



Data Science

Part-time Bootcamp

1956 Dartmouth Conference: The Founding Fathers of AI



John McCarthy



Marvin Minsky



Claude Shannon



Ray Solomonoff



Alan Newell



Herbert Simon



Arthur Samuel



Oliver Selfridge

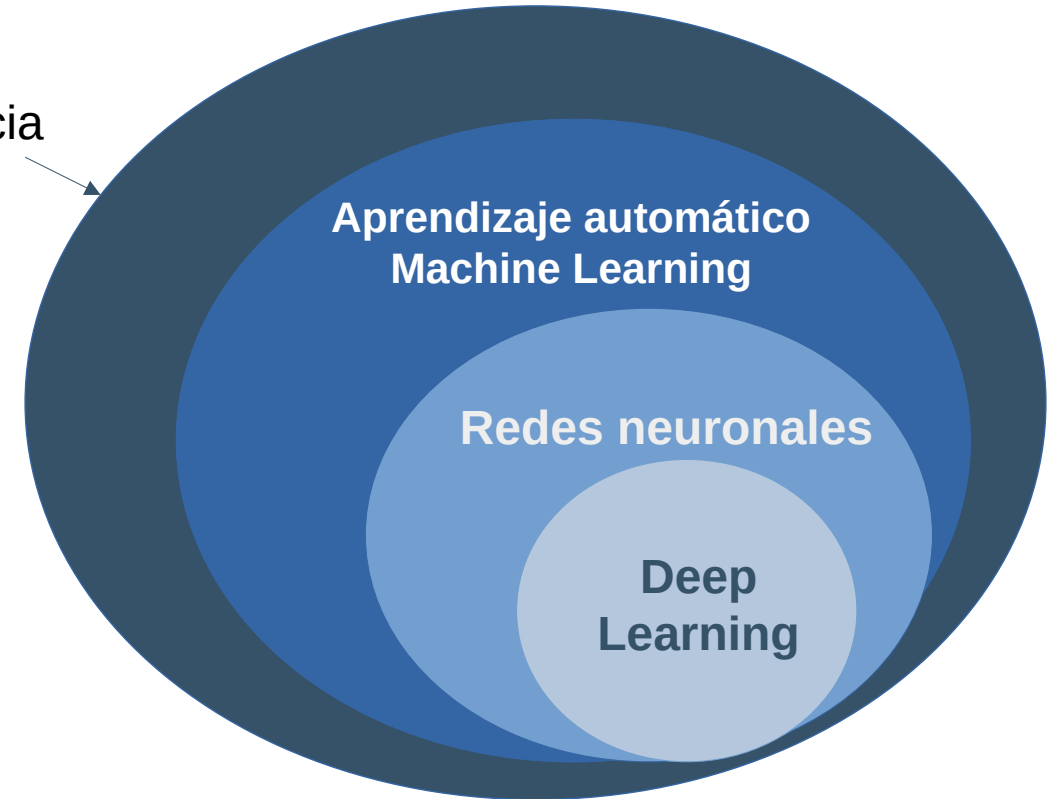


Nathaniel Rochester

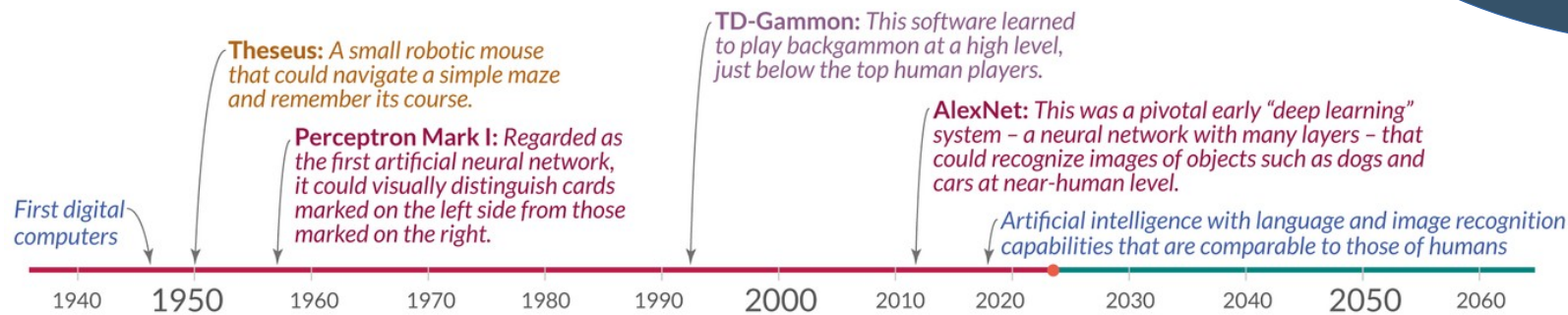


Trenchard More

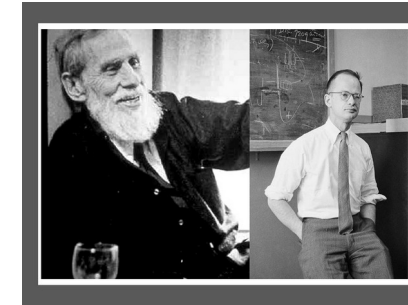
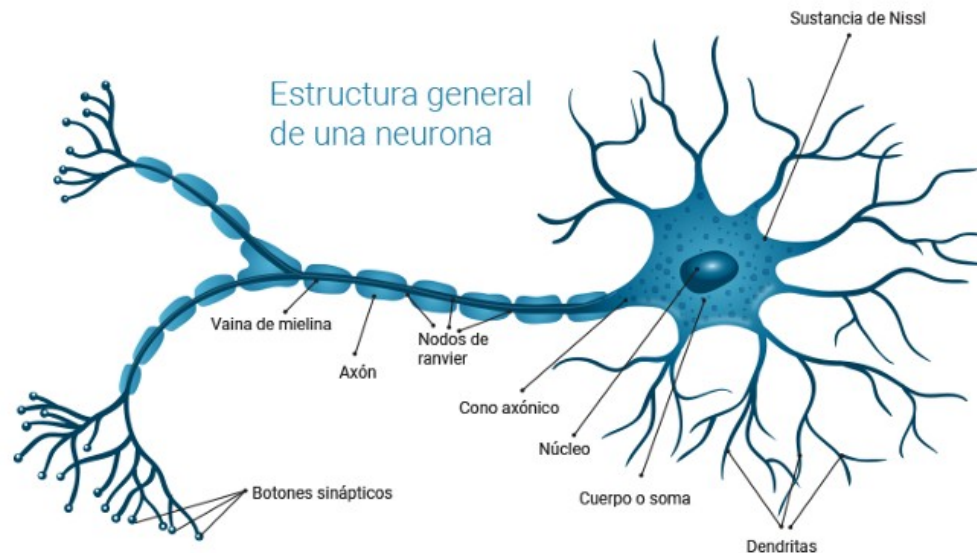
Inteligencia Artificial



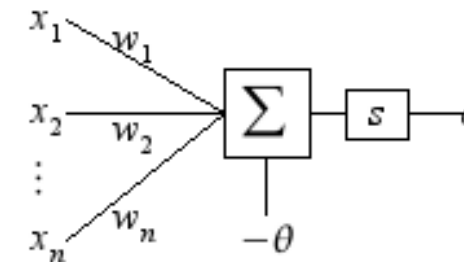
A timeline of notable artificial intelligence systems



Inspiración biológica: Las redes neuronales están inspirados en el concepto biológico de una neurona. Las neuronas recogen información de otras neuronas, a través de las dendritas. Estas señales producen cambios en el soma (cuerpo de la célula) mediante pequeñas señales eléctricas, y si la señal supera cierto umbral, se propaga a las siguientes neuronas a través del axón.



McCulloch-Pitts plantearon un modelo computacional homólogo a la neurona biológica en 1943.



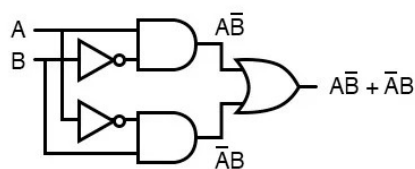
https://es.wikipedia.org/wiki/Neurona_de_McCulloch-Pitts

Perceptrón Basado en el modelo de neurona artificial anterior, Frank Rosenblatt en 1958 creo el modelo que permite la adaptación de los pesos de la neurona basado en el error residual entre el valor esperado y el valor de la neurona. De este modo podemos adaptar los pesos de las entradas y ajustarlos a una tarea concreta.

La función de activación permite la resolución de problemas no lineales.

