$F(x) = a \cdot x \quad (\text{p. ej. } a = 1)$$

- **Hay un problema:** debemos pasar por el origen.

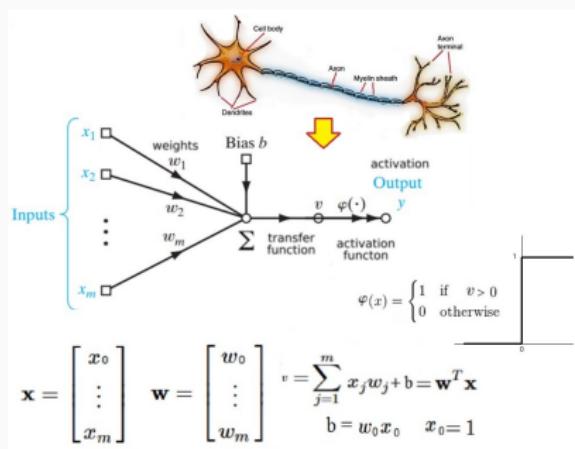
- **Solución:** añadir un término independiente.

$$F(x) = a \cdot x + b \quad (\text{p. ej. } a = 1 \text{ y } b = 500)$$

# Fundamentos del deep learning

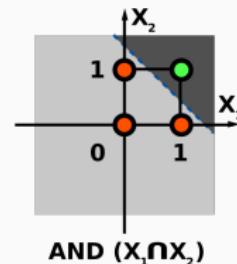
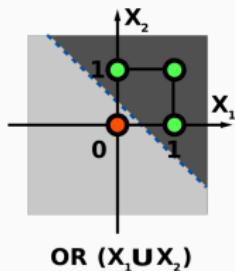
- El **perceptrón** es la unidad básica de las **redes neuronales**.

Inspirado en el funcionamiento de las neuronas biológicas, nos permite encontrar los valores de  $a$  y  $b$  que minimizan el error cometido por nuestro modelo.



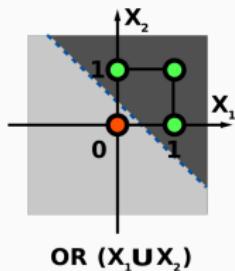
# Fundamentos del deep learning

- El **perceptrón** nos permite solucionar cualquier problema que sea **separable linealmente**:

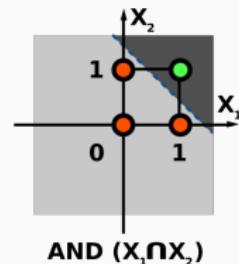


# Fundamentos del deep learning

- El **perceptrón** nos permite solucionar cualquier problema que sea **separable linealmente**:

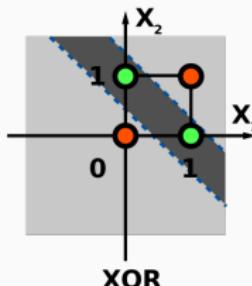


OR ( $X_1 \cup X_2$ )



AND ( $X_1 \cap X_2$ )

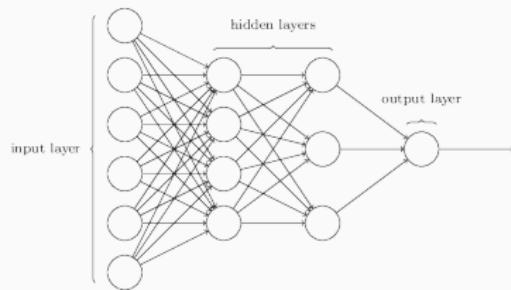
- Hay un problema: ¿Qué pasa si nuestros datos **no son separables linealmente**?



XOR

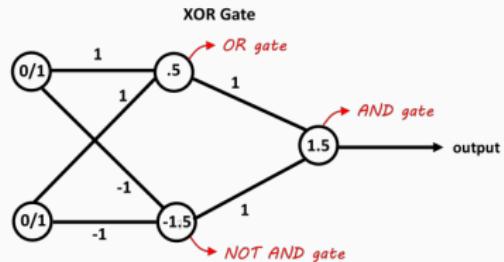
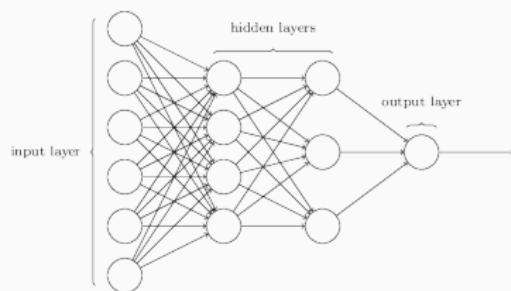
# Fundamentos del deep learning

- Solución: combinar **perceptrones** para poder combinar funciones y resolver el problema. Es decir, crear un **perceptrón multicapa** o **red neuronal artificial** (RNA).



