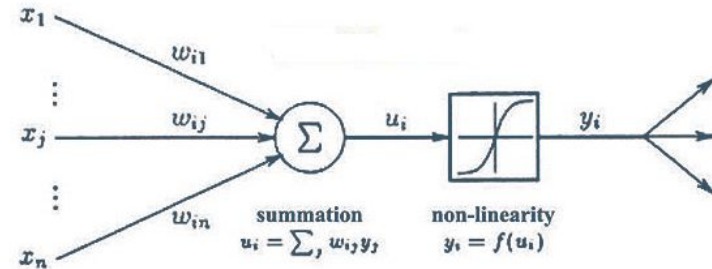
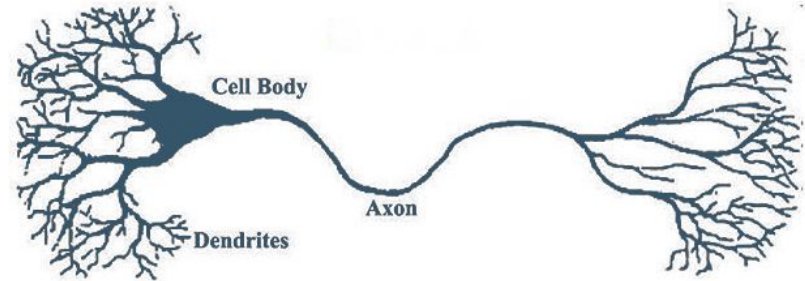
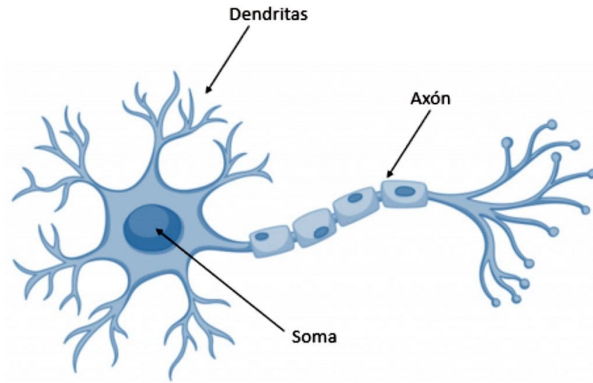




# Sesión 7: Introducción a las Redes Neuronales.

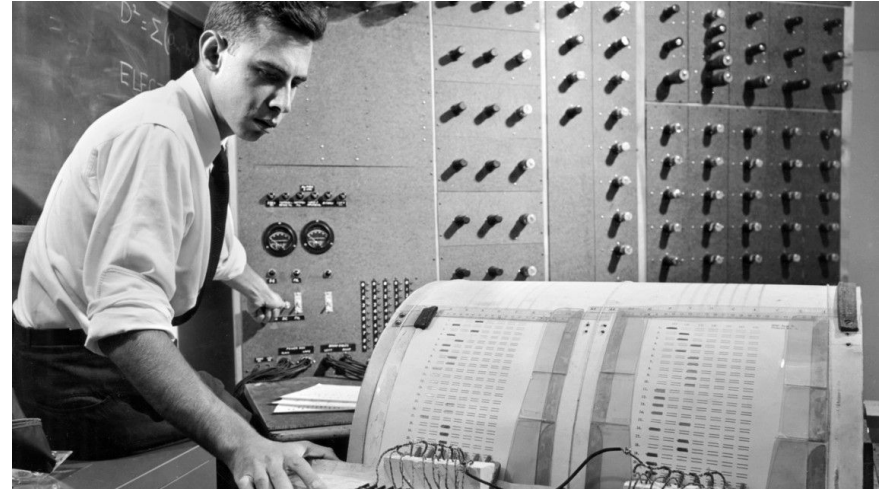


# Historia

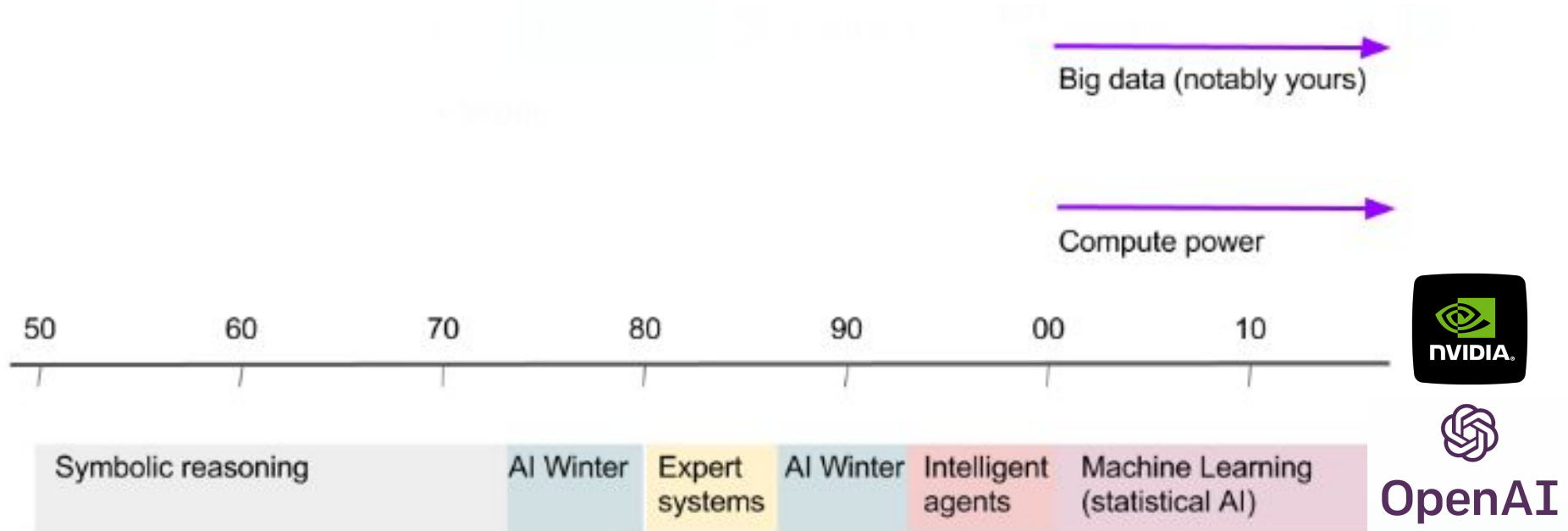
Las redes neuronales datan desde el año 1958 con Frank Rosenblatt quien es el creador del método del **Perceptrón**.

En la década del 70 la inteligencia artificial fue punto de críticas y problemas financieros. Las expectativas eran altas, pero los resultados fueron lamentables y los investigadores perdieron el financiamiento. Todos esto sucede porque los **modelos en ese entonces eran limitados**.

En los 80 se creía que se debía implementar reglas para que se tomara el pensamiento humano y para ello se implementa los “if then else”, pero para casos muy complejos estos statements eran demasiados.



