Técnicas de Inteligencia Artificial

Óscar García

Tema 6. Deep Learning

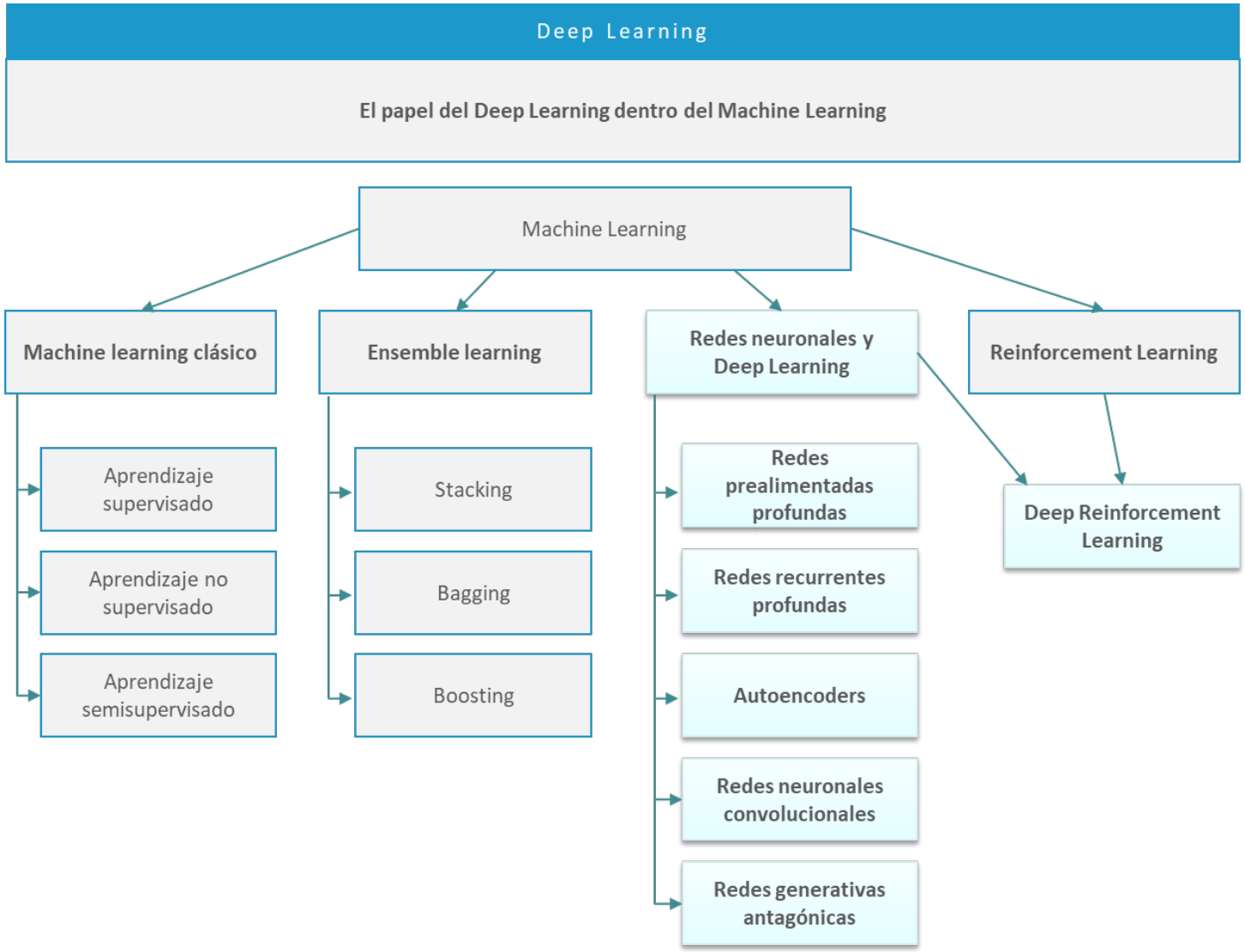
UNIVERSIDAD INTERNACIONAL DE LA RIOJA

La Universidad en Internet

Contenidos

- [6.2] Introducción
- [6.3] El papel del *Deep Learning* dentro del *Machine Learning*
- [6.4] Redes neuronales y *Deep Learning*
- [6.5] Redes prealimentadas profundas
- [6.6] Redes neuronales recurrentes profundas
- [6.7] *Autoencoders*
- [6.8] Redes neuronales convolucionales
- [6.9] Redes generativas antagónicas
- [6.10] Aprendizaje por refuerzo
- [6.11] Aprendizaje por refuerzo profundo
- [6.12] Ejemplos de implementación
- [6.13] Referencias bibliográficas

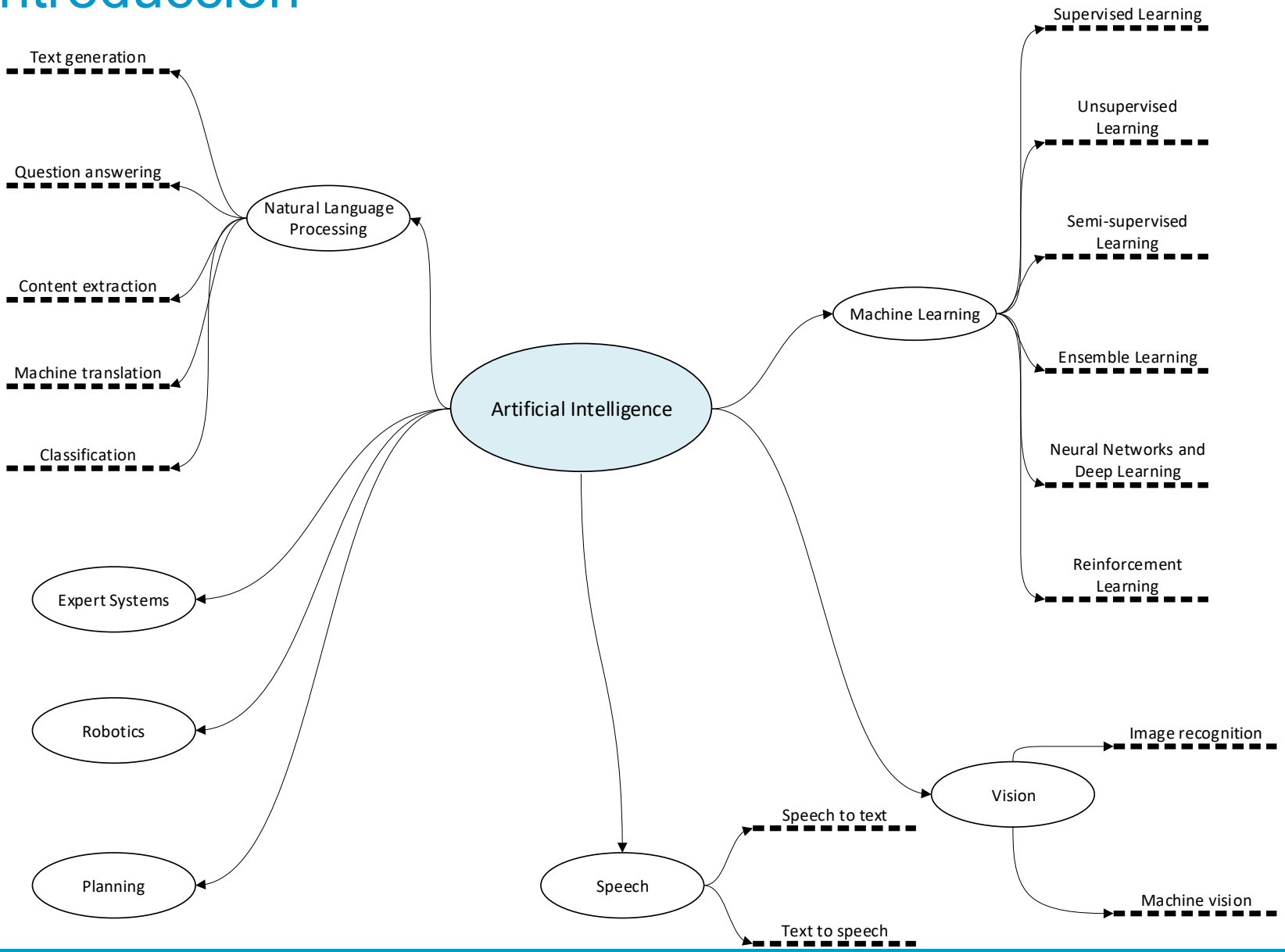
Esquema


Deep Learning
El papel del Deep Learning dentro del Machine Learning
Machine Learning
Machine learning clásico
Ensemble learning
Redes neuronales y Deep Learning
Reinforcement Learning
Aprendizaje supervisado
Aprendizaje no supervisado
Aprendizaje semisupervisado
Stacking
Bagging
Boosting
Redes prealimentadas profundas
Redes recurrentes profundas
Autoencoders
Redes neuronales convolucionales
Redes generativas antagónicas
Deep Reinforcement Learning

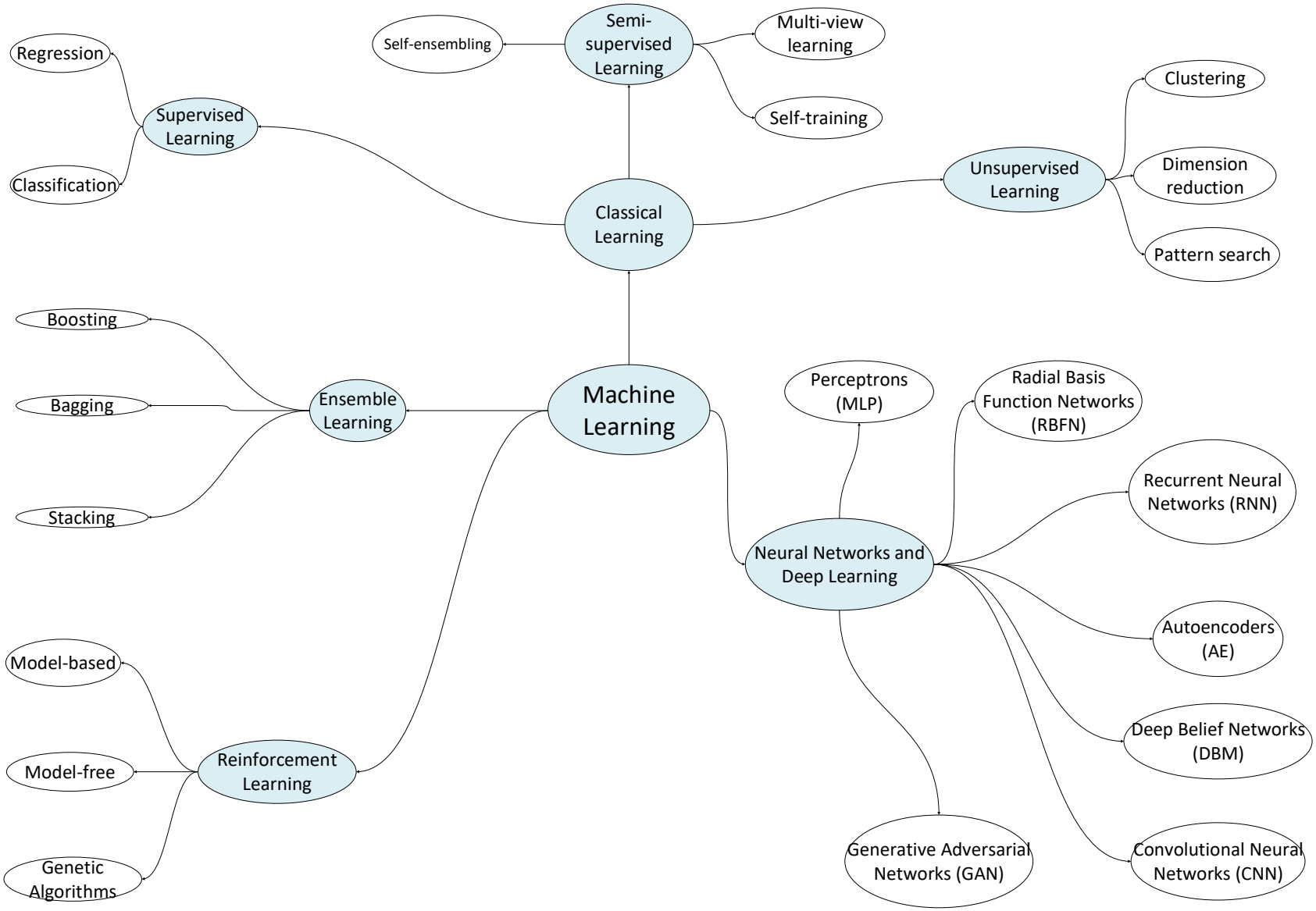
Objetivos

- Entender el lugar que ocupa el **Deep Learning** dentro del Machine Learning y éste dentro de la Inteligencia Artificial.
- Conocer la **taxonomía de los principales grupos de Redes Neuronales Profundas** y sus principales aplicaciones.
- Comprender los fundamentos básicos del **Reinforcement Learning** y del **Deep Reinforcement Learning**.
- Identificar **aplicaciones prácticas** del Deep Learning y del Deep Reinforcement Learning.
- Iniciarte en la aplicación de técnicas Deep Learning sobre datasets de pruebas utilizando Python y **TensorFlow**.

Introducción



Deep Learning dentro del Machine Learning



Redes Neuronales y Deep Learning

- Perceptrón inventado por Rosenblatt en 1958
- A finales de los 60 se abandonó el conexionismo en favor del razonamiento simbólico
- A finales de los 70 termina el primer invierno de la IA
- Años 80: invención de la retropropagación y aparición de sistemas expertos
- Finales de los 80 y principios de los 90: segunda invierno de la IA, primeras menciones a Deep Learning (1986)
- Años 90: agentes inteligentes y machine learning basado en métodos estadísticos

Redes Neuronales y Deep Learning

- ¿Por qué el éxito actual?
- Actualmente gran capacidad de computación usando CPU, GPU e incluso nuevas TPU en el cloud y el edge
  - Especialización en operaciones con tensores
- Además, contamos con gran capacidad y técnicas de almacenamiento antes no disponibles
  - NoSQL, Big Data, etc.

