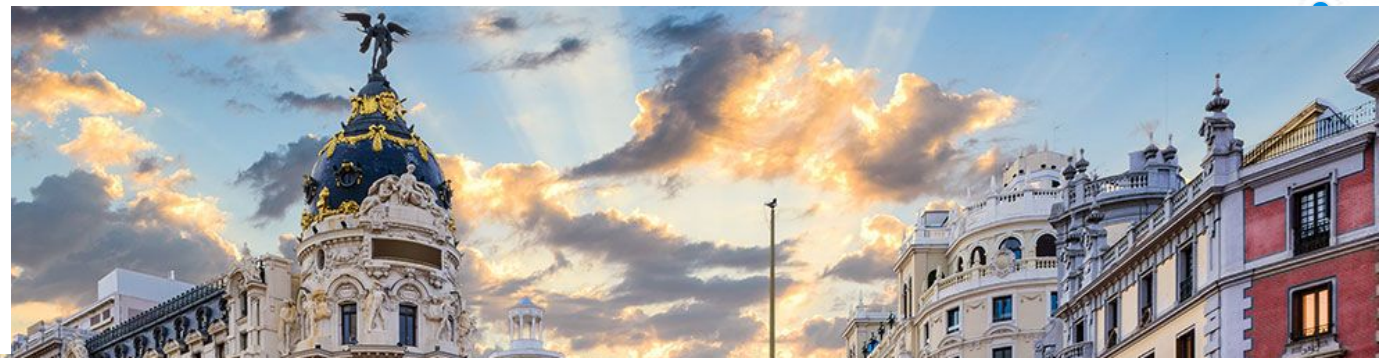
Construyendo Redes Convolucionales (CNN) con TensorFlow y Keras

Moisés Martínez

Red Hat



About me

PhD in Computer Science and AI

Big Data & AI Architect

Researcher on different universities

T3chFest and GDG Cloud Madrid Organizer

GDE in Machine Learning

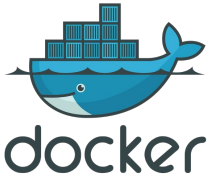
Moisés Martínez

@moisipm

@momartinm

Herramientas

Despliegue



Desarrollo

Tecnologías ML

1. Machine Learning

Red Hat

1. Machine Learning

- Definición de Aprendizaje RAE

  Adquirir el conocimiento de algo por medio del estudio o de la experiencia.

1. Machine Learning

- Definición de Aprendizaje RAE

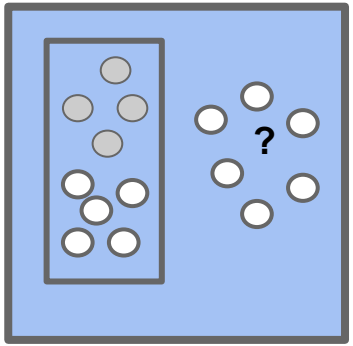
  Adquirir el conocimiento de algo por medio del estudio o de la experiencia.

- Definición de Aprendizaje Automático

  Proceso de **adquisición** de conocimiento de manera **automática** mediante la utilización de **ejemplos** (**experiencia**) de entrenamiento

1. Machine Learning

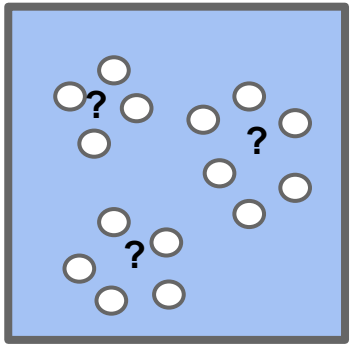
Supervisado



Ejemplos + clases

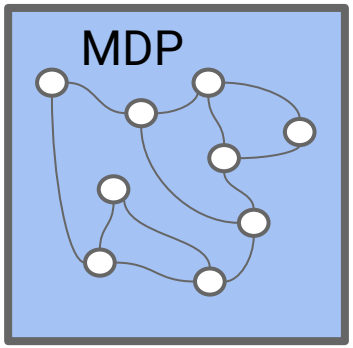
Identificar Patrones

No Supervisado



Ejemplos

Por Refuerzo

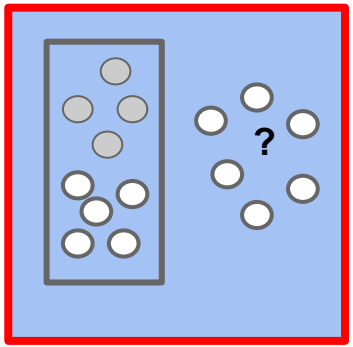


Acciones$^{\text{Refuerzo}}$ + Estados

Definir políticas

Machine Learning

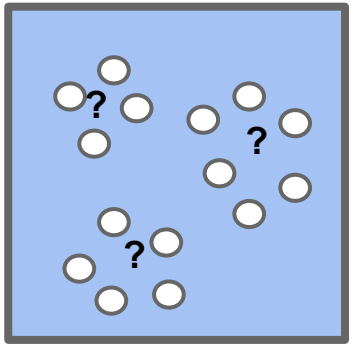
Supervisado



Ejemplos + clases

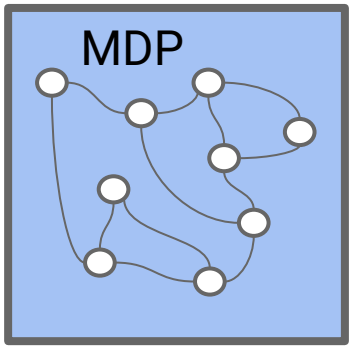
Identificar Patrones

No Supervisado



Ejemplos

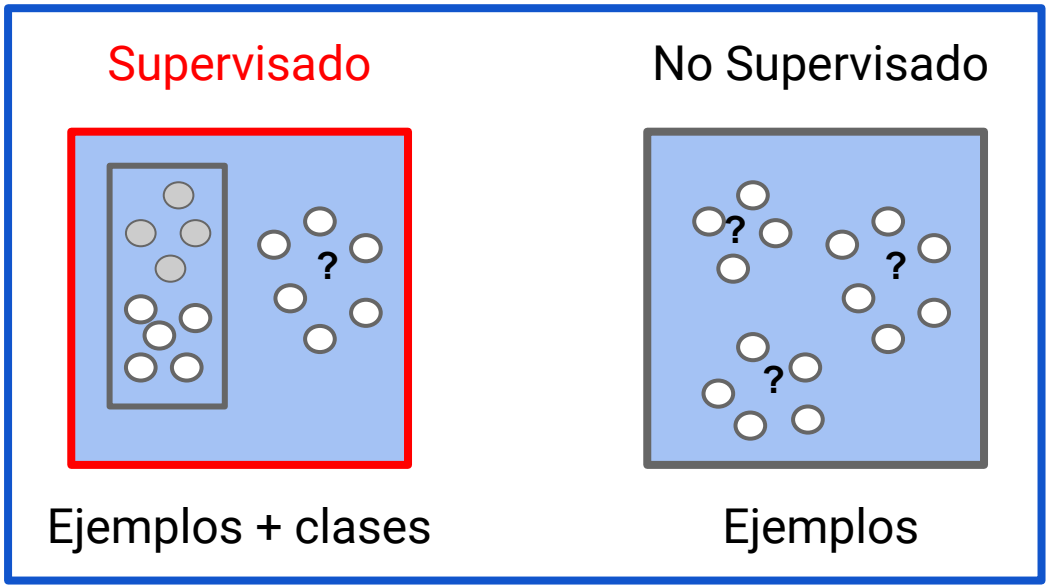
Por Refuerzo



Acciones$^{\text{Refuerzo}}$ + Estados

Definir políticas

Machine Learning



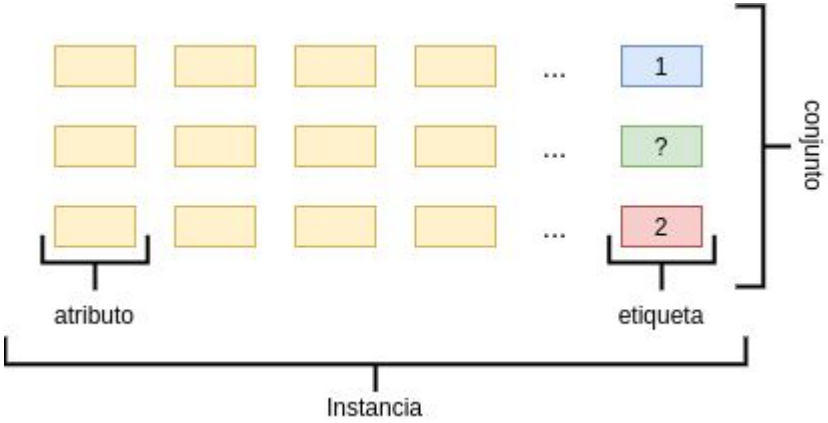
Ejemplos de aprendizaje **etiquetados previamente** (Conocemos la clases)

2. El proceso de aprendizaje

2. El proceso de aprendizaje

Entrenamiento

Datos en bruto

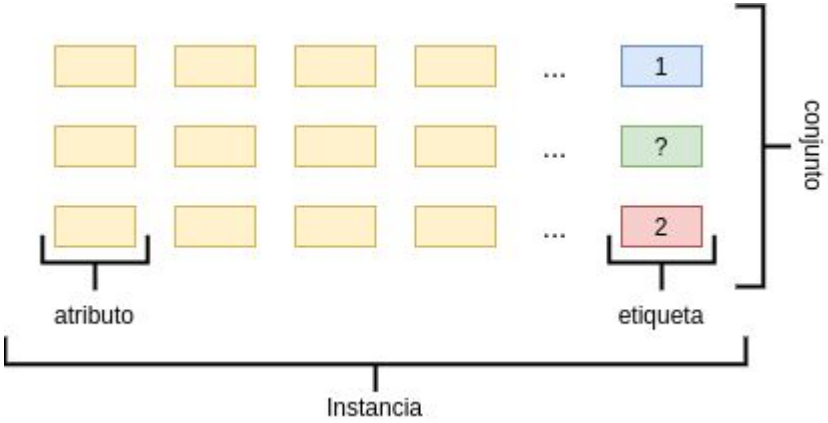


Instancia: Estructura básica para representar la información. Está compuesta por una secuencia de **atributos** que describen cada uno de los ejemplos.