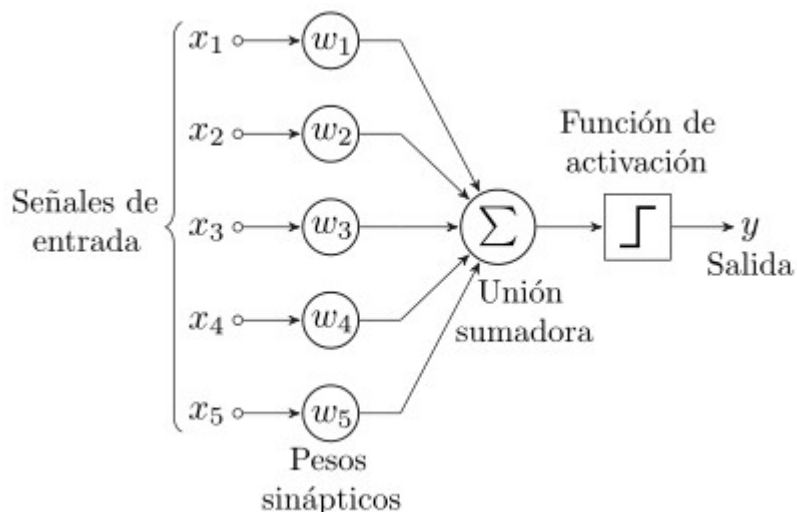


... is equivalent to ...



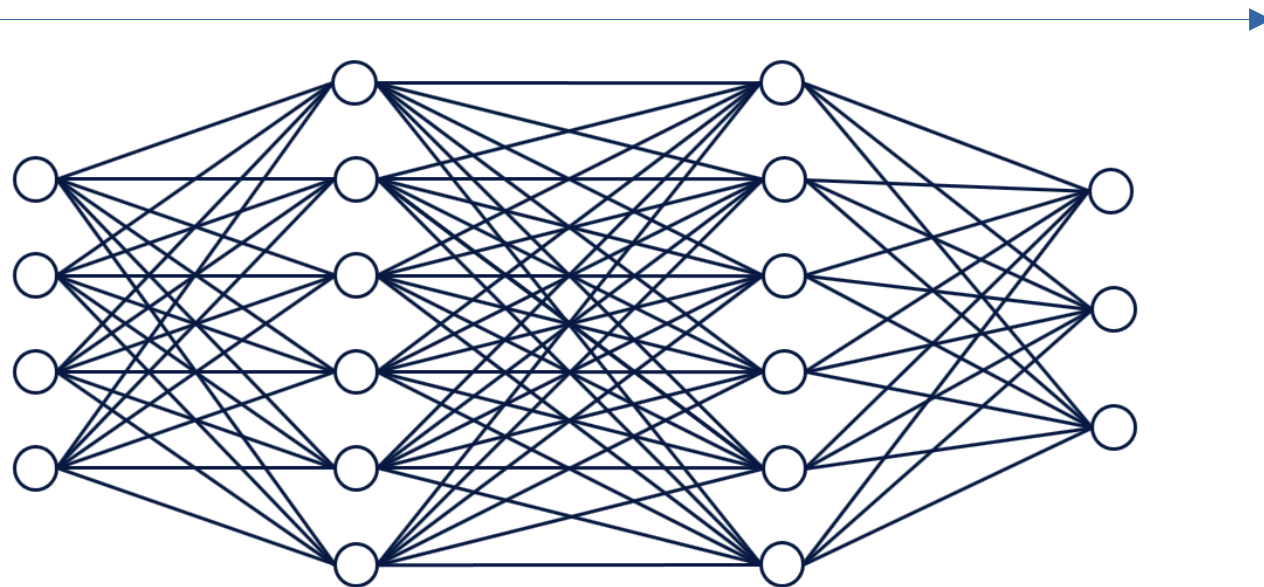
$$A \oplus B = \bar{A}B + A\bar{B}$$

Inputs		Outputs
X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0



Redes Multicapa (MLP) Basado en el modelo de perceptrón anterior organizan una serie de unidades base (neuronas) para montar una red de neuronas capaz de enfrentarse a problemas de gran complejidad.

Se consideran algoritmos de caja negra, dado que lo que sucede en las capas medias dificulta conocer la relación entre entradas y salidas. De hecho, el entrenamiento de estos modelos solo fue posible hasta que se planteó el modelo de retropropagación mediante modelos de diferenciación automática.



