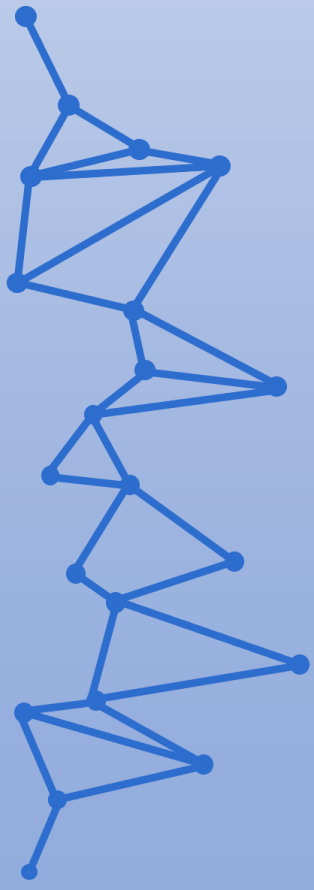
Curso de Especialización de
Inteligencia Artificial y Big Data (IABD)



Programación de Inteligencia Artificial

UD07. Algoritmos avanzados de Deep Learning.
Resumen.

JUAN ANTONIO GARCIA MUELAS

Las redes neuronales convolucionales han revolucionado el campo de la visión artificial.

En la operación convolucional, utilizamos una ventana o matriz que se va desplazando sobre la imagen y aplicando producto tensorial sobre los píxeles de la ventana en cada posición.

La principal ventaja de este tipo de redes es que se establecen las capas con un orden jerárquico, estando cada capa "especializada" en detectar un nivel de comprensión de la imagen (una puede detectar sólo bordes, otra ruedas, o puertas…)

Las **capas convolucionales** tienen **dos parámetros** principales:

- ✓ **Dimensiones de la ventana** que va aplicando la transformación: suelen ser de **3x3 o** de **5x5**.
- ✓ **Profundidad del mapa** de salida: es el **número de filtros en paralelo** que tiene la capa y que darán otros tantos mapas de salida.

En Keras, programamos este tipo de capa con estos parámetros:

```
Conv2D(output_depth, (window_height, window_width))
```

donde **`output_depth`** es la **profundidad** (recuerda, el número de **imágenes de salida**, igual que las capas neuronales tenían número de neuronas) y **`window_height`** y **`window_width`** son las **dimensiones de la matriz** de transformación o filtro.

`output_depth` hace que la red convolucional produzca un "desdoble" o aumento de número de coeficientes en el modelo.

Padding es una técnica se utiliza para evitar el efecto borde en la operación convolucional, que provoca que la imagen de salida sea de menores dimensiones que la de entrada.

Es bastante común introducir una capa de **`"pooling"`** o de **reducción entre las capas** convolucionales, para mantener bajo control el tamaño de la red.

La capa de max-pooling no afecta ni modifica la dimensión de profundidad de la red.

Otro efecto que se evita gracias a las capas max-pooling es la pérdida de información del contorno de la imagen.

Se suele aplicar una **ventana de 2x2 con un `stride`** (hacer que la ventana de la convolución no vaya recorriendo los píxeles seguidos, sino **que se vaya saltando un cierto número** de ellos) de dos elementos, **lo cual reduce el tamaño a la mitad**.

Estos dos parámetros se introducen en la capa de esta manera:

```
MaxPooling2D(window_dim,strides)
```

El modelo de una red neuronal convolucional requiere que, **tras las capas convolucionales y de `pooling`**, se añadan **capas neuronales densamente conectadas** que son las que ponderan la contribución al error que introducen las diferentes variables de salida del bloque convolucional y que hace que la propagación hacia atrás o **`"back propagation"`** haga su trabajo. Como la **salida del bloque convolucional** es un tensor 3D, es necesario "aplanarlo" con una capa de tipo **`Flatten`**.

Una de las técnicas que más se utilizan en la industria es utilizar redes pre-entrenadas con grandes datasets.