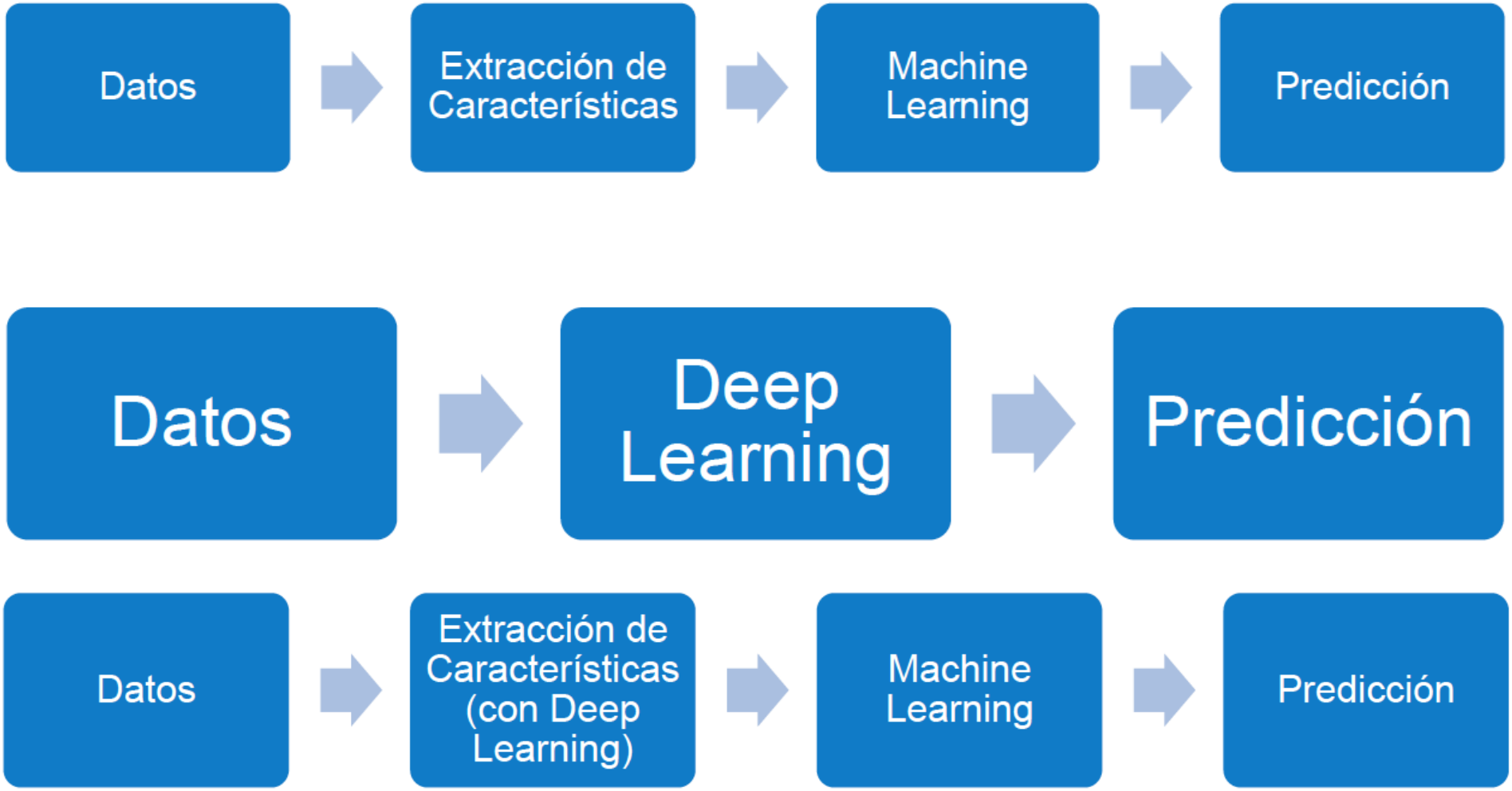
Deep Learning

- Se puede definir el a**prendizaje profundo** o ***Deep Learning*** como una clase de algoritmos de aprendizaje automático que utiliza **múltiples capas** para **extraer progresivamente características de nivel superior de la entrada bruta**.
- Esto incluye, por tanto, **una cascada de capas conectadas entre sí con unidades de procesamiento no lineal en cada una de las capas**.
- De este modo, las primeras capas (inferiores) se corresponden con niveles de abstracción menor, mientras que las últimas capas (superiores) se basan en las anteriores para formar una **representación jerárquica de conceptos**.

Deep Learning

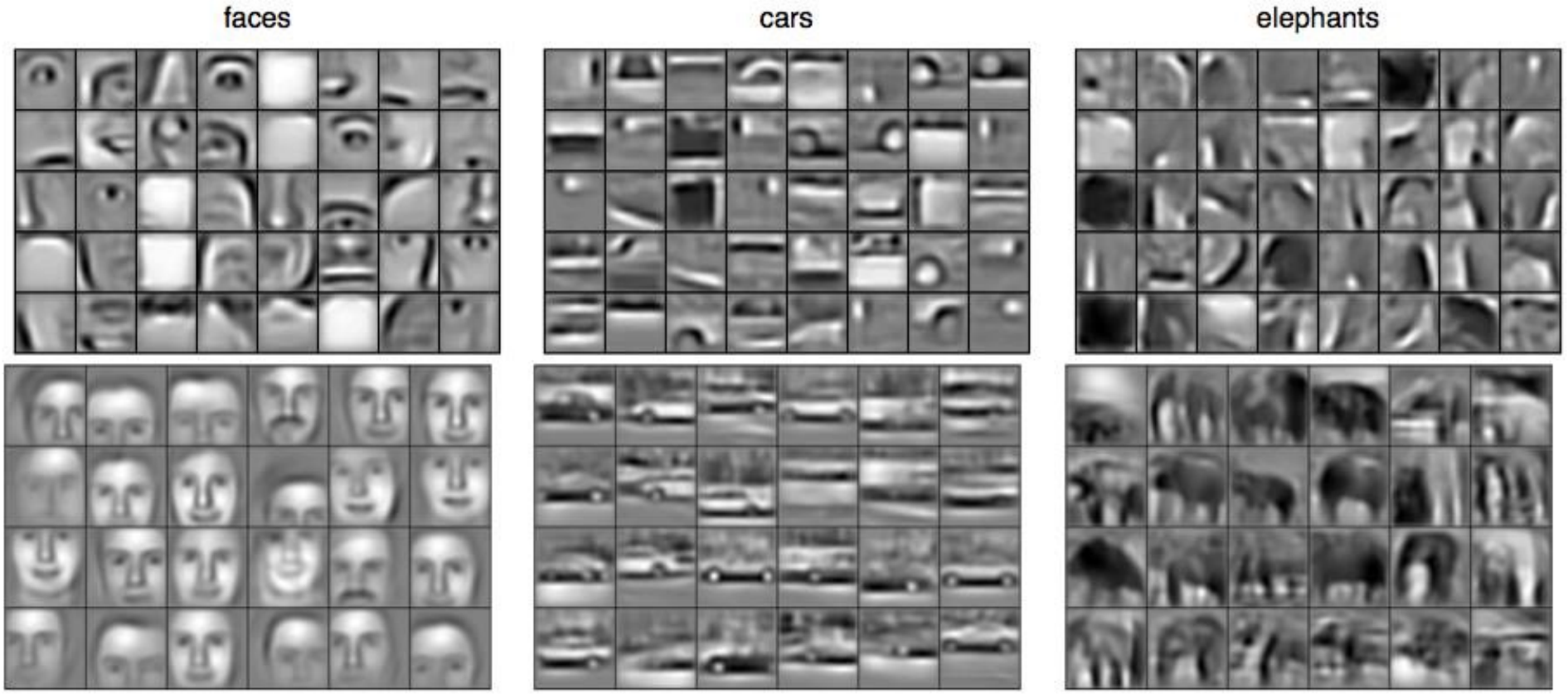
- En Machine Learning: Busca un buen **preprocesamiento** para tener una buena representación de los objetos en un vector de características.
- En **Deep Learning**: Busca directamente la red para que realice la extracción de características (detectadas en una serie de capas de procesamiento).
- Puede entenderse como tener preprocesamiento y procesamiento en la misma red – Múltiples redes

Deep Learning



Fuente de las imágenes: Raúl Arrabales

Deep Learning

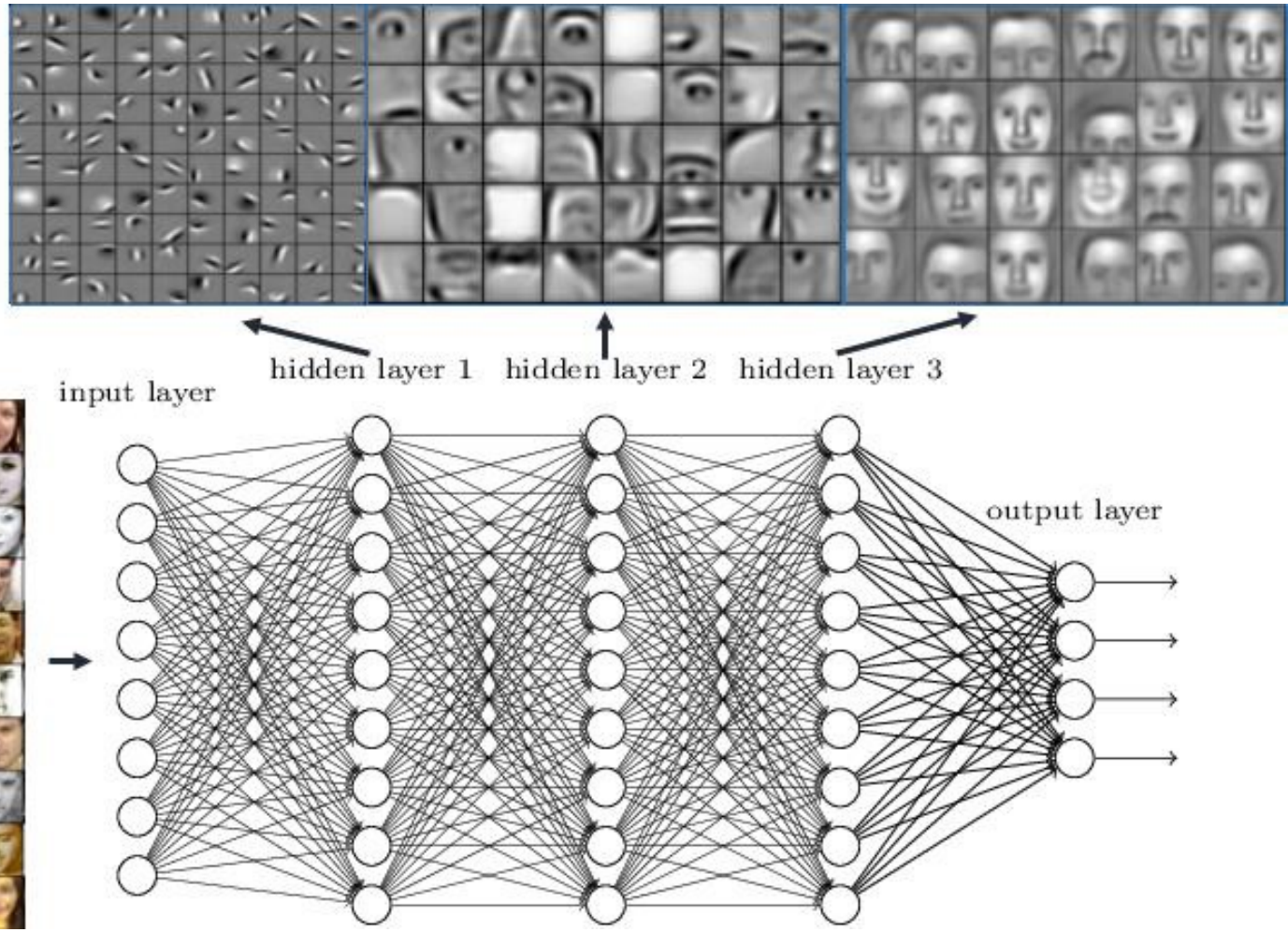


Fuente de las imágenes: Raúl Arrabales

Deep Learning

- Jerarquía en las características

Deep neural networks learn hierarchical feature representations



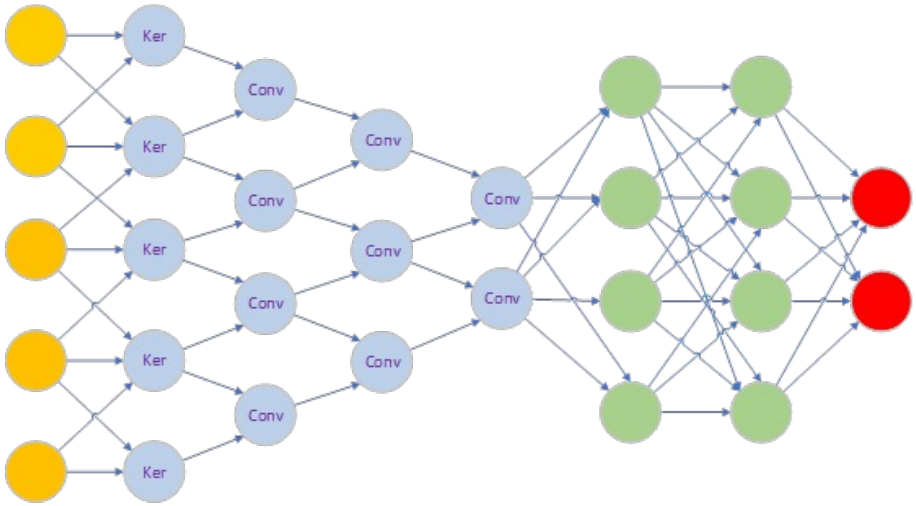
Fuente de las imágenes: Raúl Arrabales

Deep Learning

- Estrategias de entrenamiento:
  - Muchos parámetros de configuración – **hiperparametrización**.
  - Dificultad para explorar todo el espacio posible de configuraciones, aliviado con GPU, TPU, NPU y computación vectorial.
  - Opciones:
    - Tipo de variables, escalado, etc.
    - Tipo de red.
    - Número de nodos y capas ocultas.
    - Tipo de capas.
    - Inicialización de los pesos (relu, Xavier).
    - Función de activación, umbral, etc. (tangente hiperbólica, sigmoidal, relu, etc.).
    - Función de pérdida (entropía, error cuadrático medio, etc.)
    - Normalización de datos ([0,1], [-1,1]).
    - Algoritmo de optimización (descenso del gradiente, Adam, etc.)
    - Tasa de aprendizaje.
    - Epochs y batches.

Ejemplo: redes convolucionales

- Por ejemplo, en el procesamiento de imágenes empleando **redes convolucionales**, como veremos más adelante:
  - las **primeras capas (también llamadas inferiores)** pueden identificar los **bordes**,
  - mientras que las **capas finales (también llamadas superiores)** pueden identificar los **conceptos relevantes para un humano**, como los dígitos o las letras o las caras.

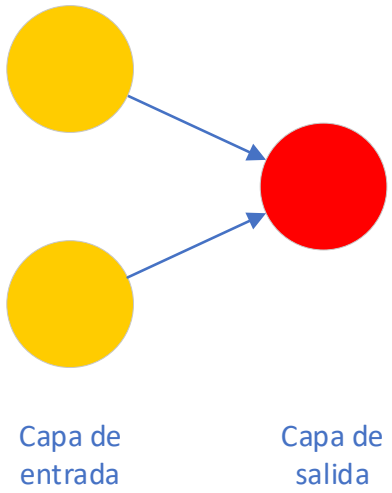


Redes más relevantes

- **Redes prealimentadas (*Feedforward networks*)** y redes prealimentadas profundas (*Deep Feedforward Networks*).
- **Redes Neuronales Recurrentes (RNN – *Recurrent Neural Networks*)** y Redes Recurrentes Profundas (*Deep Recurrent Networks*).
- ***Autoencoders* (AE)**.
- **Redes Neuronales Convolucionales (CNN – *Convolutional Neural Networks*)**.
- **Redes Generativas Antagónicas (GAN – *Generative Adversarial Networks*)**.

