

# Redes convolucionales II

Métodos Generativos, curso 2025-2026

---

Guillermo Iglesias, [guillermo.iglesias@upm.es](mailto:guillermo.iglesias@upm.es)

Jorge Dueñas Lerín, [jorge.duenas.lerin@upm.es](mailto:jorge.duenas.lerin@upm.es)

Edgar Talavera Muñoz, [e.talavera@upm.es](mailto:e.talavera@upm.es)

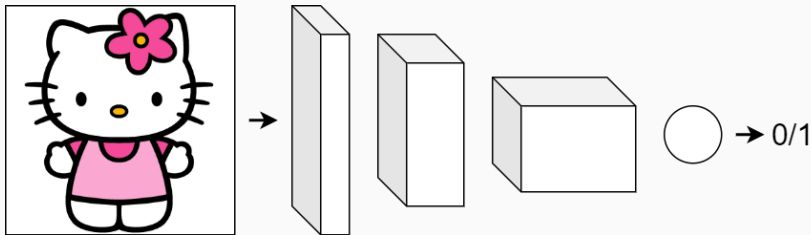
5 de noviembre de 2025

Escuela Técnica Superior de Ingeniería de Sistemas Informáticos | UPM



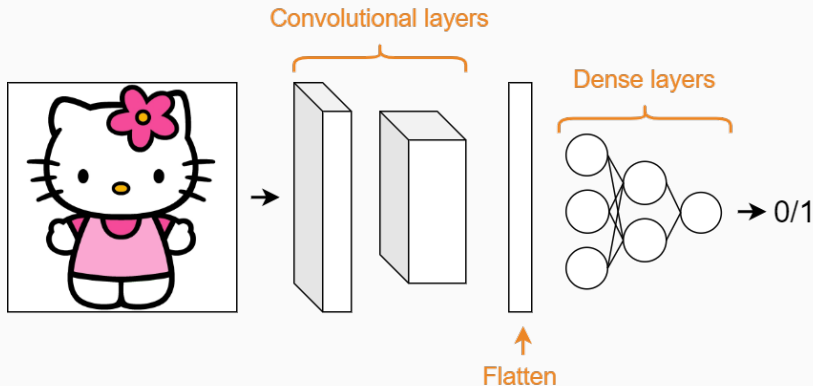
# Construcción de una CNN

La estructura de **embudo** típica de las redes neuronales **clasificadoras** también se aplica a **Convolutional Neural Network (CNN)**. Para ello el objetivo es **reducir** la dimensión de la imagen hasta generar una salida.



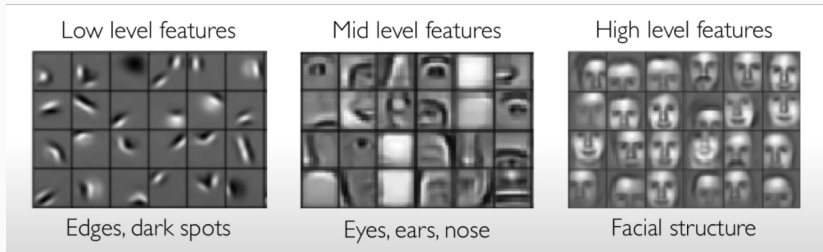
# Construcción de una CNN

Las primeras capas **convolucionales** de la red se encargan de la **extracción de características** de la imagen. Posteriormente un **perceptrón** se encarga de **clasificar** las características extraídas para generar la **salida deseada**.



# Construcción de una CNN

Es importante recordar que la **jerarquía** de capas de una red convolucional detecta características a **alto nivel** en las capas **más profundas**.



[1]

# Reducción dimensional en redes convolucionales

Para formar el “embudo” de la red se utilizan distintos mecanismos para reducir las dimensiones de la información de la red. En concreto los dos mecanismos predominantes son:

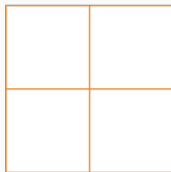
- Capas de pooling
  - MaxPooling
  - AveragePooling
- Convoluciones de strides=(2,2)

# Reducción dimensional con Pooling

**MaxPooling 2D** La capa de MaxPooling2D reduce la dimensión de un vector cogiendo el **máximo** de la ventana definida.

$\text{pool\_size} = (2, 2)$

1	2	0	2
1	4	2	0
4	3	1	0
1	2	2	1



# Reducción dimensional con Pooling

**MaxPooling 2D** La capa de MaxPooling2D reduce la dimensión de un vector cogiendo el **máximo** de la ventana definida.

$\text{pool\_size} = (2, 2)$

1	2	0	2
1	4	2	0
4	3	1	0
1	2	2	1

4	

## Reducción dimensional con Pooling

**MaxPooling 2D** La capa de MaxPooling2D reduce la dimensión de un vector cogiendo el **máximo** de la ventana definida.

