

CNN: Convolutional Neural Networks

Beatriz Remeseiro

ÍNDICE

1. Introducción
2. Filtros y convoluciones
3. Redes convolucionales
4. Capas de las redes convolucionales
5. Arquitecturas y entrenamiento
6. Casos de estudio

ÍNDICE

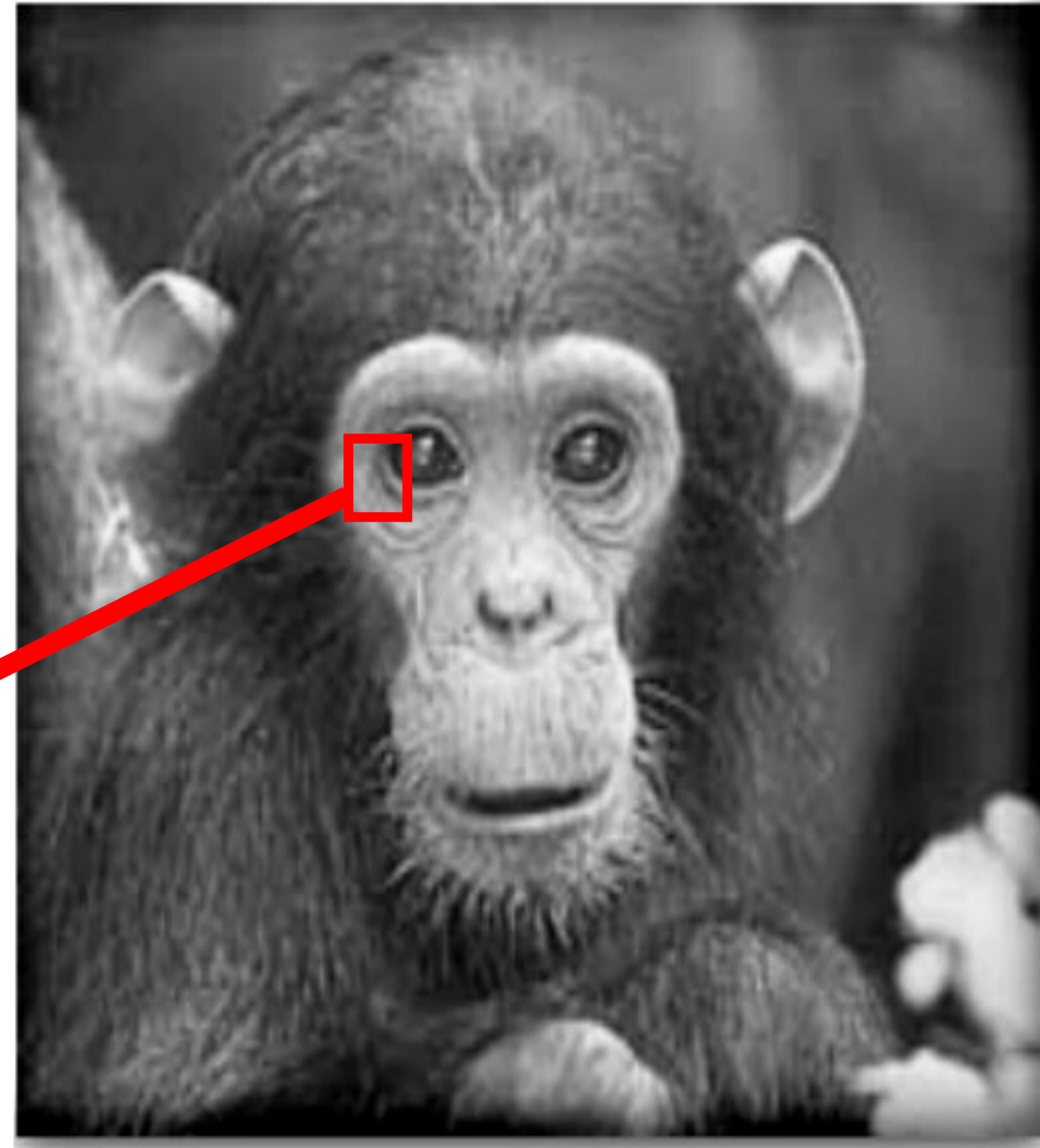
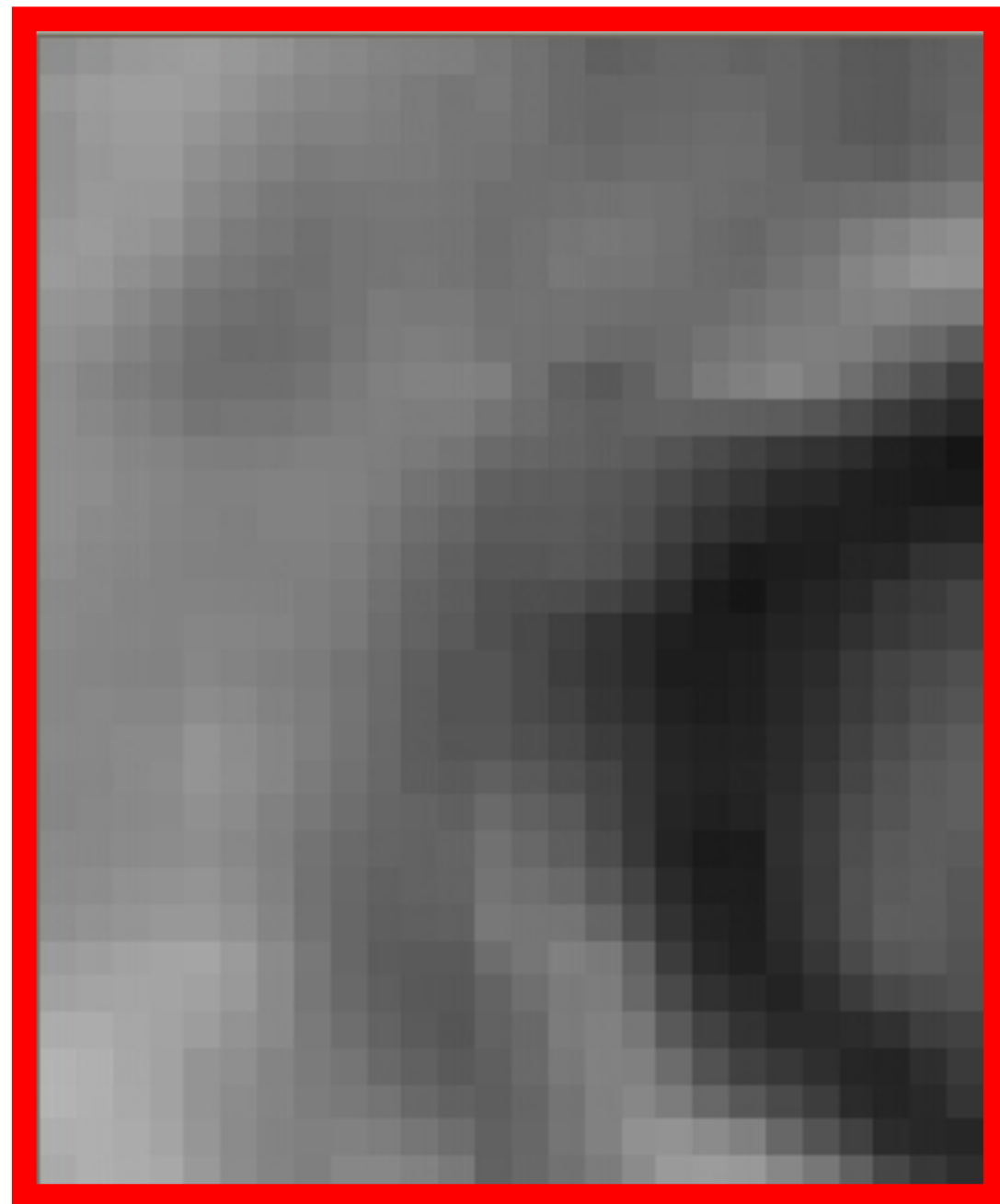
1. Introducción
2. Filtros y convoluciones
3. Redes convolucionales
4. Capas de las redes convolucionales
5. Arquitecturas y entrenamiento
6. Casos de estudio

Imágenes digitales

Intensidad: $M[i,j] \in [0,255]$

Normalizada: $M[i,j] \in [0,1]$

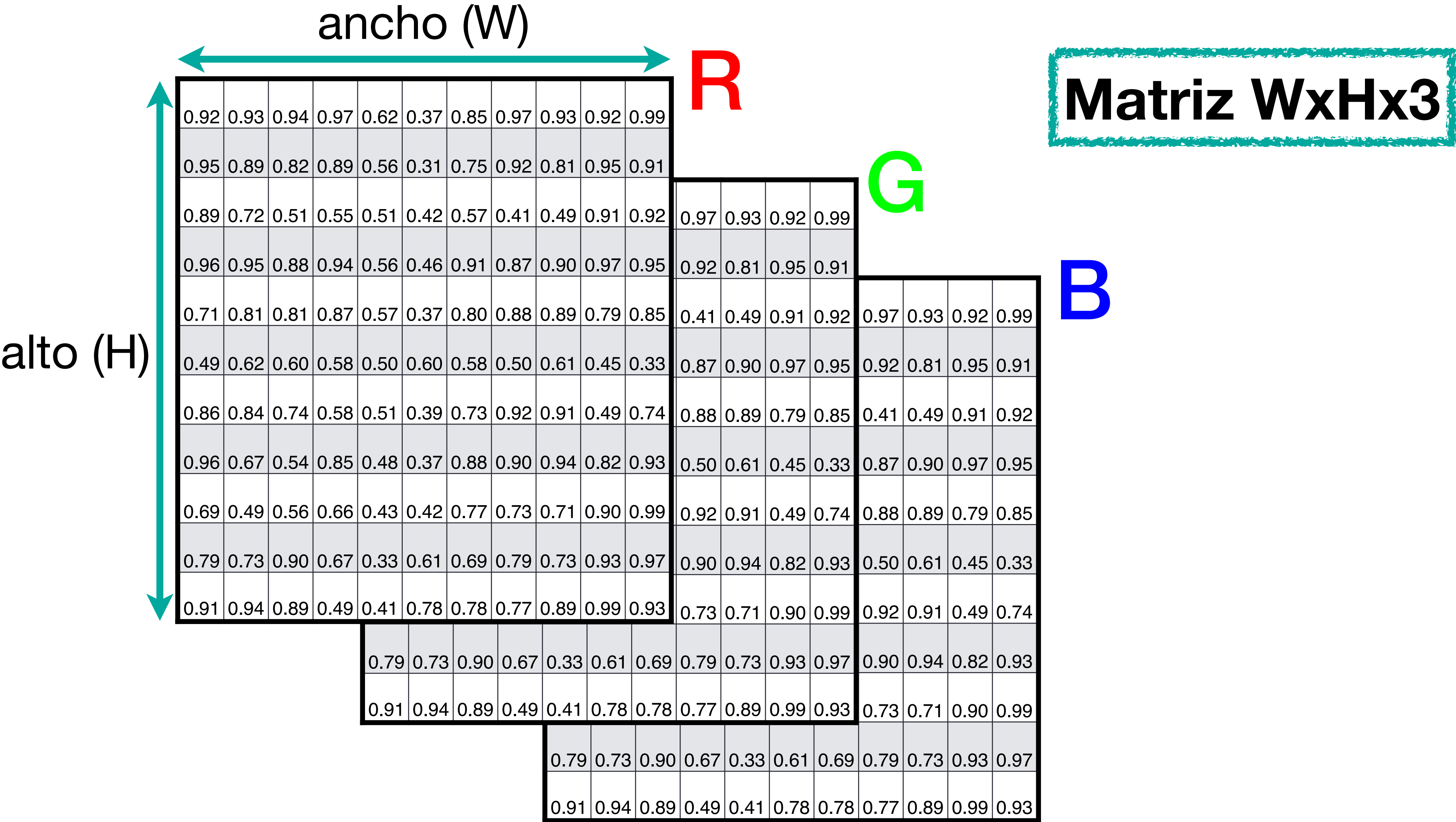
Matriz WxH



alto (H)

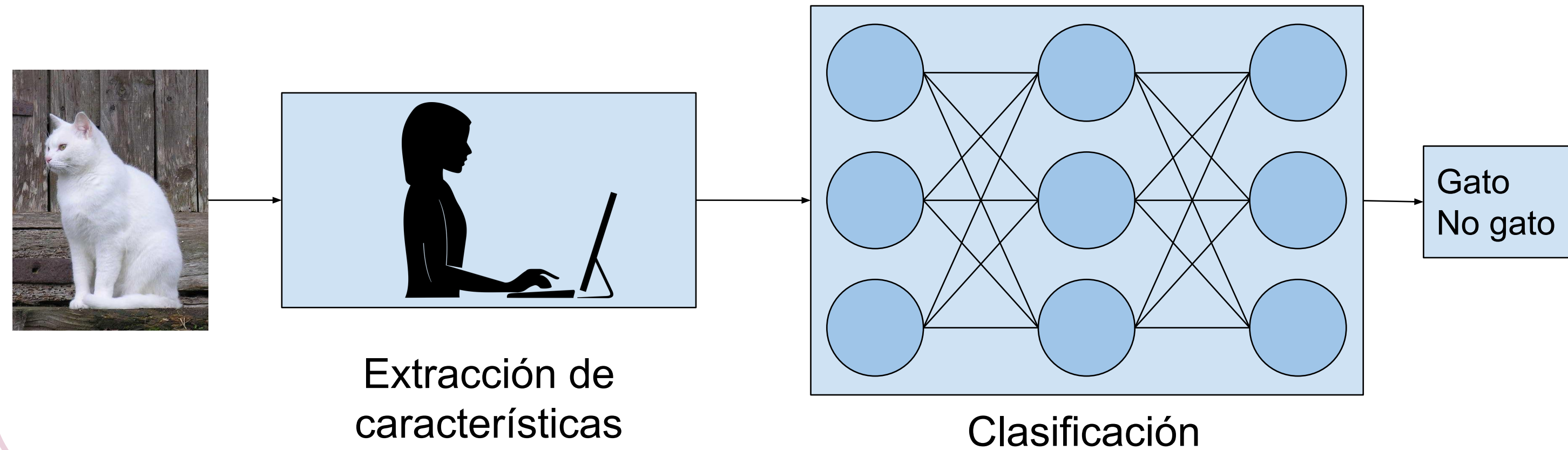
ancho (W)