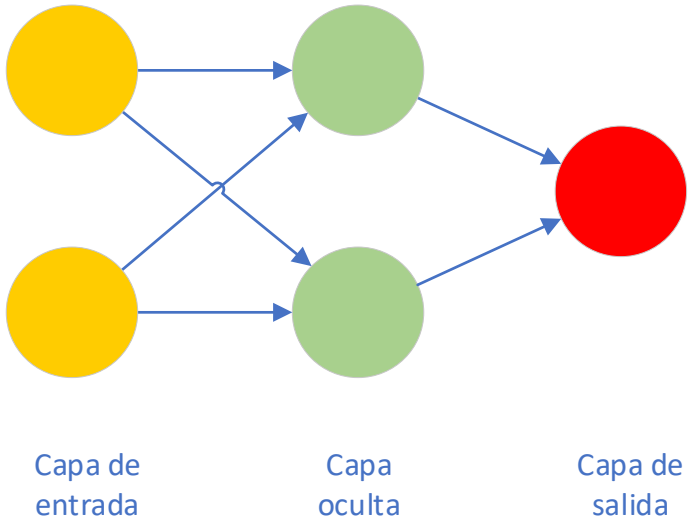
Redes prealimentadas (Feedforward Networks)

- Todas las neuronas de una capa están conectadas con todas las neuronas de la capa siguiente, no existe conexiones entre neuronas de la misma capa y no existe retroalimentación
- **Perceptrón** (Rosenblatt, 1958): discriminador lineal



Redes prealimentadas (Feedforward Networks)

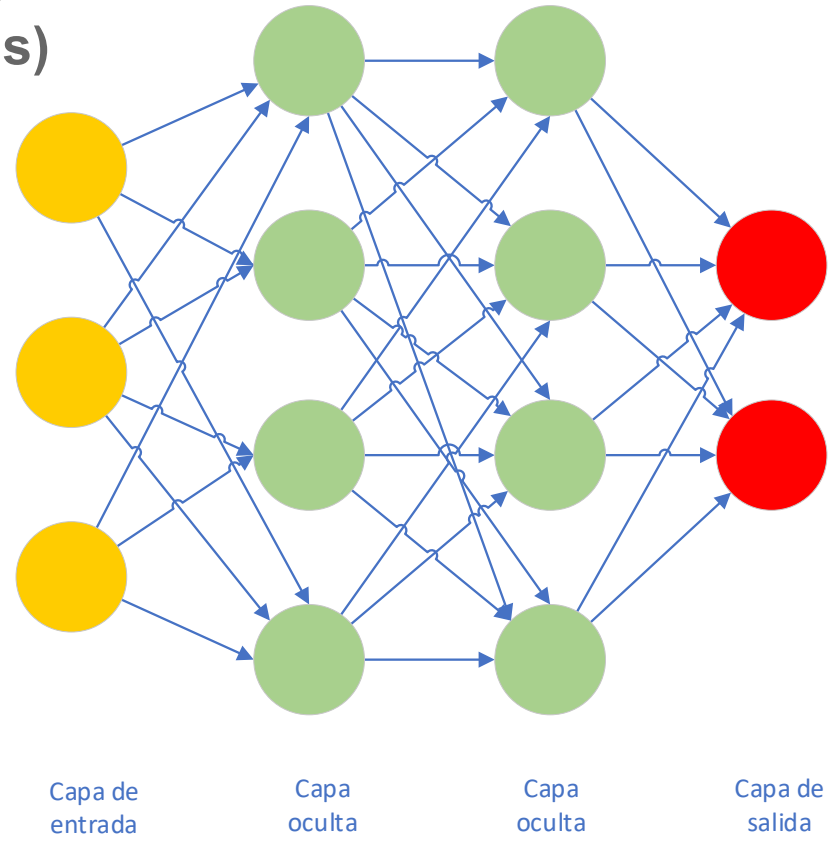
- **Perceptrón Multicapa (MLP)**
  - Funciones de activación: sigmoidal (logística, tangente hiperbólica, etc.)
  - Método de retropropagación, regla del gradiente del error
  - Aplicaciones
    - Clasificadores
    - Reconocimiento del habla
    - Reconocimiento de imágenes
    - Traducción automática
    - Etc.
  - Sustituidos por las SVM
    - Mayor simplicidad en determinadas aplicaciones



Breve introducción a SVM:
https://towardsdatascience.com/https-medium-com-pupalerushikesh-svm-f4b42800e989

Redes prealimentadas (Feedforward Networks)

- **Deep Feedforward Networks**
  - **Algoritmos de retropropagación modificados**
  - **MLP (Multilayer Perceptron)**
  - **RBF (Radial Basis Functions)**
    - Funciones de activación:
    - $\phi(r) = r = || X_j - X_k ||$
    - $\phi(r) = e^{-(\epsilon r)^2}$

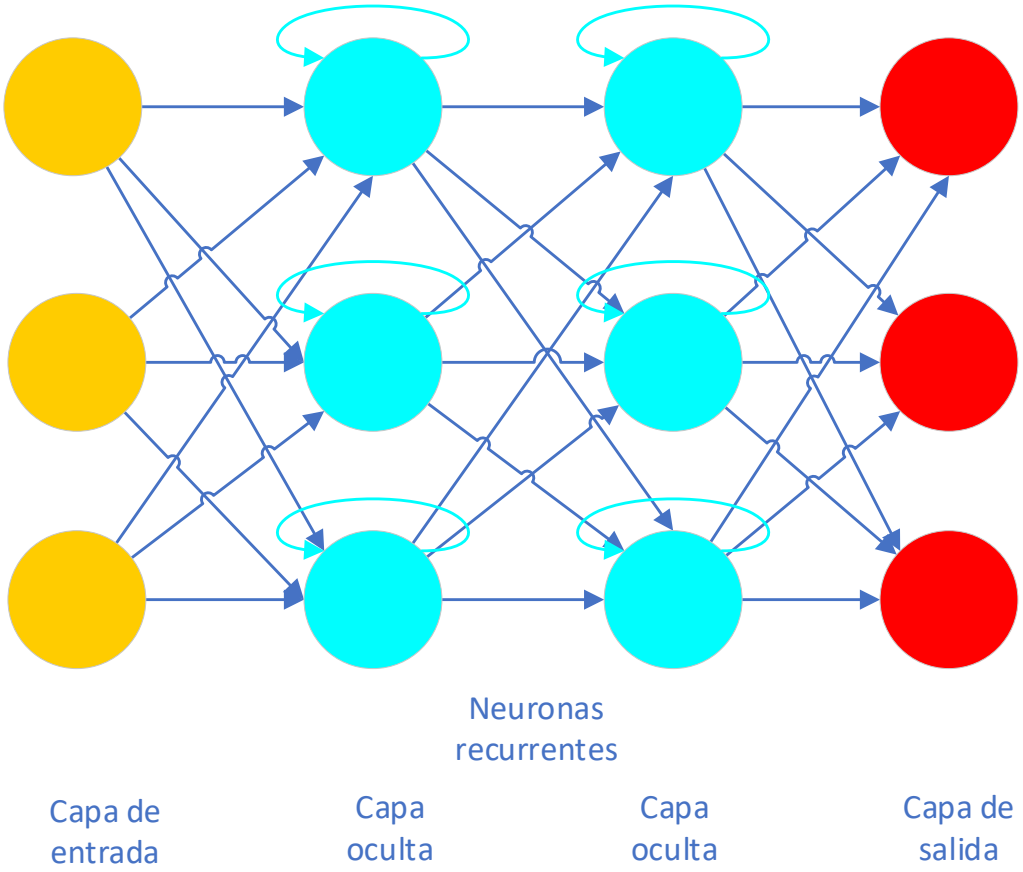


Redes neuronales recurrentes profundas

- **Redes Neuronales Recurrentes**
  - Problemas relacionados con el lenguaje
  - Voz
  - Música
  - Datos secuenciales
  - Traducción automática
  - Genómica
  - Proteómica
  - Reconocimiento de voz
  - Síntesis de voz
  - Asistentes conversacionales
  - Autocompletado de textos y voz

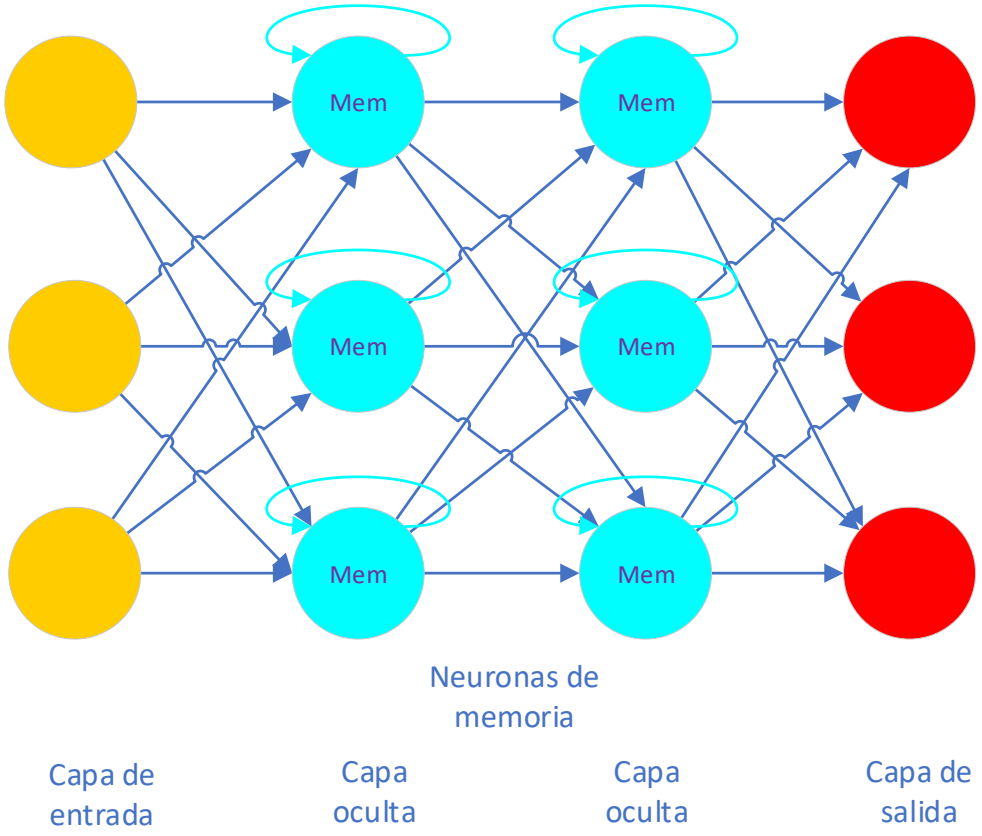
Redes neuronales recurrentes profundas

- **Redes Neuronales Recurrentes**



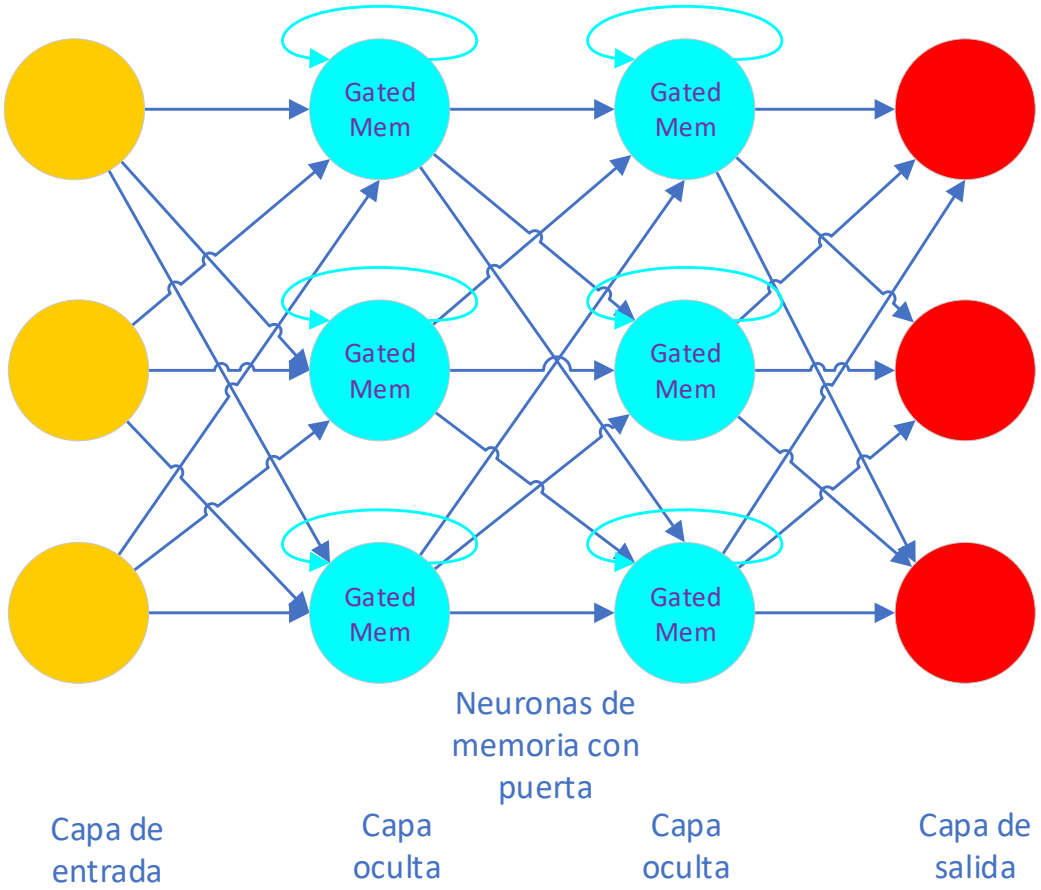
Redes neuronales recurrentes profundas

- **LSTM (Long and Short Term Memories)**
  - Mejoran el concepto de RNR
  - Celdas especiales, valor puede ser almacenado, leído o restablecido, mediante puertas de entrada, salida y olvido



Redes neuronales recurrentes profundas

- **Gated Recurrent Units (GRU)**
  - Células de memoria sólo tienen una puerta de actualización y una puerta de reajuste
  - Mejor rendimiento computacional que LSTM
  - Ligeramente menos expresivos



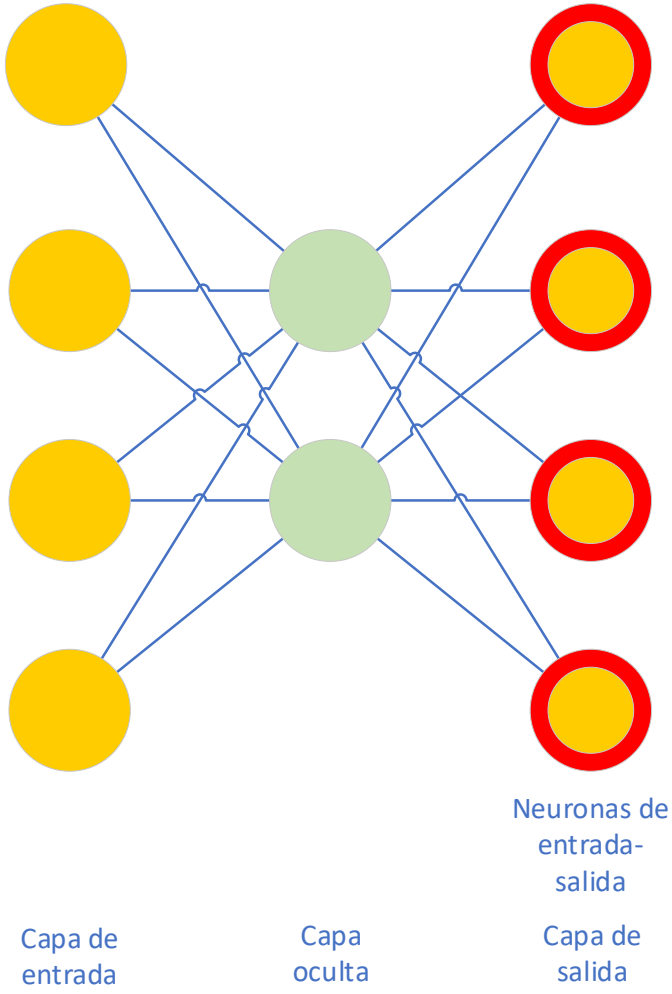
Redes neuronales recurrentes profundas

- **Redes neuronales bidireccionales recurrentes (BiRNN)**
- **Redes bidireccionales de memoria a largo y corto plazo (BiLSTM)**
- **Unidades bidireccionales de puerta recurrente (BiGRU)**

- **No sólo están conectadas al pasado, sino también al futuro**
- Tienen células de entrada y salida de correspondencia en lugar de células de salida y pueden ser entrenadas no sólo para completar los datos al final de una secuencia (final de palabra o imagen), sino para completar datos en medio de dichas secuencias (huecos en medio de palabras o imágenes).