Capas ocultas

Forward pass

$$g(x) := f^L(W^L f^{L-1}(W^{L-1} \dots f^1(W^1 x) \dots))$$

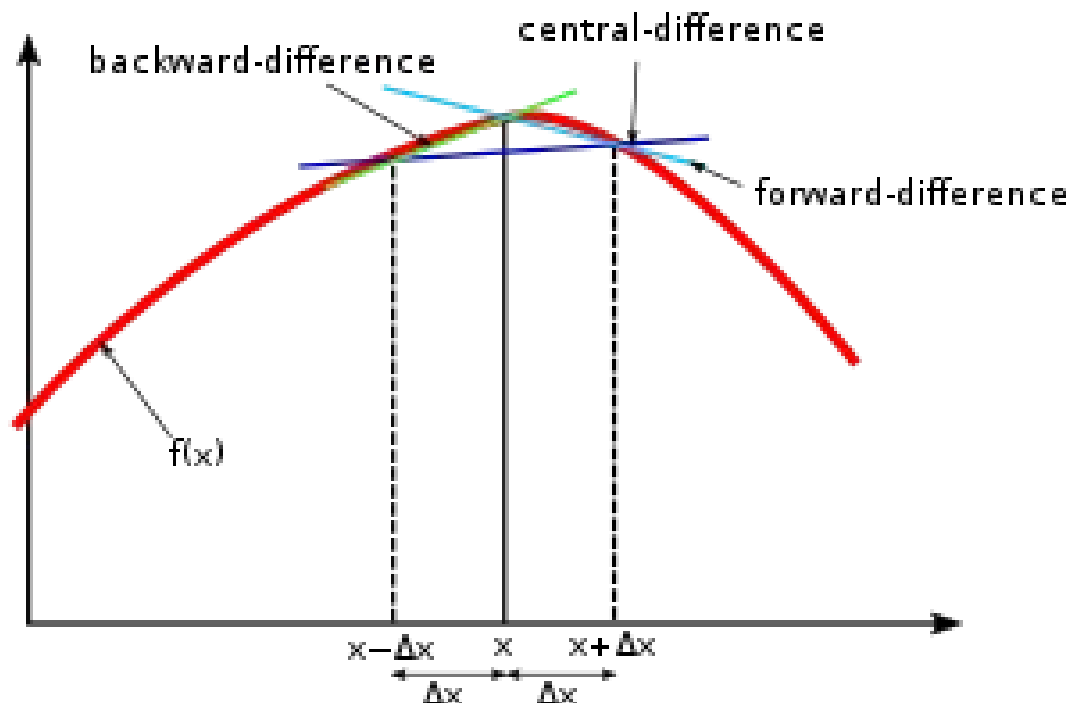
Backward pass

$$E(y, y') = \frac{1}{2} \|y - y'\|^2$$

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} = -\eta o_i \delta_j$$

Diferenciación automática Dado que para entrenar basado en gradiente estos modelos, requerimos de calcular el gradiente en muchos puntos, existen técnicas para obtener el valor numérico del gradiente en cada evaluación gracias a los conocidos como números duales.

Existen otros modelos como las diferencias finitas que con una simple perturbación permite calcular la cuesta (gradiente) de una función en un punto dado.



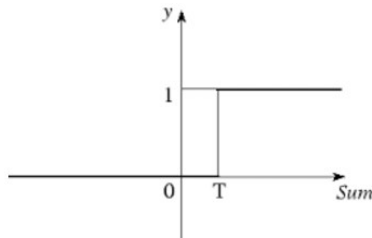
$$f'(x) = \lim_{h \rightarrow 0} \frac{f(x + h) - f(x)}{h}.$$

Lo más importante es que no debemos preocuparnos en exceso por estos cálculos, ya que existen frameworks que automatizan esta tarea:



Funciones de activación Son las funciones que podremos elegir para la parte-no lineal de nuestras neuronas.