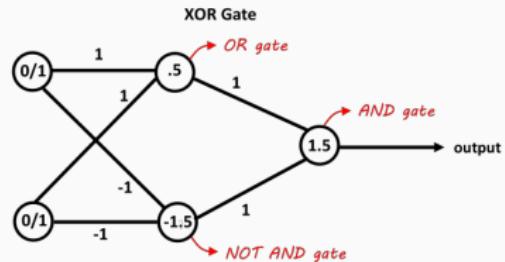
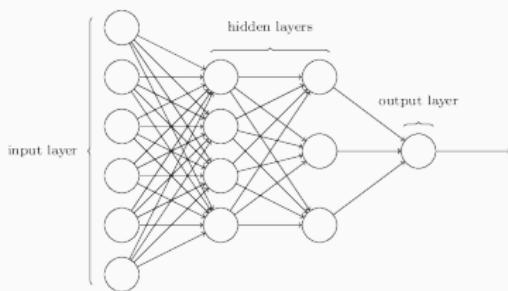
# Fundamentos del deep learning

- Solución: combinar **perceptrones** para poder combinar funciones y resolver el problema. Es decir, crear un **perceptrón multicapa** o **red neuronal artificial** (RNA).



# Fundamentos del deep learning

- Solución: combinar **perceptrones** para poder combinar funciones y resolver el problema. Es decir, crear un **perceptrón multicapa** o **red neuronal artificial** (RNA).

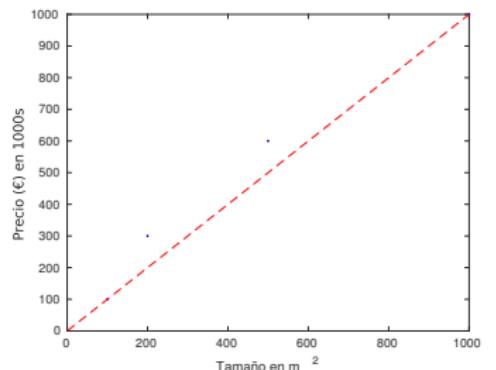


- Una **red neuronal** es un **aproximador universal de funciones**.

*jLo que andábamos buscando!*

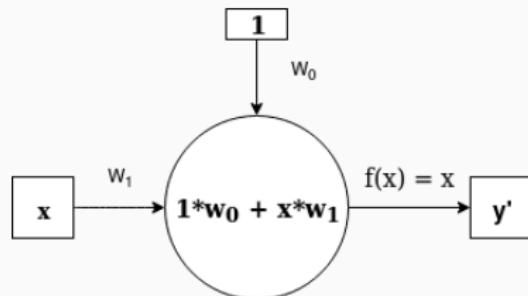
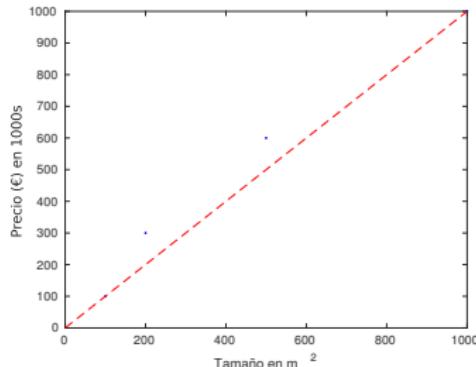
# Fundamentos del deep learning

- ¿Cómo hacer que nuestra red neuronal encuentre (aprenda) los mejores valores para todos sus parámetros (pesos)?



# Fundamentos del deep learning

- ¿Cómo hacer que nuestra red neuronal encuentre (aprenda) los mejores valores para todos sus parámetros (pesos)?

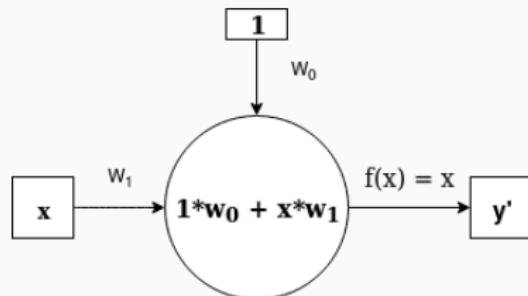
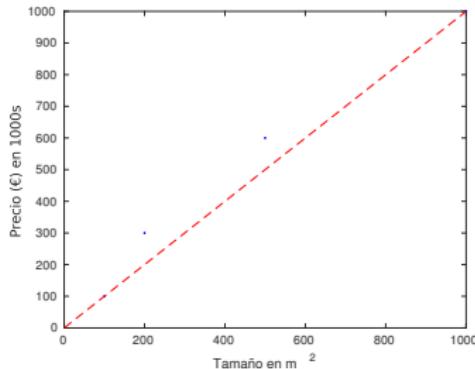


- Datos del problema:

$$x = \{100, 200, 500, 1000\}, y = \{100, 300, 600, 1000\}$$

# Fundamentos del deep learning

- ¿Cómo hacer que nuestra red neuronal encuentre (aprenda) los mejores valores para todos sus parámetros (pesos)?



