IES MARÍA DE ZAYAS
Y SOTOMAYOR

Redes Neuronales y Deep Learning

Jose Antonio Castro Grilli

Sobre mí

- **Ingeniero Informático** – UNED
- **PFG** – *Detección de cáncer de mama mediante Deep Learning*
- Surge del proyecto de investigación de la UNED: "*Cribado coste-efectivo de cáncer de mama mediante mamografía, ecografía y termografía*", financiado por el Ministerio de Ciencia e Innovación
- Desarrollador de software

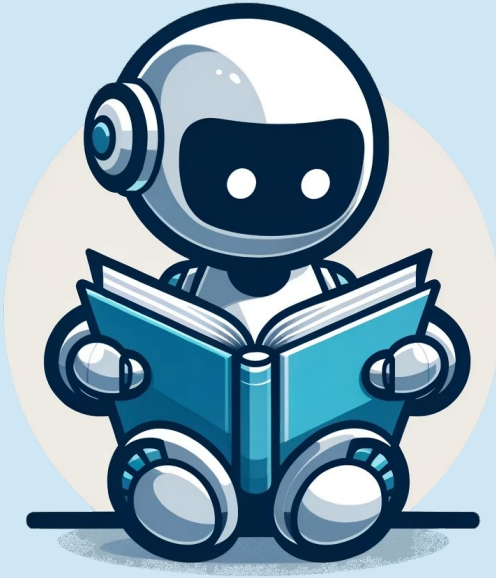
Índice

Inteligencia Artificial

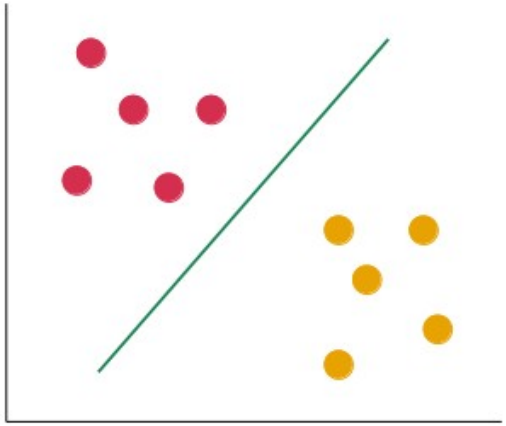
- Campo de las ciencias de la computación que estudia y desarrolla sistemas capaces de realizar tareas que requieren inteligencia.
- Subcampos:
  - Razonamiento y solución de problemas
  - Representación del conocimiento
  - Robótica
  - Procesamiento del Lenguaje Natural (NLP)
  - Percepción
  - Aprendizaje Automático (Machine Learning)

Aprendizaje Automático

- Capacidad de aprender **sin intervención humana** directa
- Algoritmos:
  - Espacio de Versiones, FOIL, k-NN, K-m, ID3, Naive-Bayes, ...
  - **Redes Neuronales**
- Tipos:
  - **Aprendizaje Supervisado**
    - Ejemplo: Clasificación de imágenes
  - **Aprendizaje No Supervisado**
    - Ejemplo: Segmentación de clientes en un mercado
  - **Aprendizaje por Refuerzo**
    - Ejemplo: Sistema de recomendación de películas