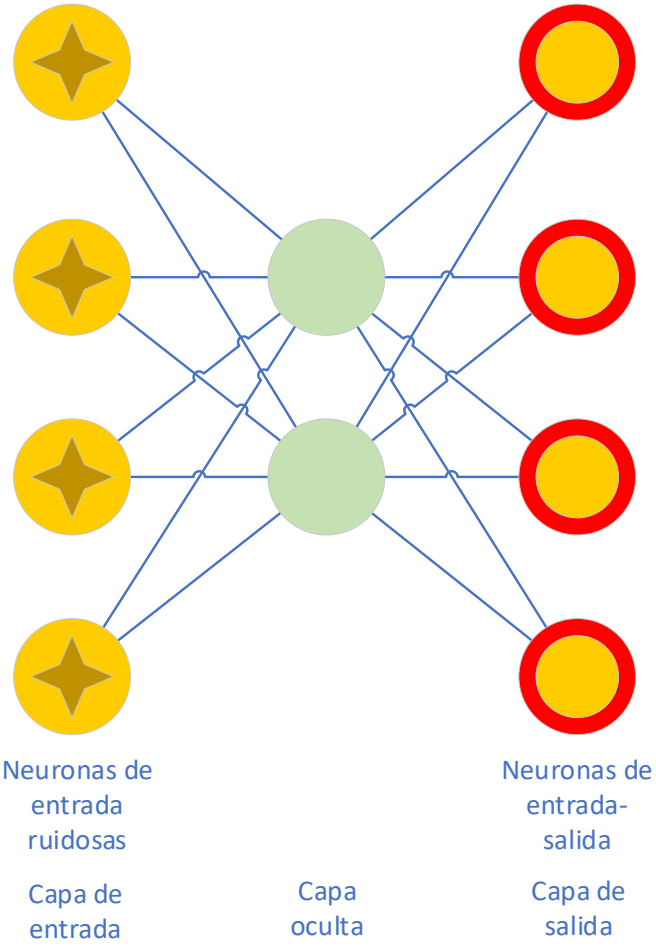
Autoencoders (AE)

- **Redes neuronales simétricas con forma de reloj de arena**
- Capas ocultas más pequeñas que capas de entrada y salida
- La capa o capas medias (depende de si número de capas impar o par) es el código, existe simetría alrededor de él
- De la entrada al código, codifican
- Del código a la salida, decodifican
- Compresión de imágenes, reducción dimensionalidad, generación imágenes, sistemas recomendación



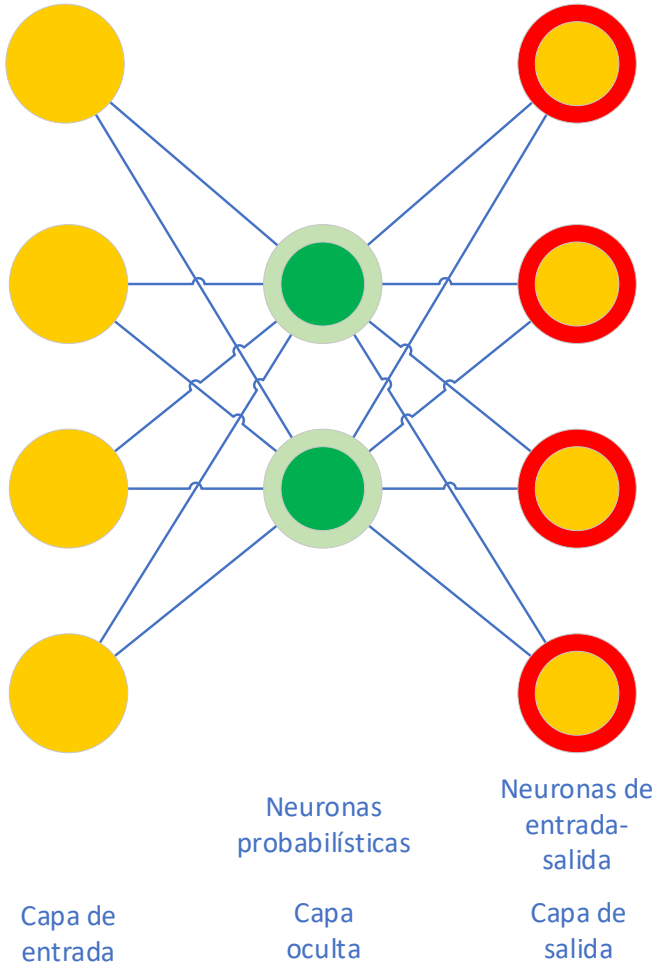
Autoencoders (AE)

- **Denoising AE (DAE) – AE de supresión de ruido**
- Se utilizan para eliminar ruido de la imagen (usando el ruido como entrada en lugar de los datos)



Autoencoders (AE)

- **Variational AE (VAE) – AE variacionales**
- Relacionados con las máquinas de Boltzmann (BM) y las máquinas de Boltzmann restringidas (RBM)
- Basadas en matemáticas bayesianas para modelar la distribución de probabilidad aproximada de las muestras de entrada
- Generación de imágenes y textos, aprendizaje semisupervisado avanzado, interpolación de textos

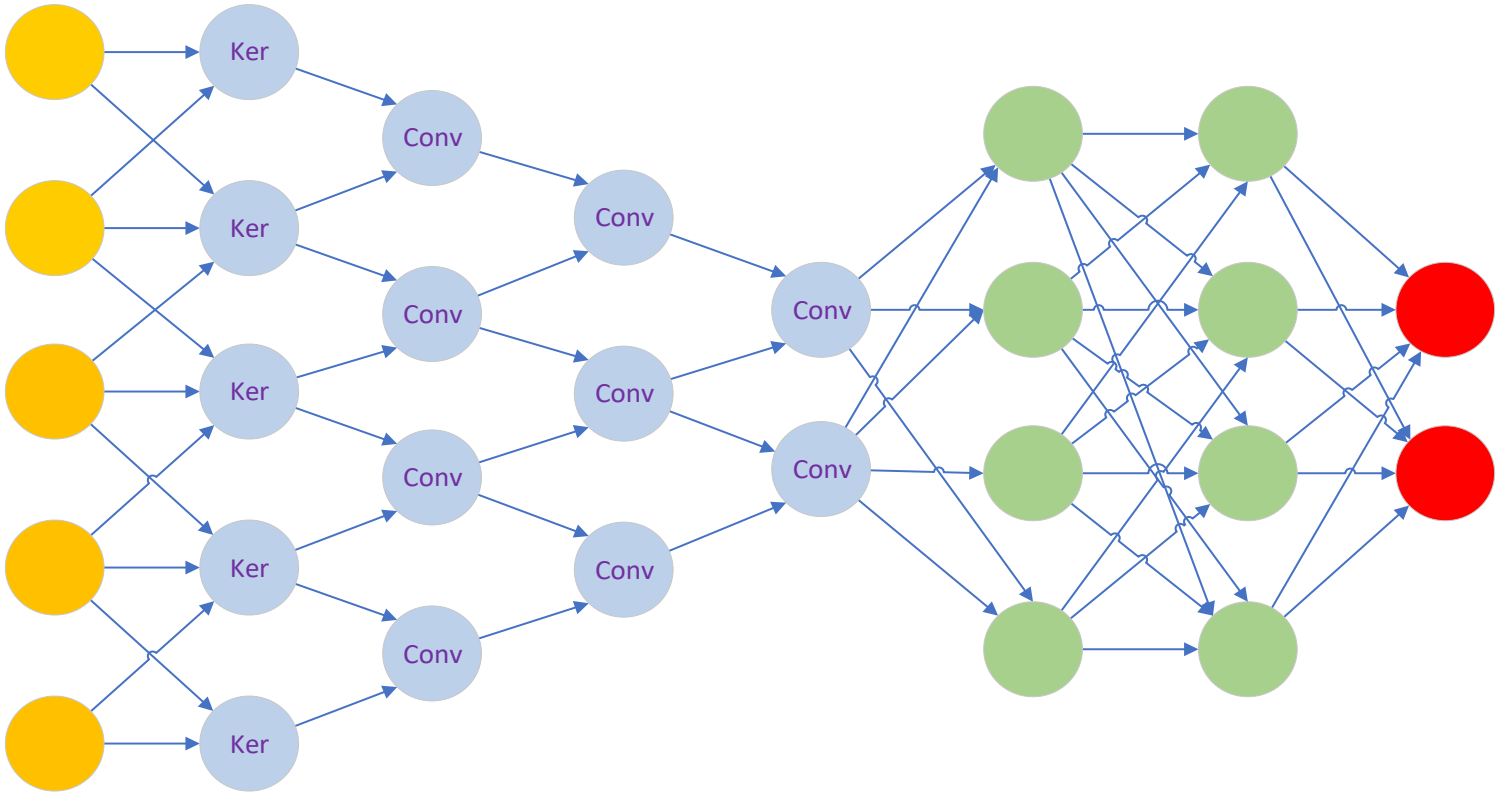


Redes neuronales convolucionales

- Los modelos lineales no funcionan bien para el reconocimiento de imágenes.
- Si queremos reconocer animales u objetos en imágenes y tomar una imagen o plantilla promedio de cada clase (por ejemplo, una para perros, otra para gatos, etc.) para usarla en los datos de entrenamiento y luego usar, por ejemplo, un algoritmo clasificador como el k-NN (u otro) esto no funciona bien.
- **Necesitamos aprovechar las características de las redes neuronales profundas para la clasificación de las imágenes utilizando capas de abstracción.**

Redes neuronales convolucionales

- **Redes neuronales convolucionales (CNN y Deep CNN)**



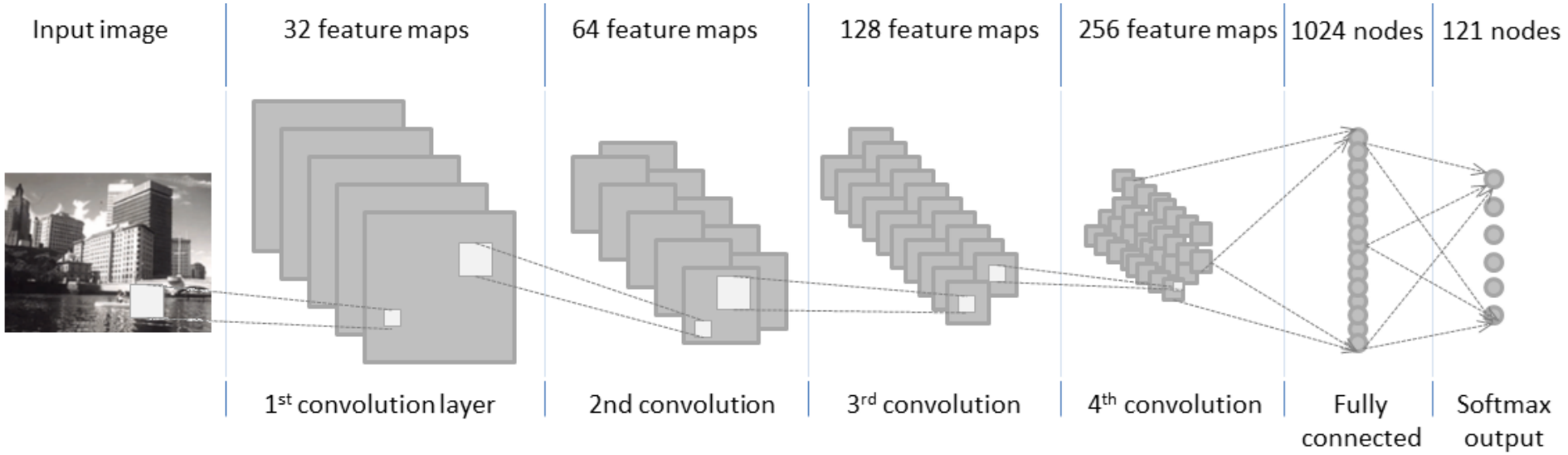
Redes neuronales convolucionales

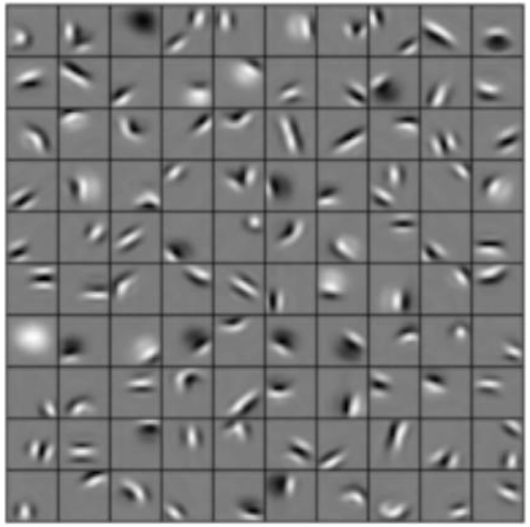
- Dividimos toda la imagen en bloques de *8* × *8* píxeles y asignamos a cada uno un tipo de línea dominante (horizontal, vertical, las dos diagonales, un bloque completamente opaco o completamente vacío, etc.).
  - El resultado es una matriz de líneas que representan los bordes de la imagen.
- En la siguiente capa tomamos de nuevo un bloque de *8* × *8* bloques obtenidos en la etapa anterior y extraemos una nueva salida con nuevas características cada vez más abstractas, repitiendo la operación una y otra vez.
- Esta operación se llama **convolución** y puede ser representada por una capa de la red neuronal, **ya que cada neurona puede actuar como cualquier función**.