# Historia







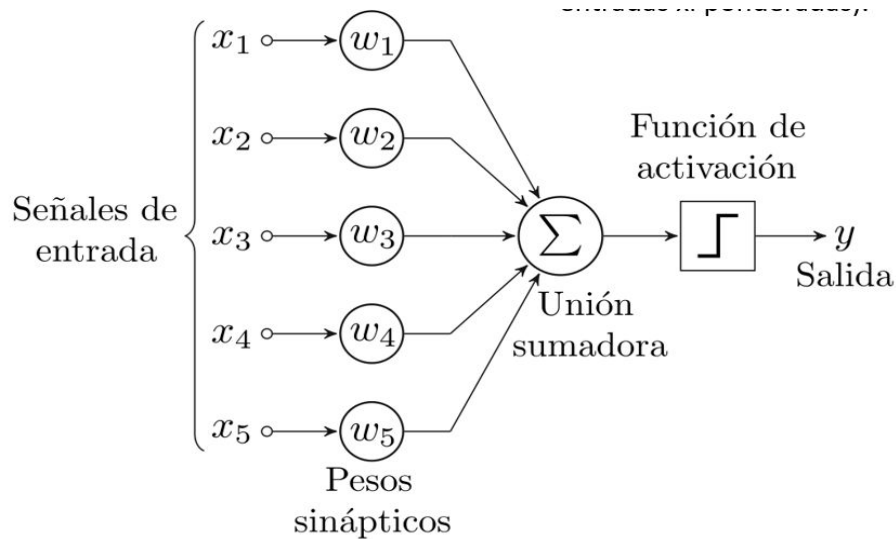




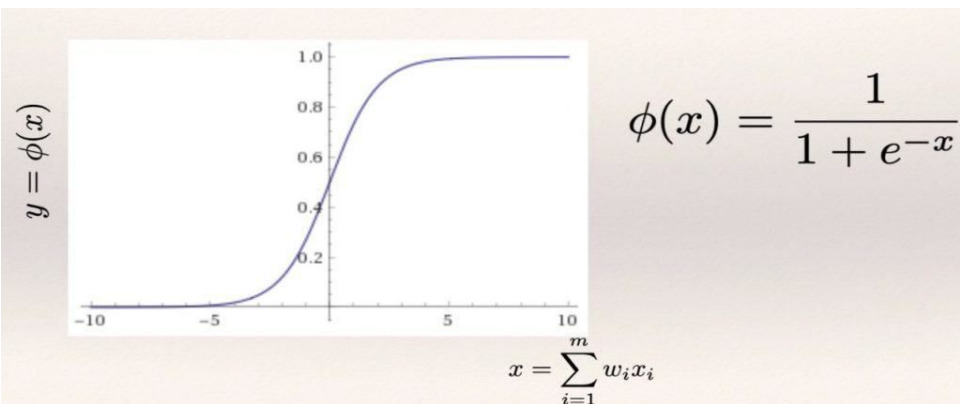
## Librerías



# Perceptron




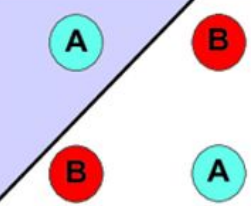
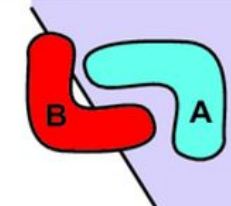
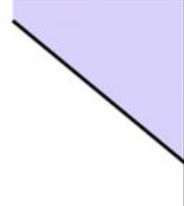
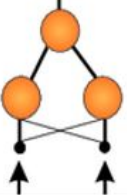
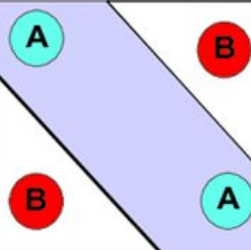
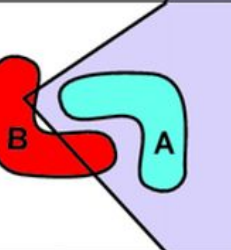
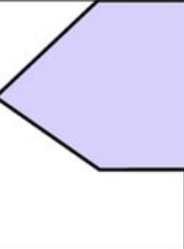
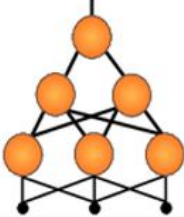
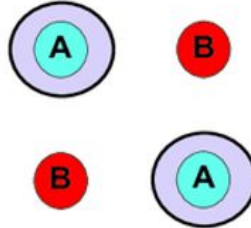
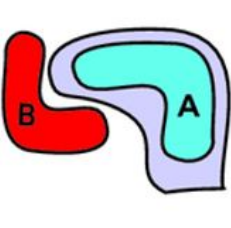

$$y = \sum_{i=1}^n w_i x_i + b$$



Es una idea de neurona muy básica que consta de una capa de **input (x)**, una capa de **pesos (w)** y la capa de **outputs**. Usualmente tiene una función de activación antes del output.



# Ejemplo

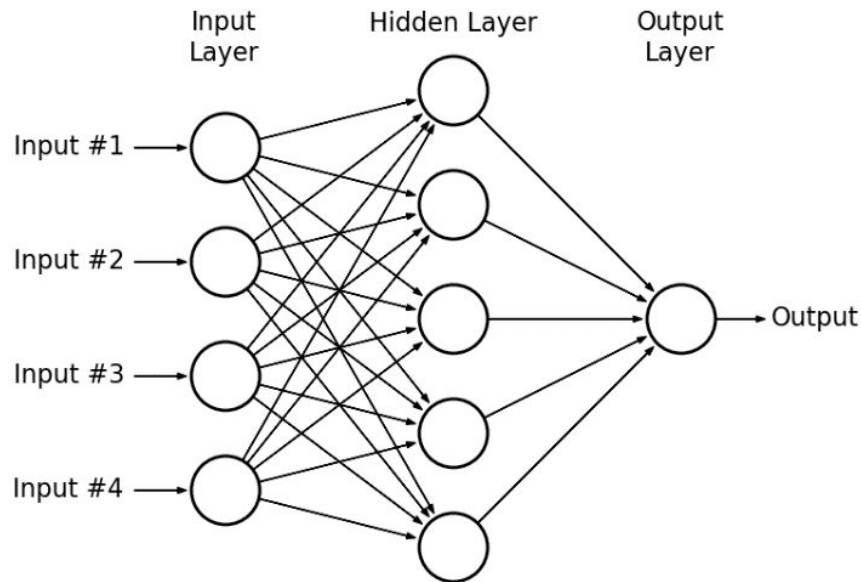
Estructura	Regiones de Decisión	Problemas de la XOR	Clases con Regiones Mezcladas	Formas de Regiones más Generales
1 Capas 	Medio plano limitado por un hiperplano			
2 Capas 	Regiones cerradas o Convexas			
3 Capas 	Complejidad arbitraria limitada por el número de neuronas			

# Multi-Layer Perceptron

**Multi-Layer Perceptron:** Es una red de perceptrones LTU que consta de diferentes capas donde cada una de ellas tiene un nodo como bias.