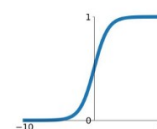
- **¿Función de activación de la entrada?** NO LLEVA
- **¿Problema de regresión?** Función de activación lineal, como ReLu o Leaky Relu
- **¿Clasificación binaria?** Sigmoid. Funciona bien. Computacionalmente es pesada y por eso solo se recomienda al final.
- **¿Clasificación multiclase?** Softmax es la mejor, acota los datos entre 0 y 1, que es lo que nos interesa, la probabilidad de pertenencia a una clase.
- **¿Capas intermedias?** ReLu es la que mejores prestaciones nos da. Buenos resultados y computacionalmente eficiente.

$$o_j = \begin{cases} 0 & \text{if } \sum_i w_{ij}x_i < T \\ 1 & \text{if } \sum_i w_{ij}x_i \geq T \end{cases}$$


Activation Functions

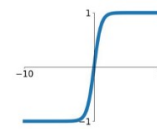
Sigmoid

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



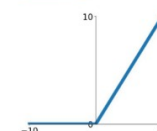
tanh

$$\tanh(x)$$



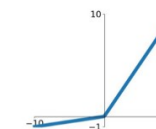
ReLU

$$\max(0, x)$$



Leaky ReLU

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

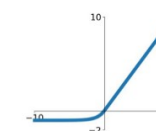


Maxout

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

ELU

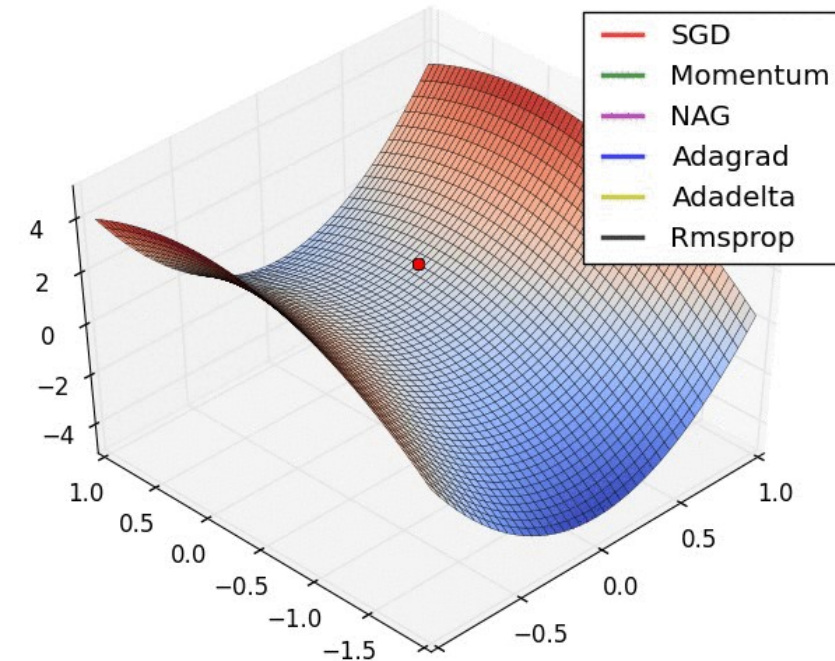
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Optimización Vimos levemente cómo podemos entrenar una función objetivo en base de la guía de los gradientes, pero existen multitud de variantes que tendremos disponibles en nuestro framework de elección.