Imágenes digitales

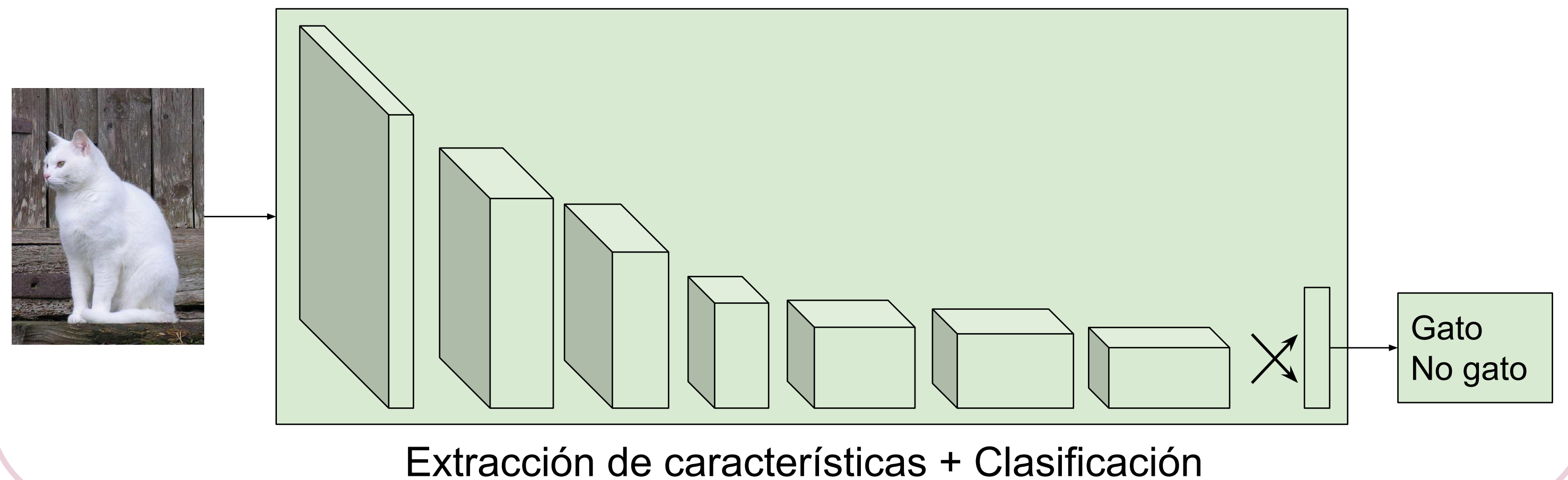


Aproximaciones

MACHINE LEARNING



DEEP LEARNING



ÍNDICE

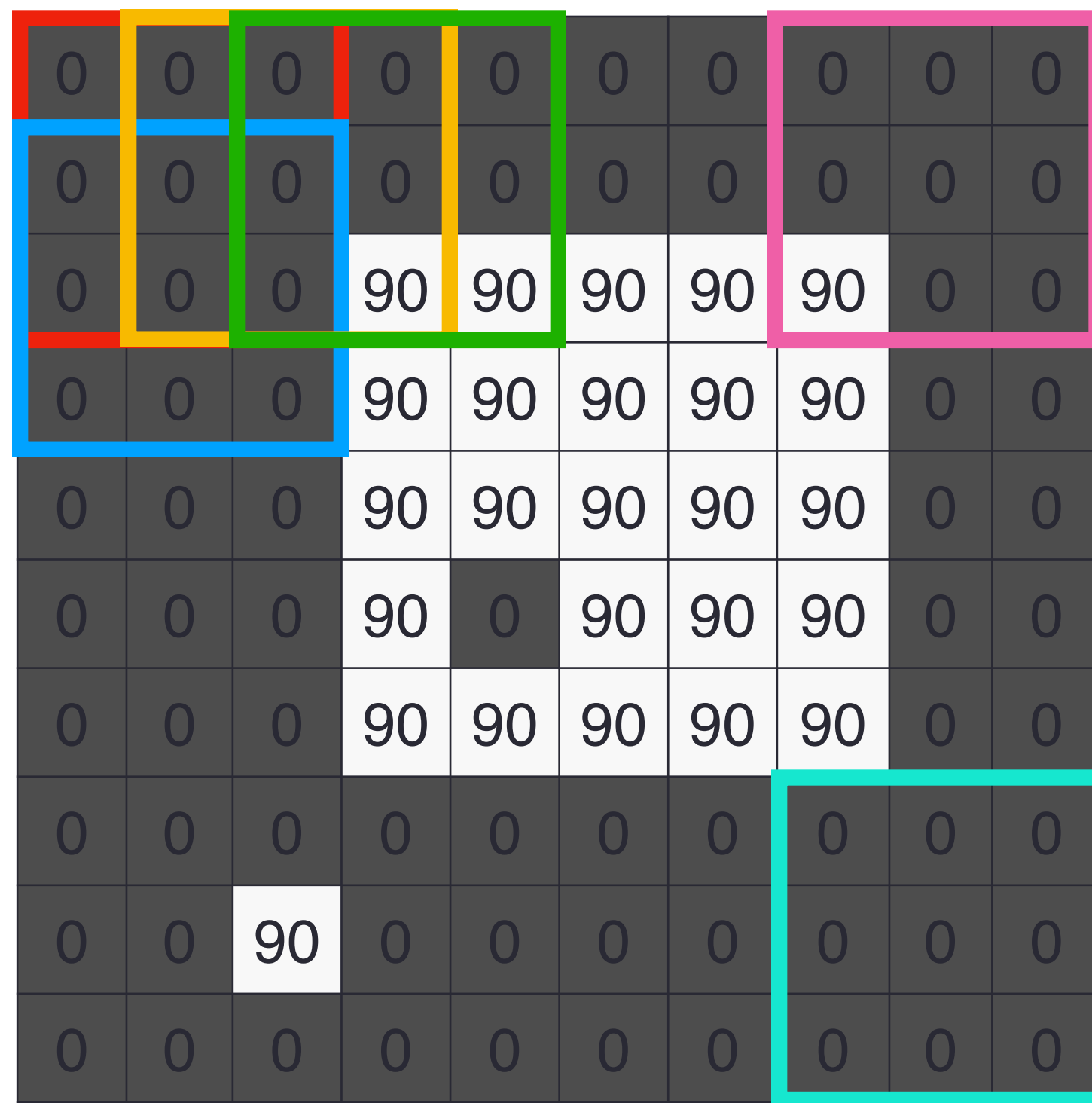
1. Introducción
- 2. Filtros y convoluciones**
3. Redes convolucionales
4. Capas de las redes convolucionales
5. Arquitecturas y entrenamiento
6. Casos de estudio

Filtrado de imágenes

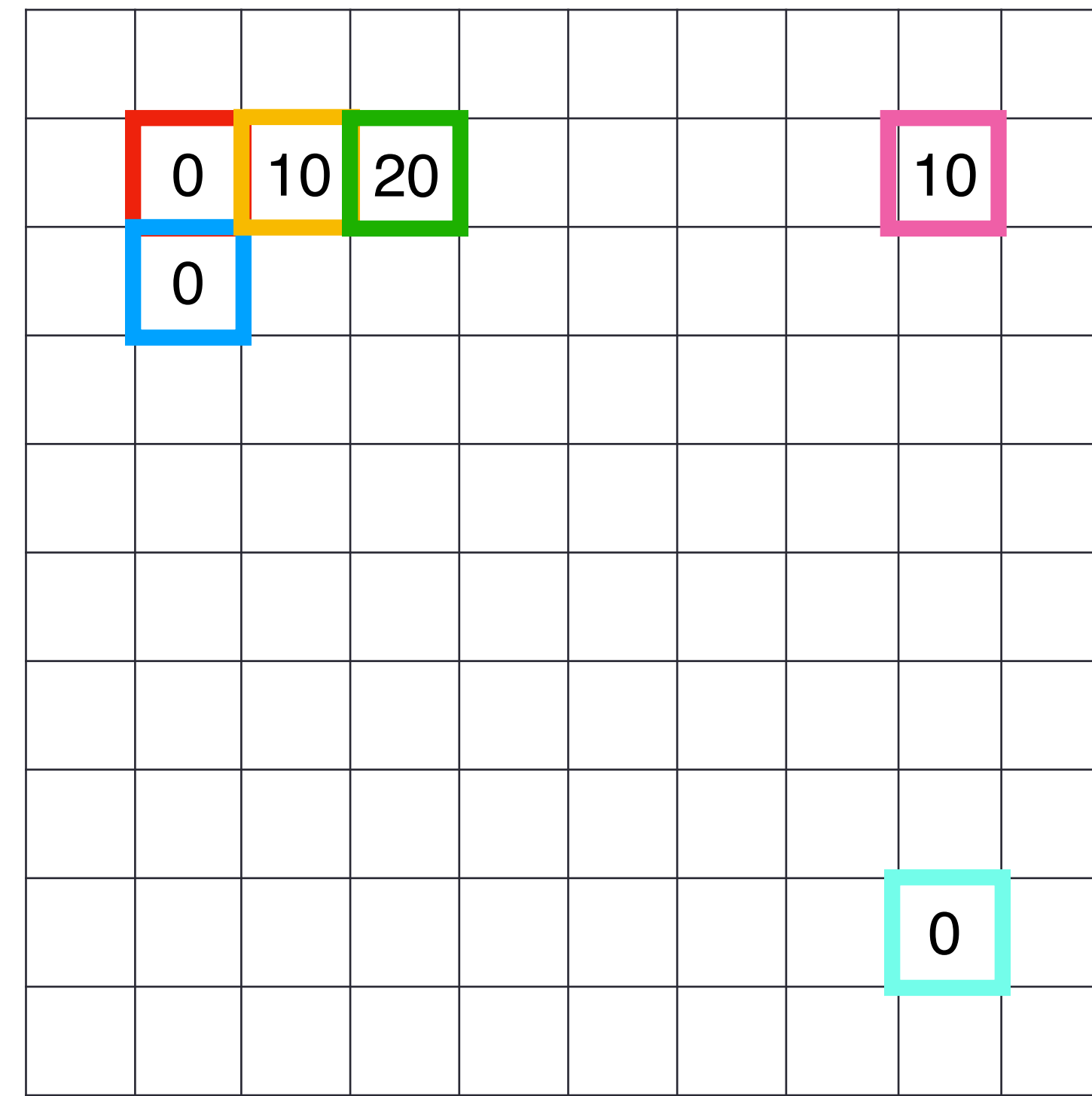
- **Filtrar una imagen:** Aplicar una función a la vecindad de cada píxel
 - Función definida por un **filtro** que indica cómo combinar los valores de los vecinos
 - Vecindad definida por el **tamaño** del filtro
- Usos de los filtros:
 - Mejora de imágenes (eliminar ruido, redimensionar, etc.)
 - Extracción de información (texturas, bordes, etc.)
 - Detección de patrones

Filtrado de imágenes

Filtro de media 3x3



$F[x,y]$



$G[x,y]$

Filtrado de imágenes

Filtro de media 3x3

| | | | | | | | | | |
|---|---|----|----|----|----|----|----|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 0 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 90 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$F[x,y]$




| | | | | | | | | | |
|--|----|----|----|----|----|----|----|----|--|
| | | | | | | | | | |
| | 0 | 10 | 20 | 30 | 30 | 30 | 20 | 10 | |
| | 0 | 20 | 40 | 60 | 60 | 60 | 40 | 20 | |
| | 0 | 30 | 60 | 90 | 90 | 90 | 60 | 30 | |
| | 0 | 30 | 50 | 80 | 80 | 90 | 60 | 30 | |
| | 0 | 30 | 50 | 80 | 80 | 90 | 60 | 30 | |
| | 0 | 20 | 30 | 50 | 50 | 60 | 40 | 20 | |
| | 10 | 20 | 30 | 30 | 30 | 30 | 20 | 10 | |
| | 10 | 10 | 10 | 0 | 0 | 0 | 0 | 0 | |
| | | | | | | | | | |

$G[x,y]$

Convolución

- Supongamos una ventana cuadrada de tamaño **$2k + 1$**

$$G[i, j] = \frac{1}{(2k + 1)^2} \sum_{u=-k}^k \sum_{v=-k}^k F[i + u, j + v]$$


Peso uniforme de cada píxel **Vecinos del píxel $F[i, j]$**

Convolución

- Expresión generalizada para **diferentes pesos**, dependiendo de la posición relativa de cada píxel en la vecindad

$$G[i, j] = \sum_{u=-k}^k \sum_{v=-k}^k \boxed{H[u, v]} F[i + u, j + v]$$



Peso no uniforme de cada píxel

Convolución

$$G[i, j] = \sum_{u=-k}^k \sum_{v=-k}^k H[u, v] F[i + u, j + v]$$

- Esta operación se denominada **convolución**: $G = H \otimes F$
- **Filtrar una imagen**: sustituir cada píxel por una combinación lineal de sus píxeles vecinos
 - El **filtro** o **máscara** $H[u, v]$ determina el peso de cada píxel vecino en la combinación lineal

Convolución

$$G = H \otimes F$$

Filtro de media 3x3

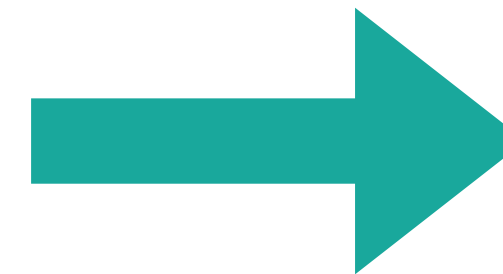
| | | | | | | | | | |
|---|---|----|----|----|----|----|----|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 0 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 90 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$F[x,y]$

\otimes

| | | |
|-----|-----|-----|
| 1/9 | 1/9 | 1/9 |
| 1/9 | 1/9 | 1/9 |
| 1/9 | 1/9 | 1/9 |

$H[u,v]$



| | | | | | | | | | |
|--|---|----|----|----|--|--|--|--|--|
| | | | | | | | | | |
| | 0 | 10 | 20 | 30 | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |

$G[x,y]$

Convolución

| | | | | | | | | | |
|---|---|----|----|----|----|----|----|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 0 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 90 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$F[x,y]$ **10x10**

\otimes

| | | |
|-----|-----|-----|
| 1/9 | 1/9 | 1/9 |
| 1/9 | 1/9 | 1/9 |
| 1/9 | 1/9 | 1/9 |

$H[u,v]$
3x3



| | | | | | | | | | |
|--|----|----|----|----|----|----|----|----|--|
| | | | | | | | | | |
| | 0 | 10 | 20 | 30 | 30 | 30 | 20 | 10 | |
| | 0 | 20 | 40 | 60 | 60 | 60 | 40 | 20 | |
| | 0 | 30 | 60 | 90 | 90 | 90 | 60 | 30 | |
| | 0 | 30 | 50 | 80 | 80 | 90 | 60 | 30 | |
| | 0 | 30 | 50 | 80 | 80 | 90 | 60 | 30 | |
| | 0 | 20 | 30 | 50 | 50 | 60 | 40 | 20 | |
| | 10 | 20 | 30 | 30 | 30 | 30 | 20 | 10 | |
| | 10 | 10 | 10 | 0 | 0 | 0 | 0 | 0 | |
| | | | | | | | | | |

$G[x,y]$ **8x8**

$$F : m \times n$$

$$H : (2k + 1) \times (2k + 1)$$

$$G = H \otimes F$$

$$G : (m - (2k + 1) + 1) \times (n - (2k + 1) + 1)$$

$$G : (m - 2k) \times (n - 2k)$$

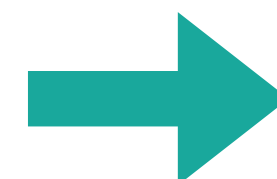
| | | | | | | | | | |
|---|---|----|----|----|----|----|----|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 0 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 90 | 90 | 90 | 90 | 90 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 90 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$F[x,y]$ **10x10**