2. El proceso de aprendizaje

Entrenamiento

Datos en bruto



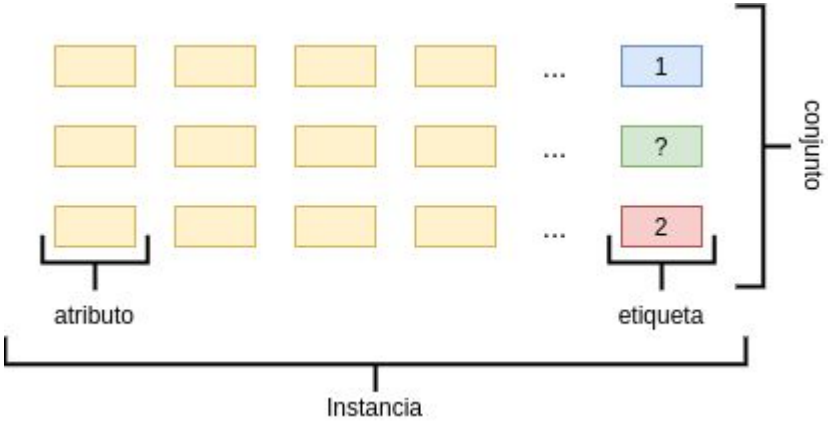
Atributo: Unidad básica para almacenar y describir la información. Suele almacenarse de dos formas:

- Continuo: Son valores numéricos de tipo continuo

2. El proceso de aprendizaje

Entrenamiento

Datos en bruto



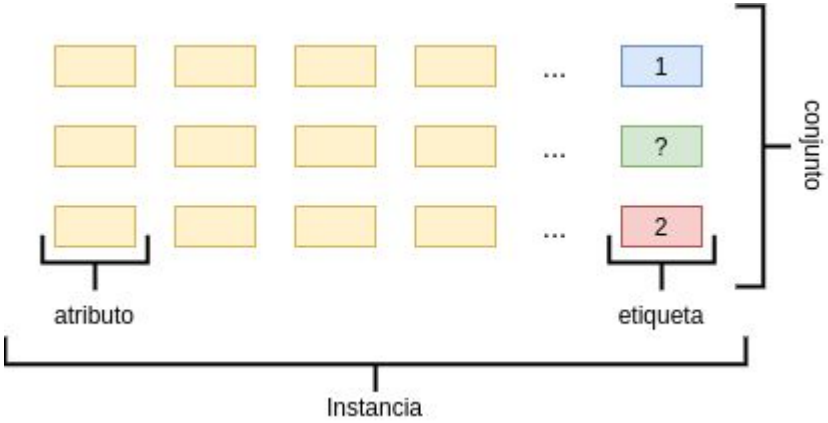
Atributo: Unidad básica para almacenar y describir la información. Suele almacenarse de dos formas:

- Continuo: Son valores numéricos de tipo continuo
- Discreto: Son valores de cualquier estructura (Cadenas de caracteres, números, etc)

2. El proceso de aprendizaje

Entrenamiento

Datos en bruto



Objetivo: Es el valor esperado para cada una de la instancias. Tiene múltiples denominaciones:

- Clase
- Etiqueta
- Valor a predecir (Numérico)

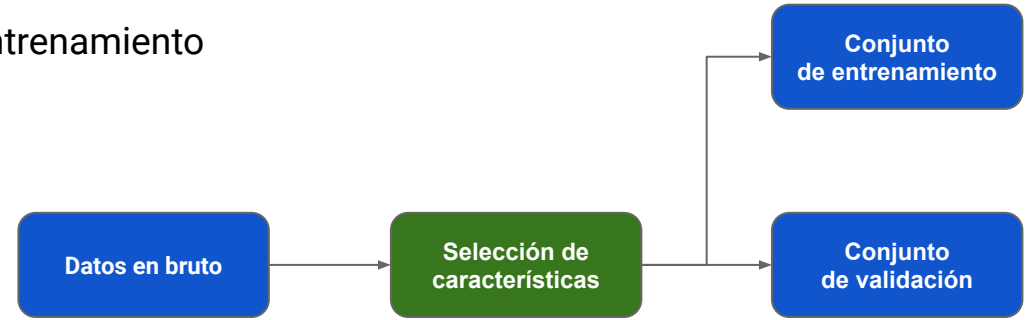
2. El proceso de aprendizaje

Entrenamiento

Conjunto de entrenamiento: Es un conjunto de instancias que son utilizadas para el proceso de entrenamiento con el objetivo de construir un modelo.

2. El proceso de aprendizaje

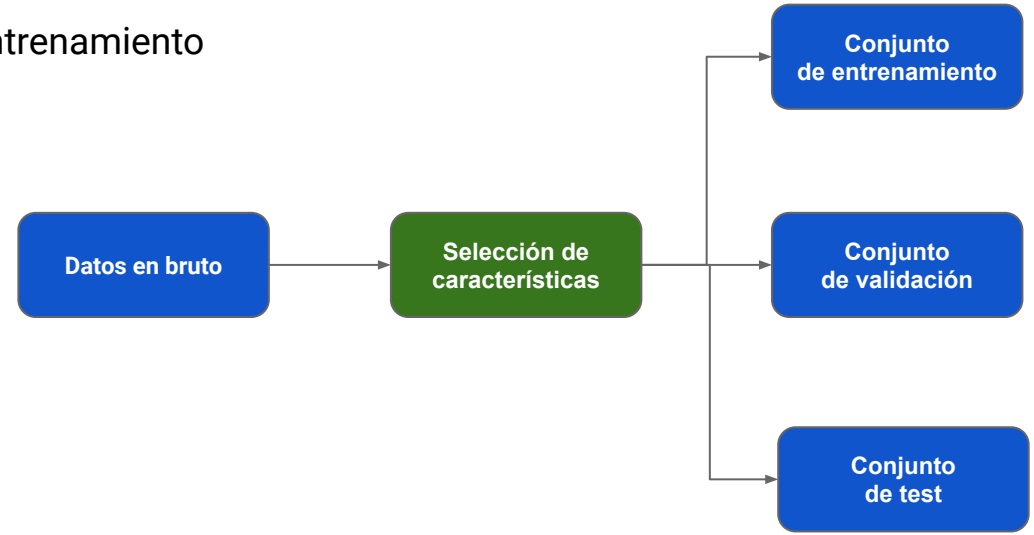
Entrenamiento



Conjunto de validación: Es un conjunto de instancias que son utilizadas para comprobar la calidad del modelo construido durante el proceso de entrenamiento

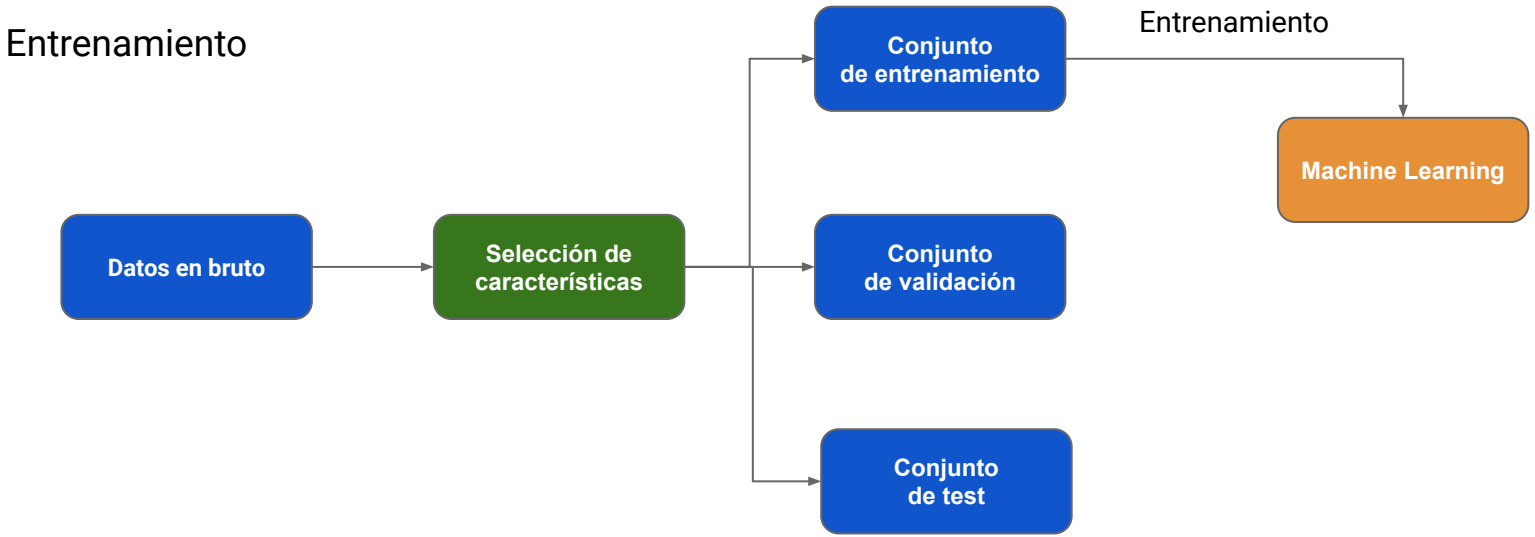
2. El proceso de aprendizaje

Entrenamiento



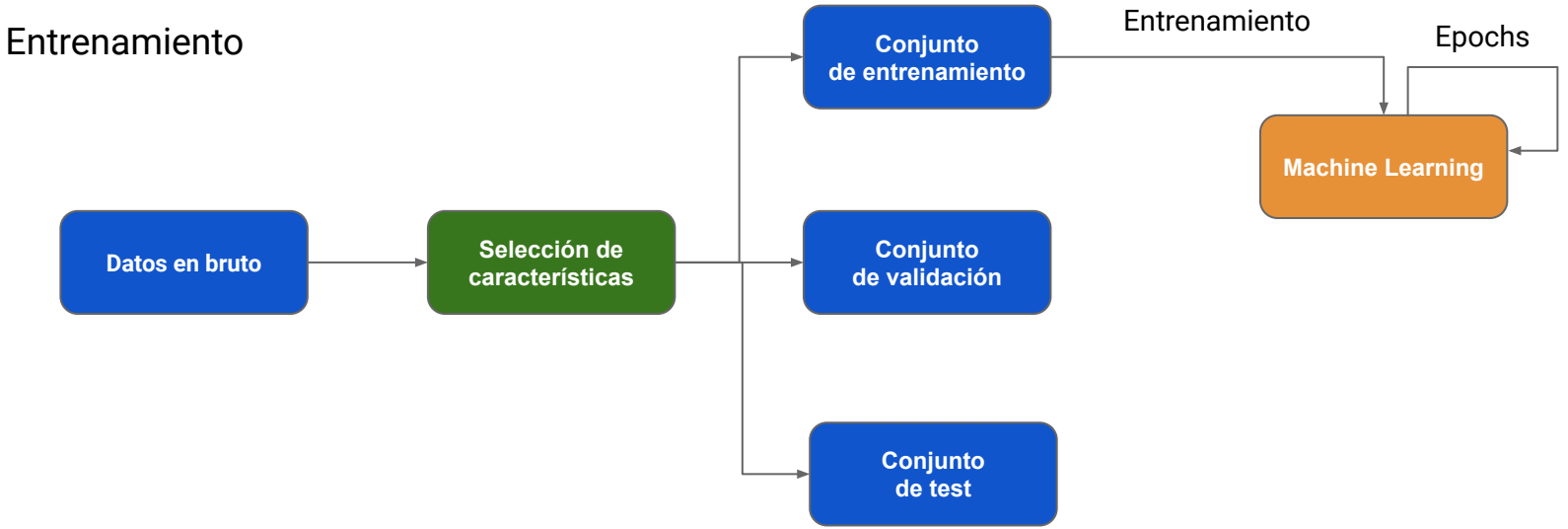
Conjunto de test: Es un conjunto de instancias que son utilizadas para comprobar la calidad del modelo final que ha sido generado.