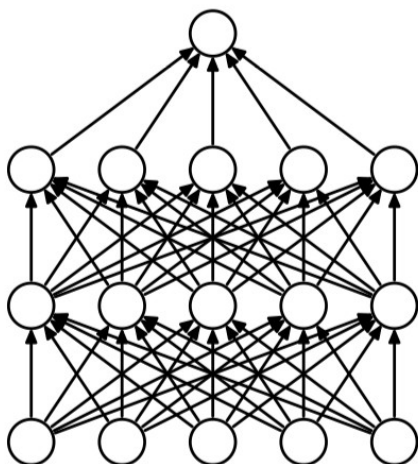
- Gradient Descent
- Stochastic Gradient Descent
- Mini-Batch Gradient Descent
- Momentum
- Nesterov Accelerated Gradient
- Adagrad
- AdaDelta
- Adam
- (...)

```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

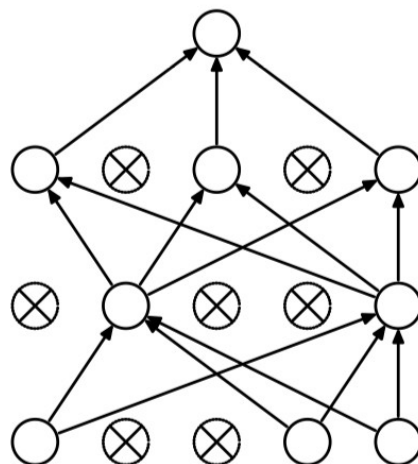


Overfitting Dada la profundidad de estos modelos, no están libres de sobreajuste. Por ello existen técnicas que nos permiten regularizar el modelo para evitar este efecto.

Una técnica extendida y con buen resultado es el Dropout, que no se trata de otra cosa que apagar algunas neuronas al azar durante el entrenamiento. Esto evita ajustarnos en exceso a las muestras que se le presentan.



(a) Standard Neural Net

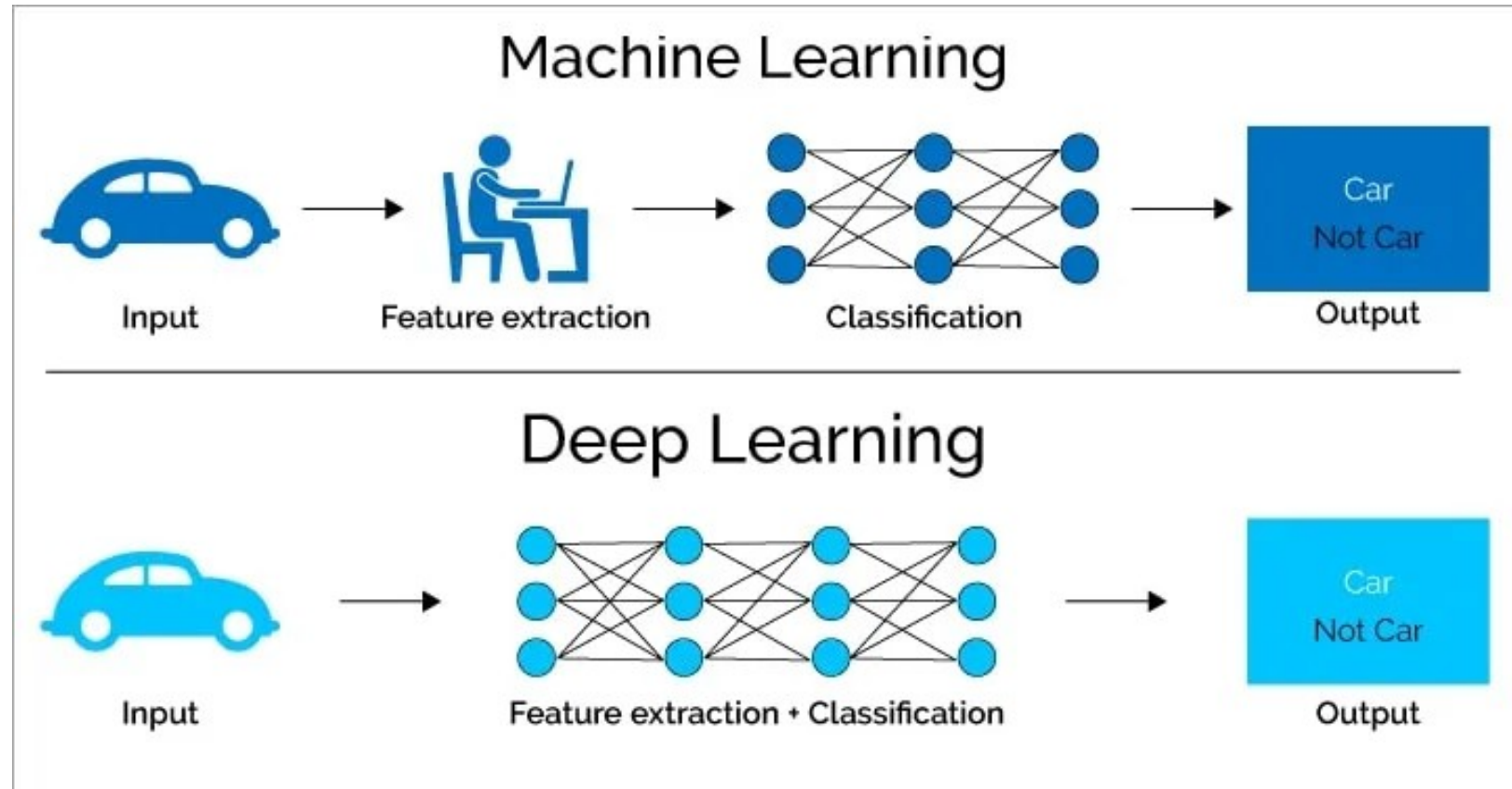


(b) After applying dropout.

Method	Test Classification error %
L2	1.62
L2 + L1 applied towards the end of training	1.60