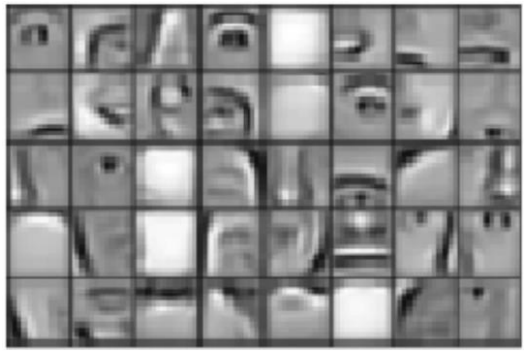
Redes neuronales convolucionales

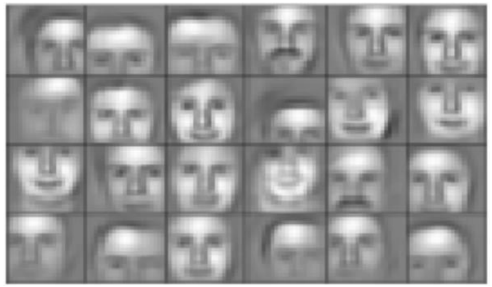
- En las primeras etapas, las neuronas se activan representando la línea dominante en cada célula de 8×8 píxeles.
- En las etapas intermedias las neuronas representan rasgos como la pata o la cabeza.
- En las etapas posteriores las neuronas se activan representando conceptos como el gato o el perro.
- La salida de la última etapa de convolución se conecta a un MLP que actúa como clasificador basado en las características más abstractas, determinando la probabilidad de pertenecer a una clase final (perro o gato, por ejemplo).

Redes neuronales convolucionales

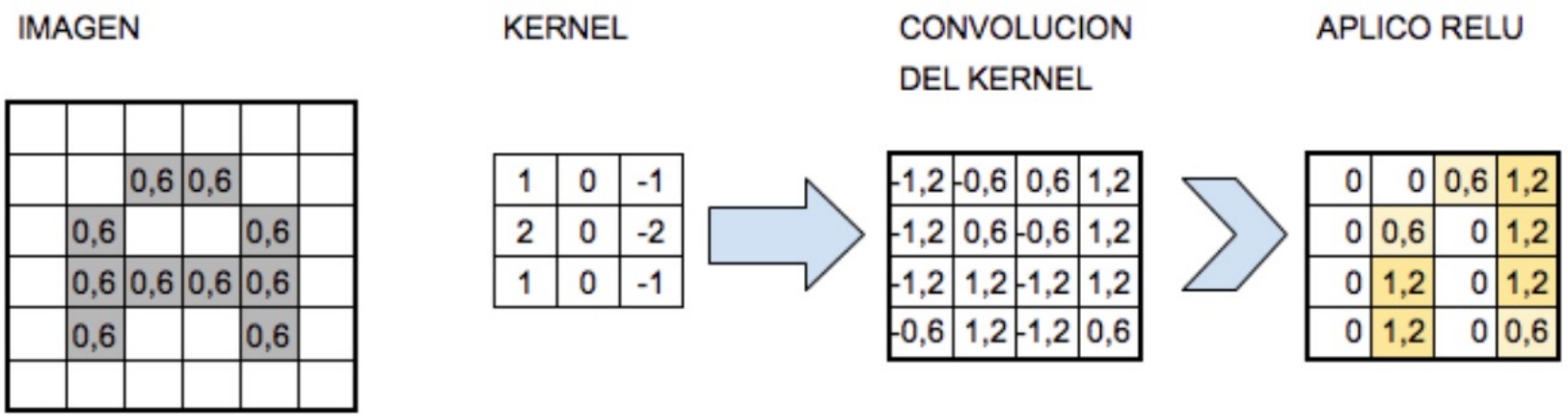




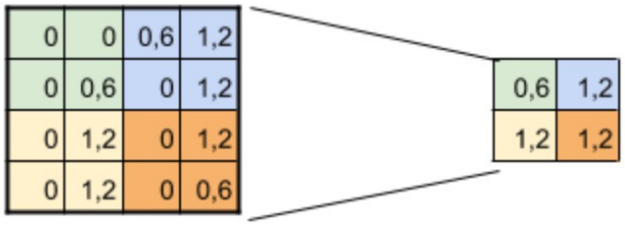


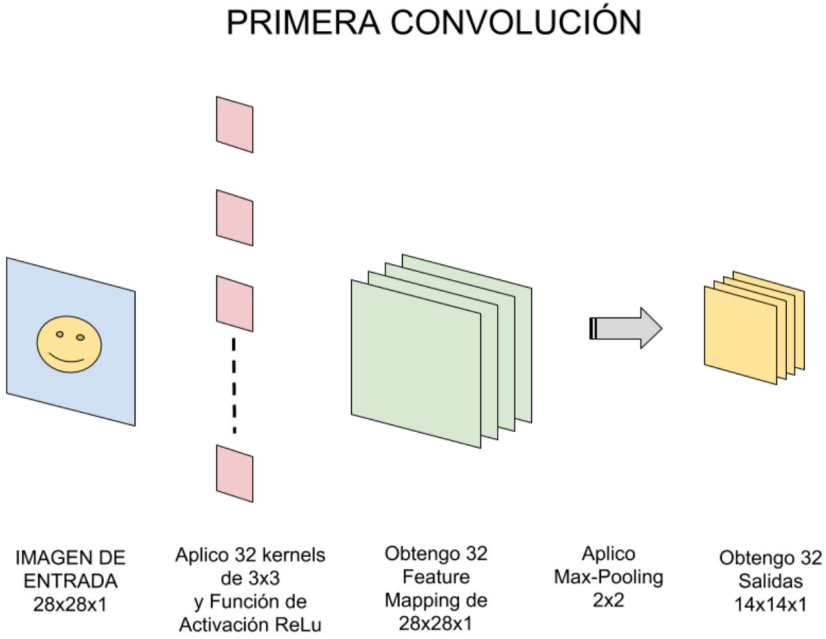


Redes neuronales convolucionales



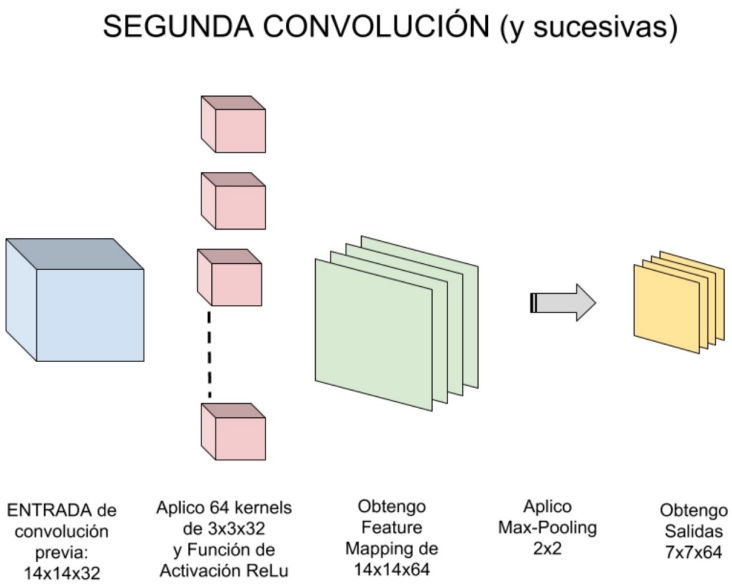
Subsampling – Max-Pooling

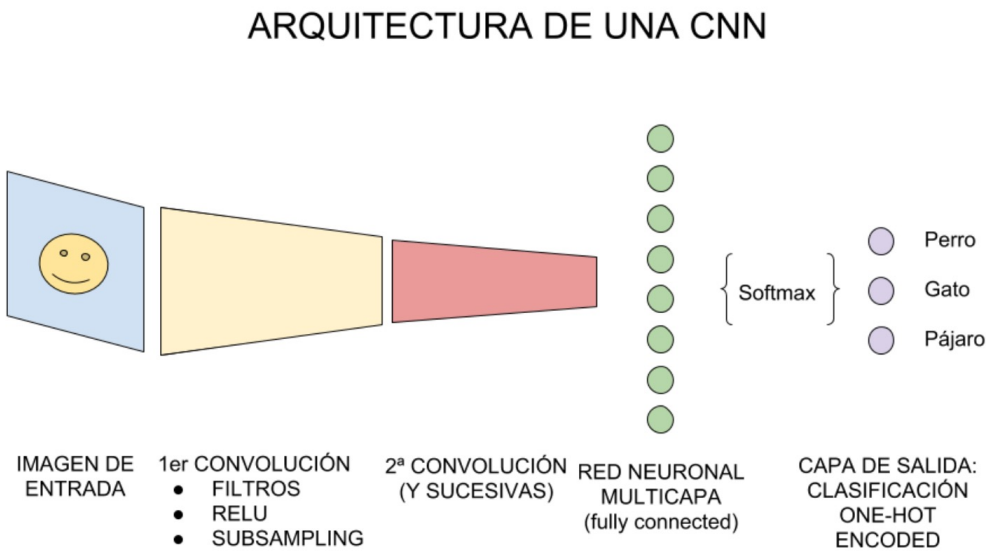




Fuente: https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/

Redes neuronales convolucionales





Fuente: https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/

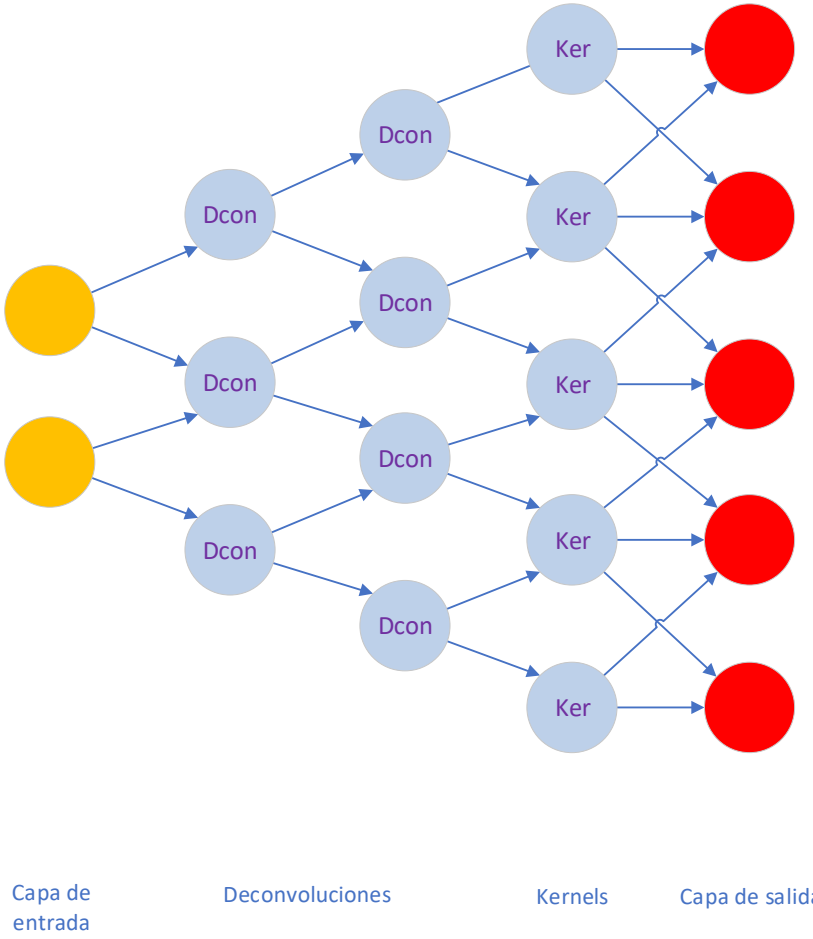
Redes neuronales convolucionales

- **Redes neuronales convolucionales (CNN y Deep CNN)**
  - Búsqueda de objetos en imágenes y vídeos
  - Reconocimiento facial
  - Transferencia de estilos
  - Mejora de calidad de imágenes
  - Etc.
- Ejemplos:
  - https://tensorspace.org/html/playground/index.html
  - Neural Network 3D Simulator:

  https://www.youtube.com/watch?v=3JQ3hYko51Y

Redes neuronales convolucionales

- **Redes deconvolucionales (DN o Deep DN – DDN).**
- **(O Redes Gráficas Inversas)**
- Entrada: “perro” o “gato”
- Salida: imagen sintética
- También para detectar cambios en imágenes



Redes neuronales convolucionales

- **Redes gráficas inversas convolucionales profundas (DCIGN)**
- **Son VAE en las que el codificador en una CNN y el decodificador es una DNN.**
- Tratan de modelar características como probabilidades
- Aplicaciones:
  - Para unir dos objetos en una imagen
  - Eliminar un objeto de una imagen
  - Rotar un objeto en una imagen 3D
  - Modificar la luz
  - Etc.

Redes Generativas Antagónicas (GAN)

- **Dos redes neuronales que trabajan juntas ("compiten")**
- **Normalmente una FFNN y una CNN**
  - **Red generativa:** genera contenido
  - **Red discriminativa:** discriminar el contenido generado por la primera
- **Otra posibilidad:**
  - **Red generativa:** DNN (red deconvolucional generando imágenes)
  - **Red discriminativa:** CNN que juzga las imágenes
  - Síntesis de imágenes artificiales, vídeos en los que se convierten caballos en cebras, personas famosas en otros cuerpos (*Deepfake*)
- https://thispersondoesnotexist.com/

Reinforcement Learning y Deep RL

- Vehículos autónomos
- Robots aspiradores
- Comercio automatizado
- Gestión de recursos empresariales
- Vídeojuegos
- Etc.

Reinforcement Learning y Deep RL

- **DeepMind (adquirida por Google en 2014)**
- 2015: AlphaGo Fan
  - Búsqueda en árbol de Monte Carlo, red de valores, red de políticas
  - Dataset: 30 millones de movimientos de humanos
  - 176 GPU
- 2016: AlphaGo Lee vence a campeones humanos por primera vez en Go
  - 48 TPU
- 2017: AlphaGo Master
  - 4 TPU
- **2017: AlphaGo Zero**
  - **¡Sin dataset de entrenamiento!**
  - **4 TPU**
  - Vence a AlphaGo en 3 días y a AlphaGo Master en 21 días
  - Red entrenada en 64 GPU y 19 CPU previamente
- **2018: Alpha Zero: go, ajedrez, shogi**
  - En 8 horas superior en go a AlphaGo Zero
  - 4 TPU para la inferencia, entrenado por 5000 TPU que competían contra él

Reinforcement Learning y Deep RL

- **Algoritmos genéticos**
- **Basados en selección natural**
- Población inicial de individuos (cromosomas)
  - Posibles soluciones a un problema, generados al azar o ciertas pautas
- Función de aptitud (bondad a la hora de resolver solución)
- Probabilidad de que los individuos se crucen (los más aptos se cruzan más)
- Probabilidad de mutación (cercana a 0, pero no 0 para no converger a soluciones locales)
- **Iteración: evaluación aptitud -> selección -> cruce -> mutación**

Reinforcement Learning y Deep RL

- **Model-based**
  - Existe un data-set previo
- **Model-free**
  - **No existe data-set previo de entrenamiento**
  - ¡Un vehículo autónomo no puede conocer toda la Tierra!
  - Se enfoca en minimizar error (o maximizar recompensa)
    - Estados
    - Acciones
    - Recompensas

Reinforcement Learning y Deep RL - AG

- **Teoría de la evolución**

  Los miembros de una población que se adapten mejor al entorno, sobrevivirán y se reproducirán más probablemente y los hijos heredarán las características de sus progenitores.