2. El proceso de aprendizaje



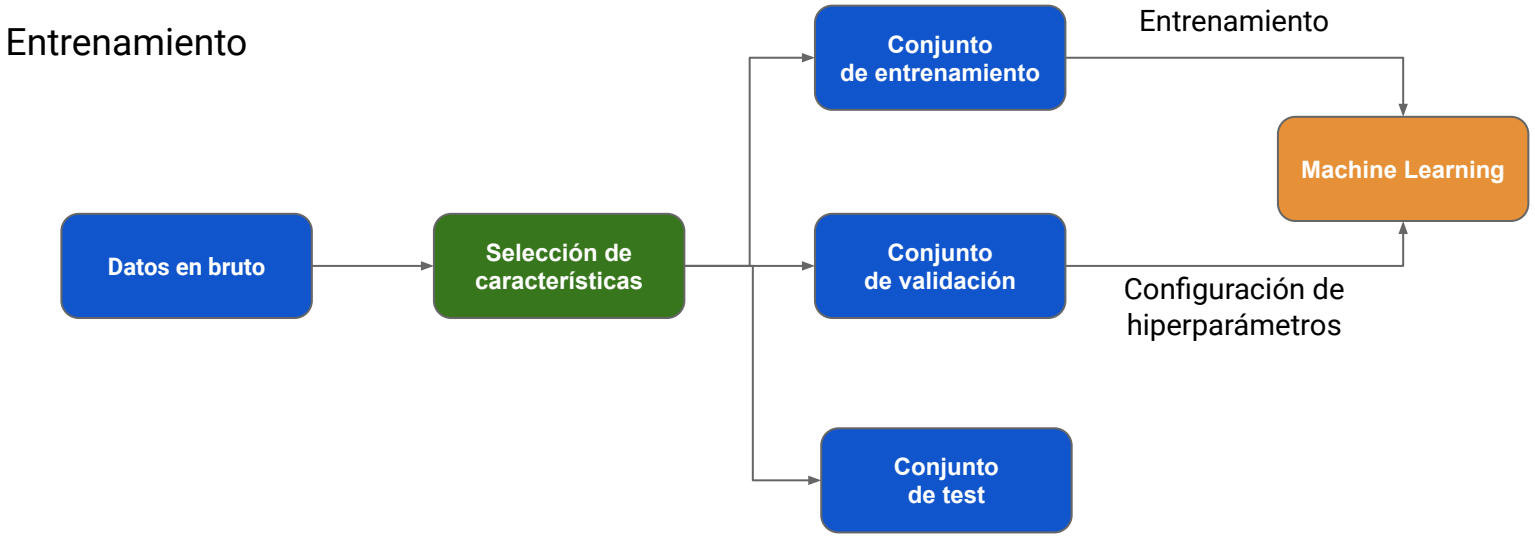
Algoritmo: Es el método de aprendizaje que se utilizar para construir el modelo de razonamiento.

2. El proceso de aprendizaje



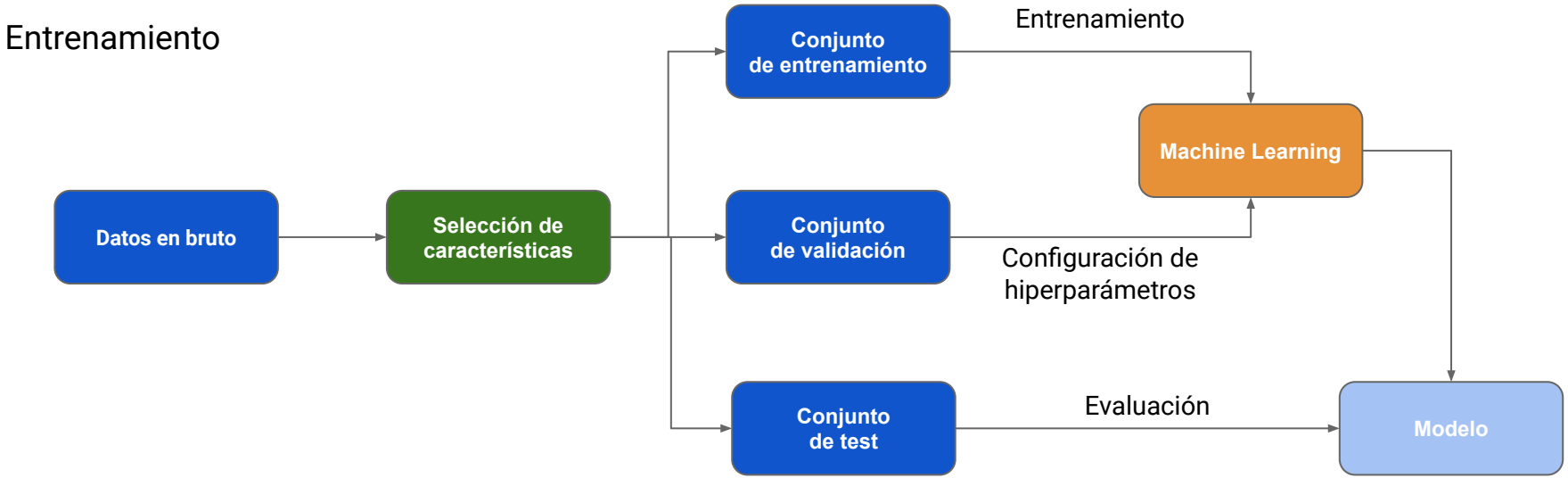
Épocas (epochs): Son el conjunto de iteraciones en las que se realiza el proceso de entrenamiento con el objetivo de construir el mejor modelo.

2. El proceso de aprendizaje



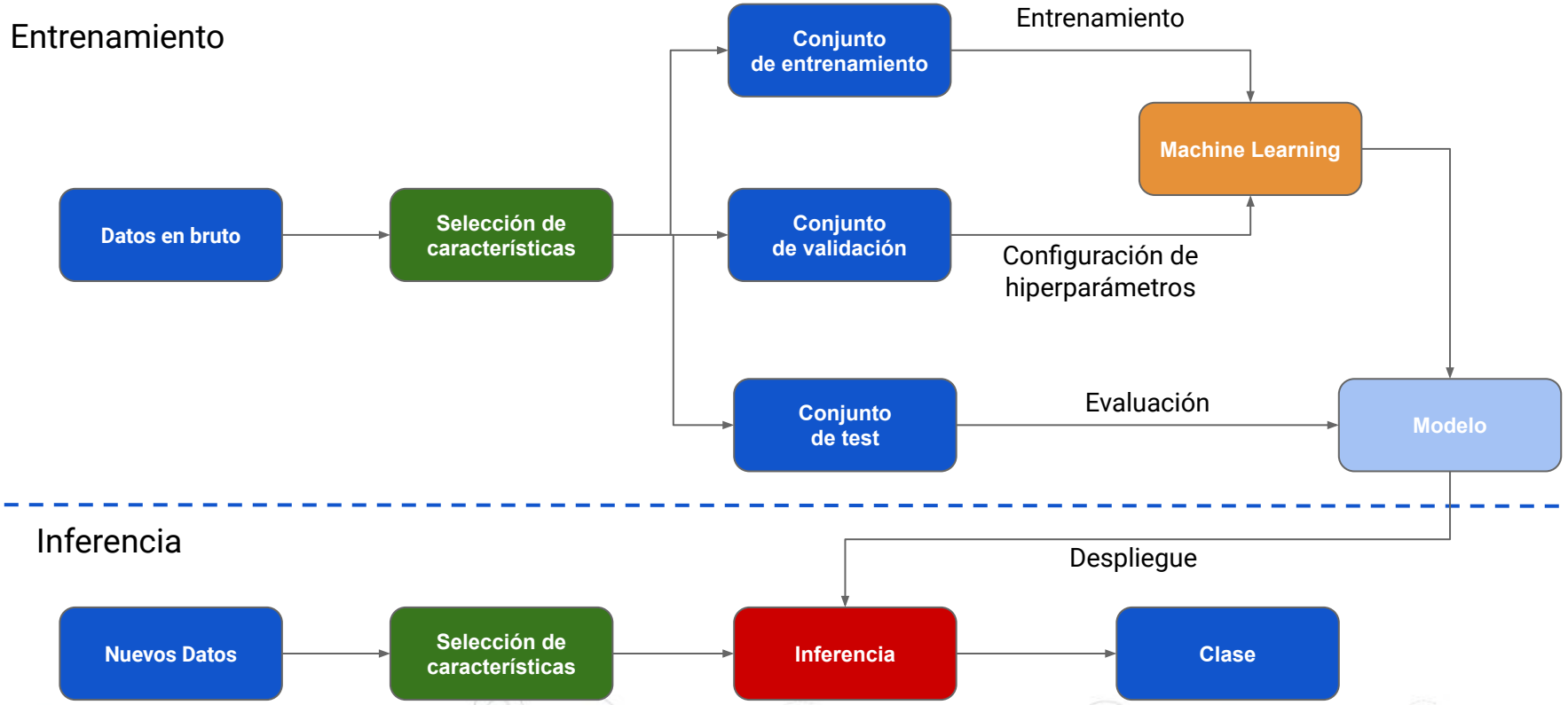
Épocas (epochs): Son el conjunto de iteraciones en las que se realiza el proceso de entrenamiento con el objetivo de construir el mejor modelo.

2. El proceso de aprendizaje



Modelo: Conjunto de reglas o patrones inferidos a partir del conjunto de entrenamiento con el objetivo de predecir, inferir o definir la agrupación de una instancia.