L2 + KL-sparsity	1.55
Max-norm	1.35
Dropout + L2	1.25
Dropout + Max-norm	1.05

Table 9: Comparison of different regularization methods on MNIST.

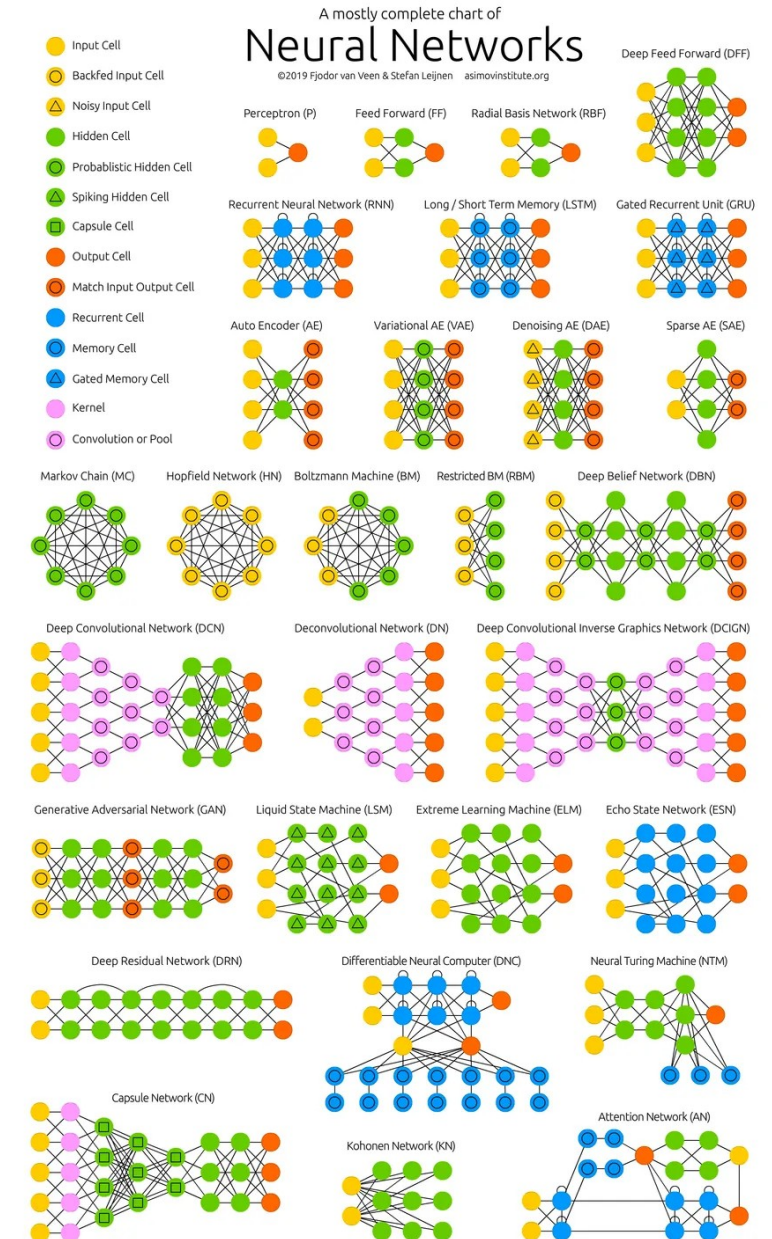
Machine Learning vs Deep Learning



Arquitecturas Las redes neuronales son muy flexibles en lo que a arquitecturas (capas, neuronas por capa, tipo de capa), con lo que deberemos de familiarizarnos con las formas comunes para distinto tipo de tareas:

- Imágenes
- Video/Texto
- Reducción de dimensionalidad
- Generación de muestras sintéticas
- Multimodalidad (texto + imágenes)
- ...

<https://www.asimovinstitute.org/neural-network-zoo/>



Bibliografía

<https://www.tensorflow.org/resources/learn-ml/theoretical-and-advanced-machine-learning?hl=es-419>

<https://www.nvidia.com/en-us/glossary/deep-learning/>

<https://www.asimovinstitute.org/neural-network-zoo/>