\otimes

| | | | | |
|------|------|------|------|------|
| 1/25 | 1/25 | 1/25 | 1/25 | 1/25 |
| 1/25 | 1/25 | 1/25 | 1/25 | 1/25 |
| 1/25 | 1/25 | 1/25 | 1/25 | 1/25 |
| 1/25 | 1/25 | 1/25 | 1/25 | 1/25 |
| 1/25 | 1/25 | 1/25 | 1/25 | 1/25 |

$H[u,v]$
5x5



| | | | | | | | | | |
|--|--|------|------|------|------|------|------|--|--|
| | | | | | | | | | |
| | | | | | | | | | |
| | | 21.6 | 32.4 | 43.2 | 54.0 | 43.2 | 32.4 | | |
| | | 25.2 | 39.6 | 54.0 | 68.4 | 54.0 | 43.2 | | |
| | | 32.4 | 50.4 | 68.4 | 86.4 | 68.4 | 54.0 | | |
| | | 25.2 | 39.6 | 54.0 | 68.4 | 54.0 | 43.2 | | |
| | | 21.6 | 32.4 | 43.2 | 50.4 | 39.6 | 32.4 | | |
| | | 14.4 | 21.6 | 28.8 | 32.4 | 25.2 | 21.6 | | |
| | | | | | | | | | |
| | | | | | | | | | |

$G[x,y]$ **6x6**

Convolución

$$\mathbf{G} = \mathbf{H} \otimes \mathbf{F}$$

$$G[i, j] = \sum_{u=-k}^k \sum_{v=-k}^k H[u, v] F[i + u, j + v]$$

- Dos parámetros importantes, que afectan a las dimensiones de G:
 - **Padding:** píxeles de relleno que se añaden a F
 - **Stride:** determina cómo se desplaza el filtro H sobre F

Convolución

- **Padding:** píxeles de relleno que se añaden alrededor de la entrada para que la salida tenga las mismas dimensiones
 - **Zero padding:** relleno de ceros

$$F : m \times n$$

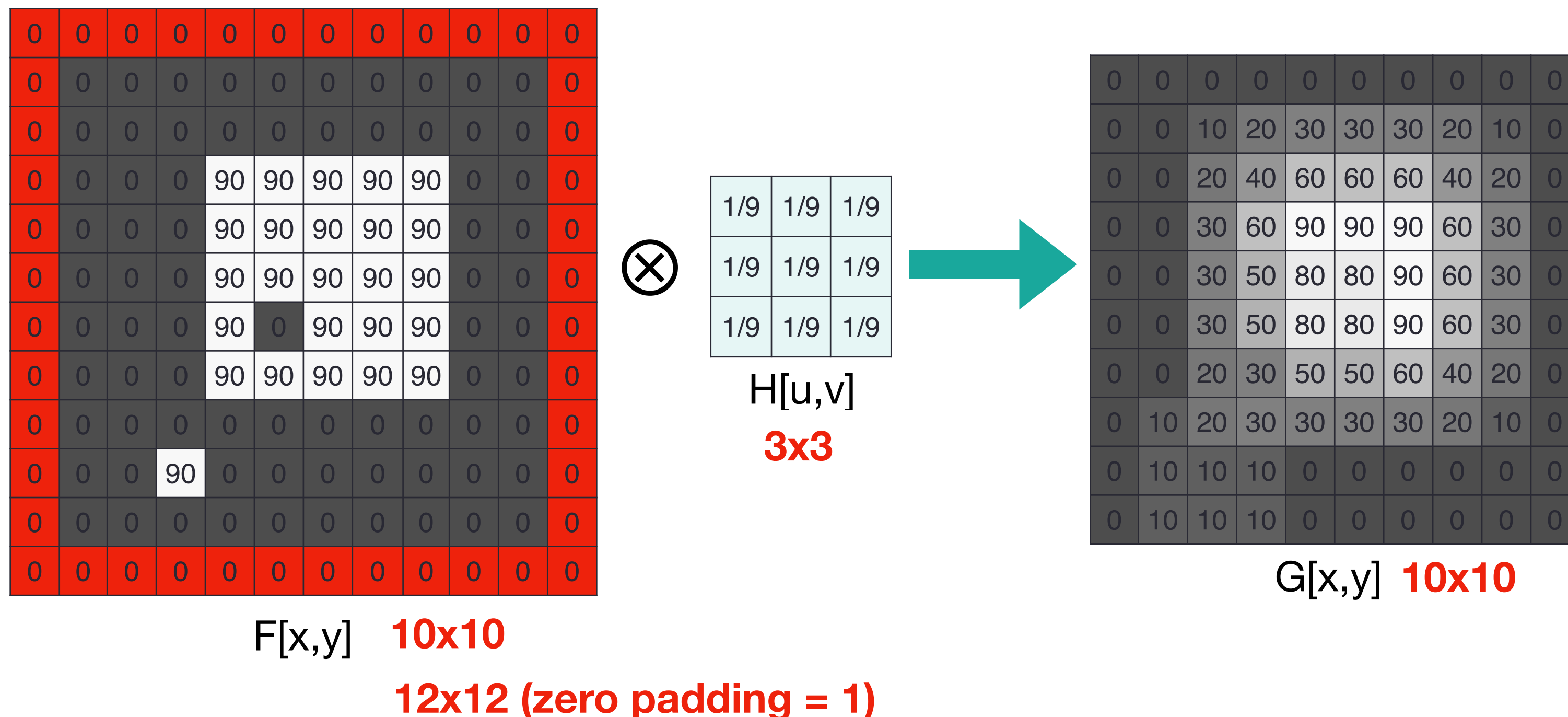
$$H : (2k + 1) \times (2k + 1)$$

$$Padding : k$$

$$G = H \otimes F$$

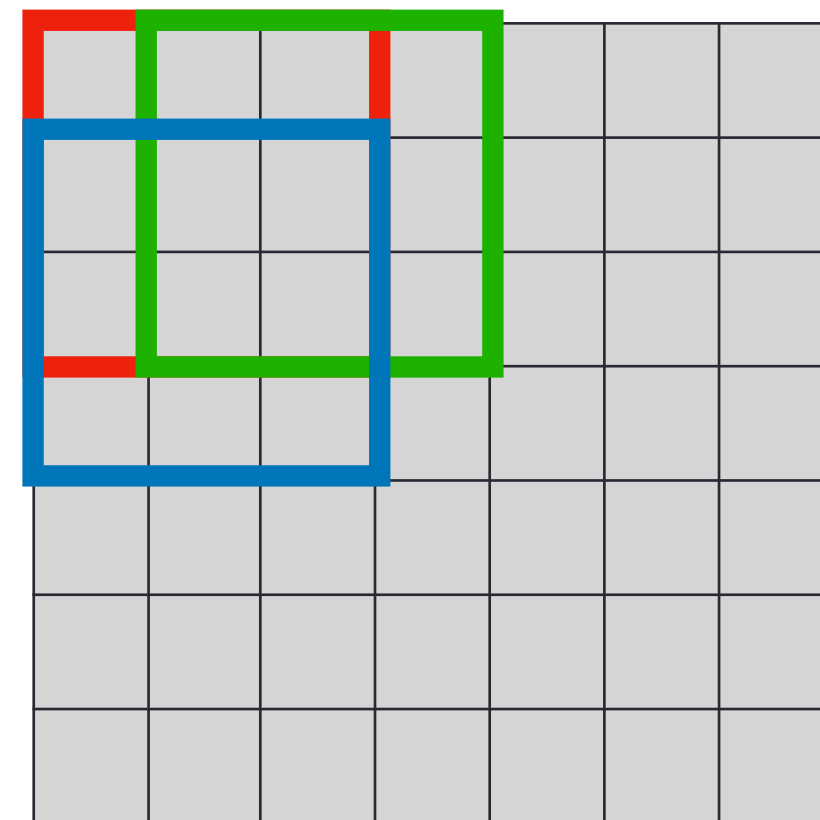
$$G : (m + 2k - (2k + 1) + 1) \times (n + 2k - (2k + 1) + 1)$$

$$G : m \times n$$



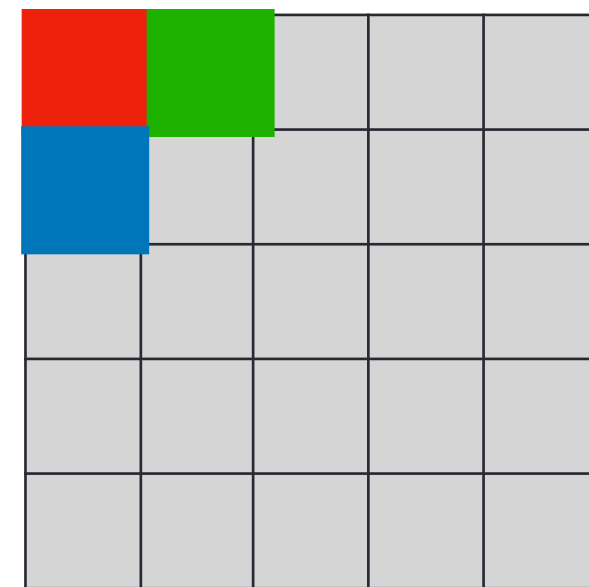
Convolución

- **Stride:** desplazamiento del filtro sobre la entrada

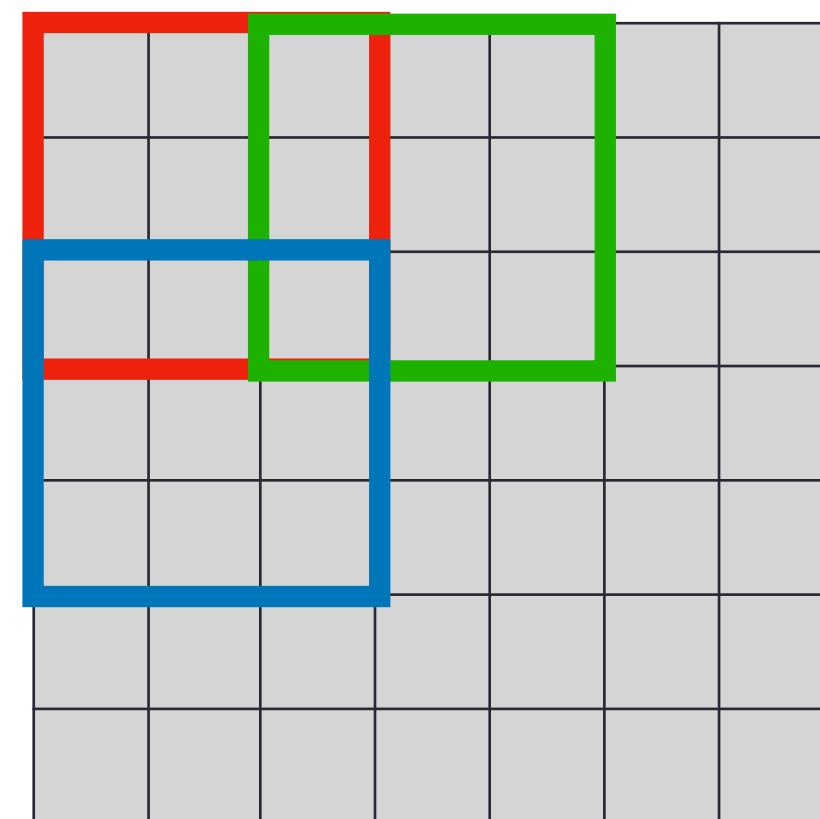


$F[x,y]$ 7x7

Filtro 3x3, stride = 1

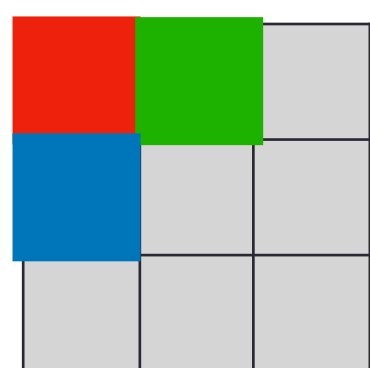


$G[x,y]$ 5x5



$F[x,y]$ 7x7

Filtro 3x3, stride = 2



$G[x,y]$ 3x3

$$F : m \times n$$

$$H : (2k + 1) \times (2k + 1)$$

$$\text{Stride} : s$$

$$G = H \otimes F$$

$$G : \left(\frac{(m - 2k - 1)}{s} + 1 \right) \times \left(\frac{(n - 2k - 1)}{s} + 1 \right)$$

Convolución

- **Padding:** píxeles de relleno que se añaden alrededor de la entrada
- **Stride:** desplazamiento del filtro sobre la entrada

$$F : m \times n$$

$$H : (2k + 1) \times (2k + 1)$$

$$\text{Stride} : s$$

$$\text{Padding} : k$$

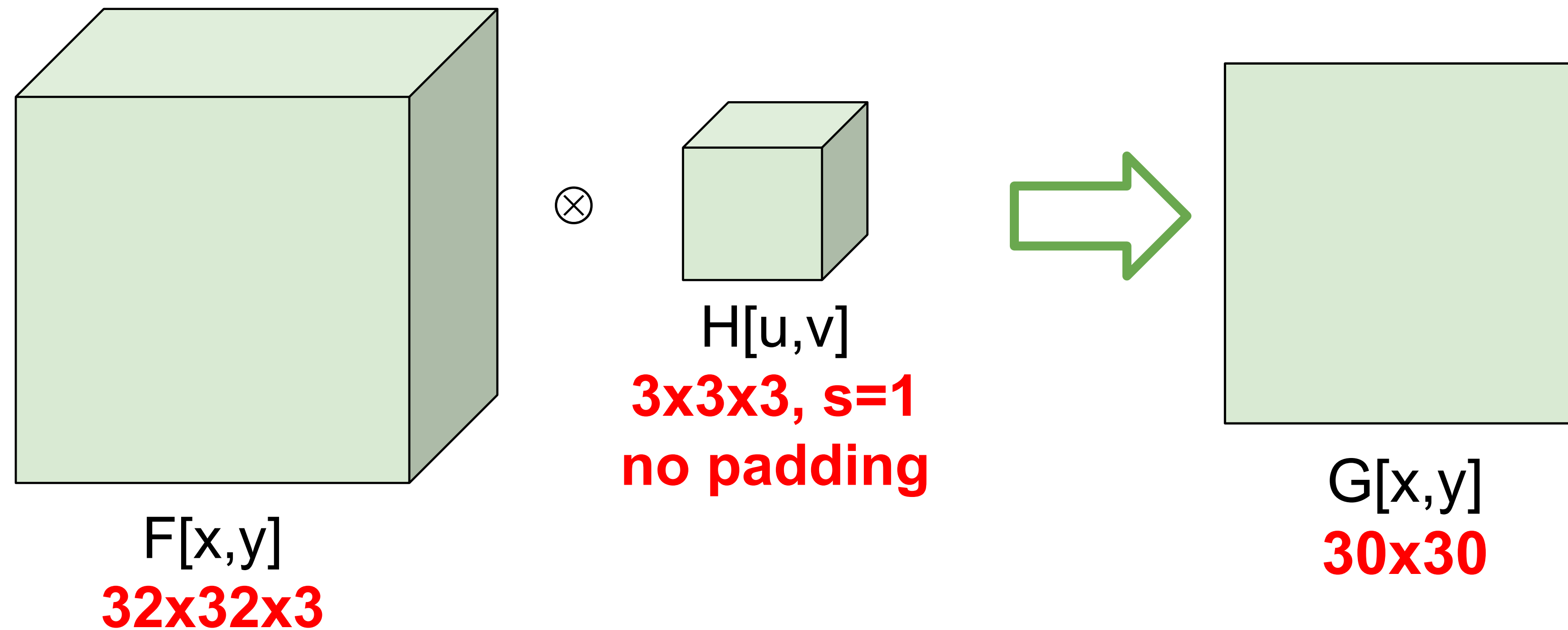
$$G = H \otimes F$$

$$G : \left(\frac{m + 2k - (2k + 1)}{s} + 1 \right) \times \left(\frac{n + 2k - (2k + 1)}{s} + 1 \right)$$

$$G : \left(\frac{m - 1}{s} + 1 \right) \times \left(\frac{n - 1}{s} + 1 \right)$$