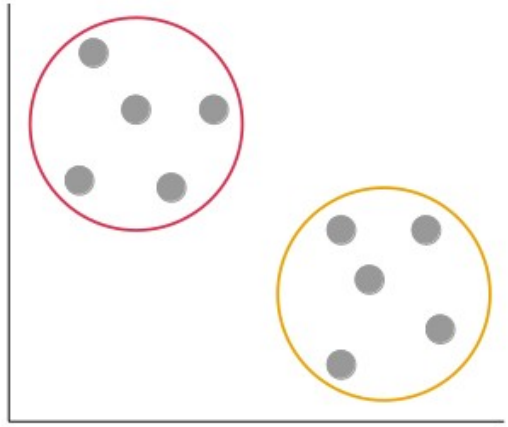


Clasificación

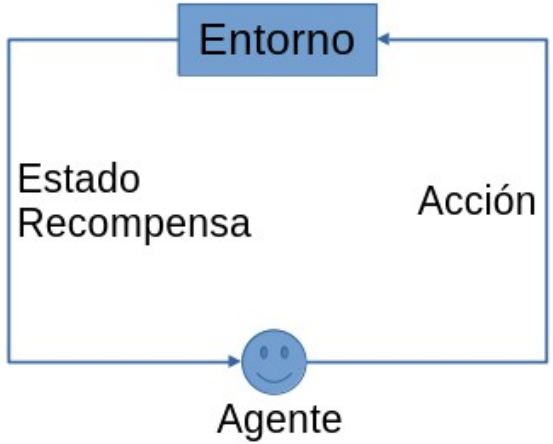


Aprendizaje Supervisado

Agrupación



Aprendizaje No Supervisado

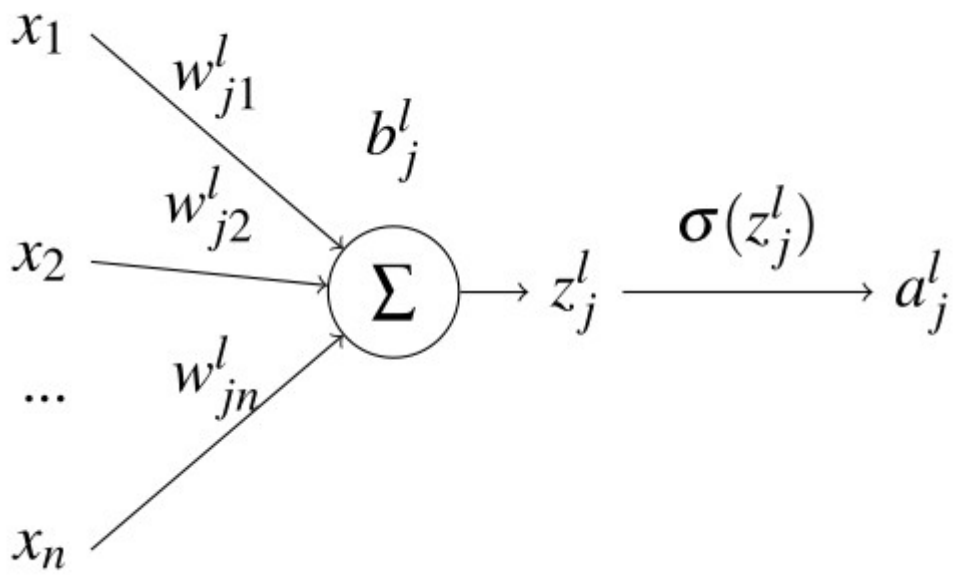


Aprendizaje por Refuerzo

Redes Neuronales

- Estructuras lógicas que se asemejan al sistema nervioso de los seres vivos
- La **neurona artificial**, o *perceptrón*, es la unidad elemental de una red neuronal
- Una **red neuronal** consiste en un conjunto de neuronas artificiales interconectadas, organizadas en capas
- Número de capas → profundidad → **Deep Learning**

Neurona Artificial

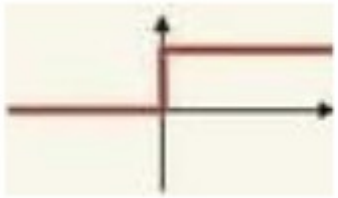


$$z^l_j = \sum_{i=1}^{n} w^l_{ji} x_i + b^l_j$$

$$a^l_j = \sigma(z^l_j)$$

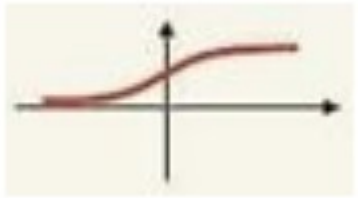
Funciones de activación

$$a_j^l = \sigma(z_j^l)$$



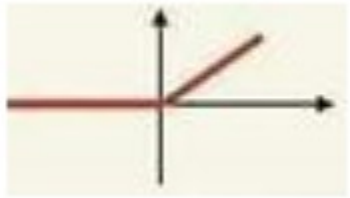
Escalonada

$$\sigma(z) = \begin{cases} 0 & si\ z \leq umbral \\ 1 & si\ z > umbral \end{cases}$$