## Proceso:

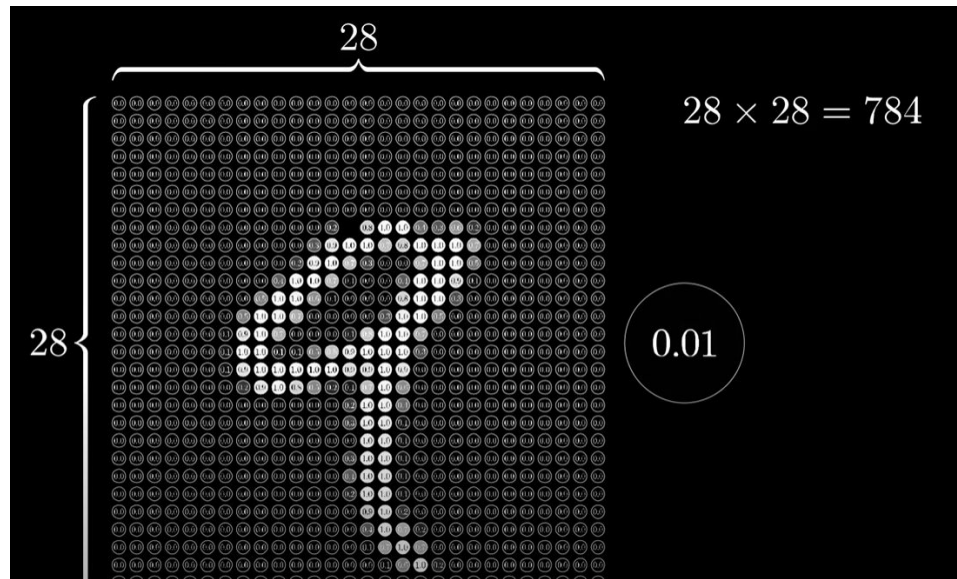
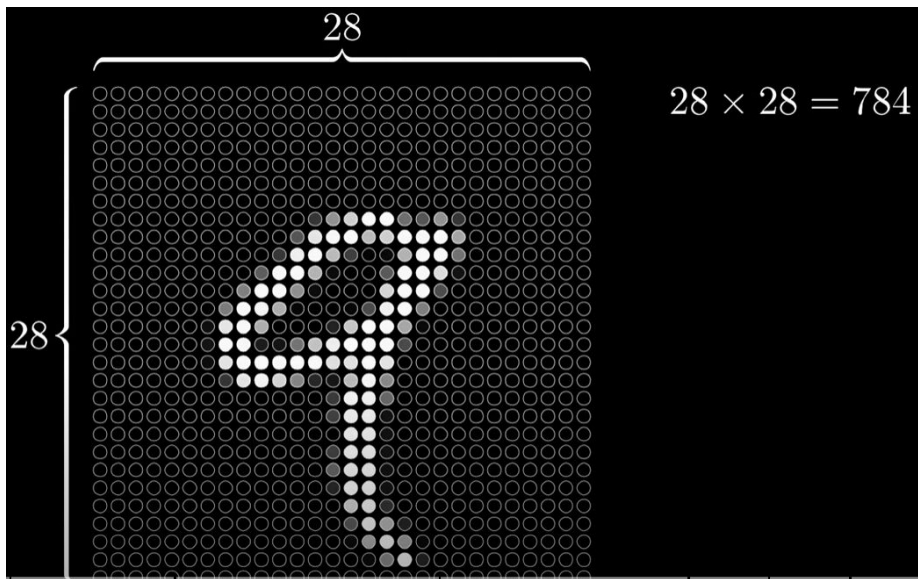
Se alimentan los datos de entrada en la capa de entrada y se toma la salida de la capa de salida. Podemos **aumentar el número de Layers en la capa oculta tanto como queramos**, para hacer que el modelo sea más complejo de acuerdo con nuestra tarea.



## Nota:

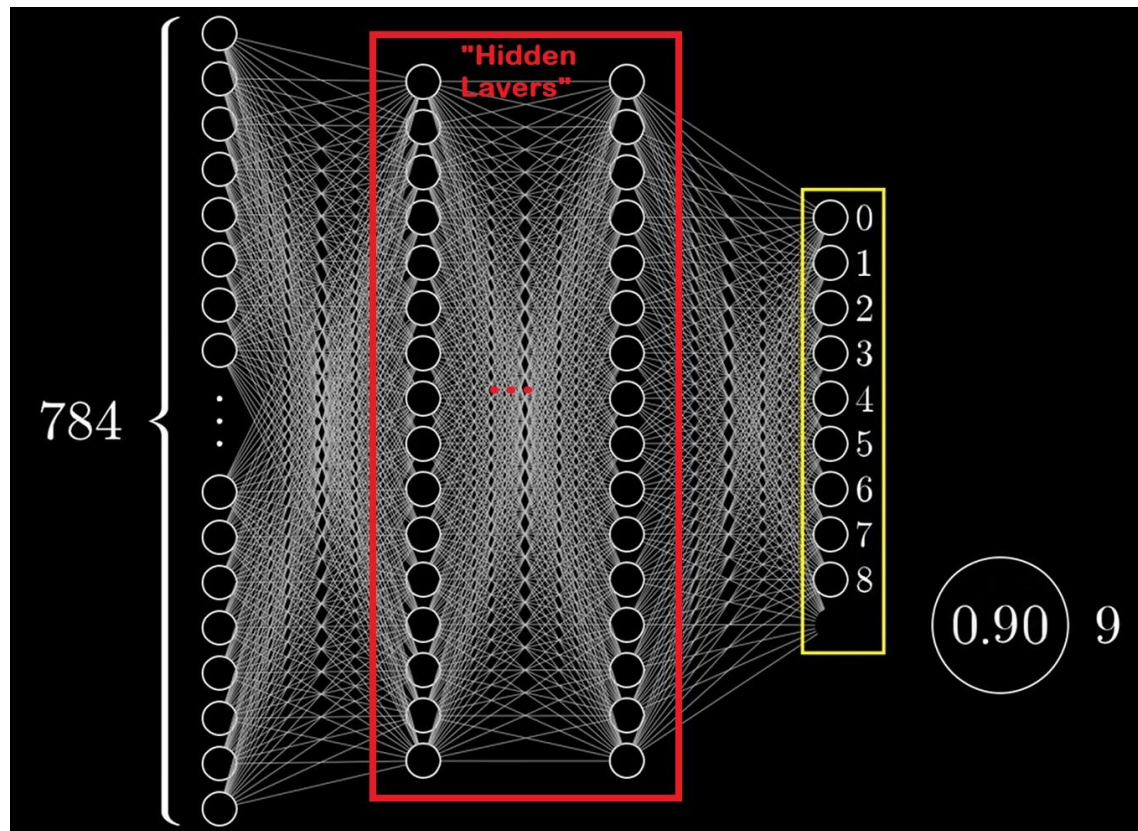
Hasta ahora se está viendo una relación lineal dentro de los perceptrones. Eso quiere decir que son imposibles de ver patrones complejos (así como la regresión logística).

# ¿Cómo procesamos imágenes con una Red Neuronal?



## Redes neuronales profundas (DNN)

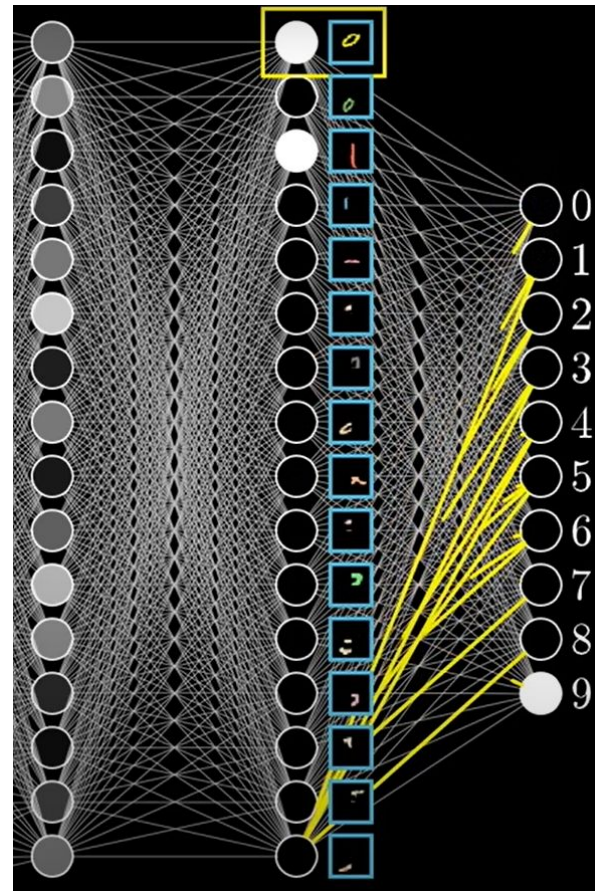
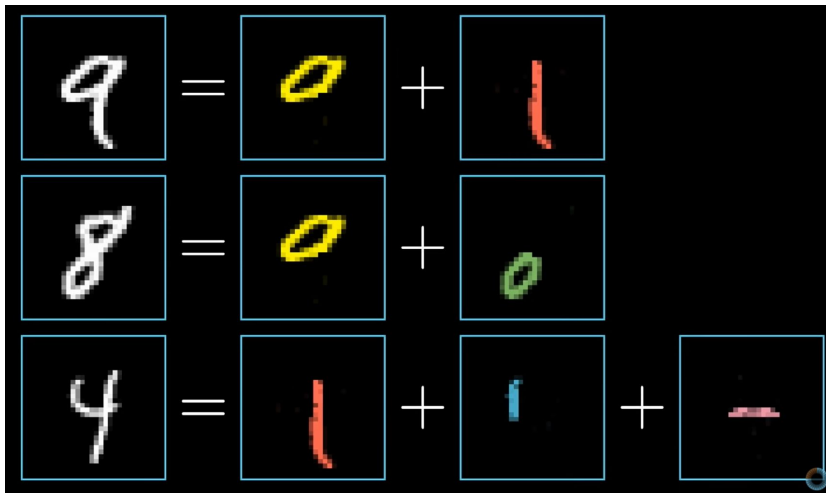
Las redes neuronales pueden tener diferentes formas que se definen en base a las capas que mantienen. De qué tipo, cuántas neuronas y qué funciones estaremos implementando.