La principal ventaja de estas redes es que tienen bien jerarquizadas sus capas (o conjuntos de capas) lo que hace que sirvan como modelo de base para todo lo que tenga que ver con el mundo visual o lingüístico. Son bastante conocidas las **arquitecturas VGG, ResNet, Xception**, etc, en el ámbito de la visión artificial.

```
tf.keras.applications.VGG16(
    include_top=True,
    weights="imagenet",
    input_tensor=None,
    input_shape=None,
    pooling=None,
    classes=1000,
    classifier_activation="softmax",
)
```

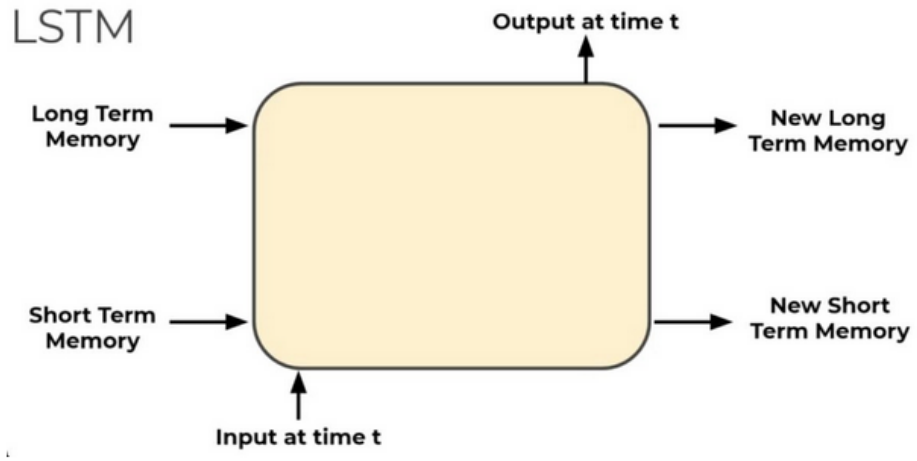
Las **redes pre-entrenadas** se pueden **utilizar** de dos maneras:

- ✓ **"Feature Extraction"** es la **técnica más rápida y menos pesada** con la que utilizar el modelo pre-entrenado **para extraer los patrones** principales que dicho **modelo detecta en las imágenes** que le estamos pasando. Después, pasaremos ese mapa de **"features"** o **características**, a un modelo sencillo de red neuronal, para su entrenamiento.
- ✓ **"Fine-tunning"** es la técnica que consiste en componer un **modelo híbrido** en el que, **al final de** la red pre-entrenada, **ponemos dos capas de red neuronal, y congelamos casi todas la capas** de la red pre-entrenada para aprovechar sus primeras jerarquías de detección**, y ya solo entrenar el último bloque y las capas que hemos añadido**.

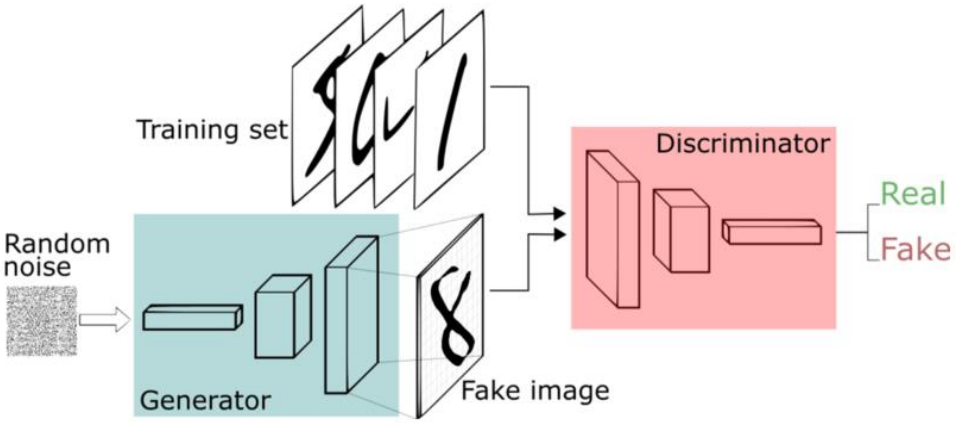
Las **redes neuronales recurrentes (RNN)** fueron modelizadas en la **década de los 80**, pero estas redes **han sido muy difíciles de entrenar por sus requerimientos en computación**. Con la llegada de los avances de estos últimos años, en materia de hardware, **se han vuelto más accesibles** y se ha popularizado su uso por la industria.

La clase de Keras que nos permite instanciar una capa neuronal recurrente es SimpleRNN.

Los **modelos Long-Short Term Memory (LSTM)** son una extensión de las redes neuronales recurrentes, que básicamente **amplían su memoria** para **tener en cuenta los estados** cronológicamente más antiguos.



Las redes **generativas adversarias o redes GAN** (**tipo de modelo que combina dos redes que compiten en un juego de suma cero**) son una arquitectura que implican, en realidad, la **interacción entre dos modelos de redes neuronales profundas**.



El sistema está listo cuando el **generador** consigue generar casos que el **discriminador** da por válidos.