- Datos del problema:  
 $x = \{100, 200, 500, 1000\}$ ,  $y = \{100, 300, 600, 1000\}$
- Solución parcial: darle valores aleatorios a todos los parámetros:

$$w_0 = 0, w_1 = 0,9 \implies Y' = \{90, 180, 450, 900\}$$

# Fundamentos del deep learning

- ¿Hay alguna forma de mejorar lo conseguido con el paso anterior?  $\equiv$  ¿Existen otros valores para los parámetros que cometan menos error?

# Fundamentos del deep learning

- ¿Hay alguna forma de mejorar lo conseguido con el paso anterior?  $\equiv$  ¿Existen otros valores para los parámetros que cometan menos error?
  - Necesitamos una medida del error. El error cuadrático medio p. ej.:

$$ECM = \frac{1}{n} \sum_{i=1}^n (Y'_i - Y_i)^2$$

# Fundamentos del deep learning

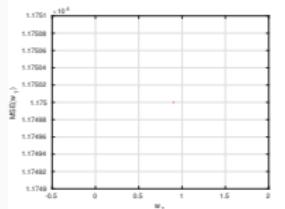
- ¿Hay alguna forma de mejorar lo conseguido con el paso anterior?  $\equiv$  ¿Existen otros valores para los parámetros que cometan menos error?
  - Necesitamos una medida del error. El error cuadrático medio p. ej.:  
$$ECM = \frac{1}{n} \sum_{i=1}^n (Y'_i - Y_i)^2$$

- Para nuestro problema tenemos:

$$w_0 = 0, w_1 = 0,9 \implies ECM = 11750$$

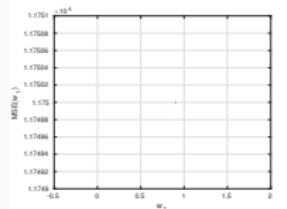
# Fundamentos del deep learning

- Viendo el error que comete, ¿Hay alguna forma de cambiar el valor de los parámetros para que este sea menor?

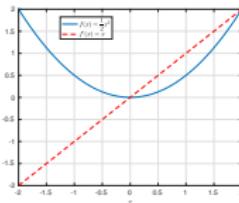


# Fundamentos del deep learning

- Viendo el error que comete, ¿Hay alguna forma de cambiar el valor de los parámetros para que este sea menor?



- **Clave:** usar la derivada.



- Para más de una dimensión utilizamos el **gradiente**:

$$W = [w_0, \dots, w_n] \quad \nabla MSE(W) = \left[ \frac{\partial}{\partial W_1} MSE(W), \dots, \frac{\partial}{\partial W_n} MSE(W) \right]$$

# Fundamentos del deep learning

- Mediante el uso del gradiente, el **algoritmo de descenso por gradiente** realiza la actualización de los parámetros:

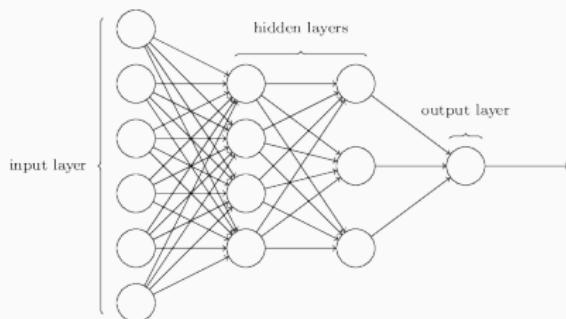
$$W' = W - \alpha \nabla_W MSE(W)$$

# Fundamentos del deep learning

- Mediante el uso del gradiente, el **algoritmo de descenso por gradiente** realiza la actualización de los parámetros:

$$W' = W - \alpha \nabla_W MSE(W)$$

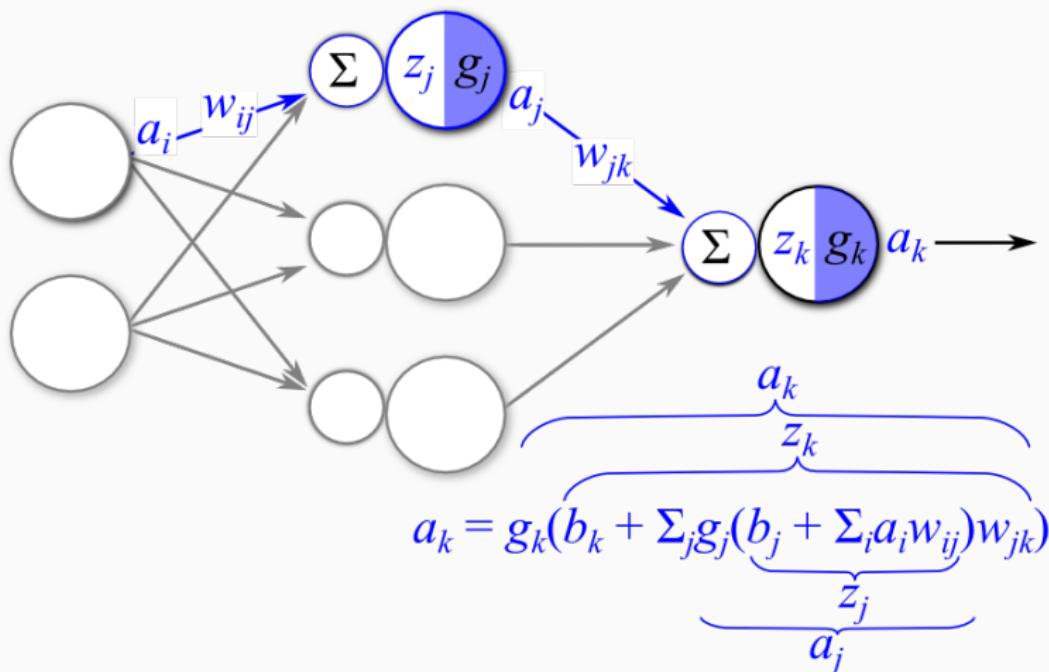
- Hemos podido calcular el error a la salida puesto que disponíamos de la **salida esperada**. ¿Qué ocurre si estamos utilizando una red neuronal con más de una capa? ¿Cuál es el error en la salida de las **unidades ocultas**?



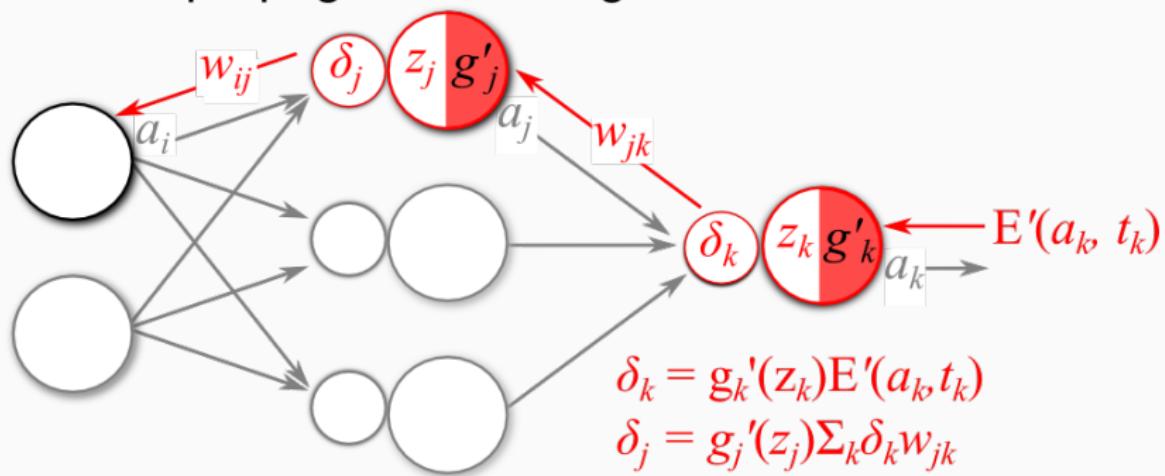
# Fundamentos del deep learning

- Como solución a este problema surge el algoritmo *backpropagation*. Mediante este algoritmo, propagamos el gradiente del error en la salida hacia atrás, para poder calcular el gradiente en las capas ocultas.
- La idea intuitiva es que cada unidad es responsable del error que tienen cada una de las unidades a las que envía su salida y lo es en la medida que marca el peso de la conexión entre ellas.