Idealmente, cada capa debería reconocer tipos de trazos. Para luego responder a la pregunta si esa suma de trazos corresponde los dígitos en sí.

Un dígito (o una imagen) al final es la suma de varios trazos.

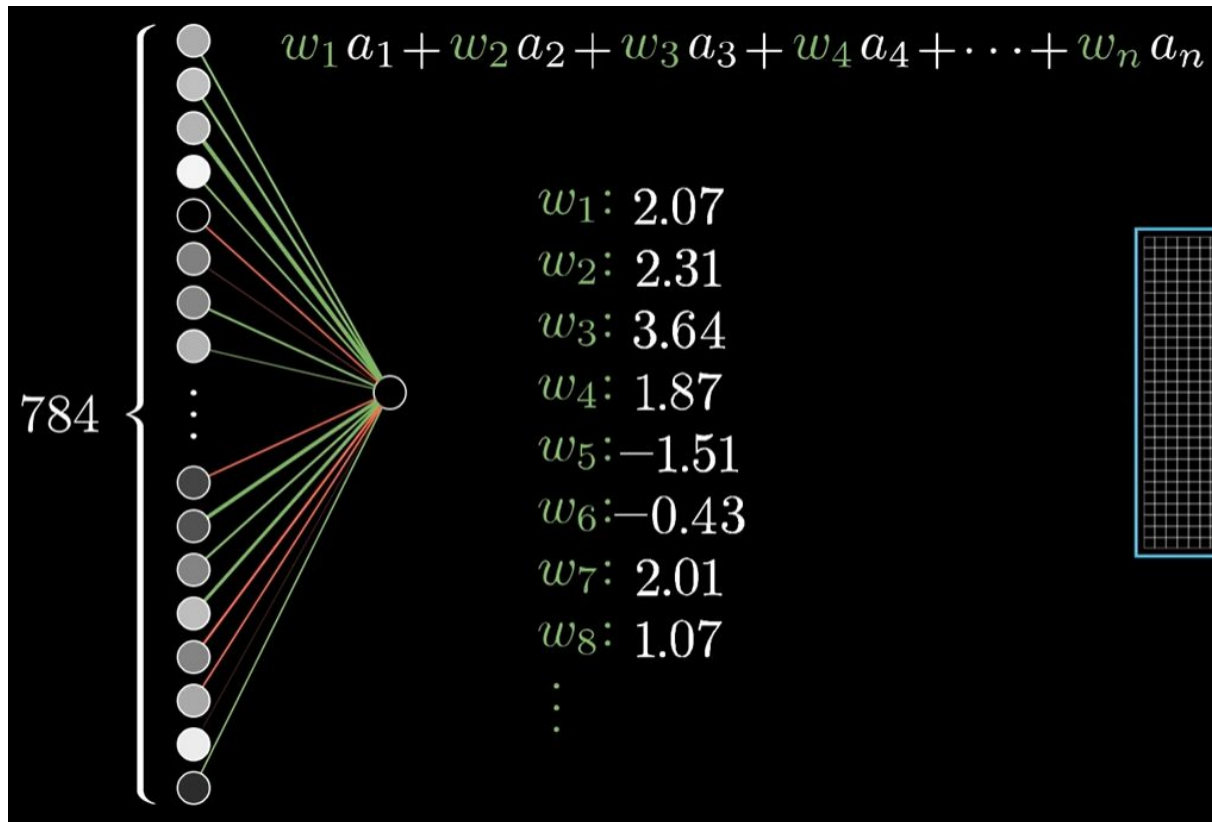




La red neuronal calculará parámetros en cada Layer, tal que dado un valor del píxel (en este caso) multiplicado por tal parámetros incrementa la probabilidad de que se prediga bien cada clase.

$$y = \sum_{i=1}^n w_i x_i + b$$

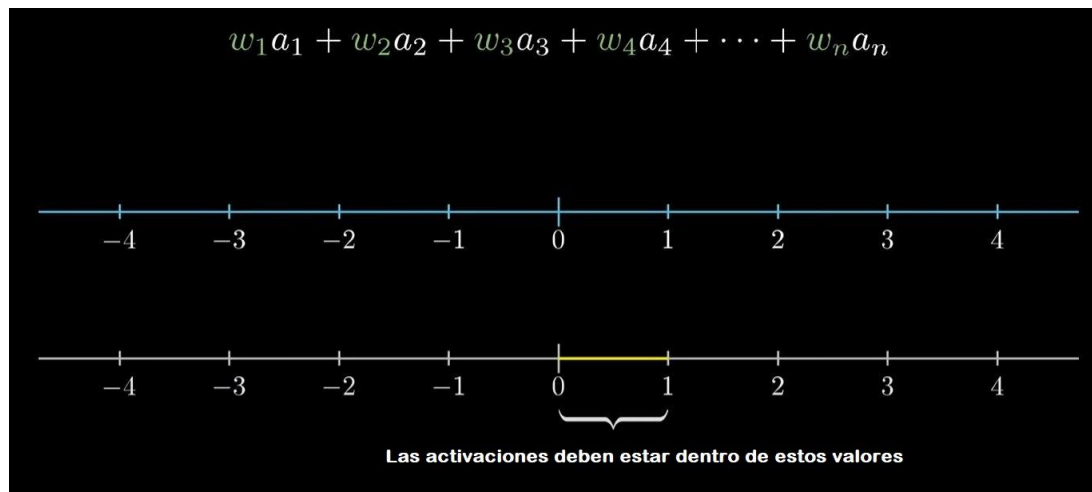
¿Qué va a calcular una red neuronal?