2. El proceso de aprendizaje

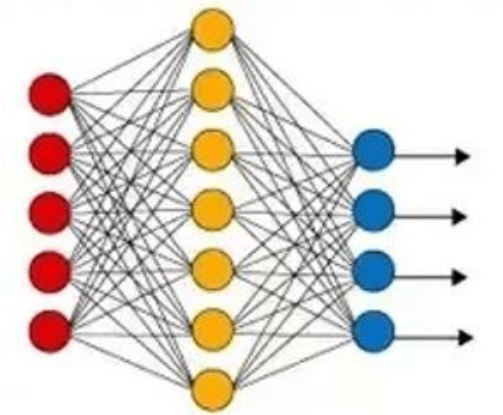


2. Redes de Neuronas

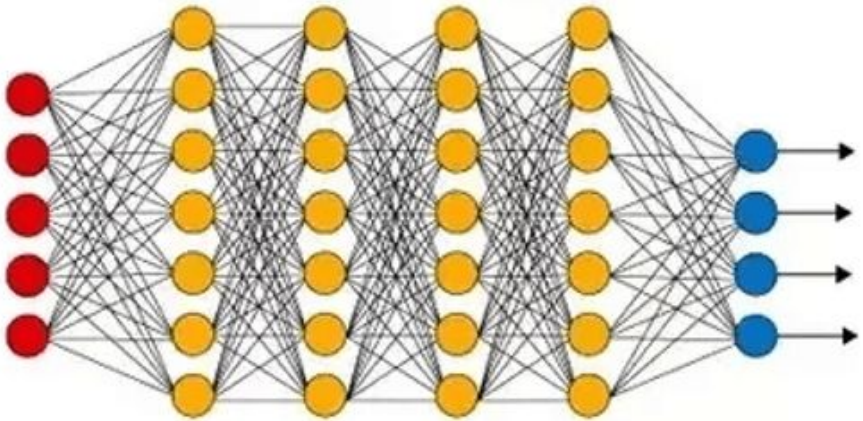
Red Hat

2. Redes de neuronas

Shallow Neural Network

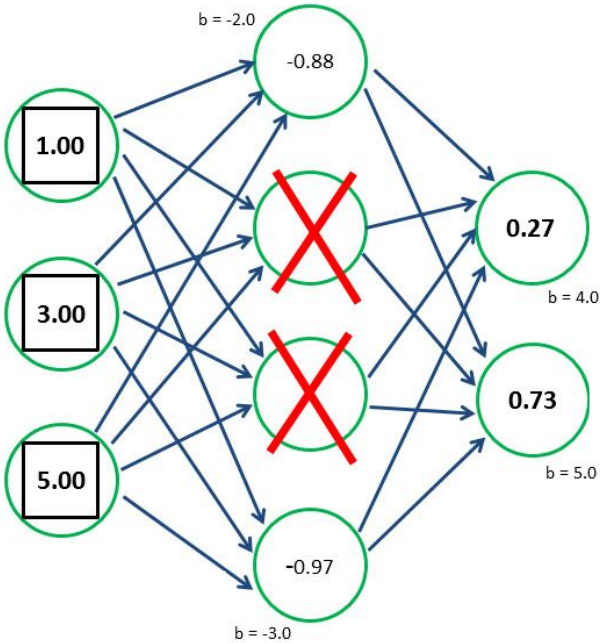


Deep Neural Network

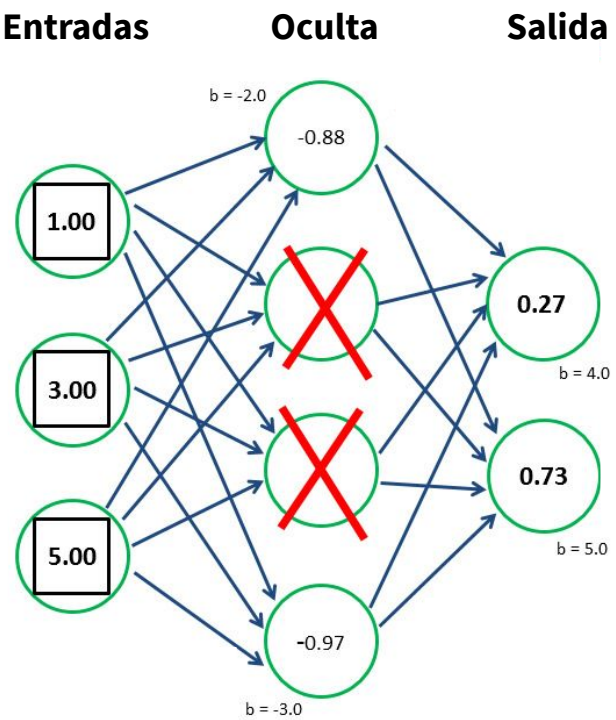


Las **diferencia** que existe entre una red “shallow” y una red “profunda” es el número de capas ocultas (amarillo). Es decir, una red de neuronas profundas es aquellas que tiene múltiples capas ocultas

2. Redes de neuronas



2. Redes de neuronas



Pesos

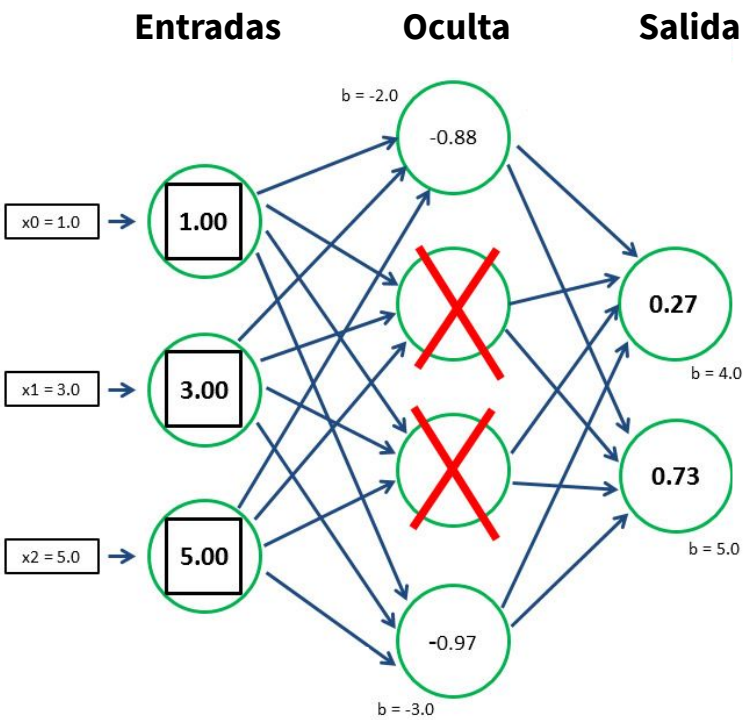
ihWeights[][]

0.01	~~0.02~~	~~0.03~~	0.04
0.05	~~0.06~~	~~0.07~~	0.08
0.09	~~0.10~~	~~0.11~~	0.12

hoWeights[][]

0.13	0.14
~~0.15~~	~~0.16~~
~~0.17~~	~~0.18~~
0.19	0.20

2. Redes de neuronas



Pesos

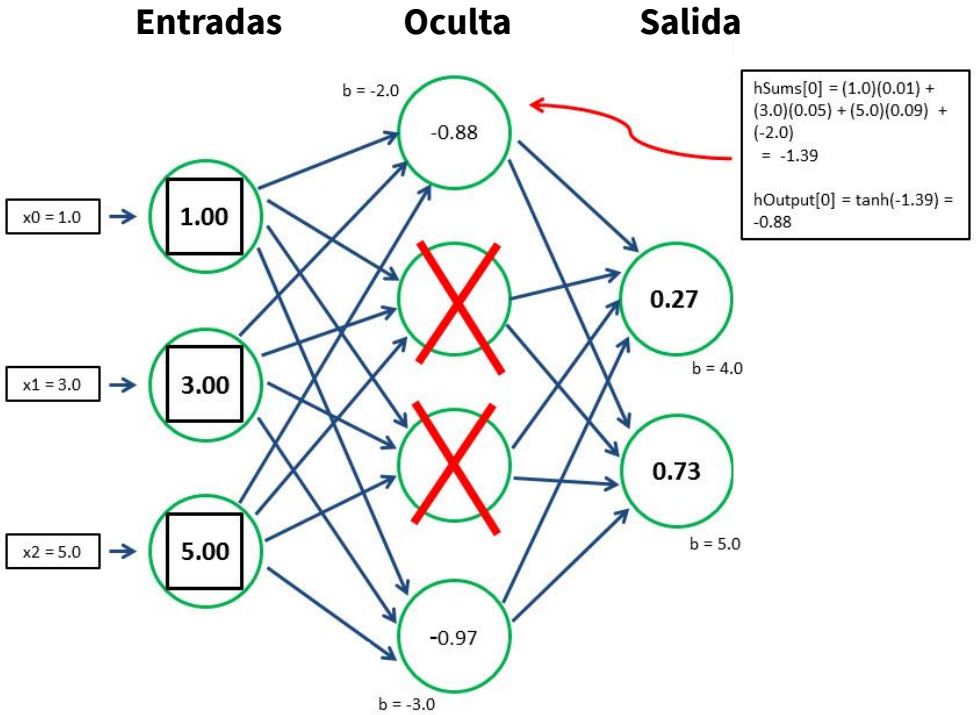
ihWeights[][]

0.01	~~0.02~~	~~0.03~~	0.04
0.05	~~0.06~~	~~0.07~~	0.08
0.09	~~0.10~~	~~0.11~~	0.12

hoWeights[][]

0.13	0.14
~~0.15~~	~~0.16~~
~~0.17~~	~~0.18~~
0.19	0.20

2. Redes de neuronas



Pesos

ihWeights[][]

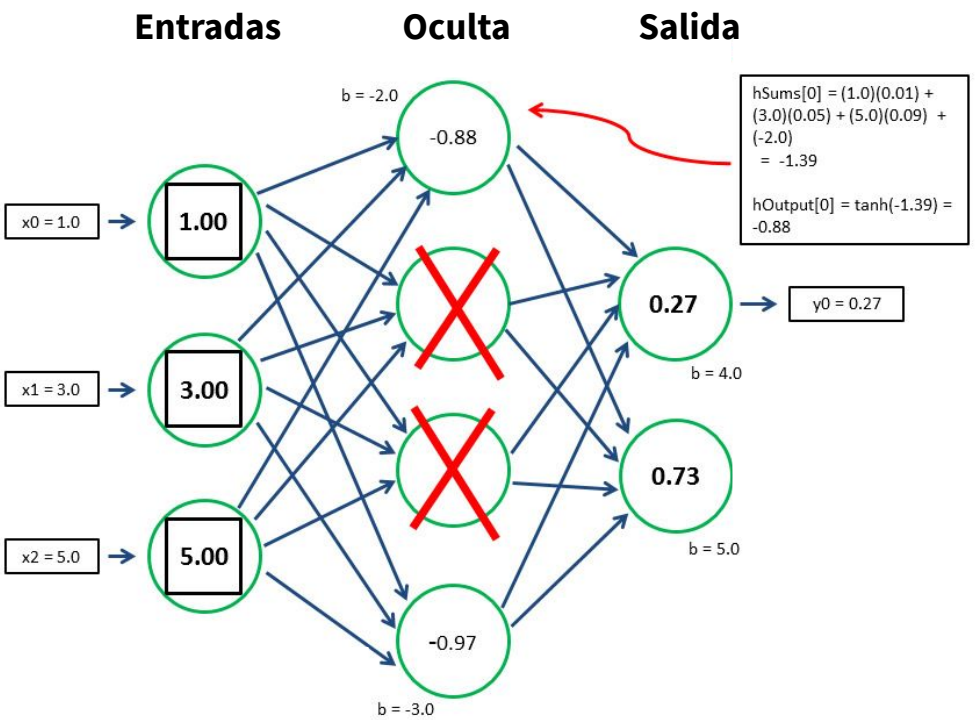
0.01	~~0.02~~	~~0.03~~	0.04
0.05	~~0.06~~	~~0.07~~	0.08
0.09	~~0.10~~	~~0.11~~	0.12

hoWeights[][]