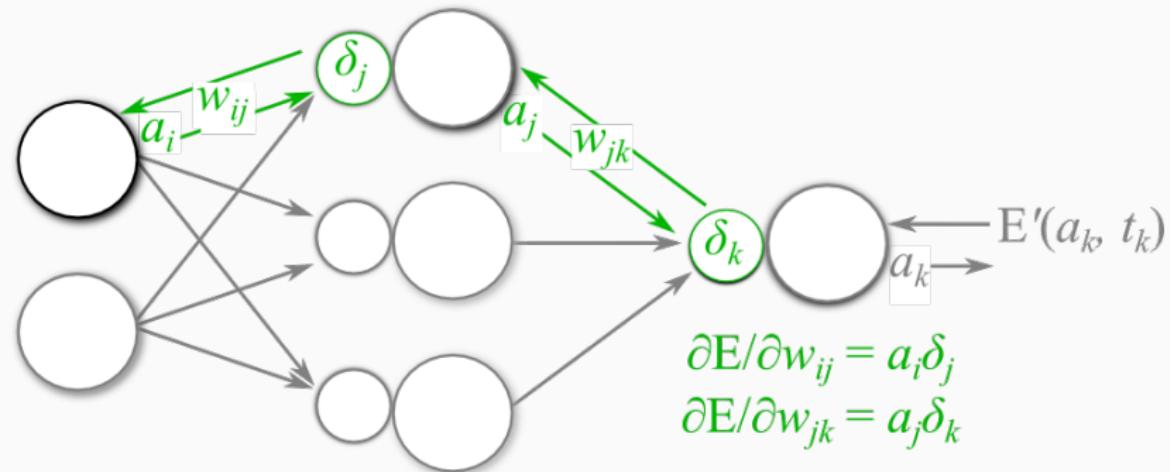
## I. Forward-propagate Input Signal



## II. Back-propagate Error Signals



## III. Calculate Parameter Gradients



## IV. Update Parameters

$$w_{ij} = w_{ij} - \eta(\frac{\partial E}{\partial w_{ij}})$$

$$w_{jk} = w_{jk} - \eta(\frac{\partial E}{\partial w_{jk}})$$

for learning rate  $\eta$

# Fundamentos del deep learning

- **Conclusión:** si disponemos de un conjunto de datos (**lo suficientemente representativo del problema**) podemos utilizar una **red neuronal** como un **aproximador universal de funciones** y, mediante el **descenso por gradiente** y el ***backpropagation***, actualizar sus parámetros para **encontrar la transformación** que hay que realizar a los datos para resolver nuestro problema.

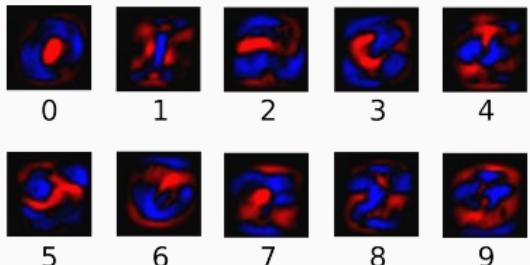
# Fundamentos del deep learning

- **Conclusión:** si disponemos de un conjunto de datos (**lo suficientemente representativo del problema**) podemos utilizar una **red neuronal** como un **aproximador universal de funciones** y, mediante el **descenso por gradiente** y el **backpropagation**, actualizar sus parámetros para **encontrar la transformación** que hay que realizar a los datos para resolver nuestro problema.
- Ejemplo de la **limitación** de las redes neuronales:



# Fundamentos del deep learning

- **Conclusión:** si disponemos de un conjunto de datos (**lo suficientemente representativo del problema**) podemos utilizar una **red neuronal** como un **aproximador universal de funciones** y, mediante el **descenso por gradiente** y el ***backpropagation***, actualizar sus parámetros para **encontrar la transformación** que hay que realizar a los datos para resolver nuestro problema.
- Ejemplo de la **limitación** de las redes neuronales:

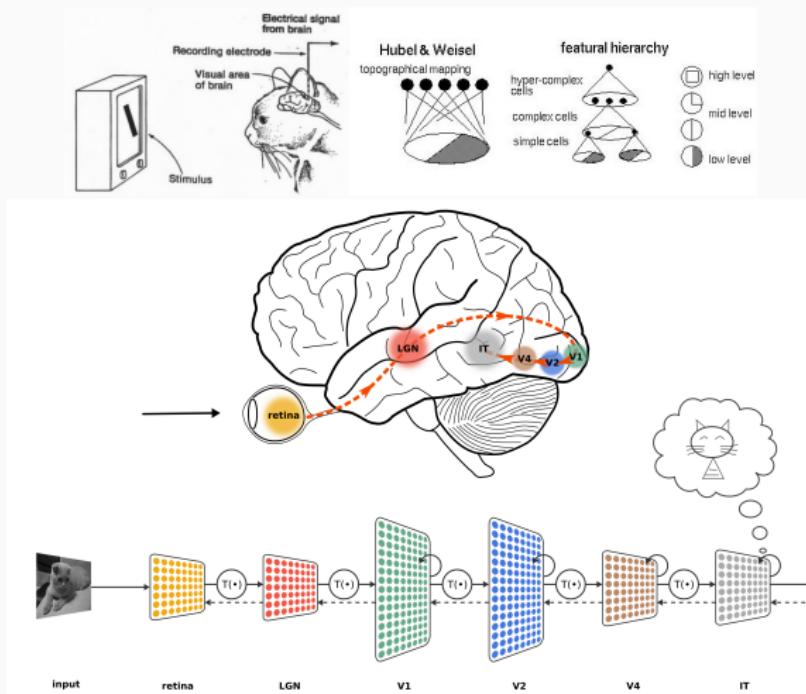


## **Algoritmo de deep learning: redes neuronales convolucionales**

---

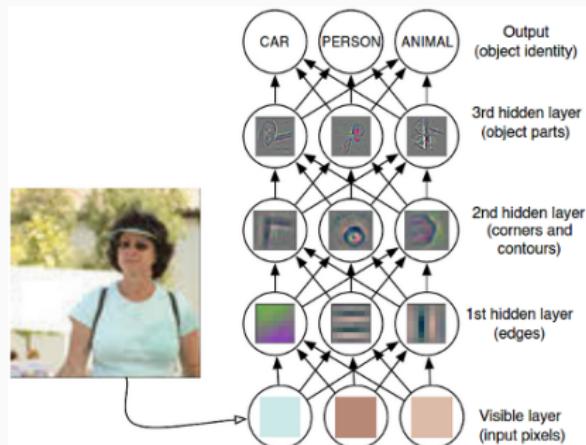
# Algoritmo de deep learning: redes neuronales convolucionales

- **Inspiradas** en nuestro sistema de visión.



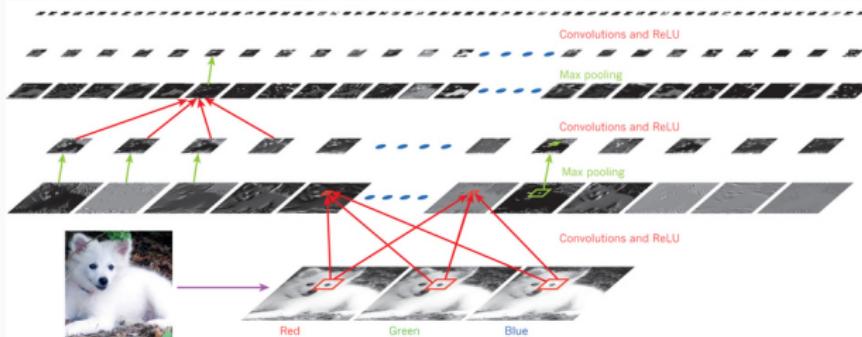
# Algoritmo de deep learning: redes neuronales convolucionales

- Aprenden una representación jerárquica de los datos e identifican patrones.



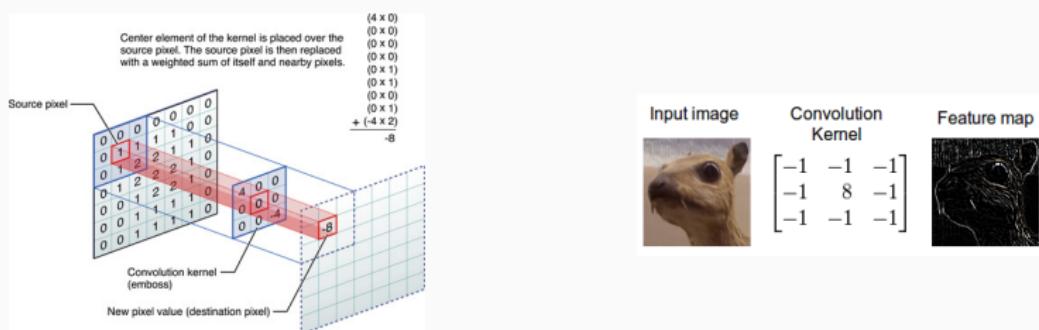
# Algoritmo de deep learning: redes neuronales convolucionales

- La representación jerárquica se consigue gracias a operaciones de **convolución, activación y submuestreo**.



# Algoritmo de deep learning: redes neuronales convolucionales

- La operación de **convolución** permite extraer características de la imagen.
- La red debe **aprender** que *kernel* de convolución le interesa aplicar.