# ¿Qué es una función de activación?

Tenemos diversas funciones de activación disponibles en Keras:

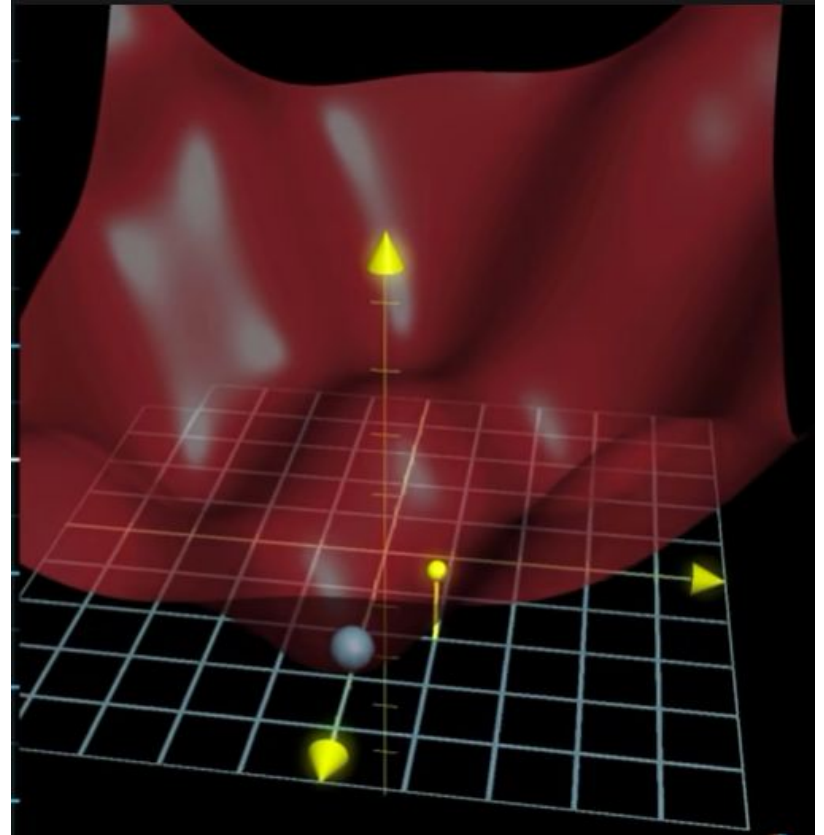
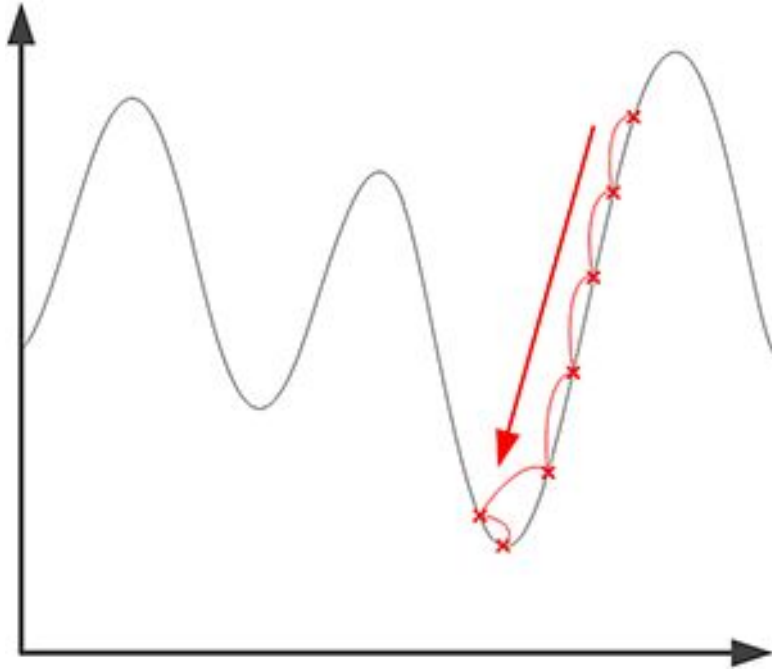
- Sigmoid function
- Softmax function
- Exponential function
- Tanh tangent function
- ReLu function



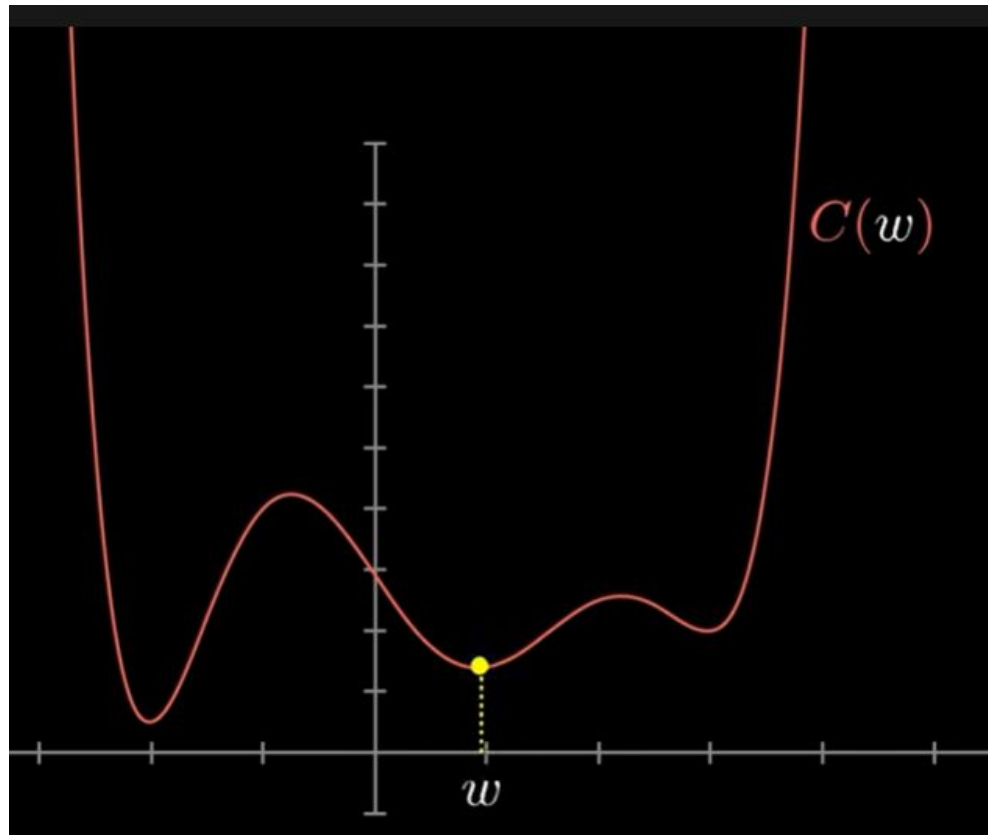
Sigmoid

$$\sigma(w_1a_1 + w_2a_2 + w_3a_3 + \dots + w_na_n)$$

# ¿Cómo se entrena una red? ¿Qué busca?



## Mínimo Local Y Mínimo Global



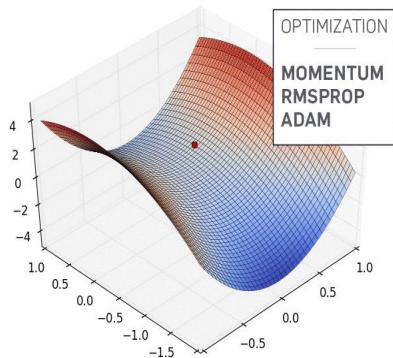


accelerate your learning

## Optimizadores

**Momentum:** Obtiene las gradientes del vector de momento, y actualiza los pesos simplemente añadiéndolo a la ecuación. En otras palabras, este optimizador es para acelerar y no para hacer más rápido el cálculo. El parámetro **beta** es un **símil de la fricción en la aceleración**.