Convolución

- **Convolución 2D:** operando con volúmenes 3D

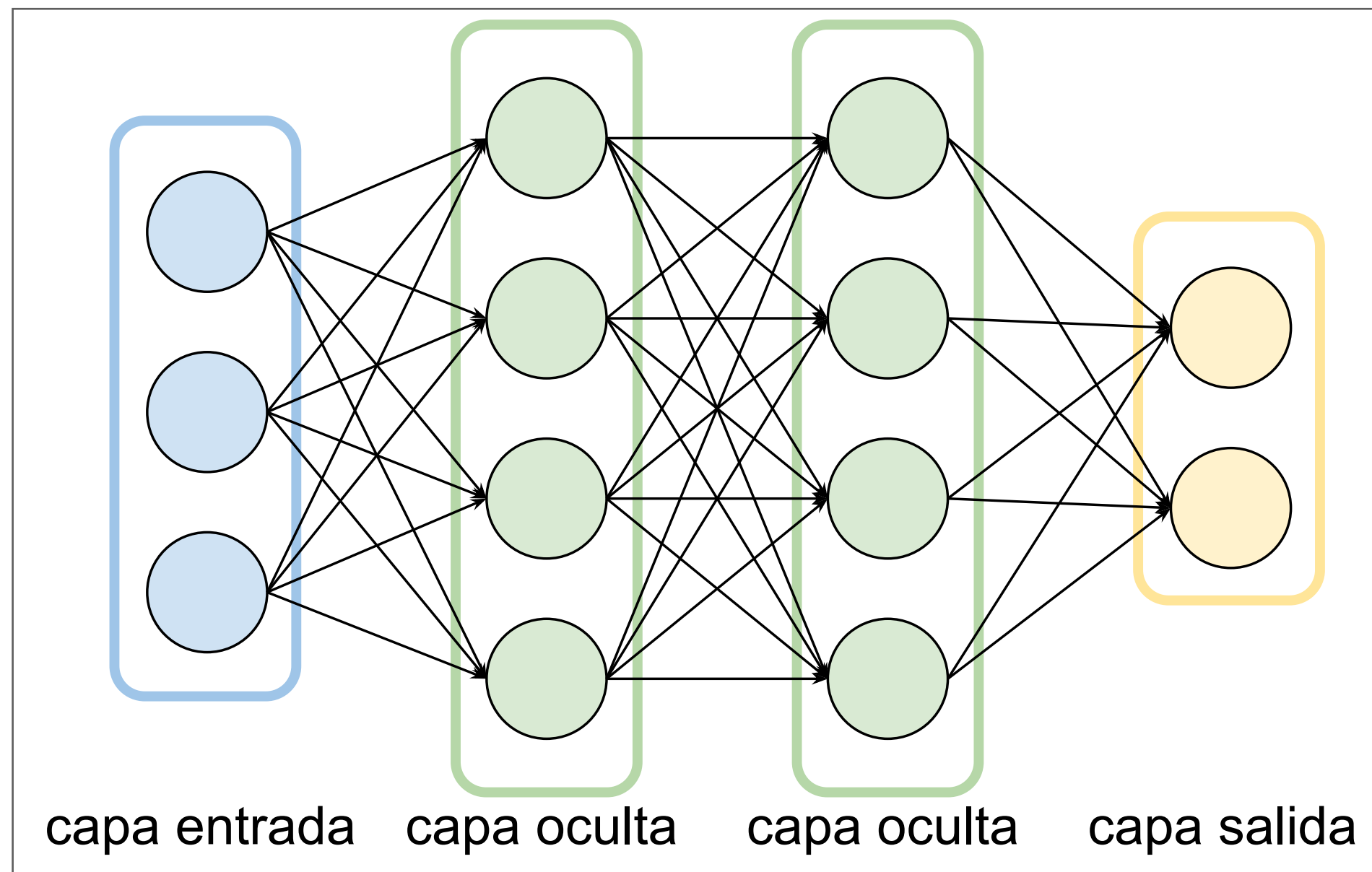


F y H: misma profundidad

ÍNDICE

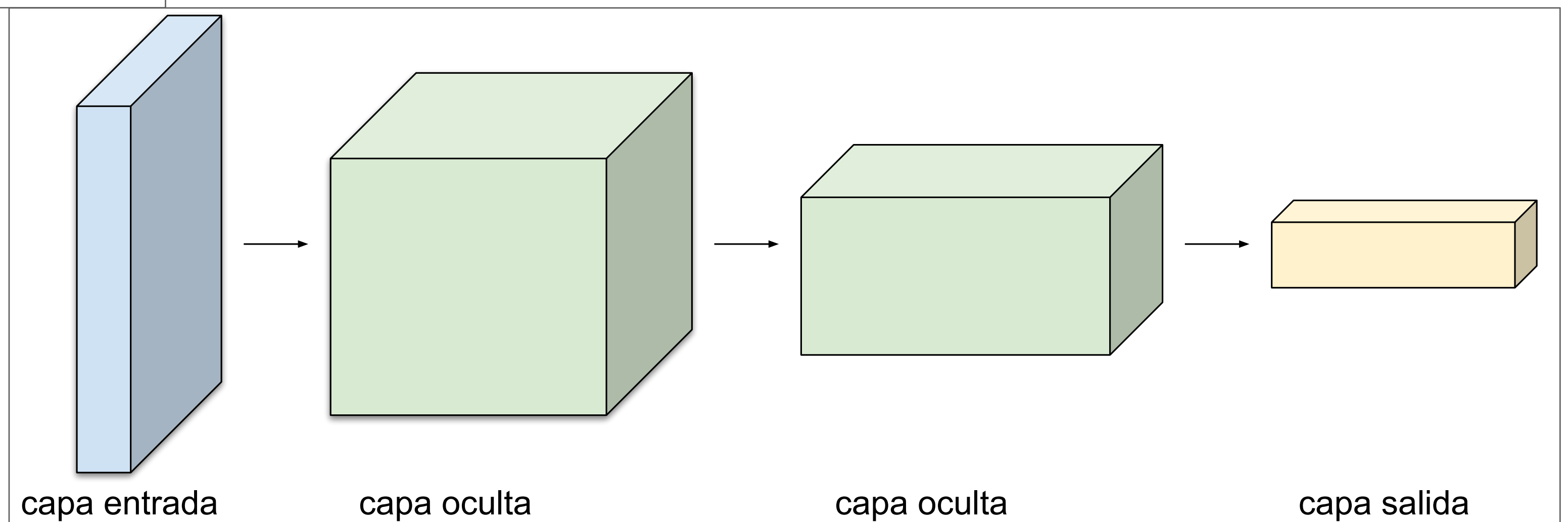
1. Introducción
2. Filtros y convoluciones
- 3. Redes convolucionales**
4. Capas de las redes convolucionales
5. Arquitecturas y entrenamiento
6. Casos de estudio

Introducción



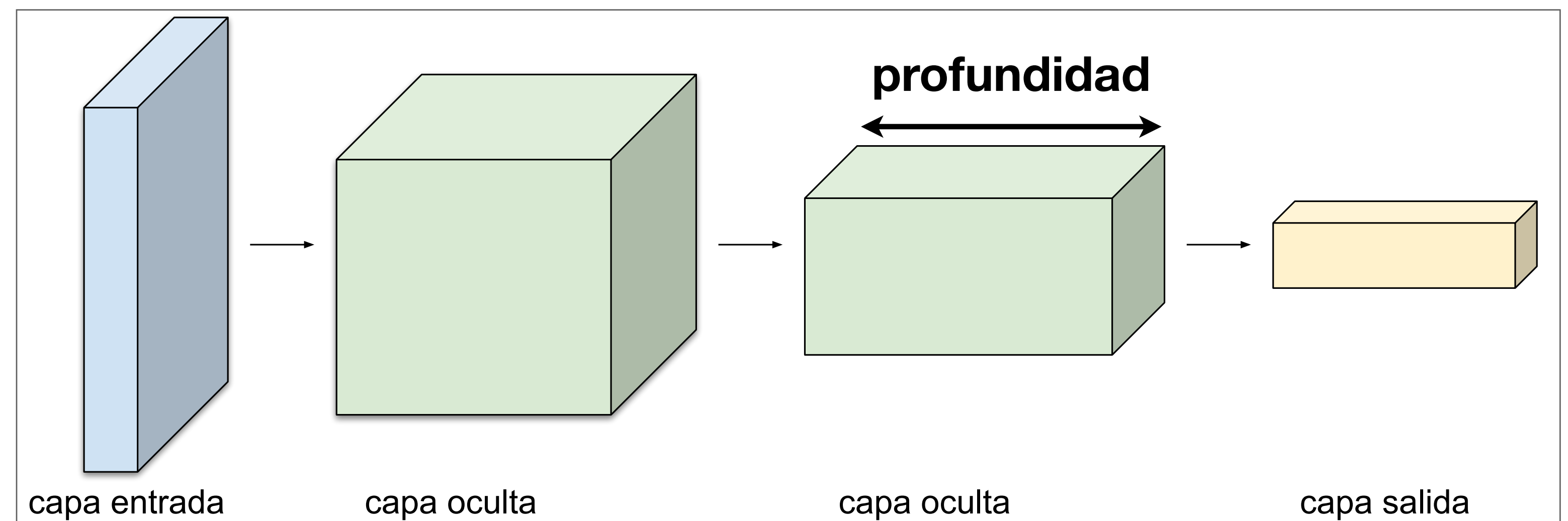
Red neuronal

Red convolucional



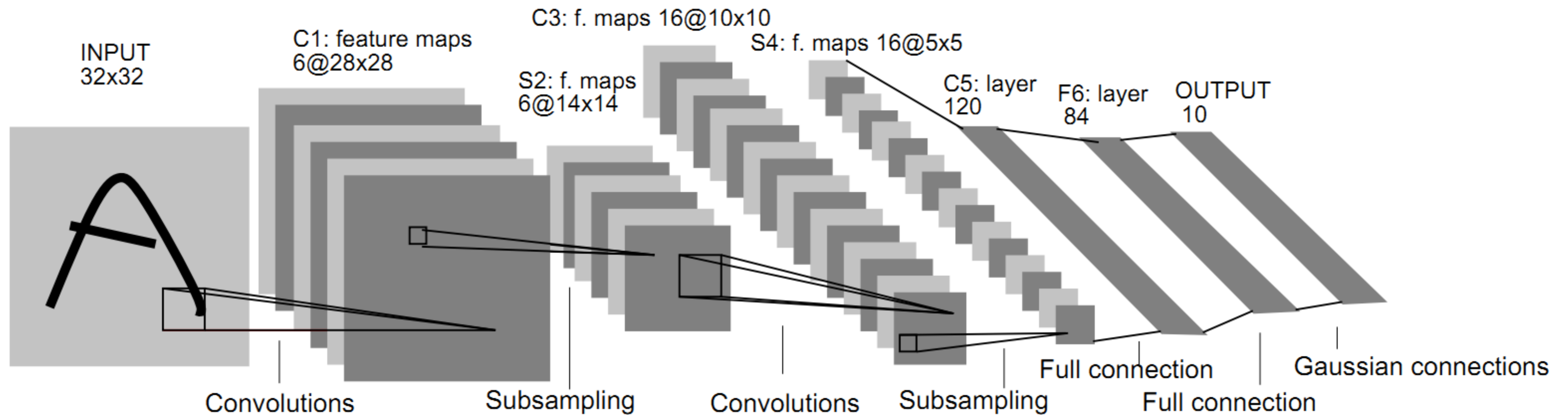
Redes convolucionales

- Redes de neuronas convolucionales
 - Tipo particular de redes neuronales profundas
 - Utilizadas en problemas de **visión artificial**
 - **CNNs, ConvNets**



Primeros intentos

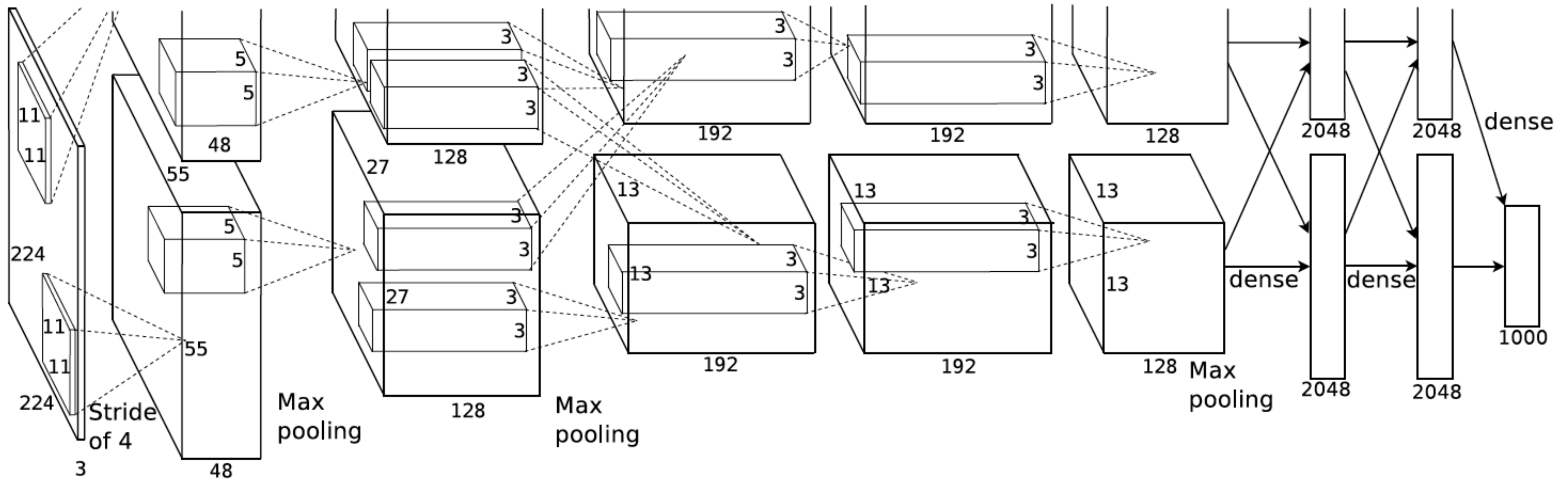
1998



LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278-2324.