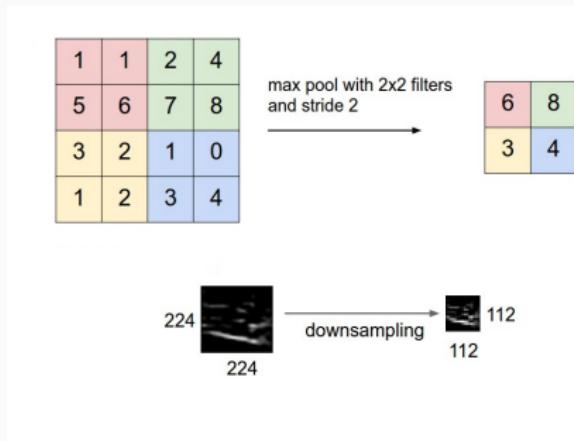
# Algoritmo de deep learning: redes neuronales convolucionales

- La operación de **activación** permite introducir **no-linealidades** en los datos.
- Se aplica, a **nivel de pixel**, alguna **función no lineal**:

Name	Plot	Equation	Derivative (with respect to x)
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a. Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
Rectified linear unit (ReLU) <sup>[9]</sup>		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$

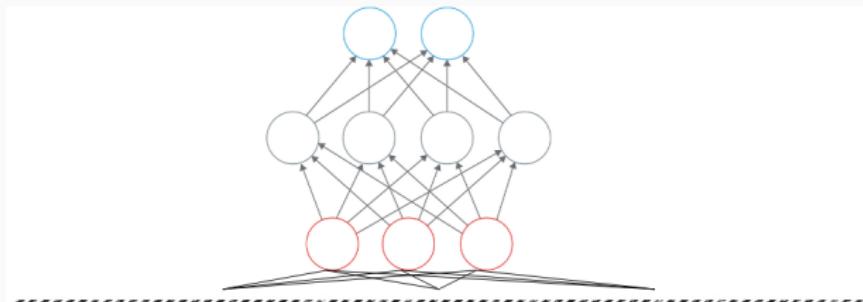
# Algoritmo de deep learning: redes neuronales convolucionales

- La operación de **submuestreo** o *pooling* consiste en realizar un **resumen** de los datos de entrada, **reduciendo** el tamaño de estos. Lo que se traduce en un **mayor rendimiento** de la red y una mayor **robustez** en la clasificación.
- La operación más usual es tomar el **máximo de una región**, aunque también se puede aplicar la media u otra operación.



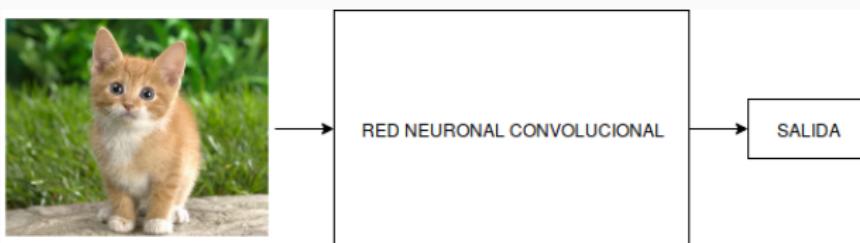
# Algoritmo de deep learning: redes neuronales convolucionales

- La **identificación de patrones** se realiza con una **red neuronal** sobre las **características de más alto nivel** de la **representación jerárquica**.



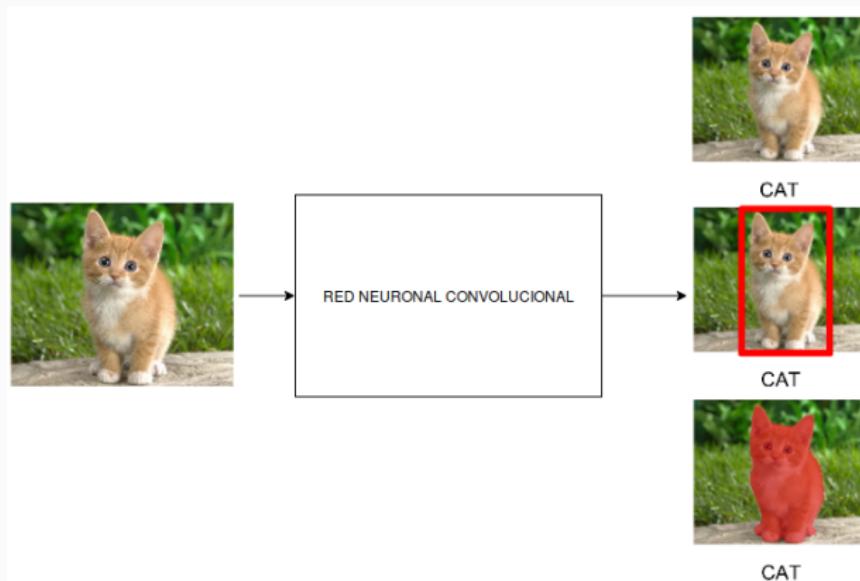
# Algoritmo de deep learning: redes neuronales convolucionales

- Estructura general de una **red neuronal convolucional**:



# Algoritmo de deep learning: redes neuronales convolucionales

- Según el tipo de **salida** de la red estaremos ante un problema de **clasiﬁcación**, **detección** o **segmentación**:



# Algoritmo de deep learning: redes neuronales convolucionales

- Redes neuronales convolucionales aplicadas a la clasificación de imágenes.
  - Entrada de la red: imágenes.