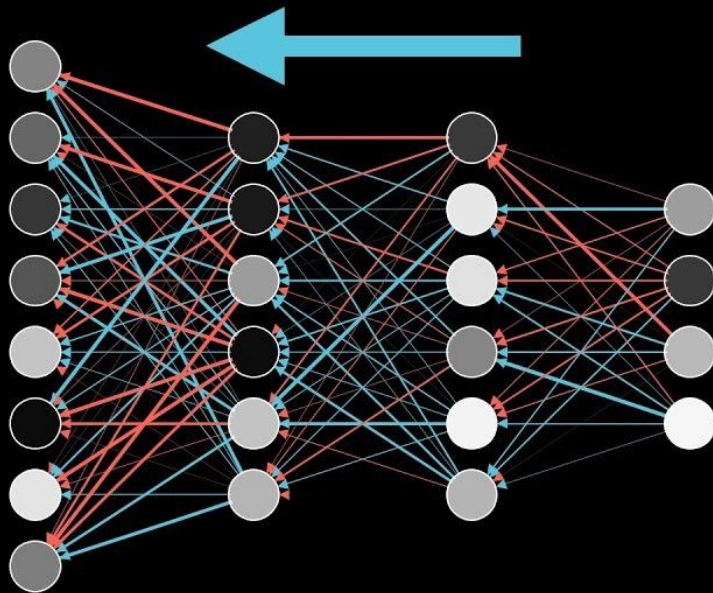
**Adam:** El optimizador de Adam es una combinación de los algoritmos Momentum (acelera la búsqueda del valor mínimo) y RMSDrop (impide las oscilaciones).





## Back Propagation

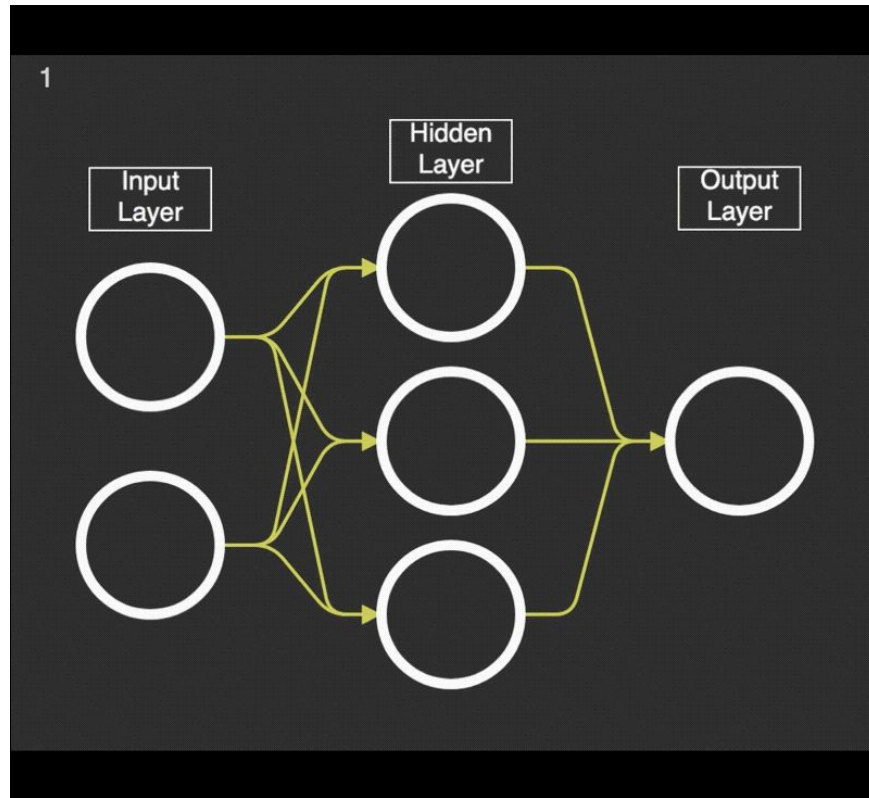
# Backpropagation



## Back Propagation

La idea de backpropagation es ver junto con derivadas parciales (autodiff en tensorflow) qué tanto afectan las variables a los errores de los outputs.

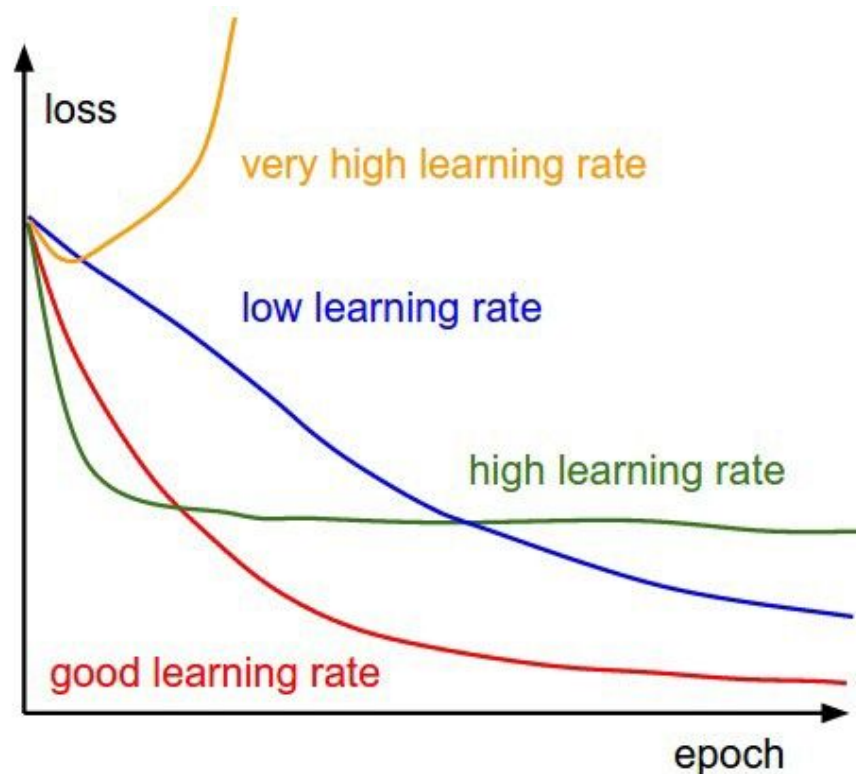
Estas gradientes se propagan hacia todas las capas anteriores y se puede ver y corregir los pesos de la variable que más ha influido en el error.



# Learning Rate

Es un hiperparámetro que controla cuánto cambiar el modelo en respuesta al error estimado cada vez que se actualizan los pesos del modelo.

Elegir el learning rate es un desafío, ya que un valor demasiado pequeño puede resultar en un largo proceso de entrenamiento que podría atascarse, mientras que un valor demasiado grande puede resultar en aprender un conjunto de pesos subóptimo demasiado rápido o un proceso de entrenamiento inestable.

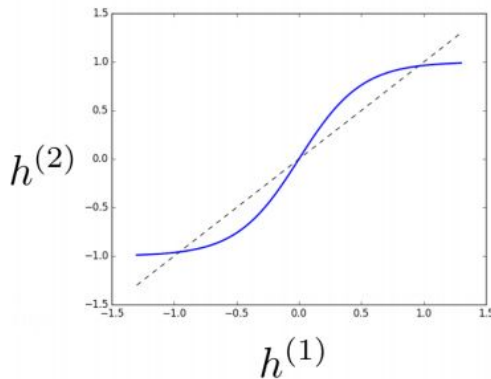


# Vanishing and Exploding Gradient

## Vanishing

Las gradientes resultan ser pequeñas en cierto punto del entrenamiento.

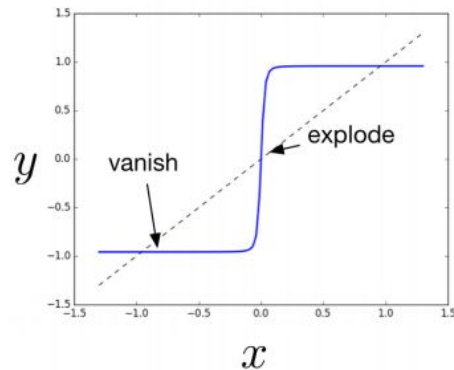
**Problema:** Los pesos pueden llegar a variar tan poco que llega quedarse en un estado neutral.