0.13	0.14
~~0.15~~	~~0.16~~
~~0.17~~	~~0.18~~
0.19	0.20

Bias: Parámetro adicional utilizado en las redes de neuronas para ajustar el bias (Ayuda al entrenamiento)

2. Redes de neuronas



Pesos

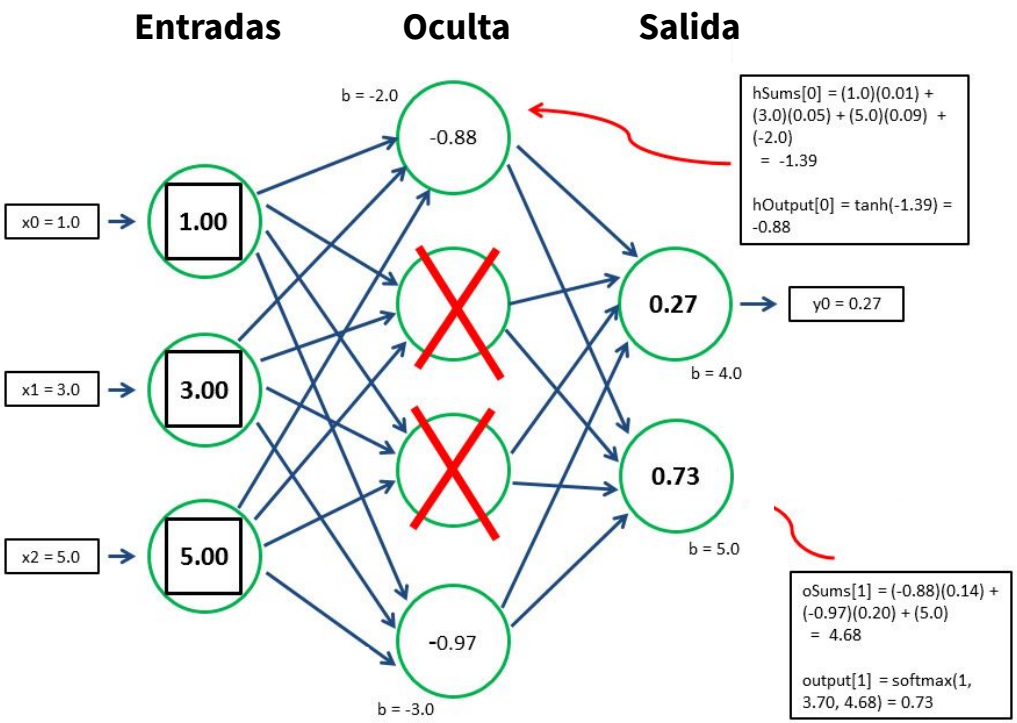
ihWeights[][]

0.01	~~0.02~~	~~0.03~~	0.04
0.05	~~0.06~~	~~0.07~~	0.08
0.09	~~0.10~~	~~0.11~~	0.12

hoWeights[][]

0.13	0.14
~~0.15~~	~~0.16~~
~~0.17~~	~~0.18~~
0.19	0.20

2. Redes de neuronas



Pesos

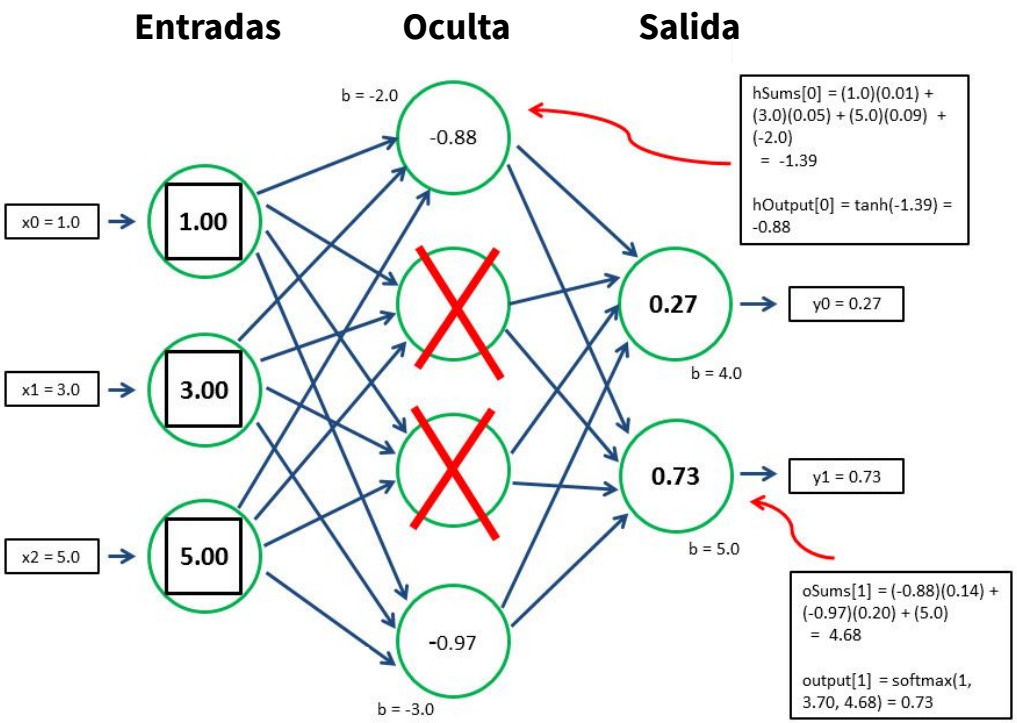
ihWeights[][]

0.01	~~0.02~~	~~0.03~~	0.04
0.05	~~0.06~~	~~0.07~~	0.08
0.09	~~0.10~~	~~0.11~~	0.12

hoWeights[][]

0.13	0.14
~~0.15~~	~~0.16~~
~~0.17~~	~~0.18~~
0.19	0.20

2. Redes de neuronas



Pesos

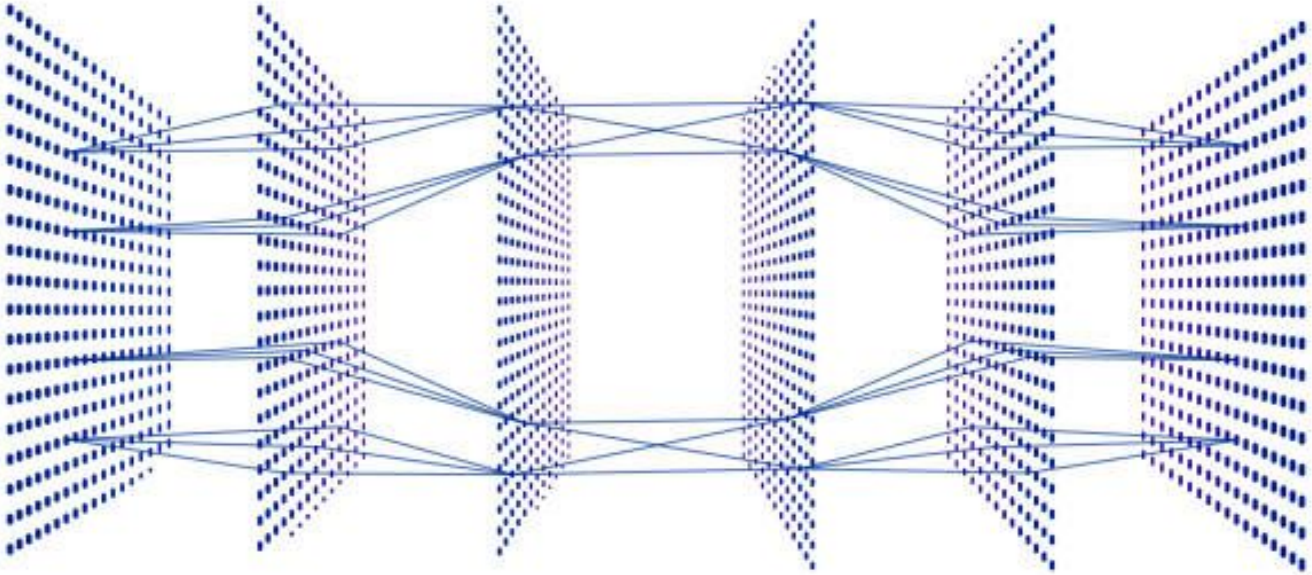
ihWeights[][]

0.01	~~0.02~~	~~0.03~~	0.04
0.05	~~0.06~~	~~0.07~~	0.08
0.09	~~0.10~~	~~0.11~~	0.12

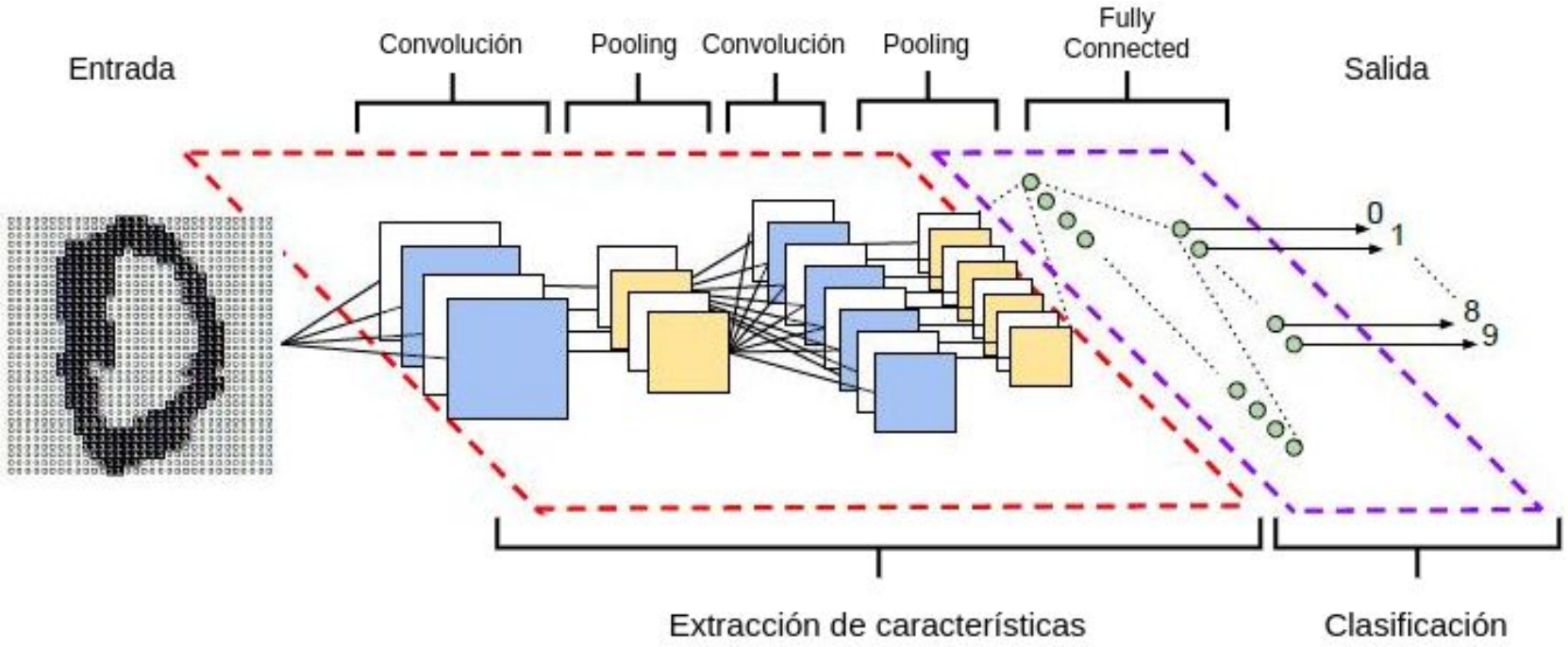
hoWeights[][]