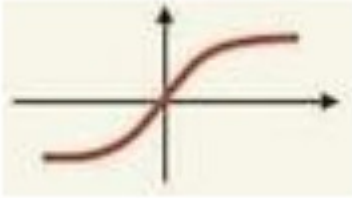
Sigmoide

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



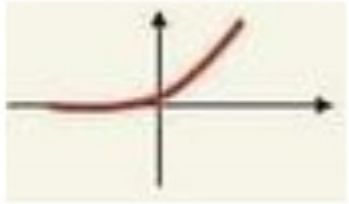
ReLU

$$\sigma(z) = max(0,z)$$



Tangente hiperbólica

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

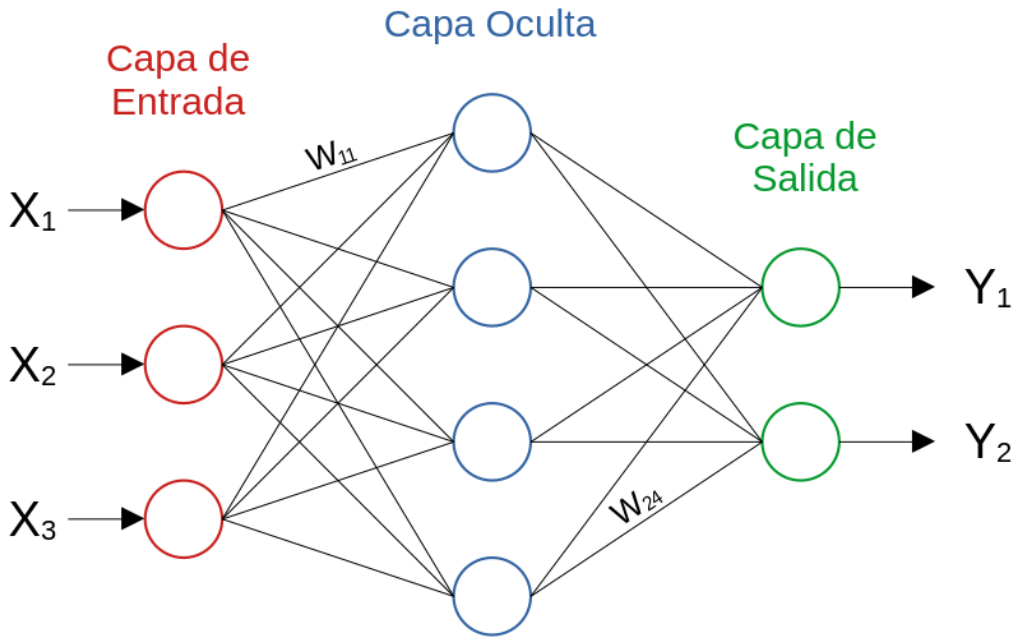


SoftPlus

$$\sigma(z) = ln(1+e^z)$$

Arquitectura

- Capas: capa de **entrada**, capas **ocultas** y capa de **salida**
- **Deep Learning**: muchas capas ocultas
- Las salidas de las neuronas de una capa se corresponden con las entradas de las neuronas de la siguiente capa
- Dos tipos principales:
  - **FNN** (Feed-forward Neural Network)
  - **RNN** (Recurrent Neural Network)
- Fases: diseño, entrenamiento y evaluación