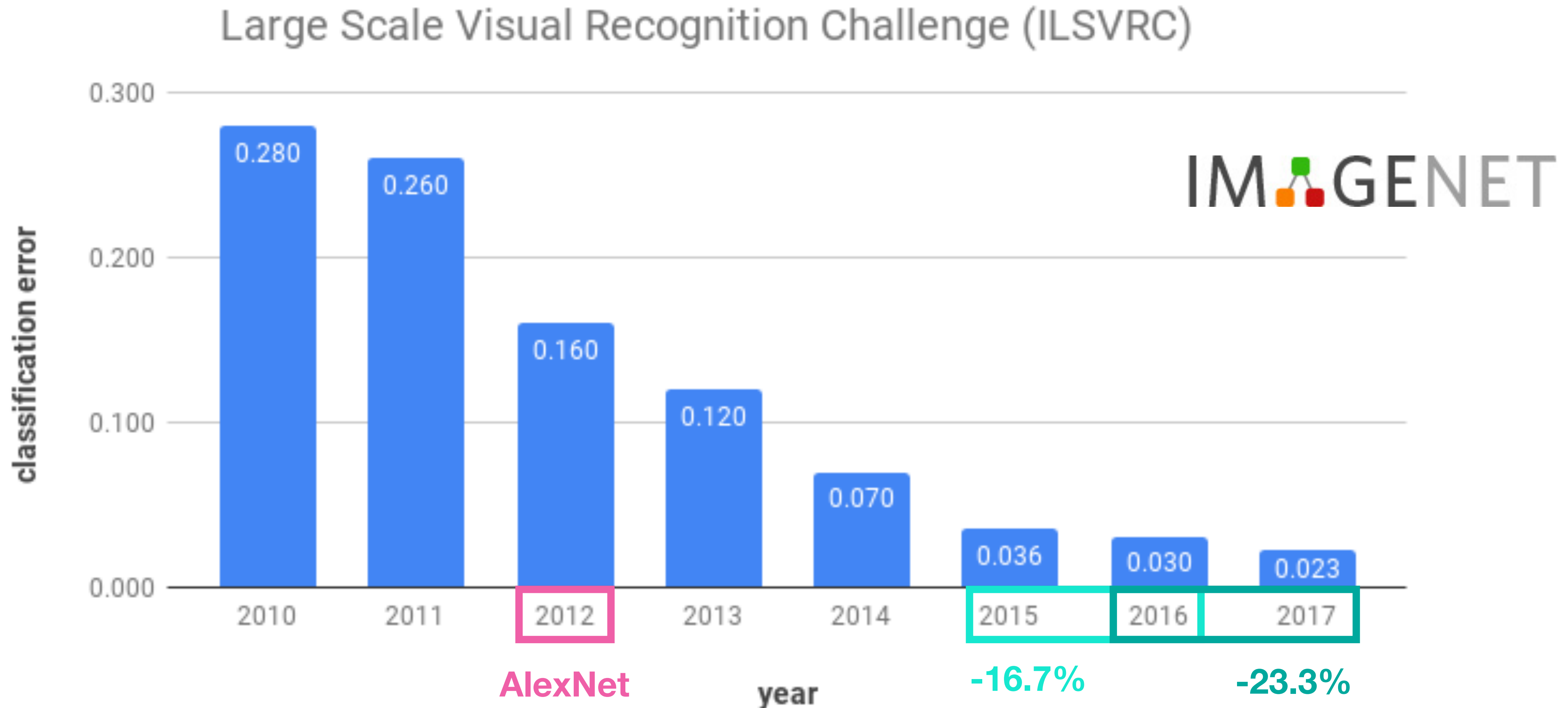
Primeros intentos

2012



Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

Evolución de las CNNs



ÍNDICE

1. Introducción
2. Filtros y convoluciones
3. Redes convolucionales
- 4. Capas de las redes convolucionales**
5. Arquitecturas y entrenamiento
6. Casos de estudio

Introducción

- La arquitectura de una CNN se define mediante una **secuencia de capas**
└─→ Cada capa transforma un volumen de activaciones en otro

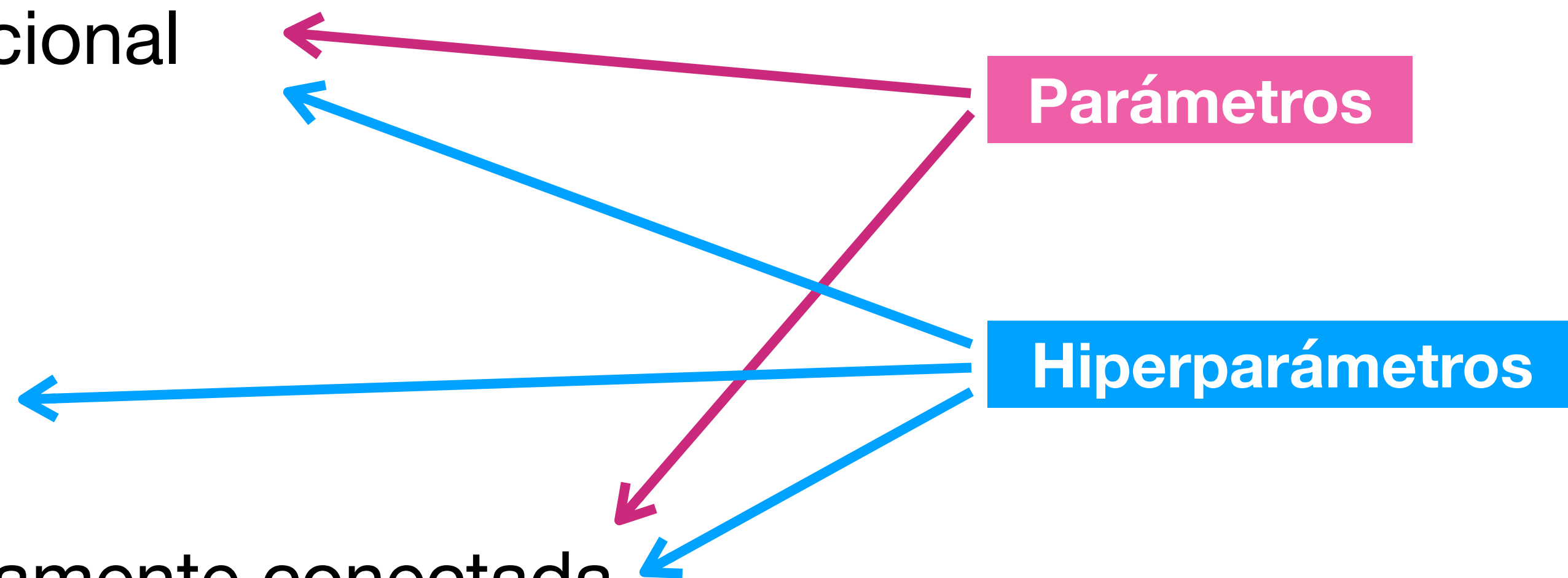
- **Principales capas** en una CNN:

- Capa convolucional

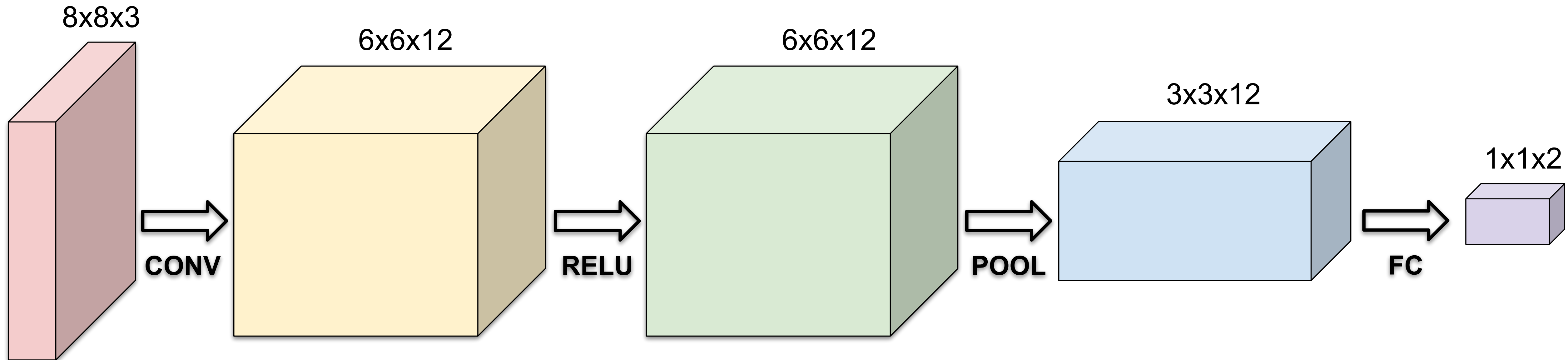
- Capa ReLU

- Capa pooling

- Capa completamente conectada



Ejemplo de una CNN



ARQUITECTURA: ENTRADA - CONVOLUCIONAL (CONV) - RELU - POOLING (POOL) - COMPLETAMENTE CONECTADA (FC)

Capa convolucional

- **Objetivo:** extraer características relevantes (bordes, colores, curvas, etc.)

- Filtros como identificadores de características

$$P = (F-1)/2, S = 1$$
$$W2 = W1, H2 = H1$$

- **Definición de la capa convolucional:**

- Hiperparámetros: número de filtros N , dimensión espacial F , padding P , stride S

- Dimensiones del volumen de entrada: $W1 \times H1 \times D$

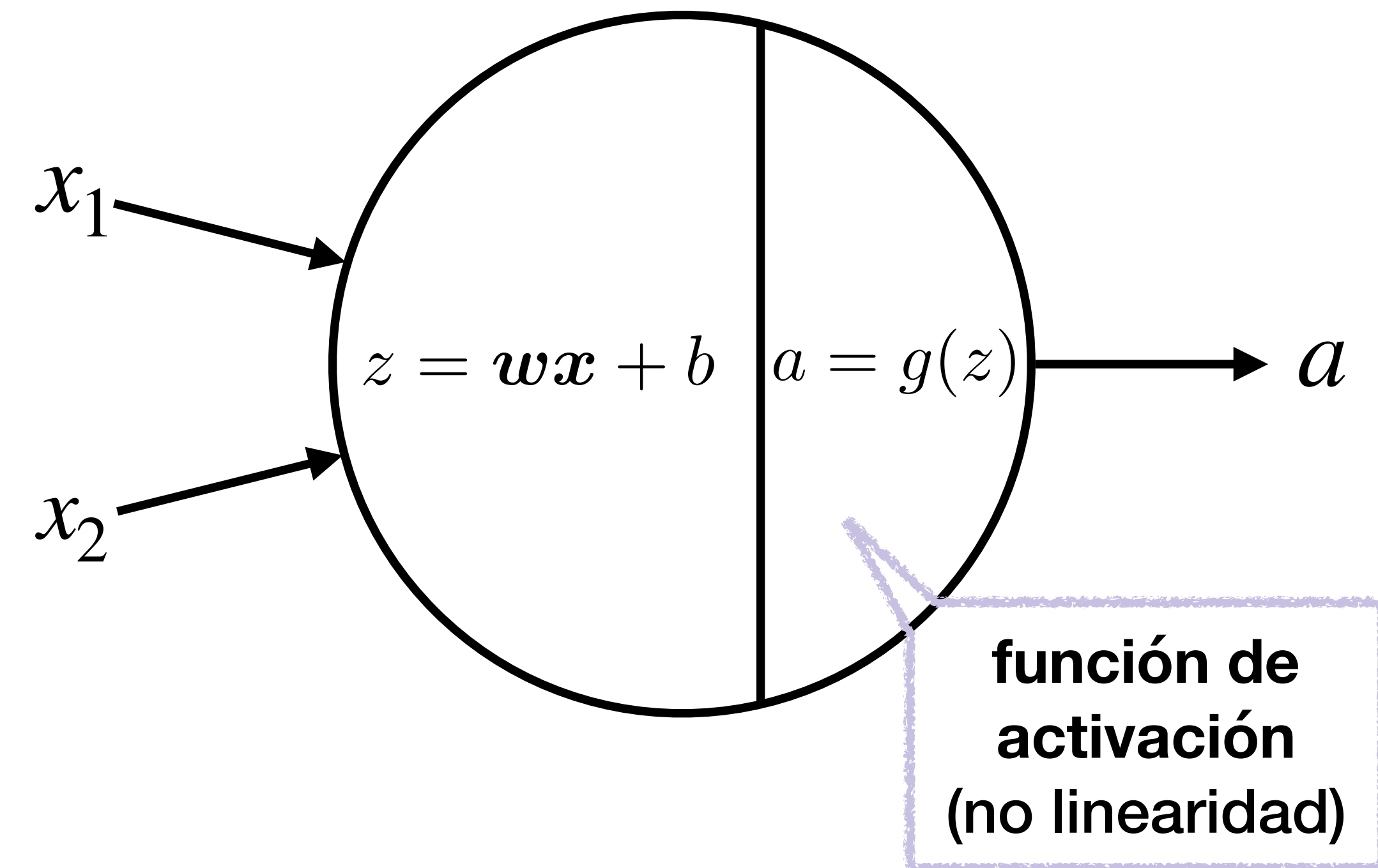
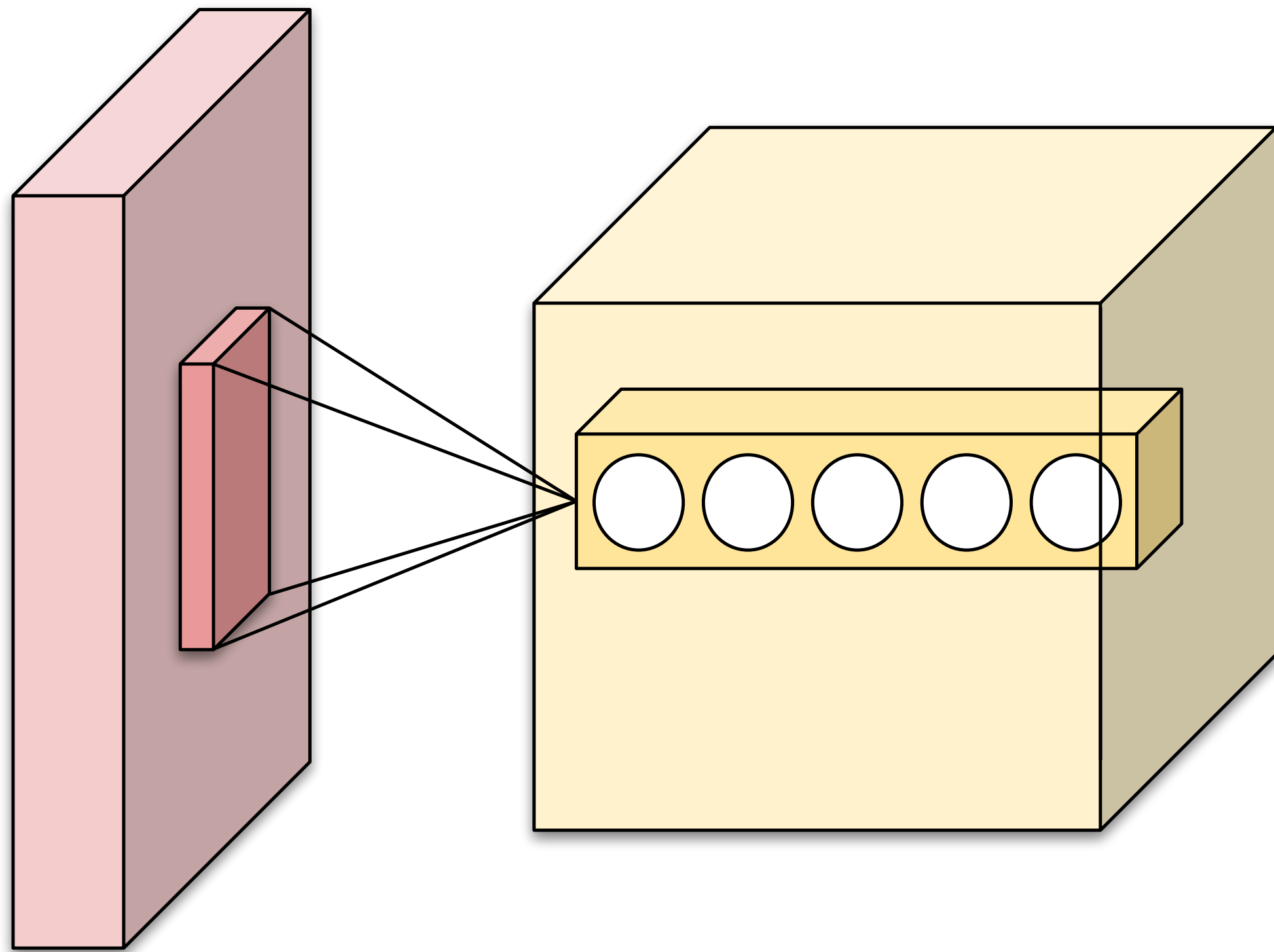
- Dimensiones del volumen de salida: $W2 \times H2 \times N$