0.13	0.14
~~0.15~~	~~0.16~~
~~0.17~~	~~0.18~~
0.19	0.20

2. Redes de neuronas

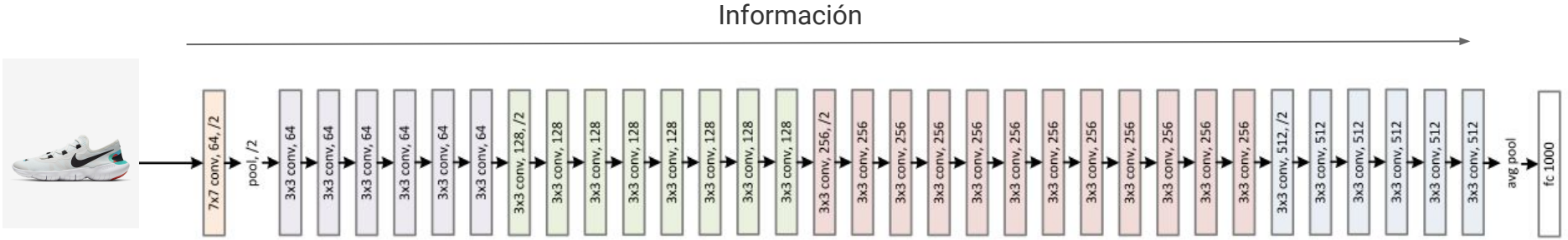


2. Redes de neuronas - Convolucionales (Feed Forward)



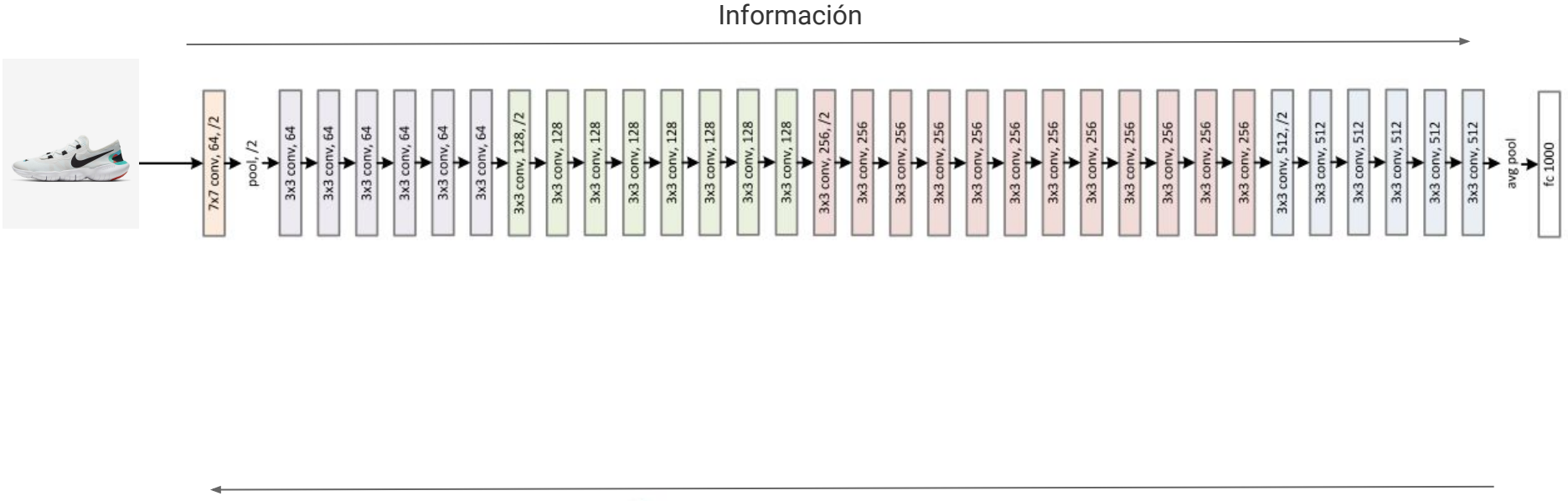
2. Redes de neuronas - Convolucionales (Feed Forward)

Las redes de neuronas **prealimentadas** son redes donde la información se mueve en una única dirección.



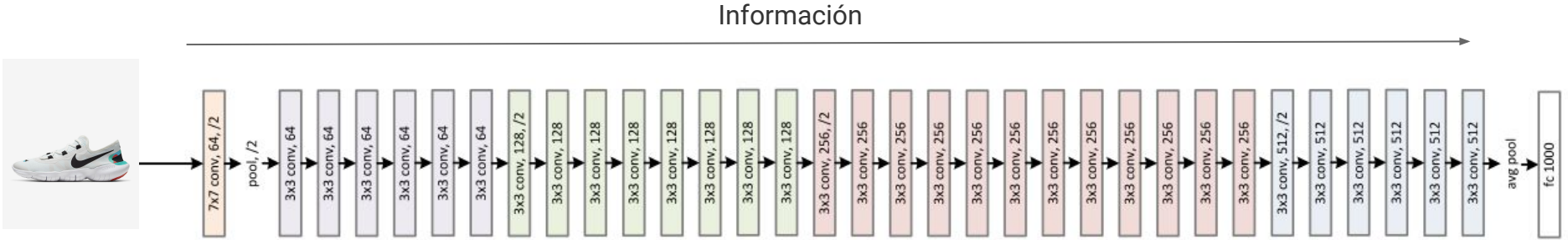
2. Redes de neuronas - Convolucionales (Feed Forward)

Las redes de neuronas **prealimentadas** son redes donde la información se mueve en una única dirección.



2. Redes de neuronas - Convolucionales (Feed Forward)

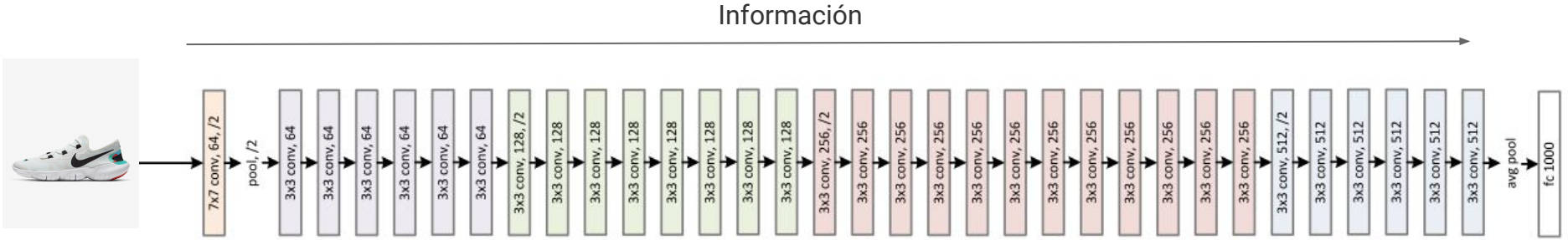
Las redes de neuronas **prealimentadas** son redes donde la información se mueve en una única dirección.



1. Comparación de la salida esperada con la salida obtenida => salida esperada - salida obtenida = error

2. Redes de neuronas - Convolucionales (Feed Forward)

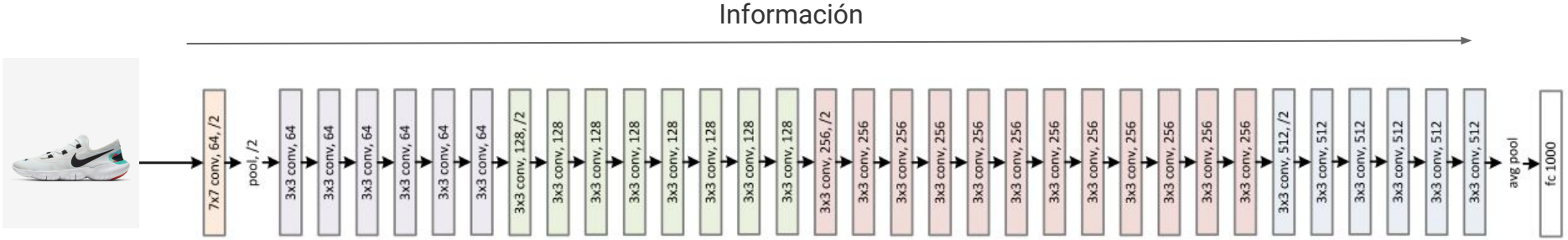
Las redes de neuronas **prealimentadas** son redes donde la información se mueve en una única dirección.



1. Comparación de la salida esperada con la salida obtenida => salida esperada - salida obtenida = error
2. Error se propaga hacia atrás mediante (retroalimentación) para recalcular los pesos de las conexiones

2. Redes de neuronas - Convolucionales (Feed Forward)

Las redes de neuronas **prealimentadas** son redes donde la información se mueve en una única dirección.

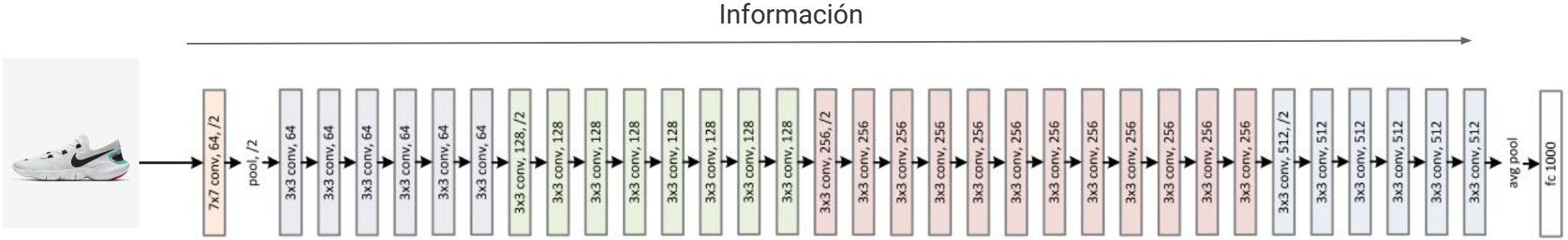


1. Comparación de la salida esperada con la salida obtenida => salida esperada - salida obtenida = error
2. Error se propaga hacia atrás mediante (retroalimentación) para recalcular los pesos de las conexiones
3. Descenso de gradiente (optimización): Derivada de la función de error con respeto a los pesos

Back propagation

2. Redes de neuronas - Convolucionales (Feed Forward)

Las redes de neuronas **prealimentadas** son redes donde la información se mueve en una única dirección.