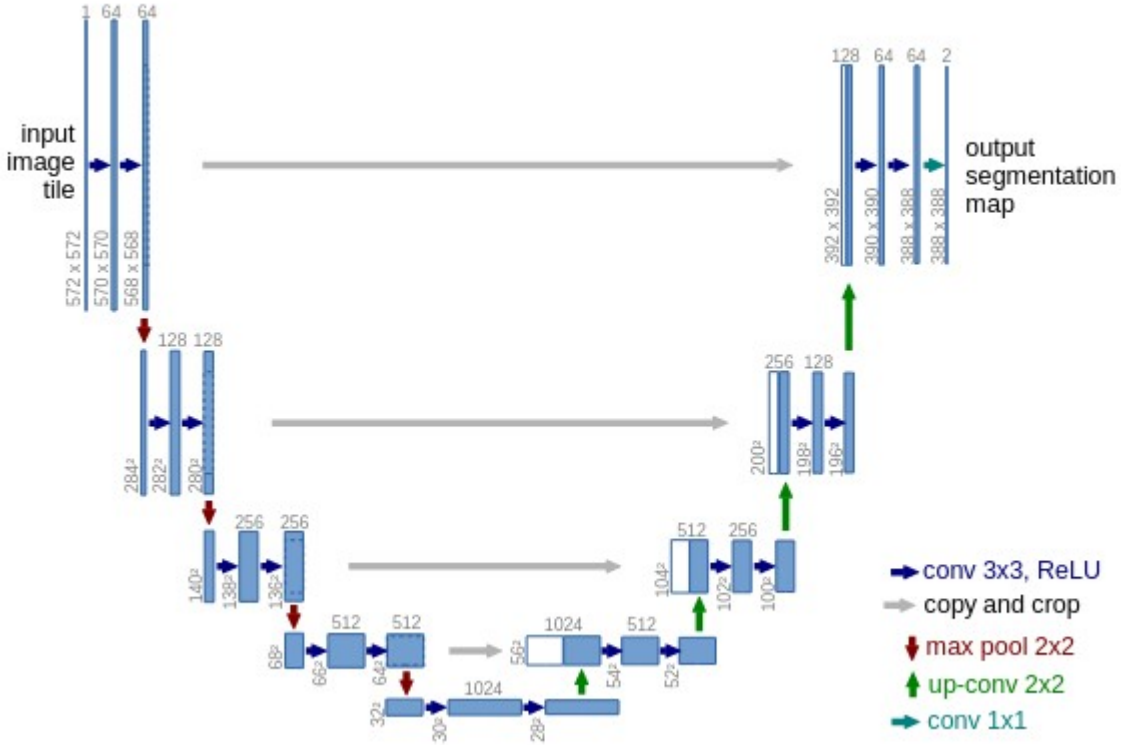
Salida de la capa oculta

$$\sigma\left(\begin{bmatrix} W_{11} & W_{12} & W_{13} \\ W_{21} & W_{22} & W_{23} \\ W_{31} & W_{32} & W_{33} \\ W_{41} & W_{42} & W_{43} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} + \begin{bmatrix} B_1 \\ B_2 \\ B_3 \\ B_4 \end{bmatrix}\right) = \begin{bmatrix} O_1 \\ O_2 \\ O_3 \\ O_4 \end{bmatrix}$$

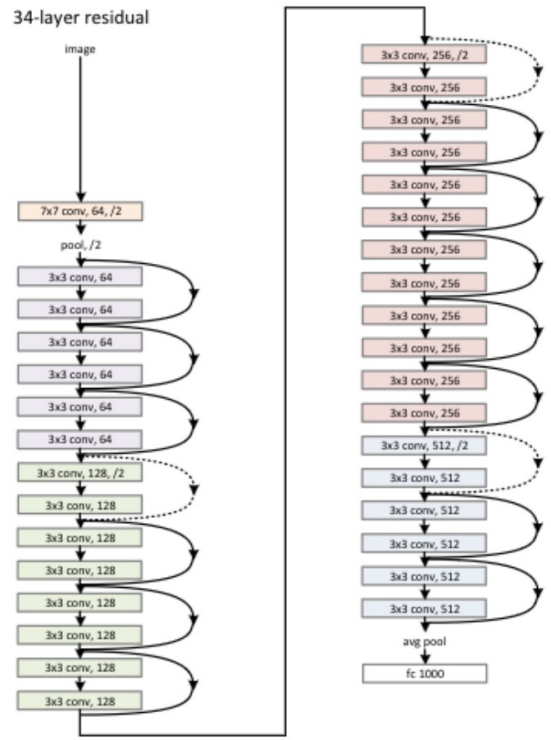
Salida de la red

$$\sigma\left(\begin{bmatrix} W_{11} & W_{12} & W_{13} & W_{14} \\ W_{21} & W_{22} & W_{23} & W_{24} \end{bmatrix} \begin{bmatrix} O_1 \\ O_2 \\ O_3 \\ O_4 \end{bmatrix} + \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}\right) = \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix}$$