$$W2 = (W1 - F + 2P)/S + 1$$

$$H2 = (H1 - F + 2P)/S + 1$$

- Parámetros: pesos de los filtros convolucionales

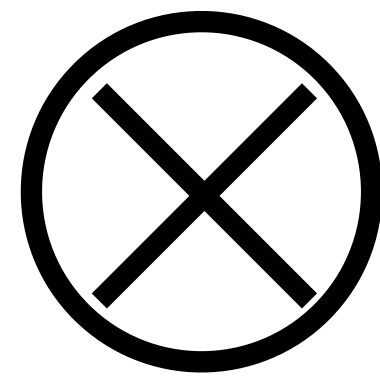
Capa convolucional



Capa convolucional

| | | | | | | | |
|----|----|----|----|---|----|----|----|
| -1 | -1 | -1 | -1 | 1 | -1 | -1 | -1 |
| -1 | -1 | -1 | -1 | 1 | -1 | -1 | -1 |
| -1 | -1 | -1 | -1 | 1 | -1 | -1 | -1 |
| -1 | -1 | -1 | -1 | 1 | -1 | -1 | -1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| -1 | -1 | -1 | -1 | 1 | -1 | -1 | -1 |
| -1 | -1 | -1 | -1 | 1 | -1 | -1 | -1 |
| -1 | -1 | -1 | -1 | 1 | -1 | -1 | -1 |

8x8



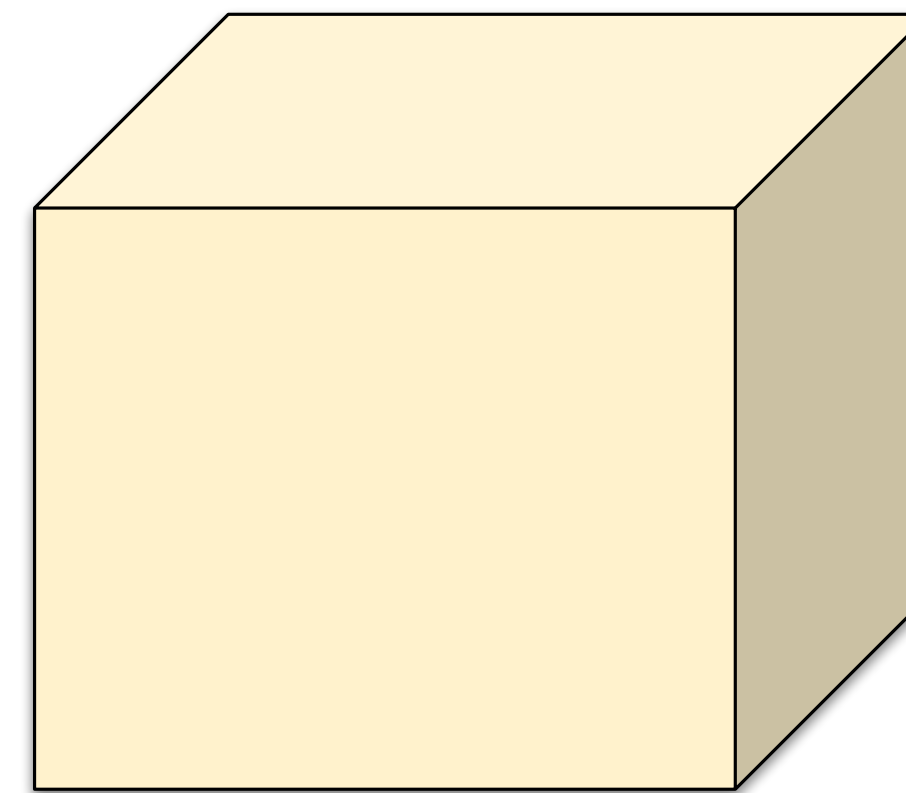
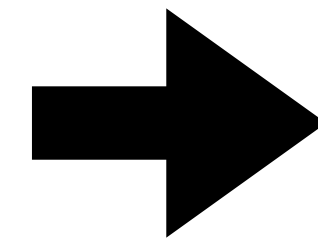
| | | |
|----|----|----|
| -1 | -1 | -1 |
| 1 | 1 | 1 |
| -1 | -1 | -1 |

| | | |
|----|---|----|
| -1 | 1 | -1 |
| -1 | 1 | -1 |
| -1 | 1 | -1 |

⋮

| | | |
|----|----|----|
| 1 | -1 | -1 |
| -1 | 1 | -1 |
| -1 | -1 | 1 |

$N = 12, F = 3$
 $P = 0, S = 1$



6x6x12

| | | | | | |
|----|----|------|------|------|----|
| 0 | 0 | -0.3 | -0.3 | -0.3 | 0 |
| 0 | 0 | -0.3 | -0.3 | -0.3 | 0 |
| -1 | -1 | -1 | -1 | -1 | -1 |
| 1 | 1 | 0.3 | 0.3 | 0.3 | 1 |
| -1 | -1 | -1 | -1 | -1 | -1 |
| 0 | 0 | -0.3 | -0.3 | -0.3 | 0 |

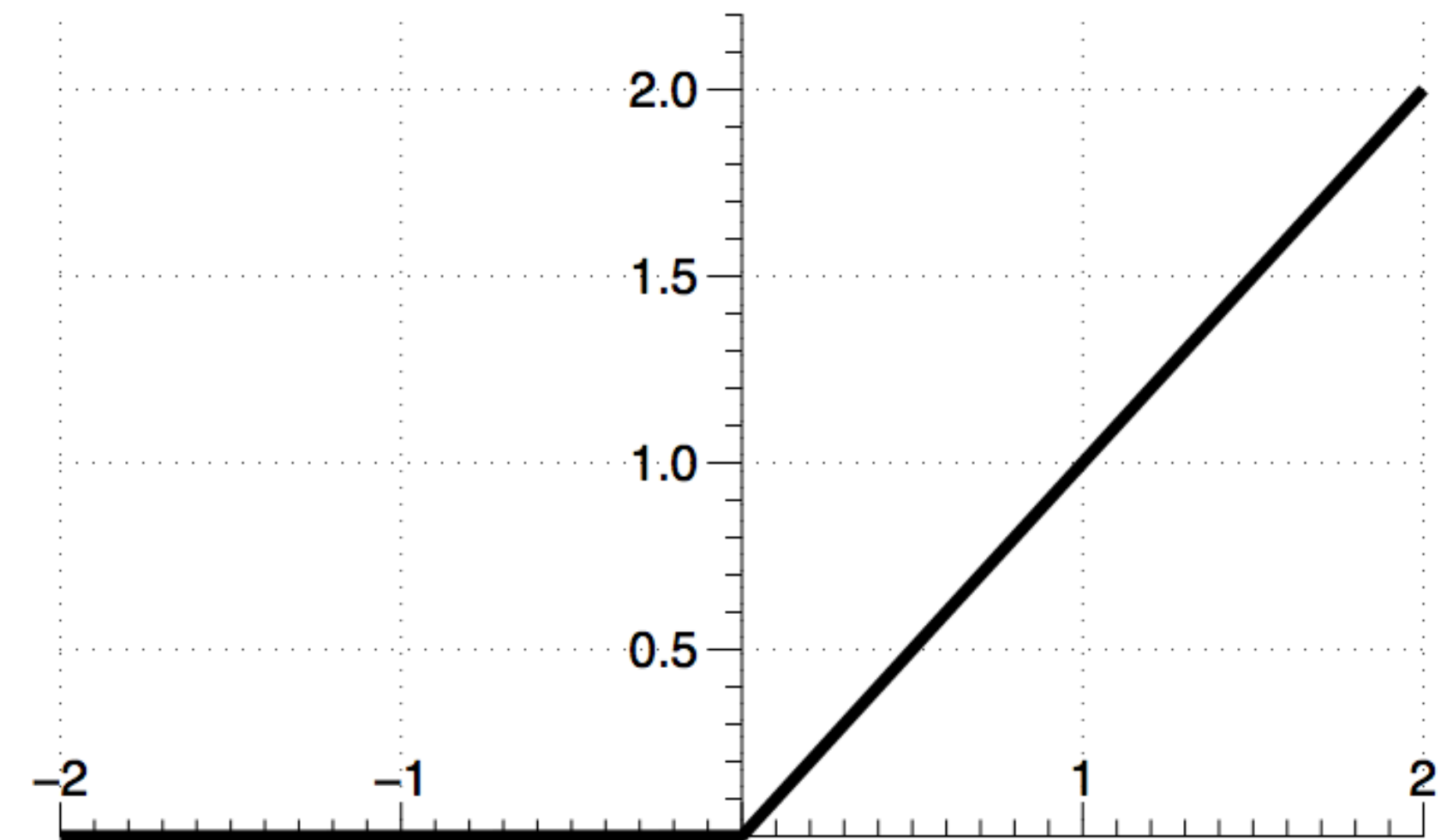
| | | | | | |
|------|------|----|-----|----|------|
| 0 | 0 | -1 | 1 | -1 | 0 |
| 0 | 0 | -1 | 1 | -1 | 0 |
| -0.3 | -0.3 | -1 | 0.3 | -1 | -0.3 |
| -0.3 | -0.3 | -1 | 0.3 | -1 | -0.3 |
| -0.3 | -0.3 | -1 | 0.3 | -1 | -0.3 |
| 0 | 0 | -1 | 1 | -1 | 0 |

⋮

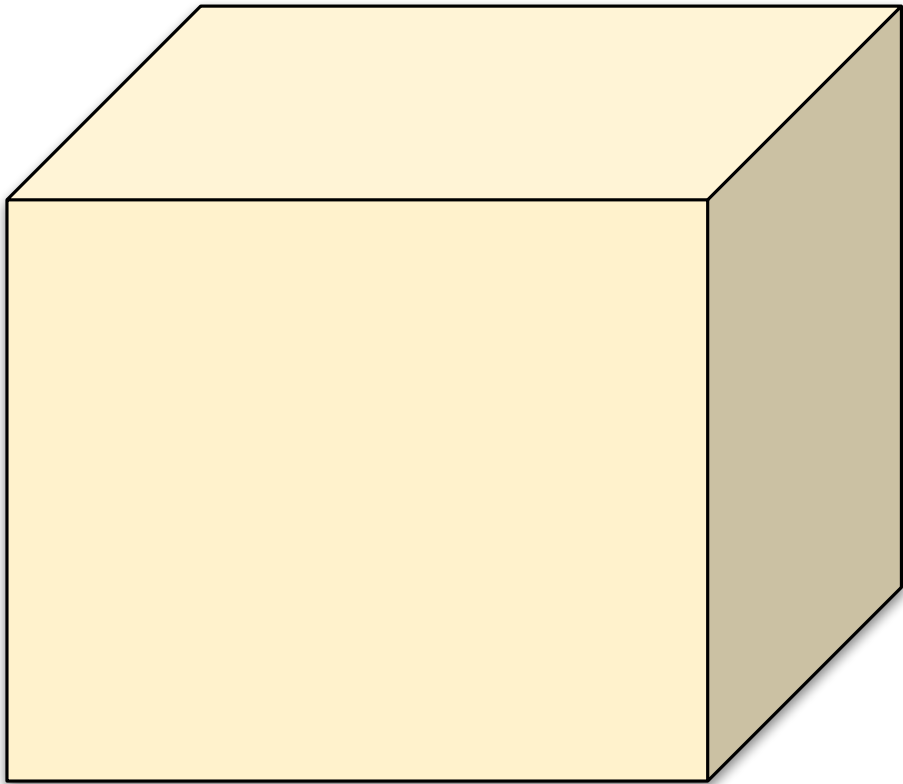
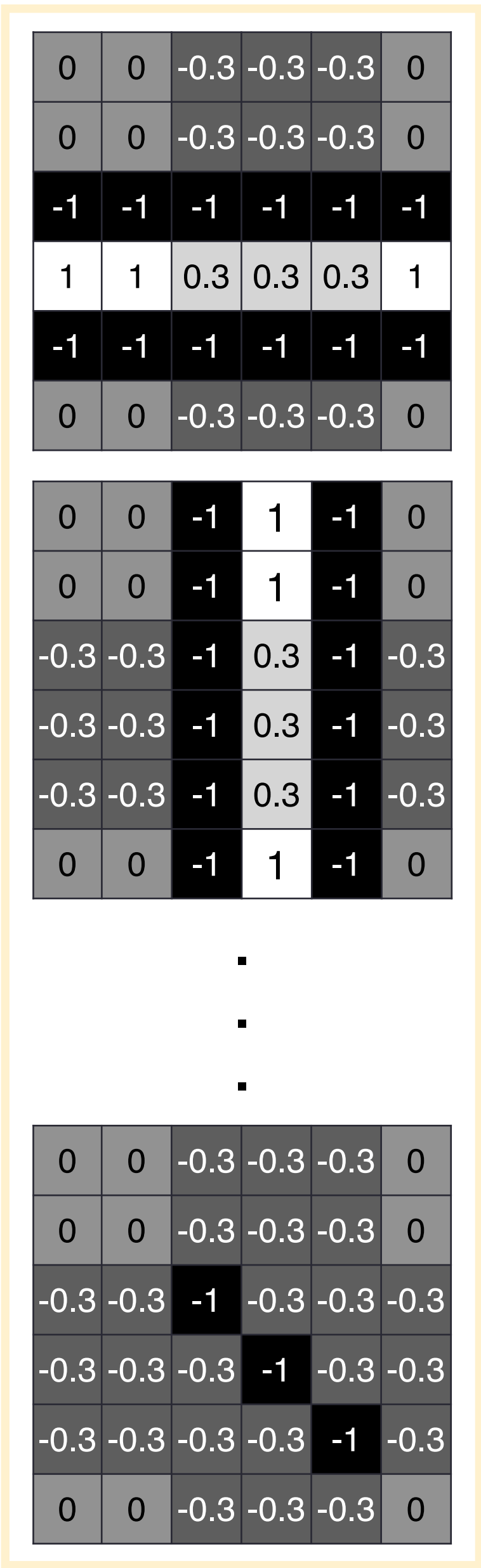
| | | | | | |
|------|------|------|------|------|------|
| 0 | 0 | -0.3 | -0.3 | -0.3 | 0 |
| 0 | 0 | -0.3 | -0.3 | -0.3 | 0 |
| -0.3 | -0.3 | -1 | -0.3 | -0.3 | -0.3 |
| -0.3 | -0.3 | -0.3 | -1 | -0.3 | -0.3 |
| -0.3 | -0.3 | -0.3 | -0.3 | -1 | -0.3 |
| 0 | 0 | -0.3 | -0.3 | -0.3 | 0 |