$\text{pool\_size} = (2, 2)$

1	2	0	2
1	4	2	0
4	3	1	0
1	2	2	1

4	2



# Reducción dimensional con Pooling

**MaxPooling 2D** La capa de MaxPooling2D reduce la dimensión de un vector cogiendo el **máximo** de la ventana definida.

$\text{pool\_size} = (2, 2)$

1	2	0	2
1	4	2	0
4	3	1	0
1	2	2	1

4	2
4	

# Reducción dimensional con Pooling

**MaxPooling 2D** La capa de MaxPooling2D reduce la dimensión de un vector cogiendo el **máximo** de la ventana definida.

$\text{pool\_size} = (2, 2)$

1	2	0	2
1	4	2	0
4	3	1	0
1	2	2	1

4	2
4	2

# Reducción dimensional con Pooling

**MaxPooling 2D** La capa de MaxPooling2D reduce la dimensión de un vector cogiendo el **máximo** de la ventana definida.

$\text{pool\_size} = (2, 2)$

1	2	0	2
1	4	2	0
4	3	1	0
1	2	2	1

4	2
4	2

*\*cabe destacar que el máximo se encarga de preservar la característica más importante*

## Reducción dimensional con Pooling

**AveragePooling 2D** La capa de AveragePooling2D reduce la dimensión de un vector cogiendo el **promedio** de la ventana definida.

$\text{pool\_size} = (2, 2)$

1	2	0	2
1	4	2	0
4	3	1	0
1	2	2	1

2	

## Reducción dimensional con Pooling

**AveragePooling 2D** La capa de AveragePooling2D reduce la dimensión de un vector cogiendo el **promedio** de la ventana definida.

$\text{pool\_size} = (2, 2)$

1	2	0	2
1	4	2	0
4	3	1	0
1	2	2	1

2	1

# Reducción dimensional con Pooling

**AveragePooling 2D** La capa de AveragePooling2D reduce la dimensión de un vector cogiendo el **promedio** de la ventana definida<sup>1</sup>.

`pool_size = (2, 2)`

1	2	0	2
1	4	2	0
4	3	1	0
1	2	2	1

2	1
2.5	1

<sup>1</sup>Pese a su similar rendimiento, el MaxPooling suele generar mejores resultados que el AveragePooling [2]

# Reducción dimensional con strides

Otra **opción** para la reducción dimensional es el uso de convoluciones con **strides** que no sean (1, 1).

- Cierta literatura apunta a esta aproximación como “**más inteligente**” ya que se le permite a la **red** que sea ella la que escoja **cómo hacer** la reducción.
- El principal **inconveniente** es que usando este método se aumenta el número de parámetros de la red.

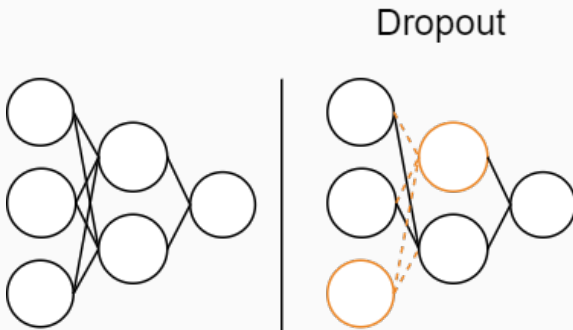
Existe un debate abierto sobre si es preferible que la red **aprenda** a realizar el *downsampling* [3], o si resulta más adecuado emplear técnicas como **MaxPooling** [4].

*\*Normalmente el stride elegido es de (2, 2) pero hay libertad para adaptarlo a distintos casos.*

# Capa de dropout

La capa de **dropout** es una capa de **regularización** que **desactiva aleatoriamente** la activación de ciertas **neuronas** durante el entreno.

Al regularizar la red previene de problemas como el **overfitting**.





# Tuneo de CNNs

---

# Los hiperparámetros en una red

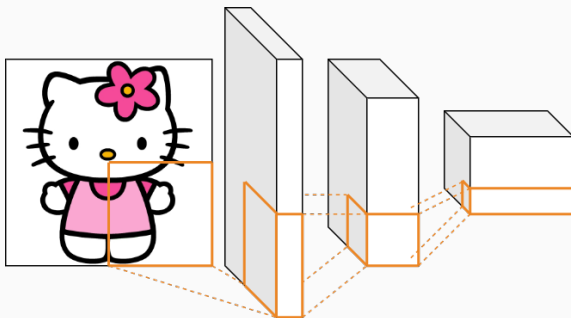
Uno de los mayores **inconvenientes** a la hora de realizar entrenamientos con redes neuronales artificiales es su **difícil configuración**. Debido a la cantidad inmensa de **hiperparámetros** a escoger.

Sin embargo, existen una serie de **prácticas comunes** a la hora de tratar con **CNNs**.

# Tamaño de imagen y filtros

A la hora de escoger el **número de filtros** de cada capa convolucional este va **ligado** al tamaño de la **matriz de datos**.