Arquitecturas destacadas de Deep Learning



U-Net

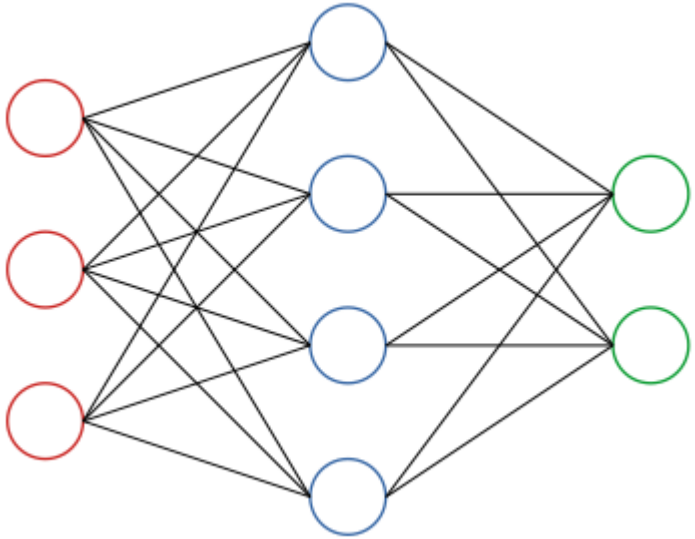


Res-Net

Aprendizaje

- Ajustar todos los pesos y sesgos de la red para que produzcan la salida deseada
- Función coste, descenso del gradiente y **retropropagación**
- División de los datos:
  - **Entrenamiento**: ajuste parámetros con retropropagación
  - **Validación**: ajuste hiperparámetros y control sobreajuste
  - **Test**: evaluación imparcial del modelo
- Otras estrategias: validación cruzada

Aprendizaje → ajuste de los parámetros



3x4+4x2 = 20 pesos

4+2 = 6 sesgos

26 parámetros

Un modelo real puede tener miles de millones de parámetros:

ResNet-50: **25** millones de parámetros

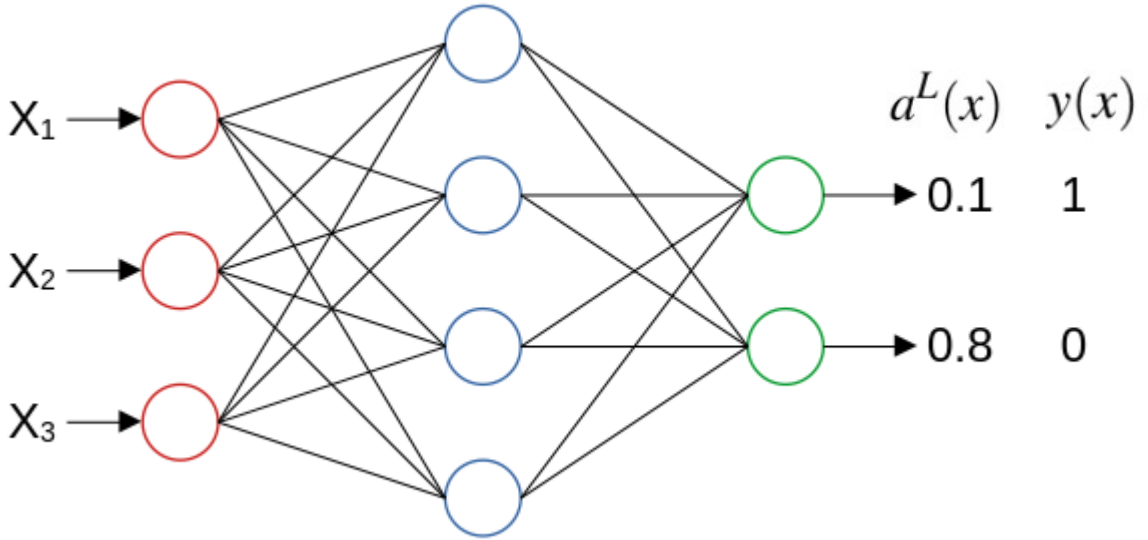
VGG-16: **138** millones de parámetros

ChatGPT 3: **175.000** millones de parámetros

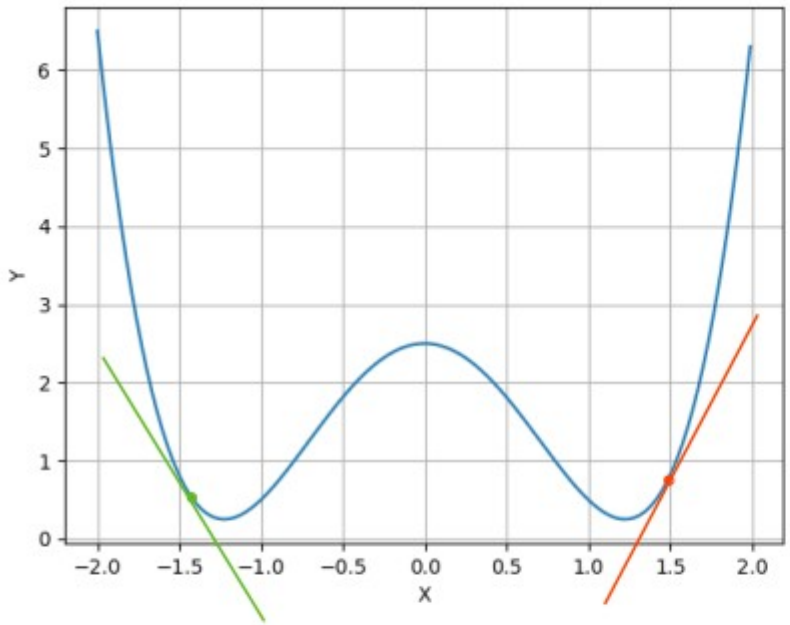
Ejemplo de función coste:

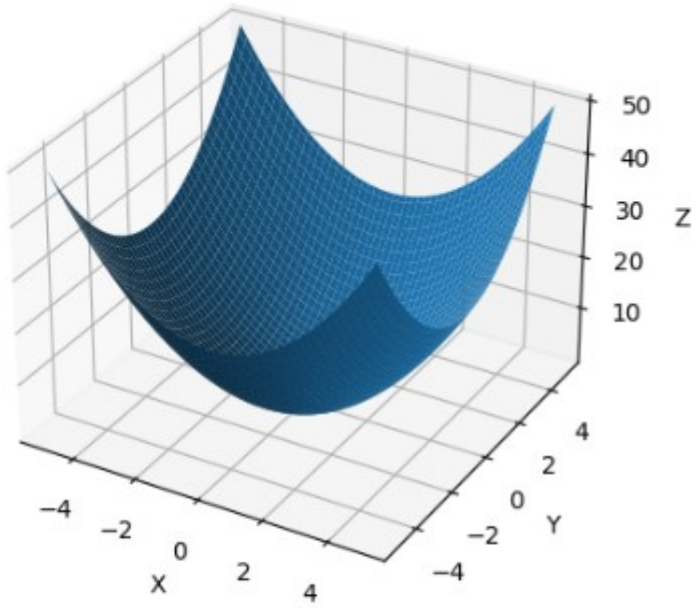
Error Cuadrático Medio

$$C = \frac{1}{2n} \sum_x \left\| y(x) - a^L(x) \right\|^2$$



Descenso del gradiente





Algoritmo de propagación hacia atrás

1. Prealimentación: para cada capa $l = 2, 3, ..., L$ calcular $z^l = w^l a^{l-1} + b^l$, y $a^l = \sigma(z^l)$.

2. Calcular el error de la salida: $\delta^L = \nabla_a C \circ \sigma'(z^L)$, donde $\circ$ representa el producto Hadamard y $\nabla_a C$ el gradiente de la función coste respecto a la salida, o $\frac{\partial C}{\partial a^l_j}$, donde j representa la neurona de esa salida.

3. Propagación hacia atrás del error: para cada capa $l = L-1, L-2, ..., 2$ calcular $\delta^l = ((w^{l+1})^T \delta^{l+1}) \circ \sigma'(z^l)$

4. Calcular el gradiente de la función coste respecto a los pesos y sesgos, $\nabla_w C$ y $\nabla_b C$, cuyos componentes consisten en $\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j$ y $\frac{\partial C}{\partial b^l_j} = \delta^l_j$, donde j representa la neurona del peso o sesgo y k la neurona de la capa anterior.

5. Descenso del gradiente: actualizar las matrices de pesos $w^l \to w^l - \eta \nabla_w C^l$ y de sesgo $b^l \to b^l - \eta \nabla_b C^l$, donde η es el ratio de aprendizaje (learning rate).

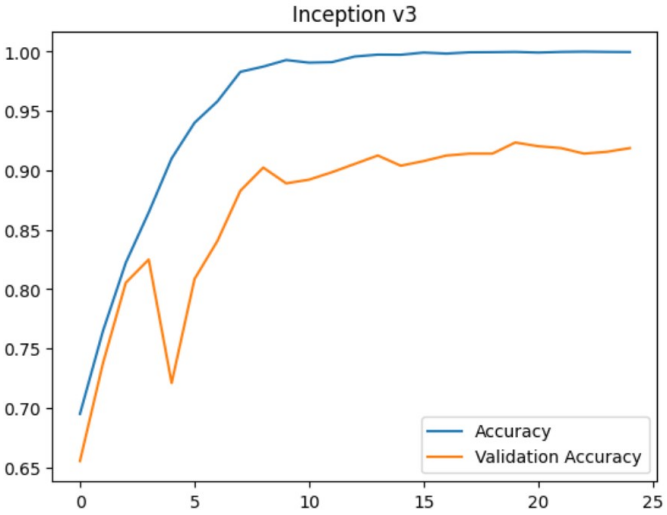
Si se aplica algoritmo del descenso del gradiente a particiones de m ejemplos, siendo x cada ejemplo, los pesos y los sesgos se actualizan de la siguiente forma:

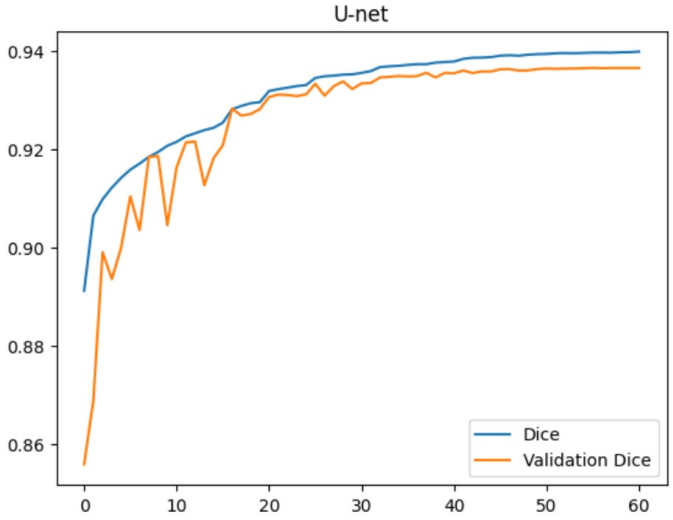
- Pesos: $w^l \to w^l - \frac{\eta}{m} \sum_x \nabla_w C^l_x$
- Sesgos: $b^l \to b^l - \frac{\eta}{m} \sum_x \nabla_b C^l_x$

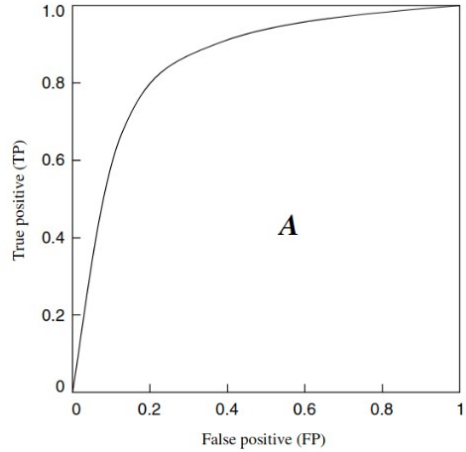
Evaluación

- Elección de **métricas** adecuadas
- Conjunto de **test**
- No se modifican los parámetros
- Verificar **sobreajuste** y generalización
- Análisis de errores
- Eficiencia y escalabilidad