Capa ReLU

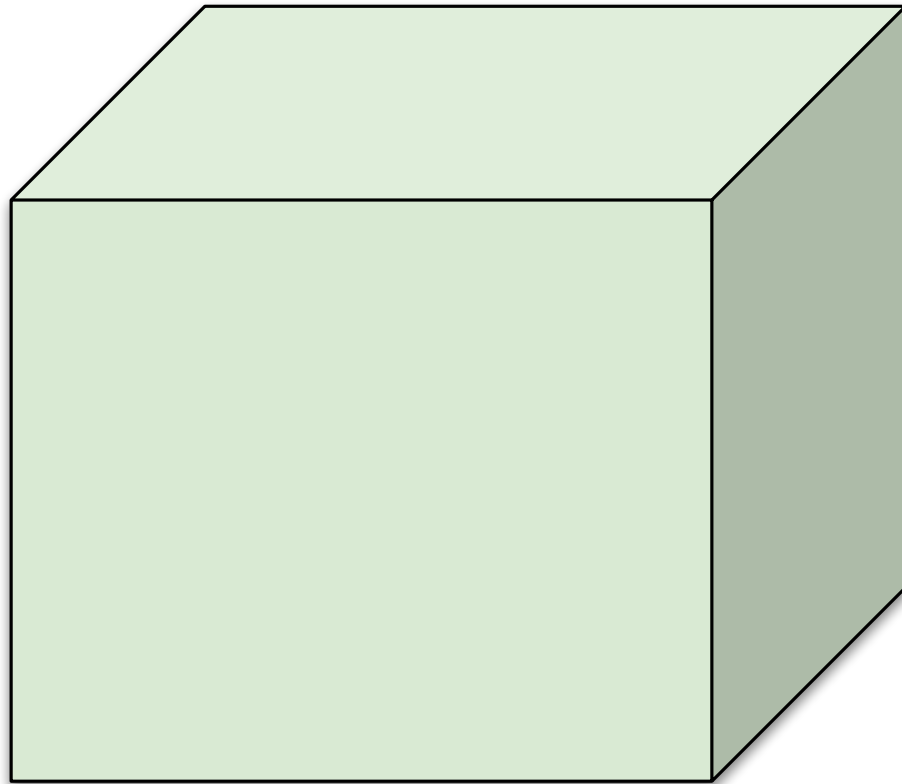
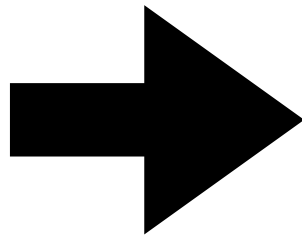
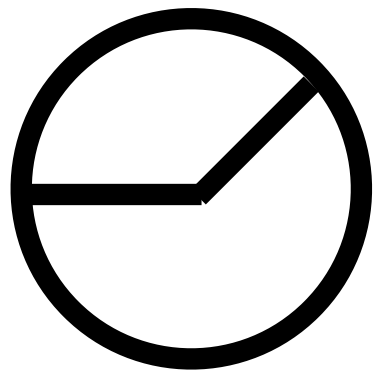
- **Objetivo:** lograr que la red sea capaz de resolver un problema no-lineal
 - Función de activación no-lineal: Rectified Linear Unit (ReLU)
- **Definición de la capa ReLU:**
 - No tiene hiperparámetros
 - No tiene parámetros
 - Función no-lineal: $a = \max(0, z)$



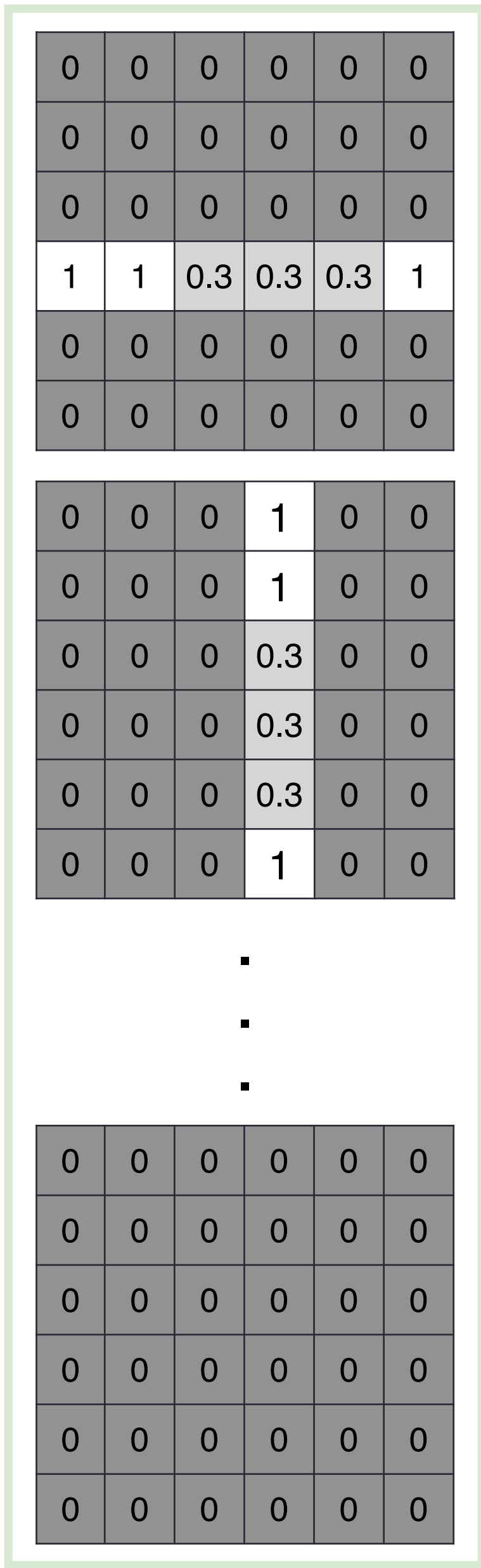
Capa ReLU



6x6x12



6x6x12



Capa pooling

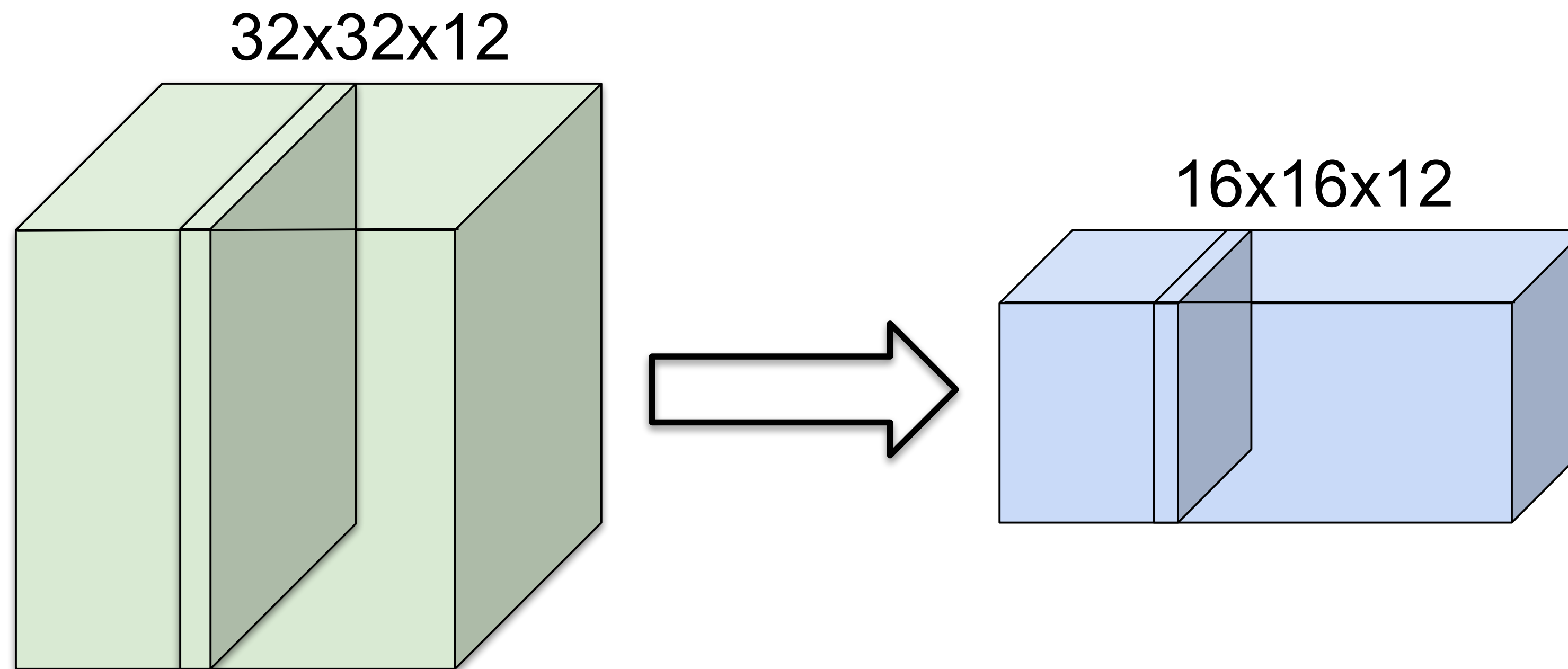
- **Objetivo:** reducir progresivamente las dimensiones espaciales
 - Menor número de parámetros, mejor control del sobreajuste
- **Definición de la capa pooling:**
 - Hiperparámetros: dimensión espacial F , stride S
 - Dimensiones del volumen de entrada: $W1 \times H1 \times D$
 - Dimensiones del volumen de salida: $W2 \times H2 \times D$
 - No tiene parámetros