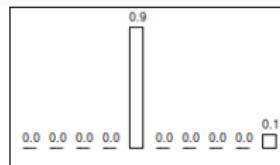
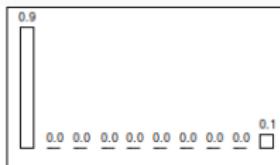
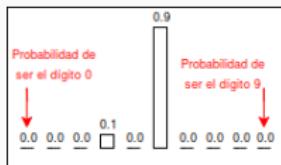


# Algoritmo de deep learning: redes neuronales convolucionales

- Redes neuronales convolucionales aplicadas a la clasificación de imágenes.
  - Entrada de la red: imágenes.



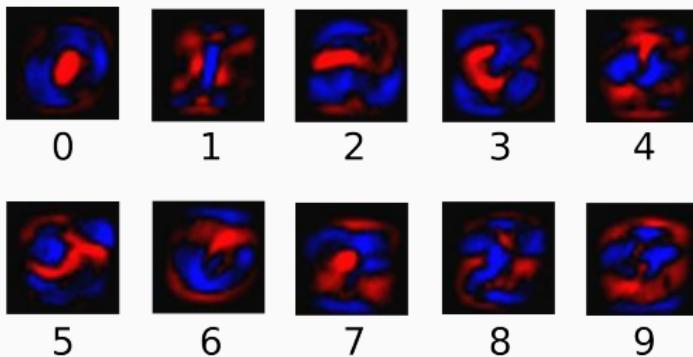
- Salida de la red: distribución de probabilidad.



# Algoritmo de deep learning: redes neuronales convolucionales

- Para obtener la distribución de probabilidad de la salida de la red neuronal se usa la capa **softmax**:
  - Utiliza la **evidencia** de que la entrada pertenezca a alguna de las clases, es decir, **la salida de la red neuronal**:

$$\text{evidence}_i = \sum_j W_{i,j} \cdot x_j + b_i$$



- Esta **evidencia** es convertida en **probabilidad** gracias a la operación que realiza la capa **softmax**:

$$y = \text{softmax}(\text{evidence}), \quad \text{softmax}(x)_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

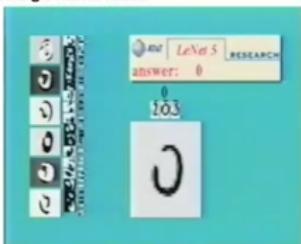
## Caso práctico: LeNet5

---

# Caso práctico: LeNet5

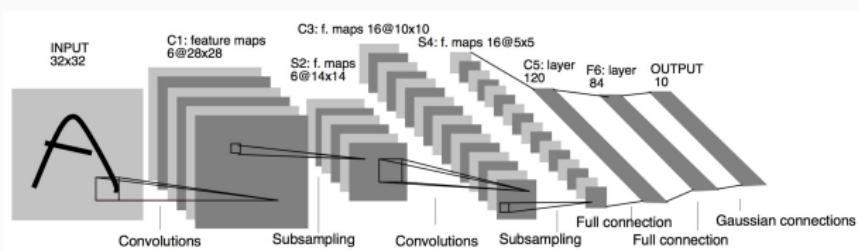
- Primera **red neuronal convolucional** de la historia, desarrollada por **Yann LeCun** y su equipo.
- Se aplicó al **reconocimiento de dígitos escritos a mano**.

Handwritten digit classification



[Courtesy of Yann LeCun]

Andrea Neri



## Caso práctico: LeNet5

- La red fue entrenada con el conjunto de datos conocido como **MNIST**.
- Cuenta con **60000** imágenes de entrenamiento y **10000** imágenes para test.
- Todas las imágenes están en **escala de grises** y con un tamaño de  **$28 \times 28$** .



## Caso práctico: LeNet5

- Intentaremos replicar la red diseñada por Yann LeCun para el dataset *MNIST*.
- Para ello vamos a usar *Tensorflow* y *Jupyter Notebook*:

```
pip install tensorflow  
pip install jupyter
```



**¡Gracias por tu atención!**  
**¿Alguna pregunta?**

isaac.perez@alu.uhu.es