## Exploding

Las gradientes resultan ser muy grandes y el algoritmo diverge.

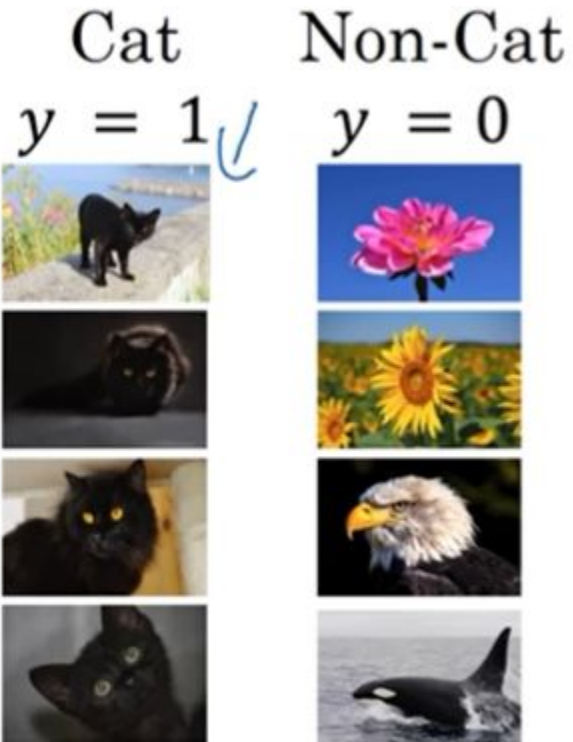
**Problema:** Los pesos varían de tal manera que resulta que el modelo sea inestable durante el training.







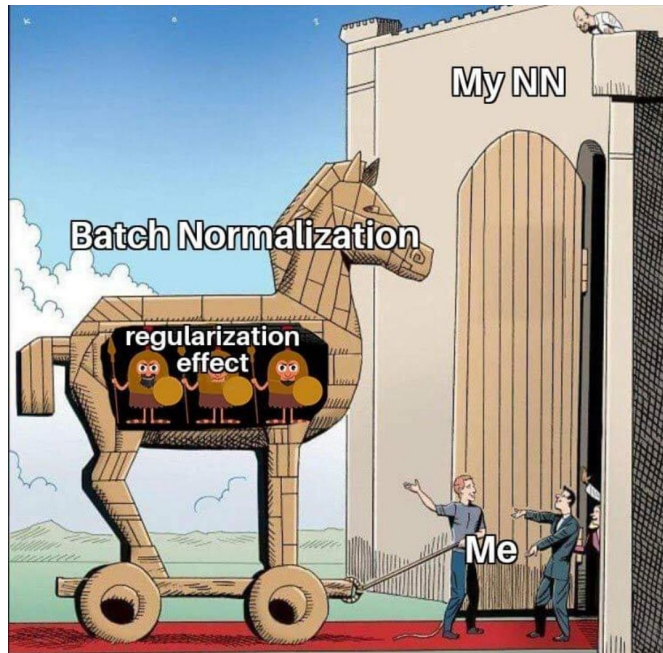
# Batch Normalization



## Batch Normalization

Dado los problemas de vanishing y exploits se recomienda la utilización del método Batch Normalization en el Deep Neural Network.

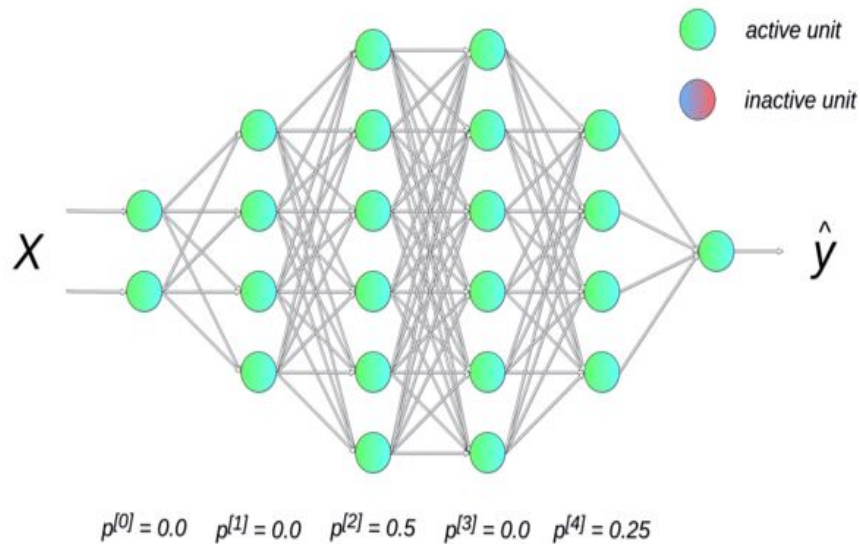
Esto Keras te lo permite fácilmente. **BN se ha convertido en la regla más usada dentro de Deep Learning**, tanto así que se sobreentiende que se usa en cada arquitectura. Sin embargo, Hongyi Zhang, ha sacado un nuevo método que overperforma al BN: **Fixup normalization**, que genera un buen performance en DNN's (10 mil layers).



## DropOut

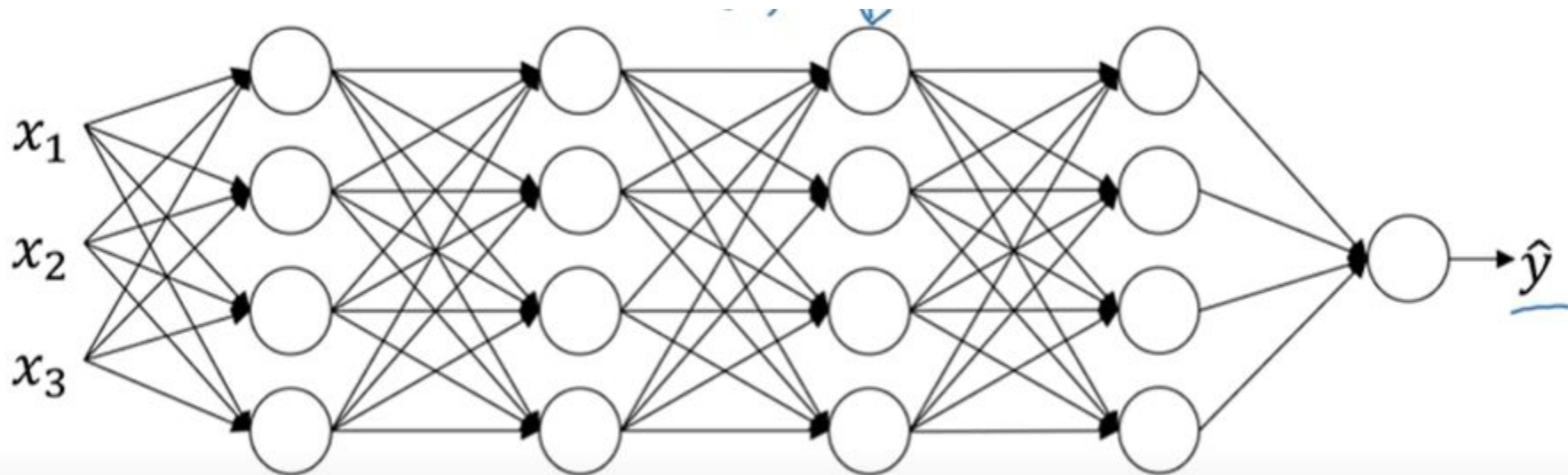
Es una de las técnicas que **más funcionan dentro de una red neuronal** y consiste en dropear nodos temporalmente en cada iteración al momento de entrenar. Estos nodos se dropean con una probabilidad  $p$ , que usualmente es en 50%.