1. Comparación de la salida esperada con la salida obtenida => salida esperada - salida obtenida = error
2. Error se propaga hacia atrás mediante (retroalimentación) para recalcular los pesos de las conexiones
3. Descenso de gradiente (optimización): Derivada de la función de error con respeto a los pesos

Back propagation

2. Redes de neuronas - Convolucionales (Feed Forward)

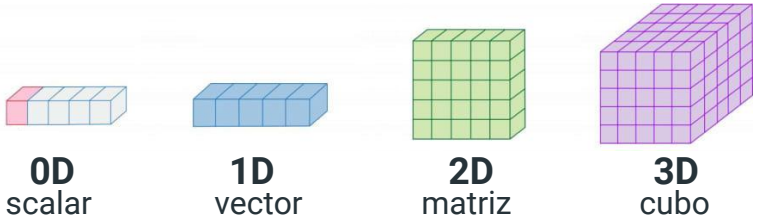
	Red Feed Forward (FFNN)	Red Convolucional (CNN)
Capa Entrada	Secuencia de características	Píxeles de una imagen
Capa Profunda	Una capa totalmente conectada	Diferentes tipos de capas: ● Convolucionales ● Subsampling ● Fully-connected
Capa Salida	Elementos a predecir	Elementos a predecir precedidos de un capa SoftMax
Aprendizaje	Supervisado	Supervisado
Interconexiones	Total entre capas	Parcial entre capas
Backpropagation	Aprendizaje de los pesos	Aprendizaje de los filtros

3. TensorFlow

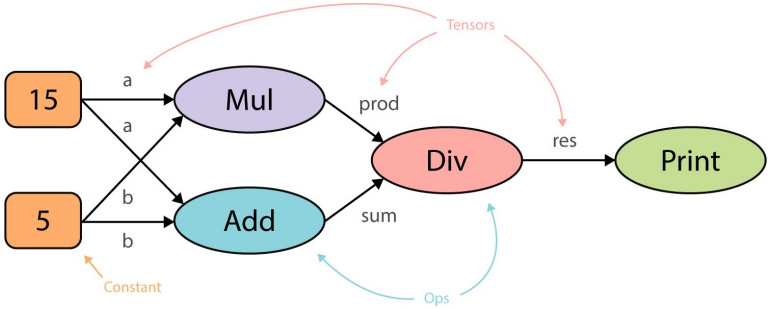
Red Hat

3. Redes de neuronas - TensorFlow

Tensor + Flow



N-dimensional array



Conjunto de operaciones

3. Redes de neuronas - TensorFlow

La información se representa mediante tres tipos de **contenedores de información**, que deben ser definidos a priori, con el objetivo de que sean incluidos en el grafo de operaciones

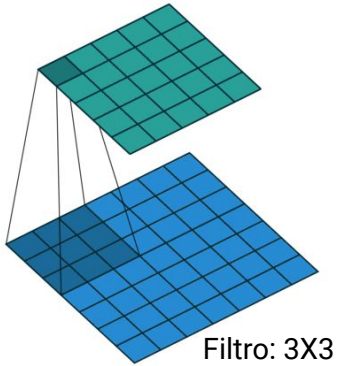
Tipo	Formato	Función	Ejemplo
Constante	Constante	tf.constant	tf.constant([None, 800, 460, 4])
Variable	Variable	tf.variable	tf.placeholder('float32', input, name='train_X')
PlaceHolder	Variable (in/out)	tf.placeholder	tf.Variable(tf.random_normal([size, size, channels, filters], stddev=0.01), name='Layer_1_weights')

4. Capas y funciones

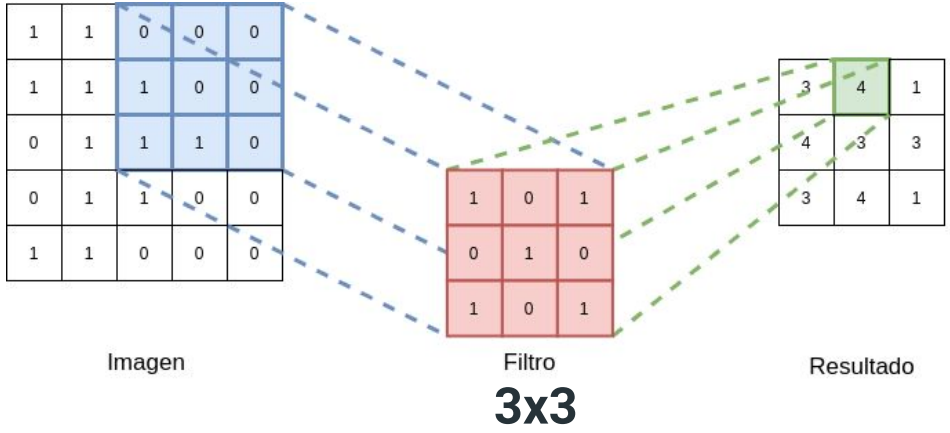
Red Hat

4. Capas y funciones - Capa convolucional

Las capas convolucionales se utilizan para la extracción de características mediante la aplicación de operaciones (productos y sumas) entre matrices. Estas operaciones se realizan mediante un filtro (kernel) cuadrado.



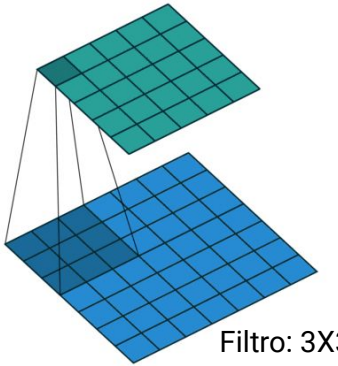
Filtro: 3X3




```
conv = tf.nn.conv2d(input, weights, strides=[1, stride, stride, 1], padding='VALID', name='layer_1_conv')
```

4. Capas y funciones - Capa convolucional

Convolución + **Bias** + **Función de activación**

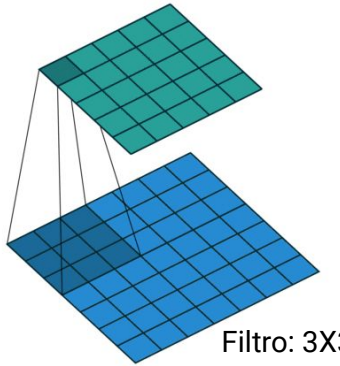


Filtro: 3X3

4. Capas y funciones - Capa convolucional

Convolución + **Bias** + **Función de activación**

```
conv_out = tf.nn.conv2d(input, weights, strides=[1, stride, stride, 1], padding='VALID', name='layer_1_conv')
```



Filtro: 3X3

4. Capas y funciones - Capa convolucional

Convolución + **Bias** + **Función de activación**

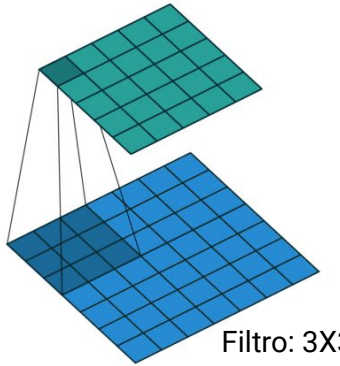
Capa anterior

Modo de aplicación del filtro

```
conv_out = tf.nn.conv2d(input, weights, strides=[1, stride, stride, 1], padding='VALID', name='layer_1_conv')
```

Pesos

Distancia entre la que se aplican los filtros



Filtro: 3X3

4. Capas y funciones - Capa convolucional

Convolución + **Bias** + **Función de activación**

Capa anterior

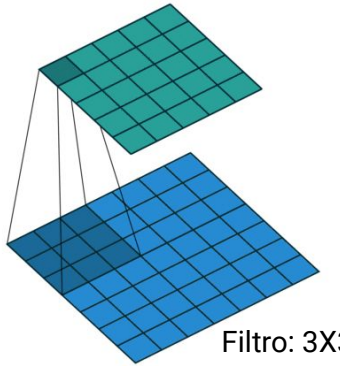
Modo de aplicación del filtro

```
conv_out = tf.nn.conv2d(input, weights, strides=[1, stride, stride, 1], padding='VALID', name='layer_1_conv')
```

Pesos

Distancia entre la que se aplican los filtros

```
bias_out= tf.add(conv_out, bias)
```



Filtro: 3X3

4. Capas y funciones - Capa convolucional

Convolución + **Bias** + **Función de activación**

Capa anterior

Modo de aplicación del filtro

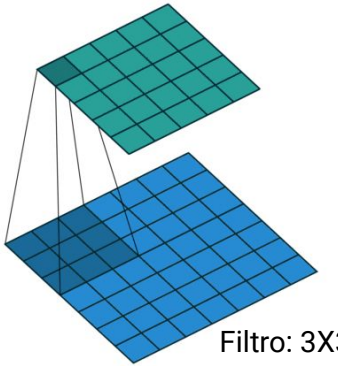
```
conv_out = tf.nn.conv2d(input, weights, strides=[1, stride, stride, 1], padding='VALID', name='layer_1_conv')
```

Pesos

Distancia entre la que se aplican los filtros

```
bias_out= tf.add(conv_out, bias)
```

bias



Filtro: 3X3

4. Capas y funciones - Capa convolucional

Convolución + **Bias** + **Función de activación**

Capa anterior

Modo de aplicación del filtro

```
conv_out = tf.nn.conv2d(input, weights, strides=[1, stride, stride, 1], padding='VALID', name='layer_1_conv')
```

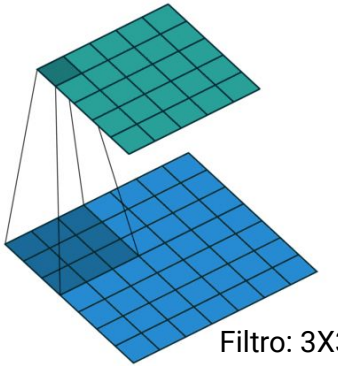
Pesos

Distancia entre la que se aplican los filtros

```
bias_out= tf.add(conv_out, bias)
```

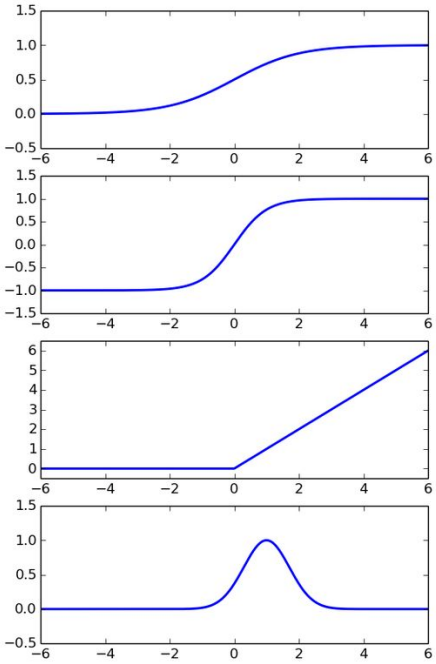
bias

```
output = tf.nn.relu(bias_out, name='Layer_1_activation_fun')
```



Filtro: 3X3

4. Capas y funciones - Funciones de activación



Sigmoid

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

Hyperbolic Tangent

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

Rectified Linear

$$\phi(z) = \begin{cases} 0 & \text{if } z < 0 \\ z & \text{if } z \geq 0 \end{cases}$$

Radial Basis Function

$$\phi(z, c) = e^{-(\epsilon \|z - c\|)^2}$$

La función de activación se encarga de generar una salida a partir de un valor de entrada, normalmente el conjunto de valores de salida está determinado mediante un rango como (0,1) o (-1,1).