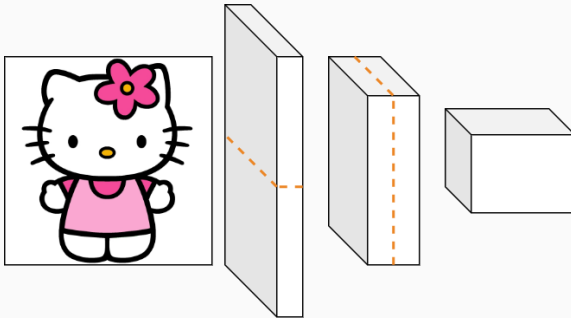
A medida que la imagen de entrada va **reduciendo** su tamaño, el número de filtros **aumenta**. Con esto se pretende extraer más **características de alto nivel** cada vez cubriendo zonas más amplias de la imagen original.



# Tamaño de imagen y filtros

Al mismo tiempo, a medida que el **número** de filtros de **multiplica por 2** las **dimensiones** de la matriz de datos se **reducen a la mitad**.

El objetivo de este **intercambio** es mantener la **misma cantidad** de información, pero tratada por la red.



## kernel\_size

El tamaño del kernel **habitualmente** es de  $(3, 3)$  o  $(5, 5)$ , en caso de imágenes muy grandes puede llegar a  $(7, 7)$ .

Para matrices de datos **más grandes** se utilizan **kernels más grandes**, en casos combinando kernels de  $(5, 5)$  para las **primeras capas** y posteriormente  $(3, 3)$  para capas más **profundas**<sup>2</sup>.

## strides

El paso de la convolución se mantiene a  $(1, 1)$  a no ser que se desee una **reducción dimensional**.

---

<sup>2</sup>Investigaciones posteriores [5] han demostrado que es más eficiente apilar dos capas convoluciones de  $(3, 3)$  que una única capa de  $(5, 5)$ .

## padding

El padding de una convolución suele ser **same** para controlar las dimensiones de la matriz de datos, pero no es extraño encontrar casos con padding **valid**.

## activation

Para las **capas ocultas** se suele utilizar la función **ReLU** o **LeakyReLU**, para la capa de **salida** la activación depende del **problema concreto**.



- 1.2\_3-CNNModa.ipynb



- 1.2\_4-CNNDigitos.ipynb

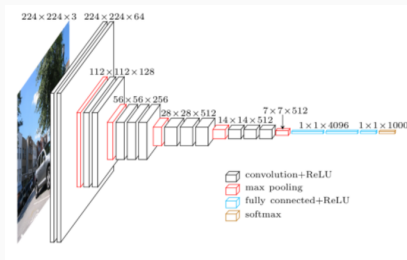


## Idea general

Usar un modelo preentrenado en un dataset grande (ej. ImageNet) para una nueva tarea.

- Enfoques comunes
  - **Feature extraction**: congelar capas y usar sus salidas como extractor de características.
  - **Fine-tuning**: reentrenar parcialmente algunas capas junto con las nuevas.
- Ventajas
  - Reduce el tiempo de entrenamiento.
  - Mejora los resultados con pocos datos.
  - Facilita el uso de arquitecturas complejas.

# Ejemplo: VGG16



- VGG16
  - Red profunda propuesta por Simonyan y Zisserman (2014).  
<https://arxiv.org/abs/1409.1556>
  - Basada en bloques de convoluciones en embudo y luego densa.
- Uso en Transfer Learning
  - Importar el modelo preentrenado. **Congelar sus pesos.**
  - Sustituir densa por nueva densa.
  - Opcionalmente: descangelar algún filtro del final.



- 1.2\_05-Cuadros.ipynb

- [1] Kathrin Melcher (Medium).  
**Convolutional hierarchy image.**  
<https://medium.com/low-code-for-advanced-data-science/introduction-to-convolutional-neural-networks-and-computer-vision-72b2d85dd1c0>.  
[Online; accessed August, 2022].
- [2] Florentin Bieder, Robin Sandkühler, and Philippe C Cattin.  
**Comparison of methods generalizing max-and average-pooling.**  
*arXiv preprint arXiv:2103.01746*, 2021.
- [3] itdxer (StackExchange).  
**Strides vs maxpooling debate.**  
<https://stats.stackexchange.com/questions/387482/pooling-vs-stride-for-downsampling>.  
[Online; accessed September, 2023].

- [4] Shuyang Sun, Jiangmiao Pang, Jianping Shi, Shuai Yi, and Wanli Ouyang.  
**Fishnet: A versatile backbone for image, region, and pixel level prediction.**  
*Advances in neural information processing systems*, 31, 2018.
- [5] Karen Simonyan and Andrew Zisserman.  
**Very deep convolutional networks for large-scale image recognition.**  
*arXiv preprint arXiv:1409.1556*, 2014.

- Autor original de las diapositivas: Guillermo Iglesias Hernández