Pero cuidado, el que se dropeen nodos temporalmente hará también que los pesos se ajusten doblemente así que tenemos dos opciones que funcionan casi igual: dividir entre  $(1-p)$  en cada corrección e iteración o multiplicar por  $(1-p)$  cada weight que derive del entrenamiento.



**Nota:** Es aplicado tanto a los inputs como a los hidden layers, pero no a outputs.

## DropOut

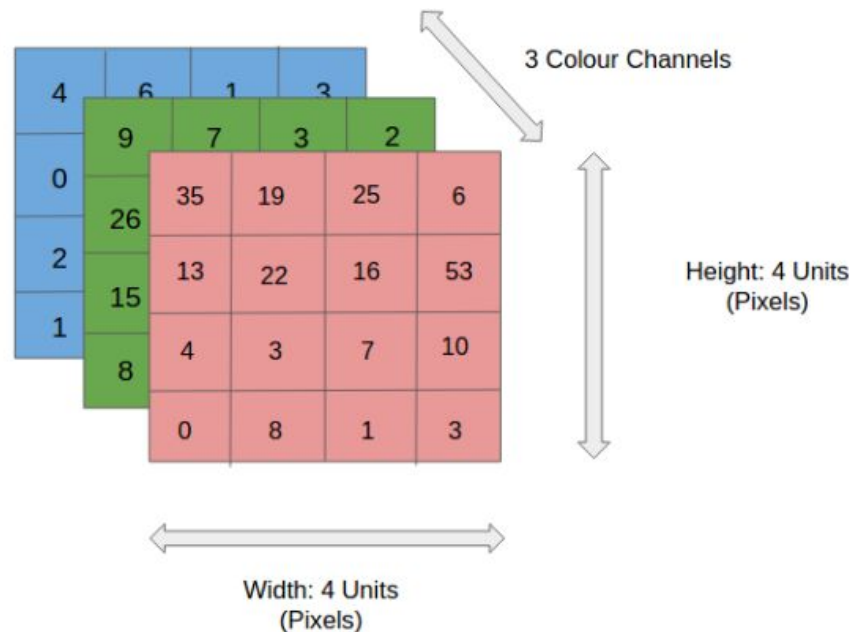


# ¿Cómo se procesa una imagen?

Para el procesamiento de una imagen lo que se realiza es la división de la imagen en píxeles.

En el caso de una imagen en blanco y negro cada píxel se convierte en un array 2D donde cada valor estará entre 0 (completamente negro) y 255 (completamente blanco)

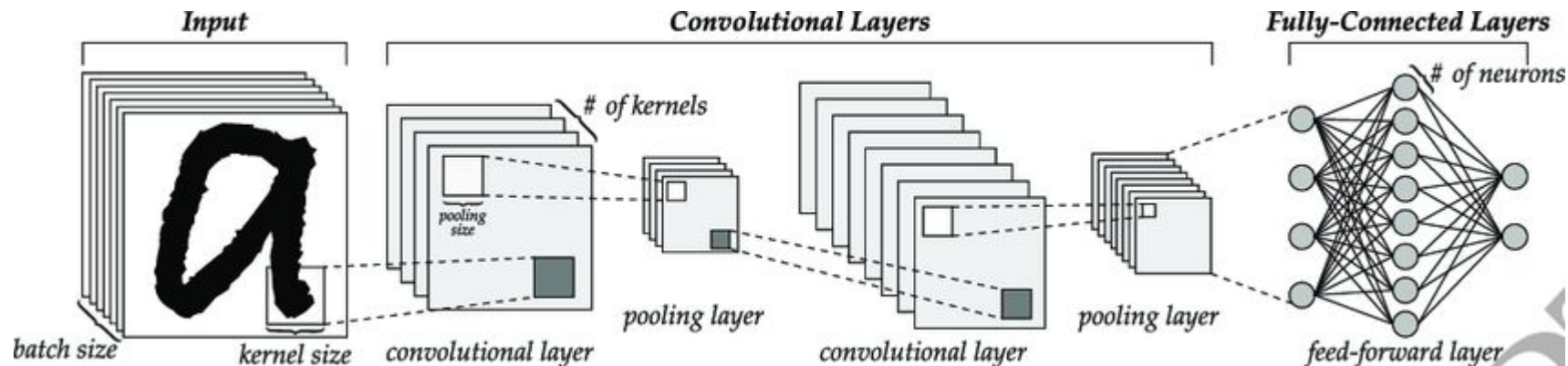
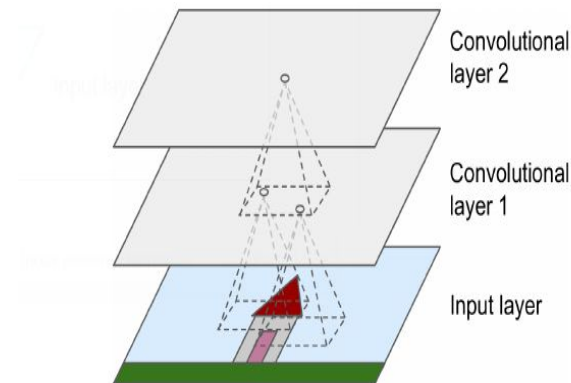
En el caso de una imagen a color se genera un array 3D en el que cada valor representa el nivel de rojo, verde y azul en cada píxel, además, los valores también van de 0 a 255.



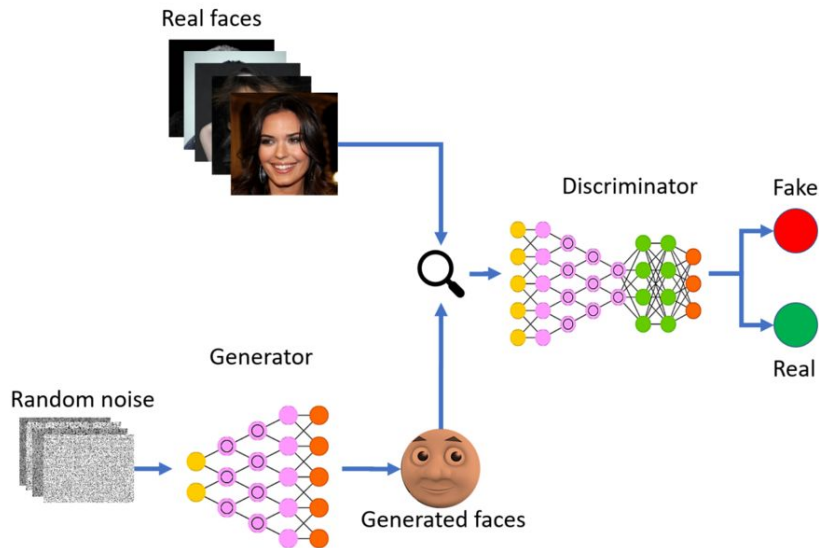


# Convolutional Neural Network (CNN)

Emergió del estudio de la corteza visual del cerebro (responsable del procesamiento visual) y ha sido usado para el reconocimiento de imágenes desde 1980.



# Generative Adversarial Networks



Son arquitecturas algorítmicas que usan dos redes neuronales, enfrentando una contra la otra (por lo tanto "adversarial") para generar nuevas instancias sintéticas de datos que pueden pasar por datos reales.

Se utilizan ampliamente en la generación de imágenes, la generación de videos y la generación de voces. Es por ello que hasta puede ser considerado como un artista.

Un mal uso que se le da es para crear contenido multimedia falso.

## Ejemplo de Gans



[www.thispersondoesnotexist.com](http://www.thispersondoesnotexist.com)