Evaluación del modelo

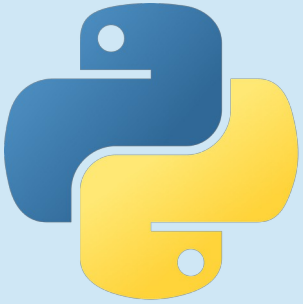


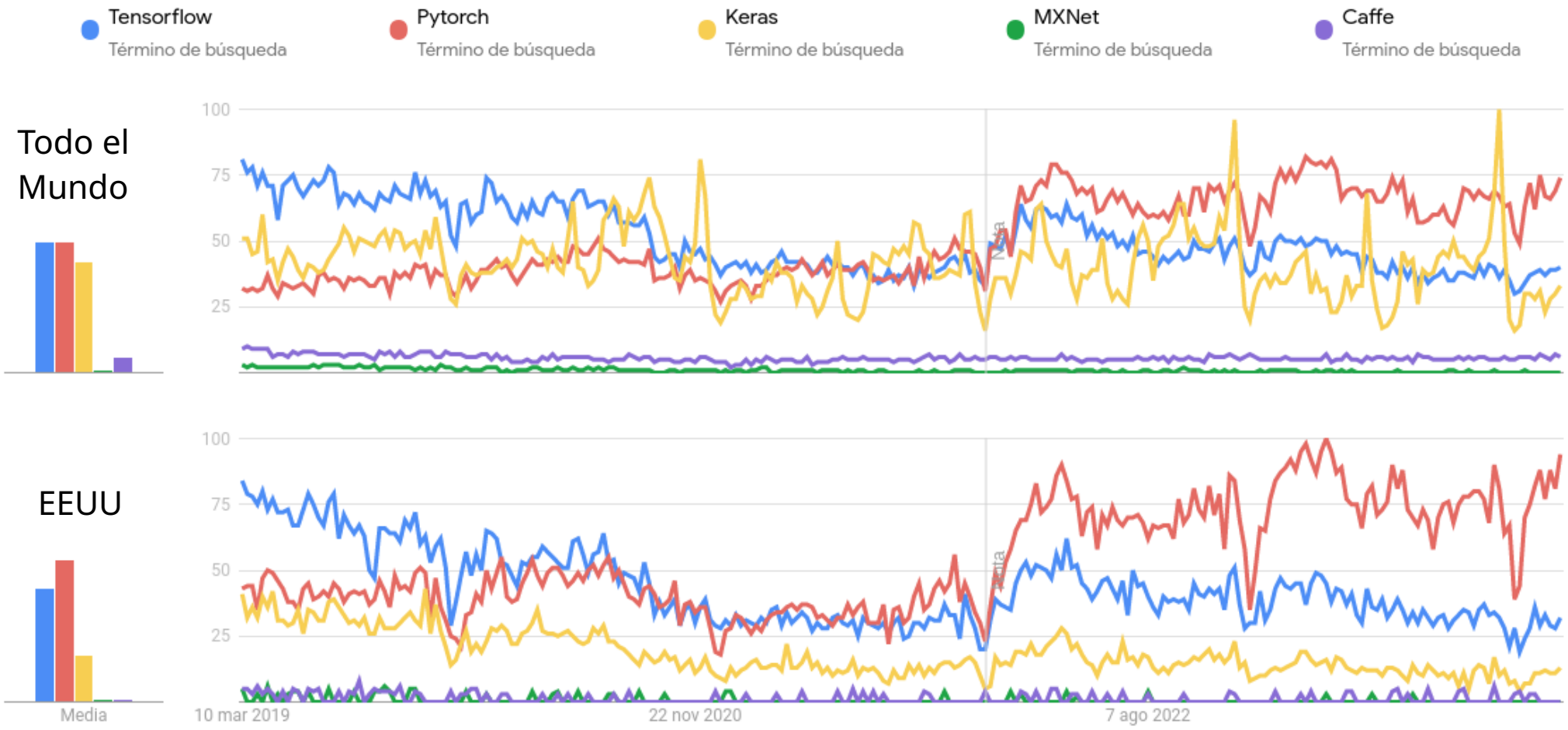




Lenguajes y Frameworks

- Lenguajes:
  - **Python**
  - R, C, C++, Java
- Frameworks:
  - **Tensorflow**
  - **Pytorch**
  - **Keras**
  - Otras opciones: MXNet, Caffe





Fuente: Google Trends

Ejemplo: Clasificación MNIST

- Datos: **70.000** imágenes de dígitos manuscritos
- Objetivo: **clasificar** cada imagen como el dígito correspondiente
- **Keras** (tensorflow)
- **Modelos**:
  - Capas totalmente conectadas
  - CNN

Ejemplo: NLP - Phi

- Procesamiento del lenguaje natural
- Large Language Models (LLMs)
- **Phi 1.5** (1300 millones de parámetros)
- **Phi 2** (2700 millones de parámetros)
- **Pytorch**
- Hugging Face (**transformers**)

Temas pendientes

- **Obtención y preparación de los datos**
- **Tipos de capas** (convolucional, submuestreo, normalización, dropout, ...)
- **Aumento de datos** (*data augmentation*)
- **Aprendizaje por transferencia** (*transfer learning*)
- **Arquitecturas especificas** (CNN, RNN, GNN, DQN, etc.)

Recursos online

- Keras: https://keras.io/
- Pytorch: https://pytorch.org/
- Hugging Face (transformers): https://huggingface.co/
- Google Colab: https://colab.research.google.com
- arXiv: https://arxiv.org/
- Google scholar: https://scholar.google.es/
- Fundamentos: http://neuralnetworksanddeeplearning.com/

¡Gracias!