  - Existe una población de individuos en un entorno con determinados recursos.
  - Aquellos individuos que mejor se adaptan al entorno se hacen con los recursos y sobreviven.
  - Los mejores individuos que sobreviven se cruzan entre ellos para tener hijos. Es posible que ocurran mutaciones.
  - Los hijos heredan características de los progenitores y dan lugar a una nueva población de individuos, que a su vez han de sobrevivir y competir por los recursos disponibles.
  - La competición por los recursos y selección de los más fuertes se sucede generación tras generación y el resultado es el aumento de la calidad de la población.

    → Conseguir soluciones de mayor calidad tras varias generaciones

Conjunto de posibles soluciones a un problema

Individuos o Cromosomas

Población inicial generada aleatoriamente

Selección

Medida de la bondad: función de fitness

Cruce

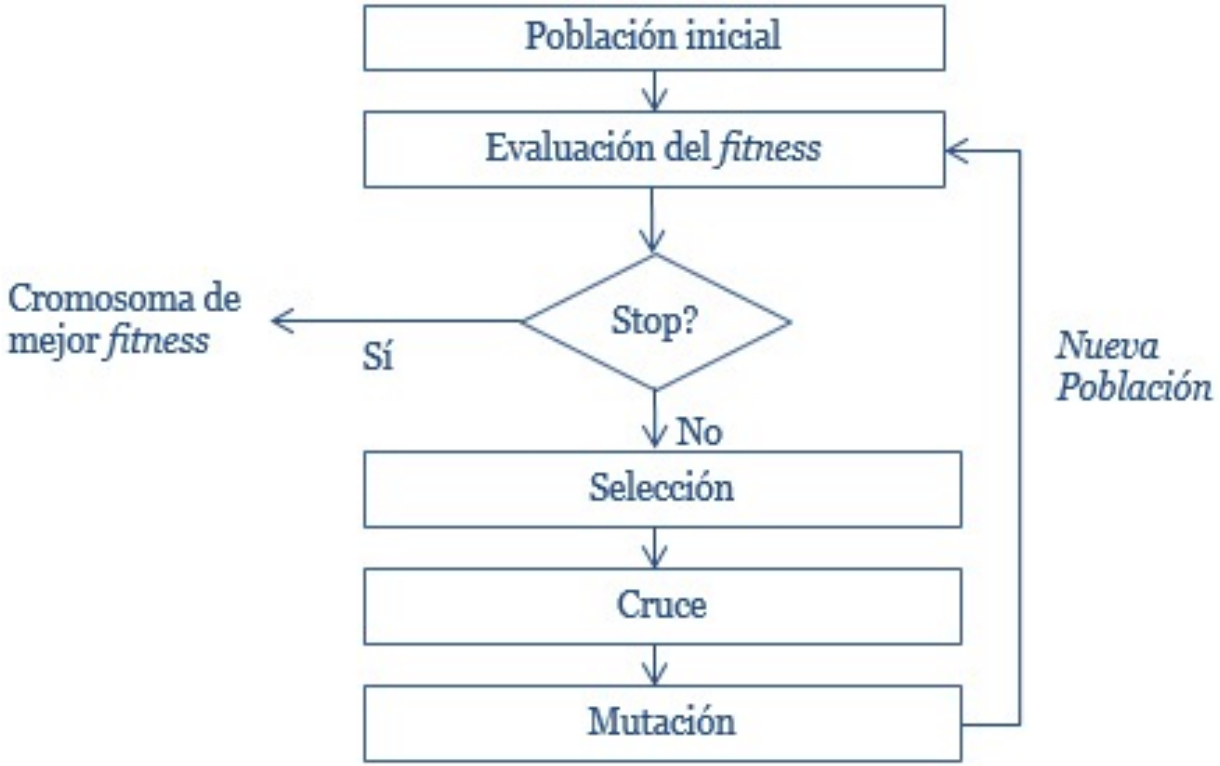
Mutación

Mayor bondad o fitness

Aumento del fitness

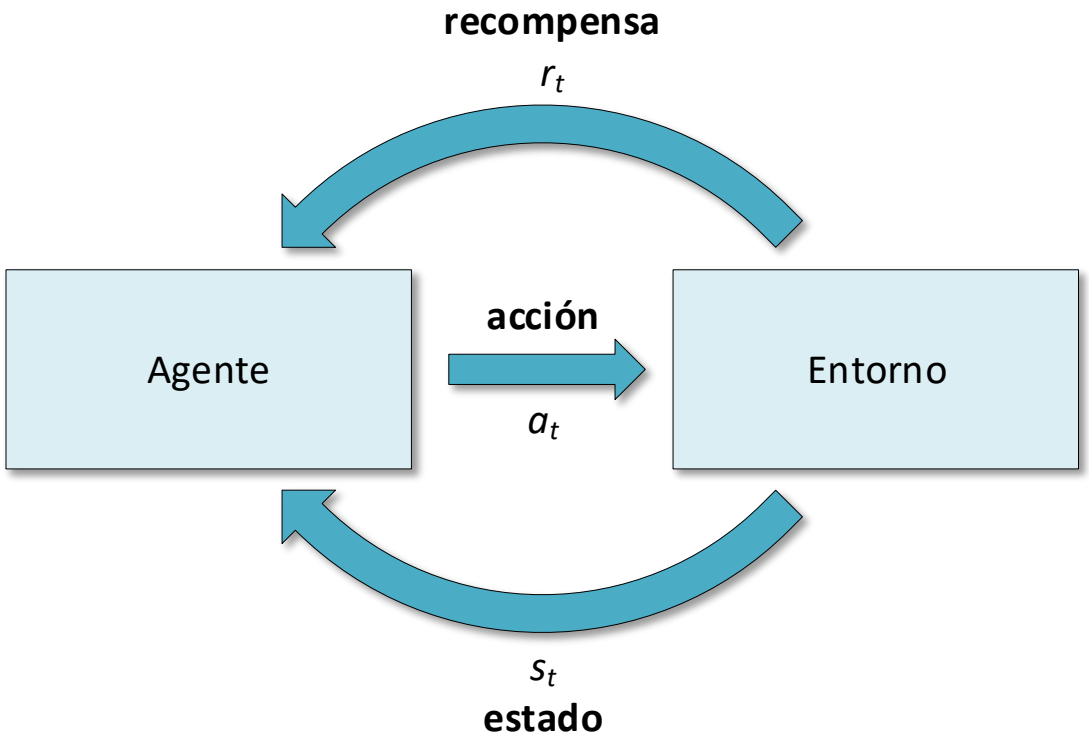
Reinforcement Learning y Deep RL - AG

- **Fases de un algoritmo genético**



Reinforcement Learning y Deep RL - Markov

- **Procesos de decisión de Markov (MDP)**



  - **Maximizar recompensa total futura:** $\sum_{t=0}^{t=\infty} \gamma^t r\left(s(t), a(t)\right)$
  - Factor de descuento **gamma**
    - Cuantifica importancia entre recompensas inmediatas y futuras

Reinforcement Learning y Deep RL

- **Balance o compromiso entre exploración y explotación (*exploration-exploitation trade-off*)**
  - Explotación de recompensas conocidas (acciones con resultados conocidos positivos)
  - Exploración en busca de nuevas recompensas (prueba nuevas acciones y observa nuevos estados)
  - **Estrategia épsilon-codiciosa**: dedicar un porcentaje **épsilon** o fracción del tiempo a explorar en lugar de a explotar
    - Se puede reducir progresivamente en función de los conocimientos adquiridos por el agente

Reinforcement Learning y Deep RL

- **Algoritmos Q-Learning y Deep Q-Networks (DQN)**
- Para cualquier proceso de decisión finito de Markov (FMDP – *Finite Markov Decision Process*), el Q-learning (Q viene de *Quality o* calidad) encuentra una política que es óptima en el sentido de que maximiza el valor esperado de la recompensa total en todos y cada uno de los pasos sucesivos, a partir del estado actual.
- Cada nuevo valor $Q^{new}(s_t, a_t)$ se actualiza en cada iteración a partir del antiguo valor $Q(s_t, a_t)$ con una ***tasa de aprendizaje* α**, ya que el valor aprendido $\left(r_t + \gamma \cdot \max_a Q\,(s_{t+1}, a)\right)$ se conoce, donde r_t es la recompensa, γ es el factor de descuento y $\max_a Q\,(s_{t+1}, a)$ es la estimación del valor óptimo futuro:

$$Q^{new}(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha\left(r_t + \gamma \cdot \max_a Q\,(s_{t+1}, a)\right)$$

TensorFlow

- https://www.tensorflow.org/
- Librería de código abierto muy popular para el cálculo numérico de alto rendimiento
- Desarrollada por el equipo de Google Brain en Google
- Definición y ejecución de cálculos que implican tensores
- *Un **tensor** es cierta clase de entidad algebraica que generaliza los conceptos de escalar, vector y matriz de una forma que sea independiente de cualquier sistema de coordenadas elegido*
- Entrenamiento y ejecución de redes neuronales profundas
- Ampliamente utilizado en el campo de la investigación y aplicación del machine learning

TensorFlow

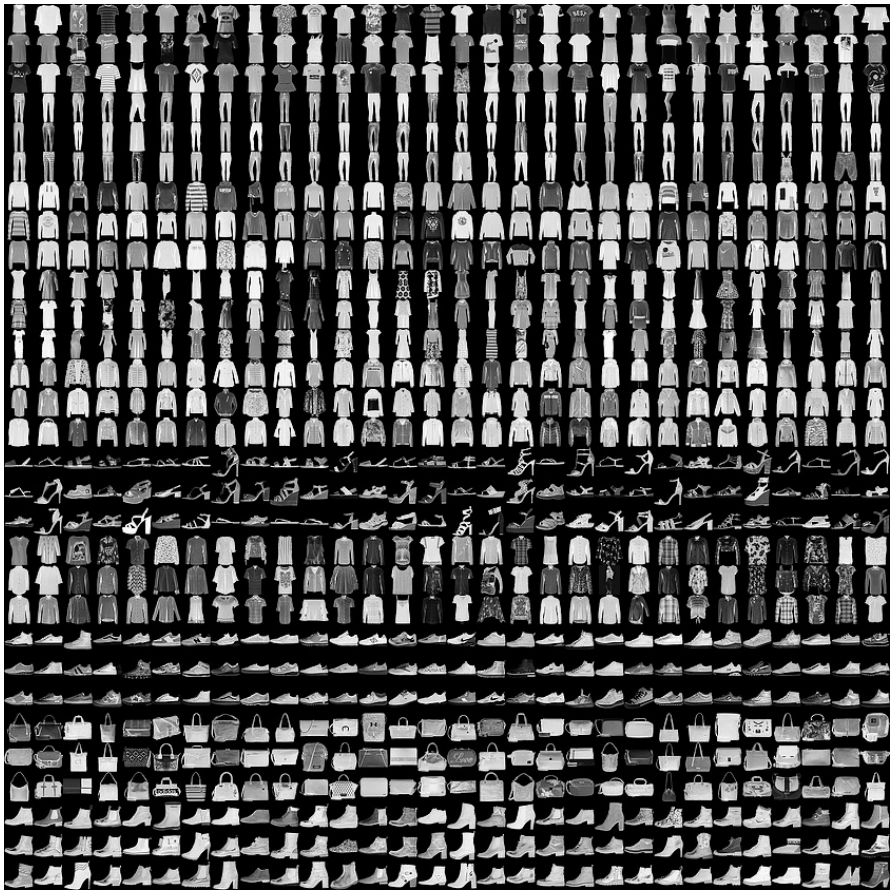
- 2015: Lanzado por Google como código abierto, basado en Python
- 2016: Google lanza las TPU
  - ASIC específico para operar con tensores
  - Aritmética de 8 bit
  - Disponibles en el Cloud
- 2018: Tensorflow.js para JavaScript / Node.js
- 2018: Google Edge TPU
- 2019: TensorFlow 2.0: C++, Haswell, Java, Go y Rust
- También bibliotecas de terceros para C#, R y Scala

TensorFlow

- **Ejemplo de reconocimiento de dígitos escritos a mano**
- MNIST es una gran base de datos de dígitos escritos a mano que se usa comúnmente para entrenar sistemas de procesamiento de imágenes
  - De la web de Lecun (Yann Lecun, h=122, es, junto a Geoffrey Hinton y Yoshua Bengio, los tres *Padrinos de la Inteligencia Artificial* o *Padrinos del Machine Learning*).
- **Nota:** es necesario instalar el paquete `tensorflow` o `tensorflow-gpu` para el funcionamiento del siguiente ejemplo (instalará más de 400MB en el ordenador, incluyendo dependencias, como parte de Keras)
- Además, bajo Windows es necesario contar con la última versión de Visual C++ Redistributable para su funcionamiento:
  - https://support.microsoft.com/en-us/help/2977003/the-latest-supported-visual-c-downloads

TensorFlow

- Utilizaremos 60.000 imágenes para entrenamiento de nuestra red neuronal
- Usaremos 10.000 restantes para el test



TensorFlow

- Exploramos los datos:
  - train_images: array con las imágenes del training dataset (28 × 28 con un entero entre 0 y 255).
  - train_labels: array con las etiquetas del training dataset, un valor de 0 a 9.
  - test_images: array con las imágenes del test dataset.
  - test_labels: array con las etiquetas del test dataset.