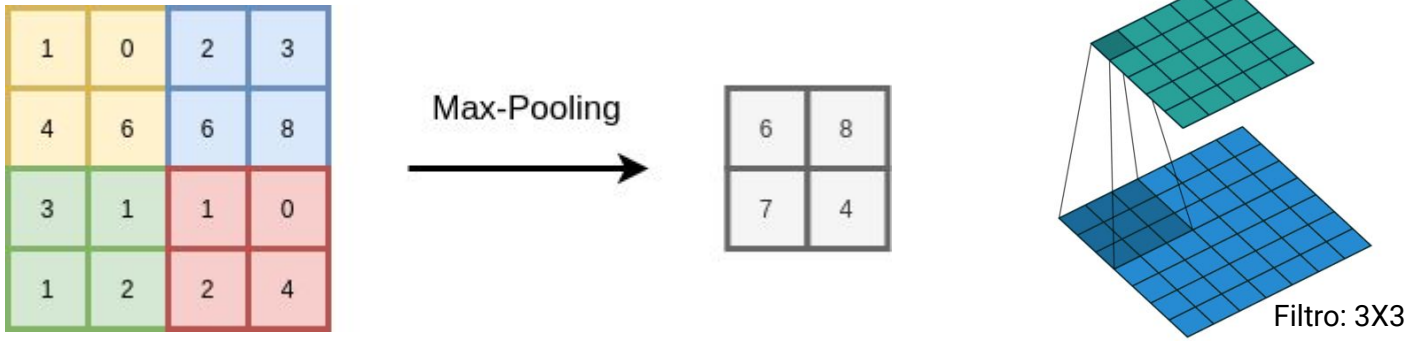
Función ReLU

La función ReLU (Rectified Linear Unit) transforma los valores de entrada anulando los valores negativos y manteniendo los positivos tal y como entran.

- Función de tipo Sparse, es decir sólo se activan los positivos.
- No está acotada.
- Buen comportamiento con valores positivos (imágenes).
- Mal comportamiento con valores negativos (muerte de neuronas).

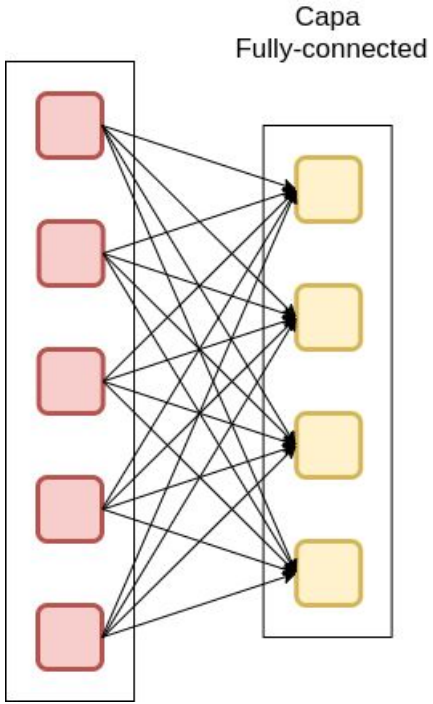
4. Capas y funciones - Capa Pooling

Las capas pooling se utilizan para la realización de una reducción de información con el objetivo de centrarse en cierta información dependiendo de la operación a realizar. Esta reducción se realiza mediante la utilización de funciones como el promedio, el máximo o el mínimo.




```
pool_out = tf.nn.max_pool(input, ksize=[1, k, k, 1], strides=[1, k, k, 1], padding='SAME')
```

4. Capas y funciones - Fully Connected



La capa completamente conectada (fully-connected) es uno de los componentes esenciales de las redes convolucionales.

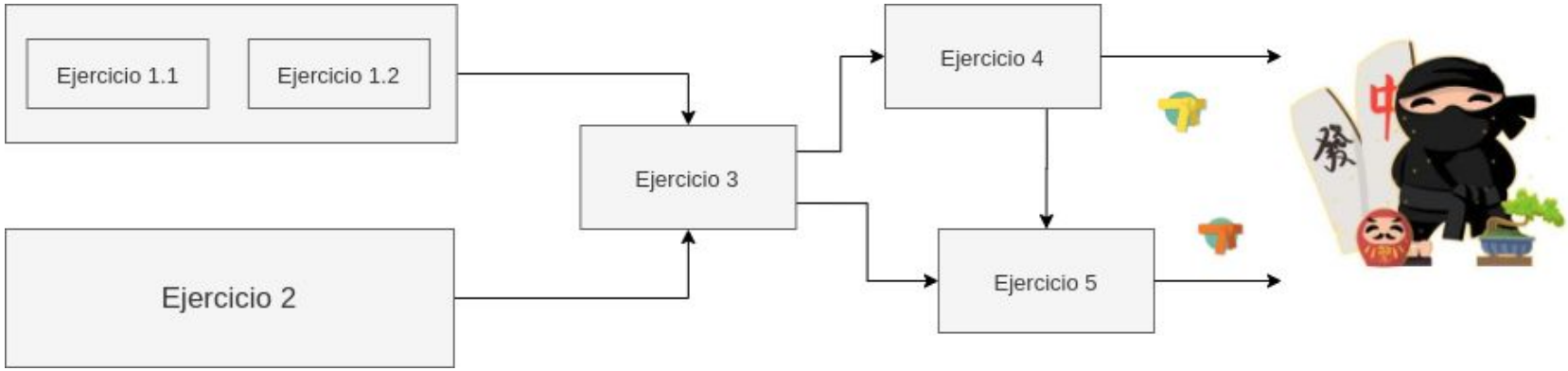
Su funcionamiento consiste en realizar una operación de “flattens” que es utilizada como salida de la red o entrada de la siguiente capa. La operación de “flatten” consiste en realizar un **aplanamiento** convirtiendo la información obtenida en un vector de **una dimensión**.

5. Itinerarios del taller

Red Hat

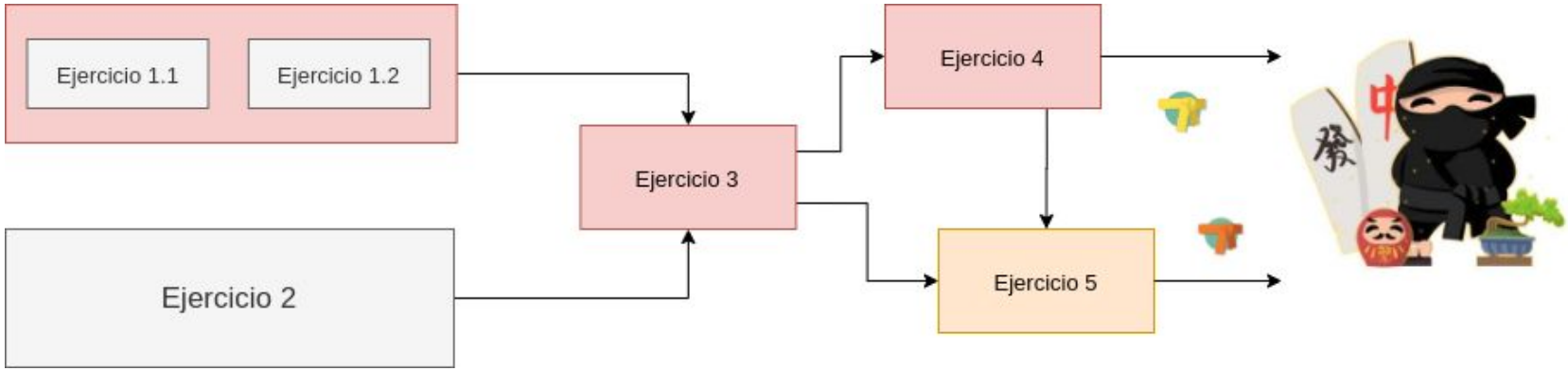
5. Itinerarios del taller

El taller tiene un conjunto de 6 ejercicios cuya realización depende de la tecnología que utilice el existente para la realización del taller.



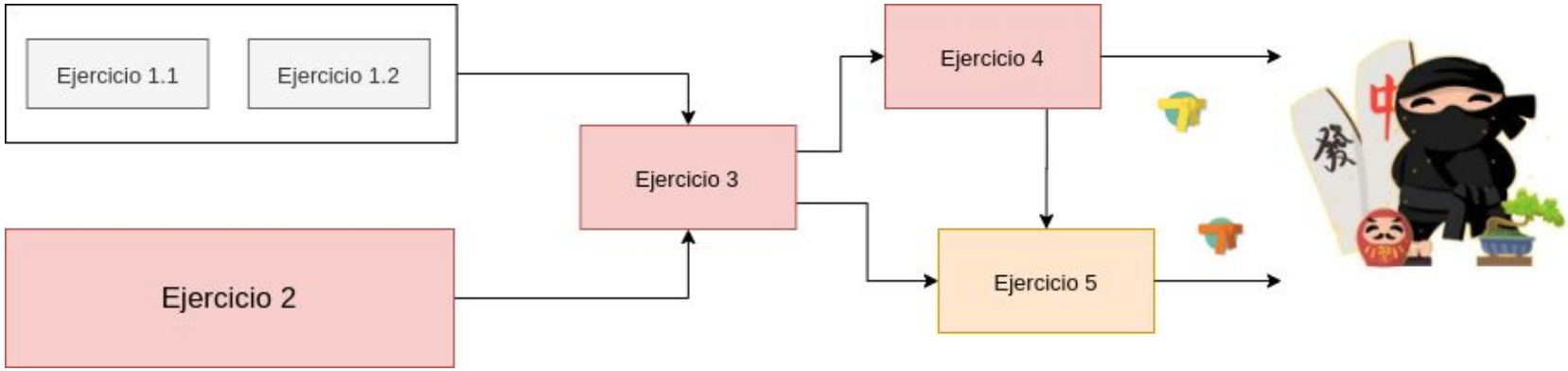
5. Itinerarios del taller

El itinerario de la dockerización



5. Itinerarios del taller

El itinerario de la colaboración en el cloud



Itinerario recomendado

¡Muchas Gracias!

¿Preguntas?

@moisipm