TensorFlow

Etiqueta	Clase
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

TensorFlow


```
# Incorporamos los nombres al dataset, pues no se incluyen
from __future__ import absolute_import, division, print_function, u
nicode_literals

# TensorFlow y tf.keras
import tensorflow as tf
from tensorflow import keras

# Librerías de ayuda
import numpy as np
import matplotlib.pyplot as plt

fashion_mnist = keras.datasets.fashion_mnist

(train_images, train_labels), (test_images, test_labels) = fashion_
mnist.load_data()

class_names = ["T-
shirt/top", "Trouser", "Pullover", "Dress", "Coat", "Sandal", "Shir
t", "Sneaker", "Bag", "Ankle boot"]
```

TensorFlow


```
# hay 60 000 imágenes en el training dataset
print(train_images.shape)
#> (60000, 28, 28)
print(len(train_labels))
#> 60000

# cada etiqueta es un número entre 0 y 9
print(train_labels)
#> [9, 0, 0, ..., 3, 0, 5]

# hay 10 000 imágenes en el test dataset
print(test_images.shape)
#> (10000, 28, 28)
print(len(test_labels))
#> 10000
```

TensorFlow

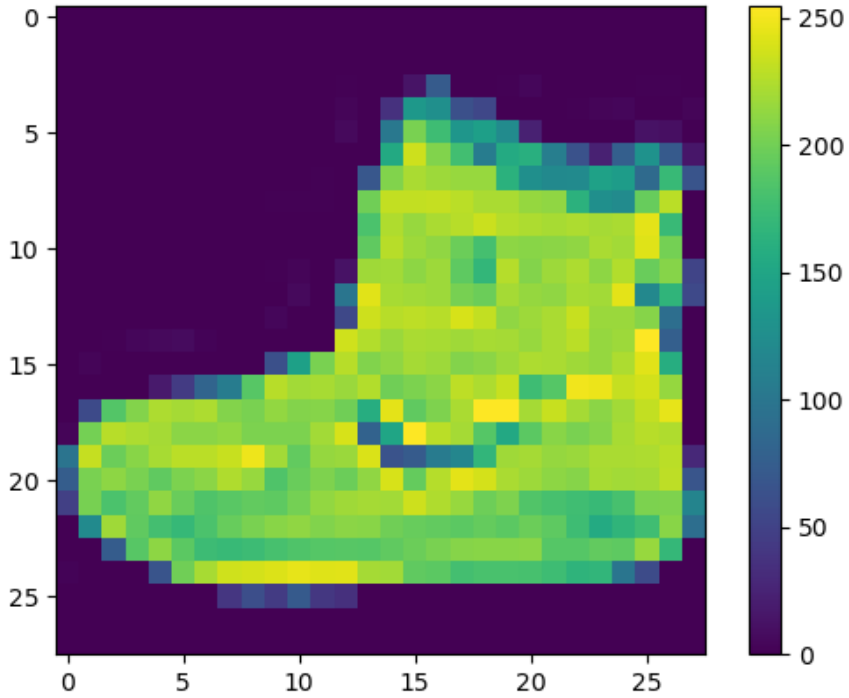

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
32768/29515 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
26427392/26421880 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
8192/5148 [===============================================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4423680/4422102 [==============================] - 0s 0us/step
```

TensorFlow


```
# Seleccionamos el primer elemento antes de preprocesar los datos
from __future__ import absolute_import, division, print_function, unicode_literals

# TensorFlow y tf.keras
import tensorflow as tf
from tensorflow import keras

# Librerías de ayuda
import numpy as np
import matplotlib.pyplot as plt

fashion_mnist = keras.datasets.fashion_mnist

(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data(
)

class_names = ["T-
shirt/top", "Trouser", "Pullover", "Dress", "Coat", "Sandal", "Shirt", "Sneaker", "
Bag", "Ankle boot"]

plt.figure()
plt.imshow(train_images[0])
plt.colorbar()
plt.grid(False)
plt.show()
```

TensorFlow

- Veremos una imagen como la siguiente, en la que vemos los 28 × 28 píxeles, cada uno con un valor entre 0 y 255:



TensorFlow


```
# Normalizamos los datos y mostramos las primeras 25 imágenes

from __future__ import absolute_import, division, print_function, u
nicode_literals

# TensorFlow y tf.keras
import tensorflow as tf
from tensorflow import keras

# Librerías de ayuda
import numpy as np
import matplotlib.pyplot as plt

fashion_mnist = keras.datasets.fashion_mnist

(train_images, train_labels), (test_images, test_labels) = fashion_
mnist.load_data()
```

TensorFlow


```
# normalizamos
train_images = train_images / 255.0
test_images = test_images / 255.0

class_names = ["T-
shirt/top", "Trouser", "Pullover", "Dress", "Coat", "Sandal", "Shir
t", "Sneaker", "Bag", "Ankle boot"]

plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i], cmap=plt.cm.binary)
    plt.xlabel(class_names[train_labels[i]])
plt.show()
```

TensorFlow


Ankle boot
T-shirt/top
T-shirt/top
Dress
T-shirt/top
Pullover
Sneaker
Pullover
Sandal
Sandal
T-shirt/top
Ankle boot
Sandal
Sandal
Sneaker
Ankle boot
Trouser
T-shirt/top
Shirt
Coat
Dress
Trouser
Coat
Bag
Coat

TensorFlow


```
from __future__ import absolute_import, division, print_function
, unicode_literals

# TensorFlow y tf.keras
import tensorflow as tf
from tensorflow import keras

# Librerías de ayuda
import numpy as np
import matplotlib.pyplot as plt

fashion_mnist = keras.datasets.fashion_mnist

(train_images, train_labels), (test_images, test_labels) = fashi
on_mnist.load_data()
```

TensorFlow

```
# normalizamos
train_images = train_images / 255.0
test_images = test_images / 255.0

class_names = ["T-
shirt/top", "Trouser", "Pullover", "Dress", "Coat", "Sandal", "S
hirt", "Sneaker", "Bag", "Ankle boot"]

# Creamos el modelo:
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation="relu"),
    keras.layers.Dense(10, activation="softmax")
])
```

TensorFlow


```
# Compilamos el modelo:
model.compile(optimizer="adam", loss="sparse_categorical_crossen
tropy", metrics=["accuracy"])

# Entrenamos el modelo:
model.fit(train_images, train_labels, epochs=10)

# Evaluamos exactitud:
test_loss, test_acc = model.evaluate(test_images,  test_labels,
verbose=2)
print("\nTest accuracy:", test_acc)
#> Test accuracy: 0.8848999738693237
```

TensorFlow


```
# Realizamos predicciones, mostrando la predicción sobre el prim
er elemento
predictions = model.predict(test_images)
# mostramos las 10 probabilidades
print(predictions[0])
#> [4.3090558e-09 5.8588463e-09 2.3414977e-09 1.4041760e-
07 5.7719642e-09
#> 1.0011349e-04 4.5159254e-07 2.3861572e-02 1.9545018e-
07 9.7603750e-01]
# nos quedamos con la más elevada
print(np.argmax(predictions[0]))
#> 9
```

TensorFlow

```
# Mostremos de forma gráfica todo el conjunto de las predicciones de las
 10 clases
def plot_image(i, predictions_array, true_label, img):
  predictions_array, true_label, img = predictions_array, true_label[i],
 img[i]
  plt.grid(False)
  plt.xticks([])
  plt.yticks([])

  plt.imshow(img, cmap=plt.cm.binary)

  predicted_label = np.argmax(predictions_array)
  if predicted_label == true_label:
    color = "blue"
  else:
    color = "red"
  plt.xlabel("{} {:2.0f}% ({})".format(class_names[predicted_label],
                                       100*np.max(predictions_array),
                                       class_names[true_label]),
                                       color=color)
```

TensorFlow


```
def plot_value_array(i, predictions_array, true_label):
  predictions_array, true_label = predictions_array, true_label[
i]
  plt.grid(False)
  plt.xticks(range(10))
  plt.yticks([])
  thisplot = plt.bar(range(10), predictions_array, color="#77777
7")
  plt.ylim([0, 1])
  predicted_label = np.argmax(predictions_array)

  thisplot[predicted_label].set_color("red")
  thisplot[true_label].set_color("blue")
```

TensorFlow

TensorFlow

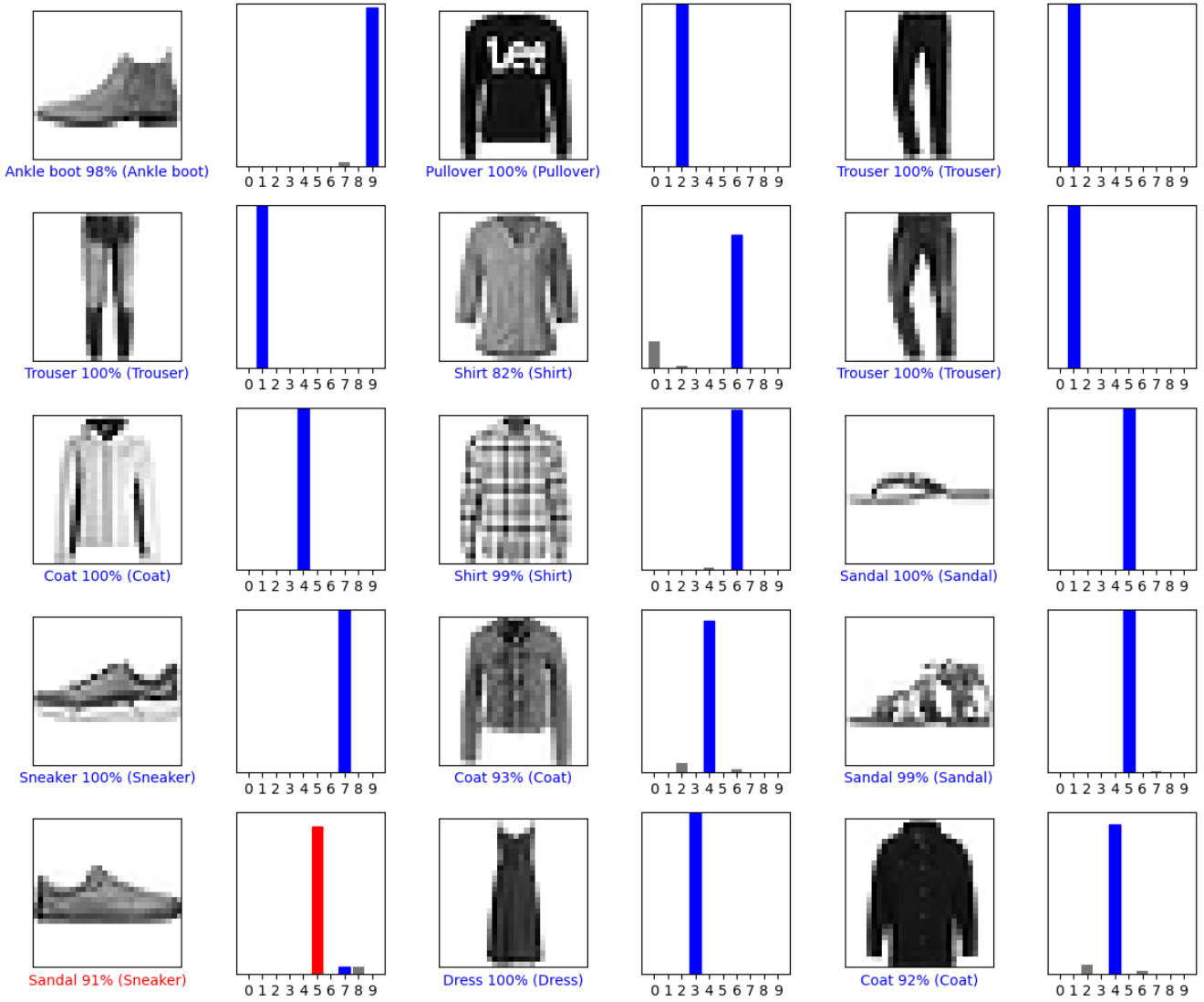
```
# Mostraremos las primeras 15 imágenes de test con sus predicciones y sus etiquetas reales
# En azul las predicciones correctas y en rojo las incorrectas
num_rows = 5
num_cols = 3
num_images = num_rows*num_cols
plt.figure(figsize=(2*2*num_cols, 2*num_rows))
for i in range(num_images):
  plt.subplot(num_rows, 2*num_cols, 2*i+1)
  plot_image(i, predictions[i], test_labels, test_images)
  plt.subplot(num_rows, 2*num_cols, 2*i+2)
  plot_value_array(i, predictions[i], test_labels)
plt.tight_layout()
plt.show()
```

TensorFlow

Al entrenar la red veremos un proceso como el siguiente, para las 10 épocas que se han seleccionado:

```
Epoch 1/10
1875/1875 [==============================] - 5s 3ms/step - loss:
0.4985 - accuracy: 0.8248
Epoch 2/10
1875/1875 [==============================] - 6s 3ms/step - loss:
0.3716 - accuracy: 0.8666
0.2772 - accuracy: 0.8975
...
Epoch 9/10
1875/1875 [==============================] - 4s 2ms/step - loss:
0.2463 - accuracy: 0.9074
Epoch 10/10
1875/1875 [==============================] - 4s 2ms/step - loss:
0.2375 - accuracy: 0.9103
313/313 - 0s - loss: 0.3325 - accuracy: 0.8849
```

TensorFlow


Ankle boot 98% (Ankle boot)
Pullover 100% (Pullover)
Trouser 100% (Trouser)
Trouser 100% (Trouser)
Shirt 82% (Shirt)
Trouser 100% (Trouser)
Coat 100% (Coat)
Shirt 99% (Shirt)
Sandal 100% (Sandal)
Sneaker 100% (Sneaker)
Coat 93% (Coat)
Sandal 99% (Sandal)
Sandal 91% (Sneaker)
Dress 100% (Dress)
Coat 92% (Coat)
0 1 2 3 4 5 6 7 8 9

Referencias bibliográficas

- Alcantarilla, P.F., Stent, S., Ros, G., Arroyo, R., Gherardi, R. (2018). Street-view change detection with deconvolutional networks. *Autonomous Robots* 42, 1301–1322.
- Alex, V., Vaidhya, K., Thirunavukkarasu, S., Kesavadas, C., & Krishnamurthi, G. (2017). Semisupervised learning using denoising autoencoders for brain lesion detection and segmentation. *Journal of Medical Imaging*, *4*(4), 041311.
- Alonso, R. S., Tapia, D. I., Bajo, J., García, Ó., De Paz, J. F., & Corchado, J. M. (2013). Implementing a hardware-embedded reactive agents platform based on a service-oriented architecture over heterogeneous wireless sensor networks. *Ad Hoc Networks*, *11*(1), 151-166.
- Arulkumaran, K., Deisenroth, M.P., Brundage, M., Bharath, A.A. (2017). Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine* 34, 26–38.

Referencias bibliográficas

- Bengio, Y., Lamblin, P., Popovici, D., & Larochelle, H. (2007). Greedy layer-wise training of deep networks. In *Advances in neural information processing systems* (pp. 153-160).
- Bora, T.C., Mariani, V.C., dos Santos Coelho, L. (2019). Multi-objective optimization of the environmental-economic dispatch with reinforcementlearning based on non-dominated sorting genetic algorithm. *Applied Thermal Engineering* 146, 688–700.
- Broomhead, D. S., & Lowe, D. (1988). *Radial basis functions, multi-variable functional interpolation and adaptive networks* (No. RSRE-MEMO-4148). Royal Signals and Radar Establishment Malvern (United Kingdom).
- Cao, R., Freitas, C., Chan, L., Sun, M., Jiang, H., & Chen, Z. (2017). ProLanGO: protein function prediction using neural machine translation based on a recurrent neural network. *Molecules*, *22*(10), 1732.

Referencias bibliográficas

- Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.
- Costa-jussà, M. R., Allauzen, A., Barrault, L., Cho, K., & Schwenk, H. (2017). Introduction to the special issue on deep learning approaches for machine translation. *Computer Speech & Language*, 46, 367-373.
- De Paz, J. F., Tapia, D. I., Alonso, R. S., Pinzón, C. I., Bajo, J., & Corchado, J. M. (2013). Mitigation of the ground reflection effect in real-time locating systems based on wireless sensor networks by using artificial neural networks. *Knowledge and information systems*, *34*(1), 193-217.

Referencias bibliográficas

- Dechter, R. (1986). *Learning while searching in constraint-satisfaction problems* (pp. 178-183). University of California, Computer Science Department, Cognitive Systems Laboratory.
- Deng, L., & Liu, Y. (Eds.). (2018). *Deep learning in natural language processing*. Springer.
- Faia, R., Pinto, T., Vale, Z., & Corchado, J. M. (2018). Case-based reasoning using expert systems to determine electricity reduction in residential buildings. In *2018 IEEE Power & Energy Society General Meeting* (PESGM) (pp. 1-5). IEEE.
- Fayek, H. M., Lech, M., & Cavedon, L. (2017). Evaluating deep learning architectures for Speech Emotion Recognition. *Neural Networks*, 92, 60-68.
- García, Ó., Prieto, J., Alonso, R. S., & Corchado, J. M. (2017). A framework to improve energy efficient behaviour at home through activity and context monitoring. *Sensors*, *17*(8), 1749.

Referencias bibliográficas

- Gatys, L. A., Ecker, A. S., & Bethge, M. (2016). Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2414-2423).
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y. (2014). Generative adversarial nets, in: *Advances in neural information processing systems*, pp. 2672–2680.
- Hu, B., Shi, C., Liu, J. (2017). Playlist recommendation based on reinforcement learning, in: *International Conference on Intelligence Science*, Springer. pp. 172–182
- Jouppi, N., Young, C., Patil, N., & Patterson, D. (2018). Motivation for and evaluation of the first tensor processing unit. *IEEE Micro*, *38*(3), 10-19.
- Karpathy, A. (2016). Cs231n convolutional neural networks for visual recognition. *Neural networks*, *1*, 1.

Referencias bibliográficas

- Kim, J., Kim, J., Lee, S., Park, J., & Hahn, M. (2016, November). Vowel based voice activity detection with LSTM recurrent neural network. In *Proceedings of the 8th International Conference on Signal Processing Systems* (pp. 134-137).
- Kulkarni, T.D., Whitney, W.F., Kohli, P., Tenenbaum, J. (2015). Deep convolutional inverse graphics network, in: Advances in neural informationprocessing systems, pp. 2539–2547
- Lample, G., Chaplot, D.S., 2017. Playing fps games with deep reinforcement learning, in: *Thirty-First AAAI Conference on Artificial Intelligence*.
- LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436-444.
- Leike, J., Krueger, D., Everitt, T., Martic, M., Maini, V., Legg, S. (2018). Scalable agent alignment via reward modeling: a research direction. arXiv preprint arXiv:1811.07871 .

Referencias bibliográficas

- Lima, A. C. E., de Castro, L. N., & Corchado, J. M. (2015). A polarity analysis framework for Twitter messages. *Applied Mathematics and Computation*, 270, 756-767.
- Liu, Y.J., Cheng, S.M., Hsueh, Y.L. (2017). enb selection for machine type communications using reinforcement learning based markov decision process. *IEEE Transactions on Vehicular Technology* 66, 11330–11338.
- Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., & Alsaadi, F. E. (2017). A survey of deep neural network architectures and their applications. *Neurocomputing*, *234*, 11-26.
- Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, O.P., Mordatch, I. (2017). Multi-agent actor-critic for mixed cooperative-competitive environments, in: *Advances in Neural Information Processing Systems*, pp. 6379–6390.
- Luan, F., Paris, S., Shechtman, E., & Bala, K. (2017). Deep photo style transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 4990-4998).

Referencias bibliográficas

- Ma, L., Jia, X., Sun, Q., Schiele, B., Tuytelaars, T., Van Gool, L. (2017). Pose guided person image generation, in: *Advances in Neural Information Processing Systems*, pp. 406–416.
- Marvin, M., & Seymour, A. P. (1969). Perceptrons.
- Mesnil, G., Dauphin, Y., Yao, K., Bengio, Y., Deng, L., Hakkani-Tur, D., ... & Zweig, G. (2014). Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *23*(3), 530-539.
- Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., Kavukcuoglu, K. (2016). *Asynchronous methods for deep reinforcement learning, in: International conference on machine learning*, pp. 1928–1937
- Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature* 518, 529

Referencias bibliográficas

- Nachum, O., Norouzi, M., Xu, K., & Schuurmans, D. (2017). Bridging the gap between value and policy based reinforcement learning. In *Advances in Neural Information Processing Systems* (pp. 2775-2785).
- Naraei, P., Abhari, A., & Sadeghian, A. (2016). Application of multilayer perceptron neural networks and support vector machines in classification of healthcare data. In *2016 Future Technologies Conference (FTC)* (pp. 848-852). IEEE.
- Pal, S. K., & Wang, P. P. (2017). *Genetic algorithms for pattern recognition*. CRC press.
- Park, D. H., & Chiba, R. (2017). A neural language model for query auto-completion. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 1189-1192).
- Panesar, A. (2019). What Is Machine Learning?. In *Machine Learning and AI for Healthcare* (pp. 75-118). Apress, Berkeley, CA.

Referencias bibliográficas

- Partila, P., Tovarek, J., Voznak, M., Rozhon, J., Sevcik, L., & Baran, R. (2018). Multi-classifier speech emotion recognition system. In *2018 26th Telecommunications Forum (TELFOR)* (pp. 1-4). IEEE.
- Phon-Amnuaisuk, S. (2011). Learning chasing behaviours of non-player characters in games using SARSA, in: *European Conference on the Applications of Evolutionary Computation*, Springer. pp. 133–142.
- Phon-Amnuaisuk, S. (2017). What does a policy network learn after mastering a pong game?, in: *International Workshop on Multi-disciplinary Trends in Artificial Intelligence*, Springer. pp. 213–222.
- Ramchoun, H., Idrissi, M. A. J., Ghanou, Y., & Ettaouil, M. (2016). Multilayer Perceptron: Architecture Optimization and Training. *IJIMAI*, *4*(1), 26-30.
- Razzak, M. I., Naz, S., & Zaib, A. (2018). Deep learning for medical image processing: *Overview, challenges and the future. In Classification in BioApps* (pp. 323-350). Springer, Cham.

Referencias bibliográficas

- Rivas, A., Chamoso, P., González-Briones, A., & Corchado, J. M. (2018). Detection of cattle using drones and convolutional neural networks. *Sensors*, *18*(7), 2048.
- Rodríguez, L. P., Crespo, A. G., Lara, M. P., & Mezcua, B. R. (2008). Study of different fusion techniques for multimodal biometric authentication. In *2008 IEEE international conference on wireless and mobile computing, networking and communications* (pp. 666-671). IEEE.
- Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, *65*(6), 386.
- Rosenblatt, F. (1960). Perceptron simulation experiments. *Proceedings of the IRE*, *48*(3), 301-309.
- Rumelhart, D. E., & McClelland, J. L. (1986). The PDP Research Group: Parallel distributed processing: Explorations in the microstructure of cognition. *Foundations*, *1*, 3-44.

Referencias bibliográficas

- Russell, S., & Norvig, P. (2002). Artificial intelligence: a modern approach.
- Sallab, A.E., Abdou, M., Perot, E., Yogamani, S. (2017). Deep reinforcement learning framework for autonomous driving. *Electronic Imaging* 2017,70–76.
- Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, *3*(3), 210-229.
- Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al. (2019). *Mastering atari, go, chess and shogi by planning with a learned model*. arXiv preprint arXiv:1911.08265
- Semeniuta, S., Severyn, A., & Barth, E. (2017). A hybrid convolutional variational autoencoder for text generation. *arXiv preprint arXiv:1702.02390*.
- Seo, J., Han, S., Lee, S., & Kim, H. (2015). Computer vision techniques for construction safety and health monitoring. *Advanced Engineering Informatics*, *29*(2), 239-251.

Referencias bibliográficas

- Shah, S. A. A., Bennamoun, M., & Boussaid, F. (2016). Iterative deep learning for image set based face and object recognition. *Neurocomputing*, 174, 866-874.
- Shin, H. C., Roth, H. R., Gao, M., Lu, L., Xu, Z., Nogues, I., ... & Summers, R. M. (2016). Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning. *IEEE transactions on medical imaging*, *35*(5), 1285-1298.
- Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., ... & Dieleman, S. (2016). Mastering the game of Go with deep neural networks and tree search. *nature*, *529*(7587), 484.
- Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al., (2017). Mastering the game of go without human knowledge. *Nature* 550, 354–359

Referencias bibliográficas

- Sledge, I.J., Príncipe, J.C. (2017). Balancing exploration and exploitation in reinforcement learning using a value of information criterion, in: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE. pp. 2816–2820.
- Tan, C. C., & Eswaran, C. (2011). Using autoencoders for mammogram compression. *Journal of medical systems*, *35*(1), 49-58.
- Turing, I. B. A. (1950). Computing machinery and intelligence-AM Turing. *Mind*, *59*(236), 433.
- Van Engelen, J. E., & Hoos, H. H. (2020). A survey on semi-supervised learning. *Machine Learning*, *109*(2), 373-440.
- Van Hasselt, H. (2010). Double Q-learning. In *Advances in neural information processing systems* (pp. 2613-2621).

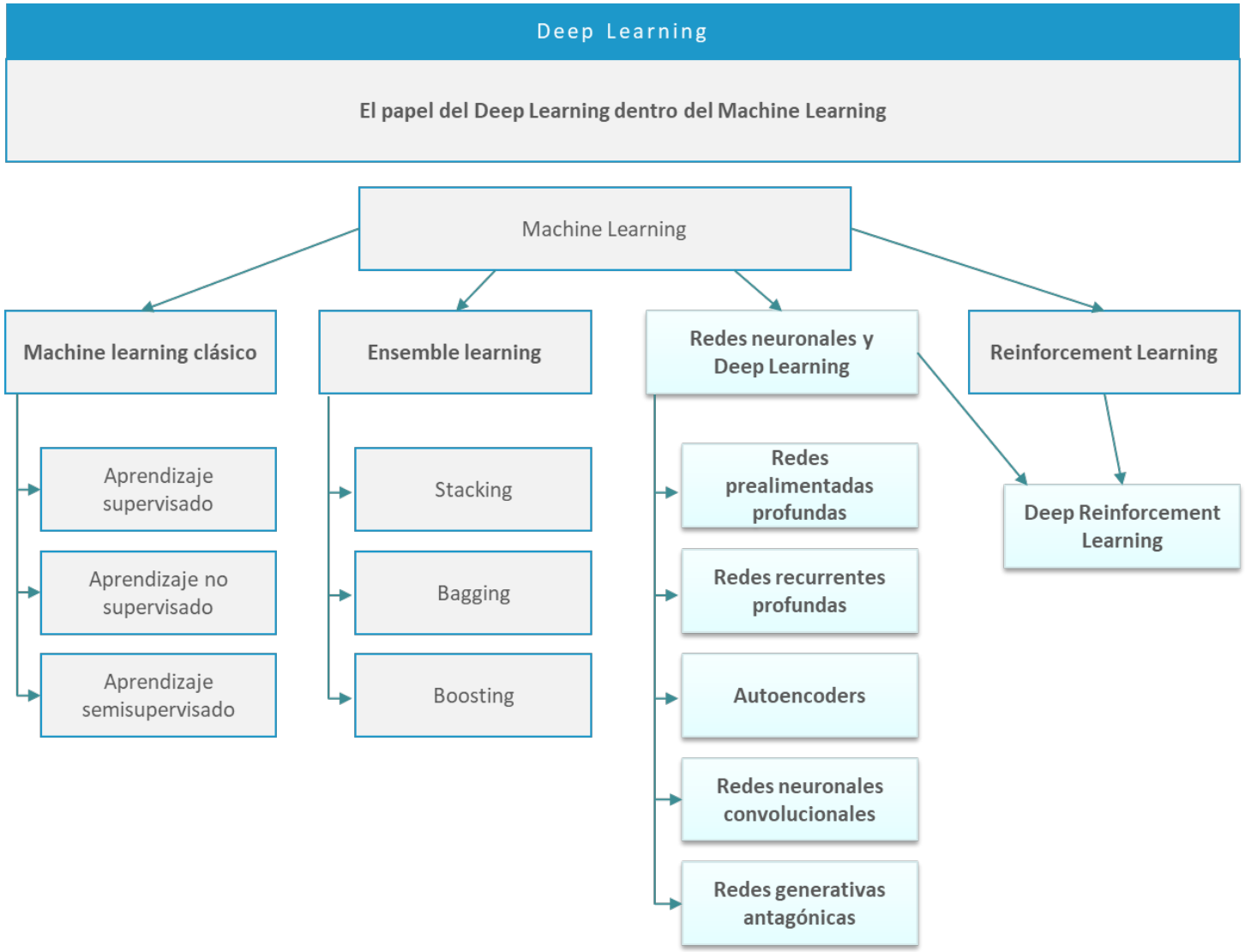
Referencias bibliográficas

- Van Hasselt, H., Guez, A., & Silver, D. (2016). Deep reinforcement learning with double q-learning. In *Thirtieth AAAI conference on artificial intelligence*.
- Watkins, C.J., Dayan, P., 1992. Q-learning. *Machine learning* 8, 279–292
- Wu, Z., Shen, C., & Van Den Hengel, A. (2019). Wider or deeper: Revisiting the resnet model for visual recognition. *Pattern Recognition*, 90, 119-133.
- Wu, K., Wang, H., Esfahani, M. A., & Yuan, S. (2019). BND*-DDQN: Learn to Steer Autonomously through Deep Reinforcement Learning. *IEEE Transactions on Cognitive and Developmental Systems*.
- Yasnitsky, L. N. (2019). Whether Be New “Winter” of Artificial Intelligence?. In *International Conference on Integrated Science* (pp. 13-17). Springer, Cham.

Referencias bibliográficas

- Young, T., Hazarika, D., Poria, S., & Cambria, E. (2018). Recent trends in deep learning based natural language processing. *IEEE Computational intelligenCe magazine*, *13*(3), 55-75.
- Zabalza, J., Ren, J., Zheng, J., Zhao, H., Qing, C., Yang, Z., ... & Marshall, S. (2016). Novel segmented stacked autoencoder for effective dimensionality reduction and feature extraction in hyperspectral imaging. *Neurocomputing*, *185*, 1-10.
- Zhang, Z., Geiger, J., Pohjalainen, J., Mousa, A. E. D., Jin, W., & Schuller, B. (2018). Deep learning for environmentally robust speech recognition: An overview of recent developments. *ACM Transactions on Intelligent Systems and Technology* (TIST), 9(5), 1-28.

Resumen


Deep Learning
El papel del Deep Learning dentro del Machine Learning
Machine Learning
Machine learning clásico
Ensemble learning
Redes neuronales y Deep Learning
Reinforcement Learning
Aprendizaje supervisado
Aprendizaje no supervisado
Aprendizaje semisupervisado
Stacking
Bagging
Boosting
Redes prealimentadas profundas
Redes recurrentes profundas
Autoencoders
Redes neuronales convolucionales
Redes generativas antagónicas
Deep Reinforcement Learning

Gracias por vuestra atención
¿Dudas?

Imagen por Peggy und Marco Lachmann-Anke
Licencia: Creative Commons Zero

www.unir.net