$$W2 = (W1 - F) / S + 1$$

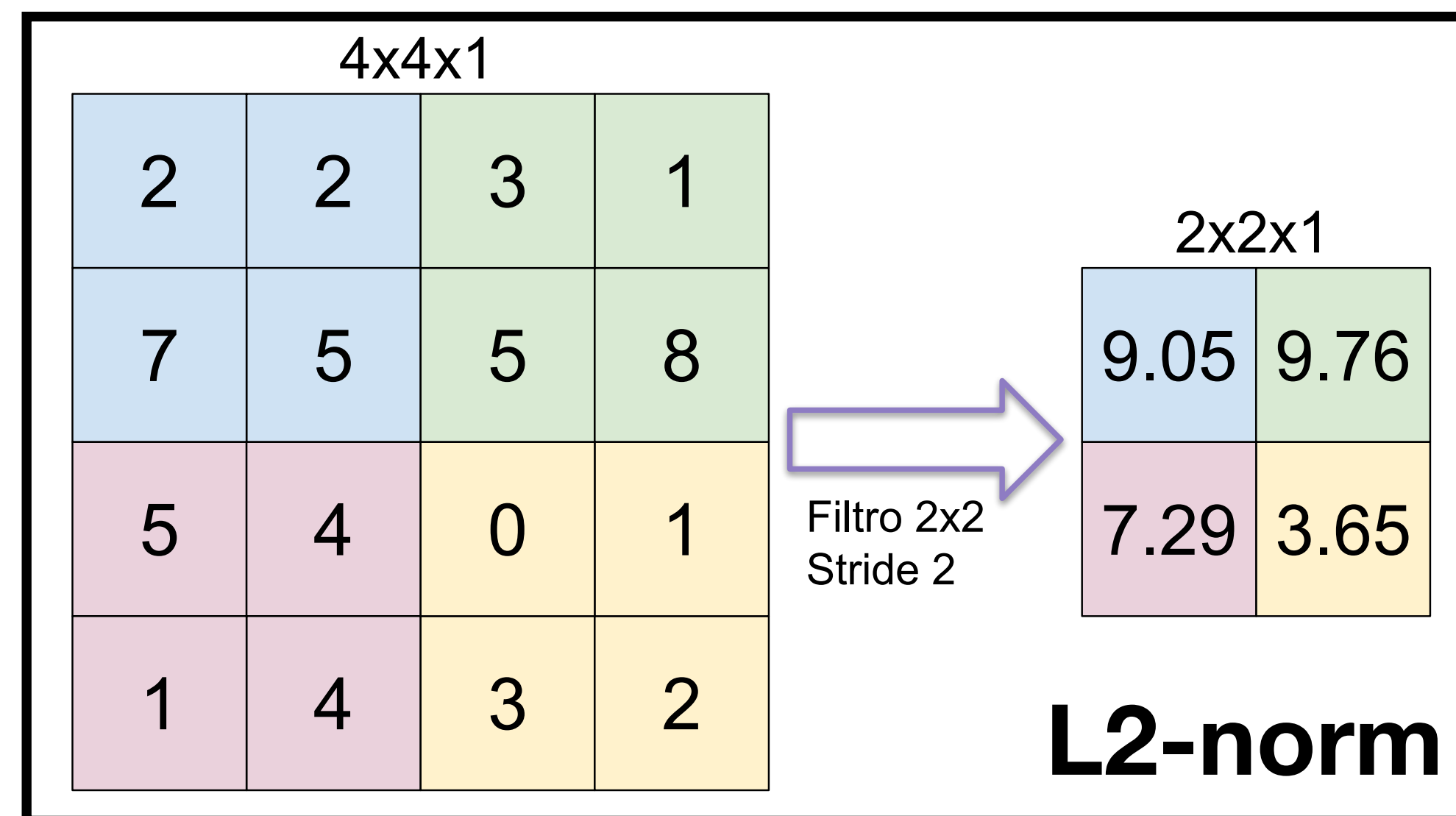
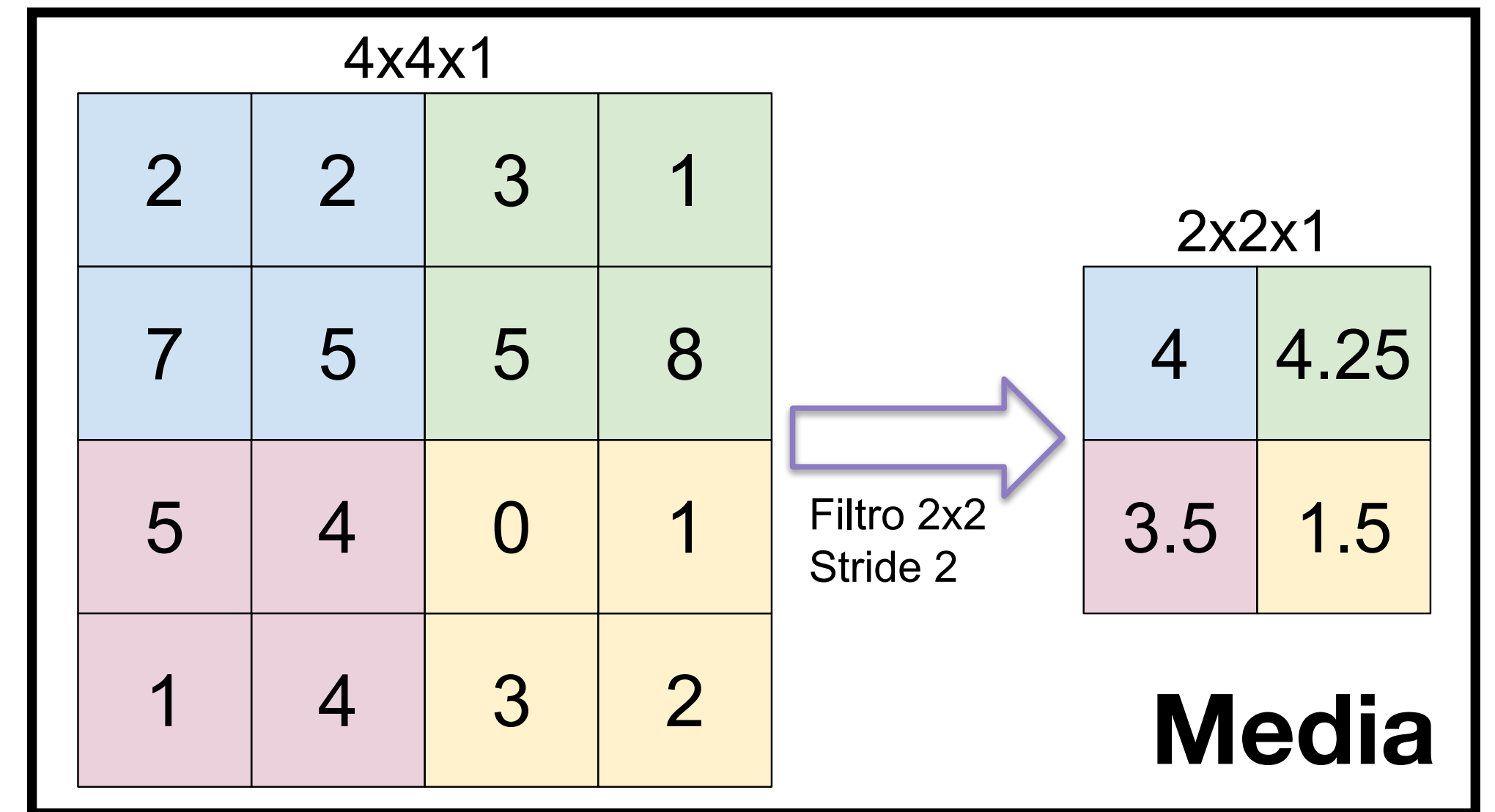
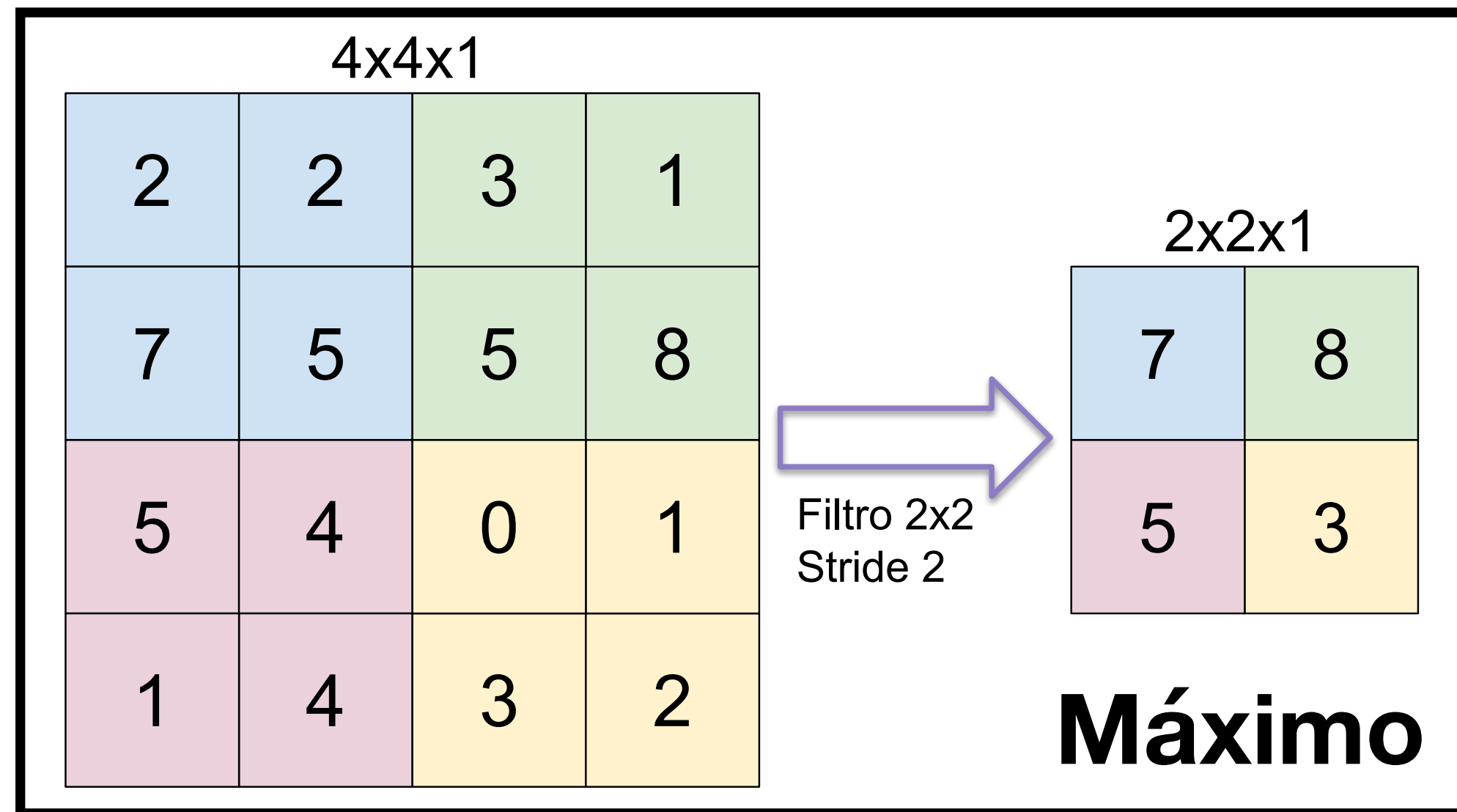
$$H2 = (H1 - F) / S + 1$$

Capa pooling

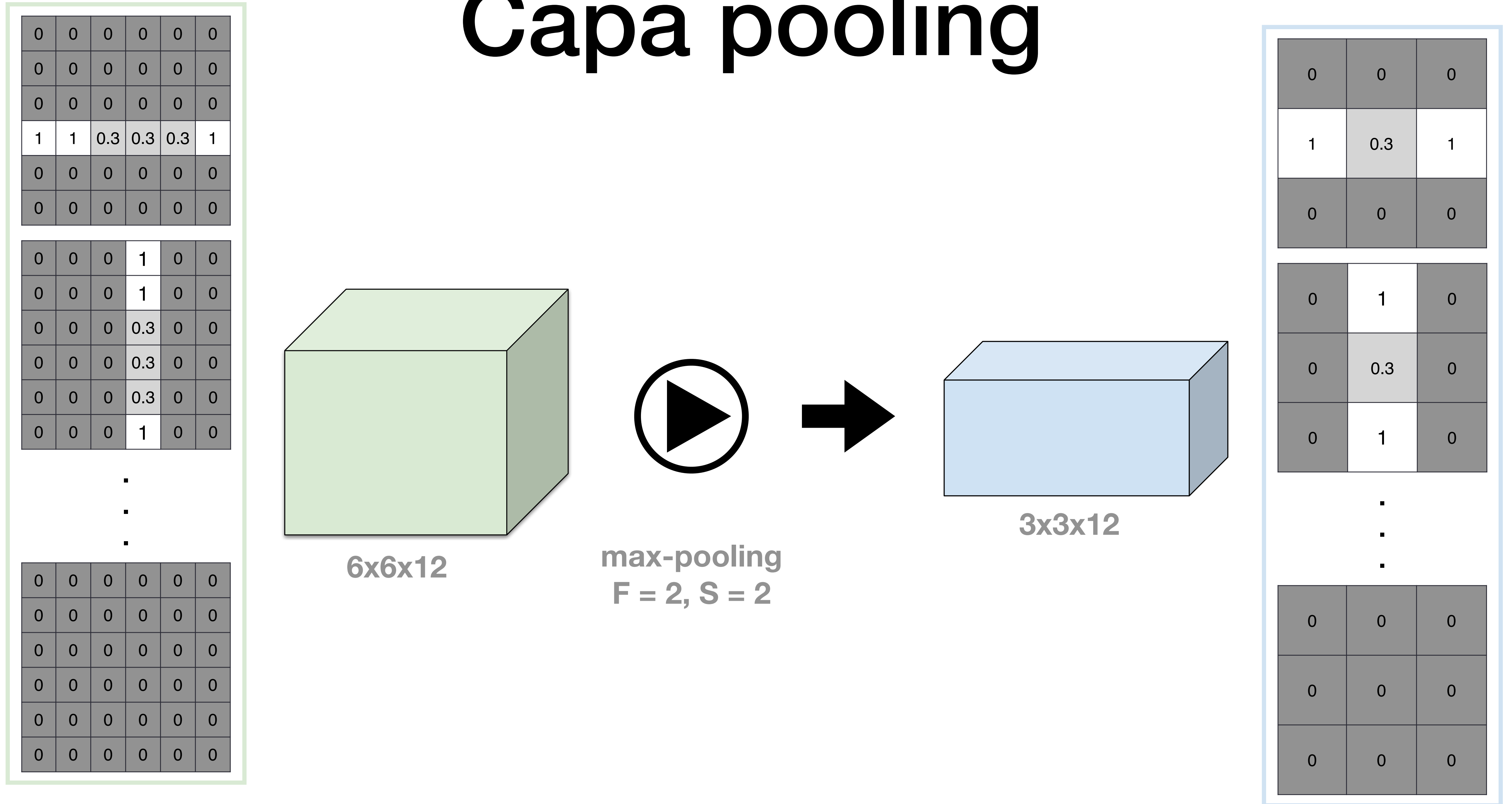
- Capa pooling más común: filtros de tamaño $F = 2$, stride $S = 2$
 - Descarta 75% de las activaciones \rightarrow reduce a la mitad ancho y alto



Capa pooling



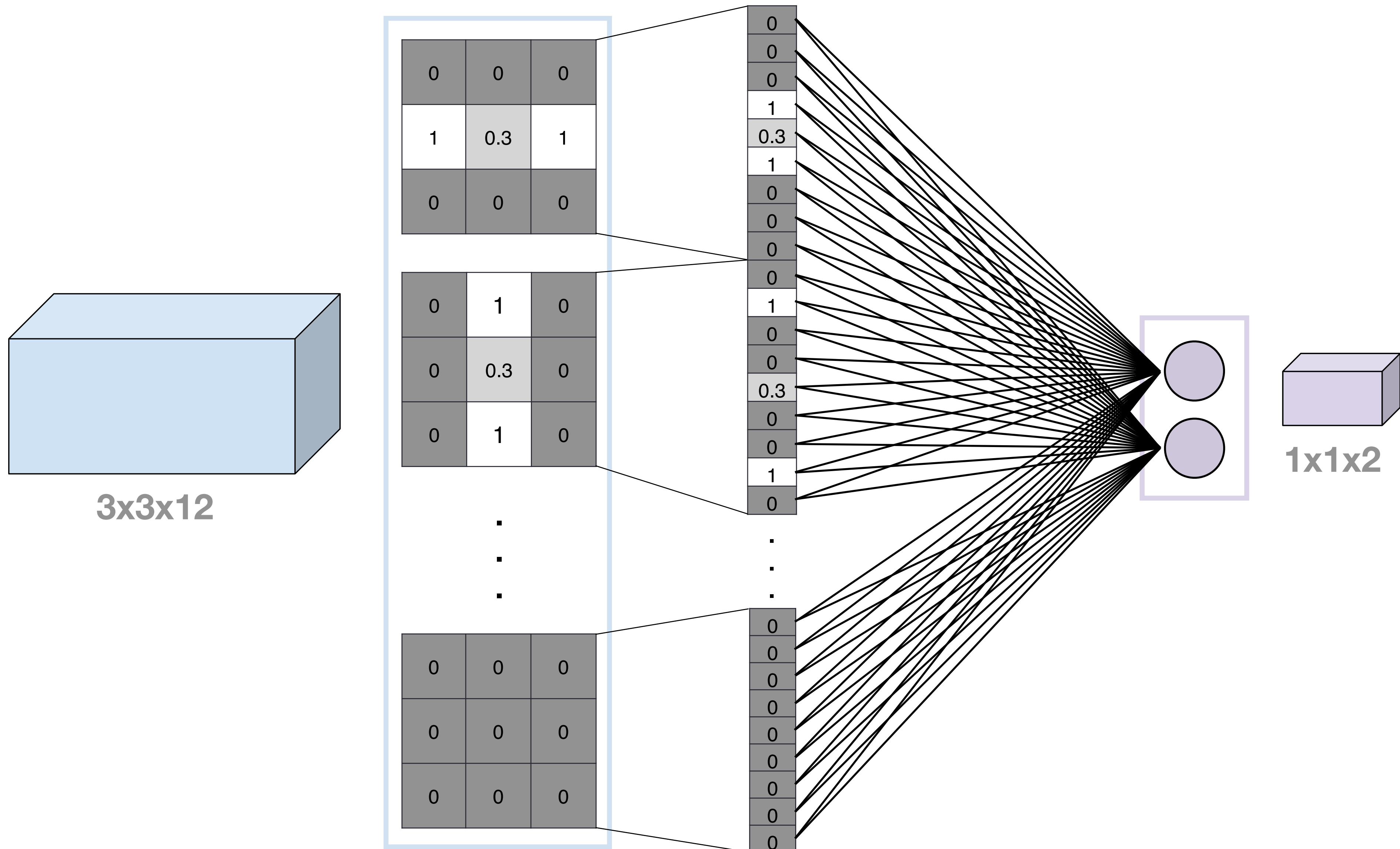
Capa pooling



Capa completamente conectada

- **Objetivo:** determinar qué características correlacionan más con cada clase
 - Conexiones completas con todas las neuronas de la capa anterior
- **Definición de la capa completamente conectada:**
 - Hiperparámetro: número de neuronas (K)
 - Dimensiones del volumen de entrada: $W1 \times H1 \times D1$
 - Dimensiones del volumen de salida: $1 \times 1 \times K$
 - Parámetros: pesos de las conexiones entre neuronas

Capa completamente conectada



ÍNDICE

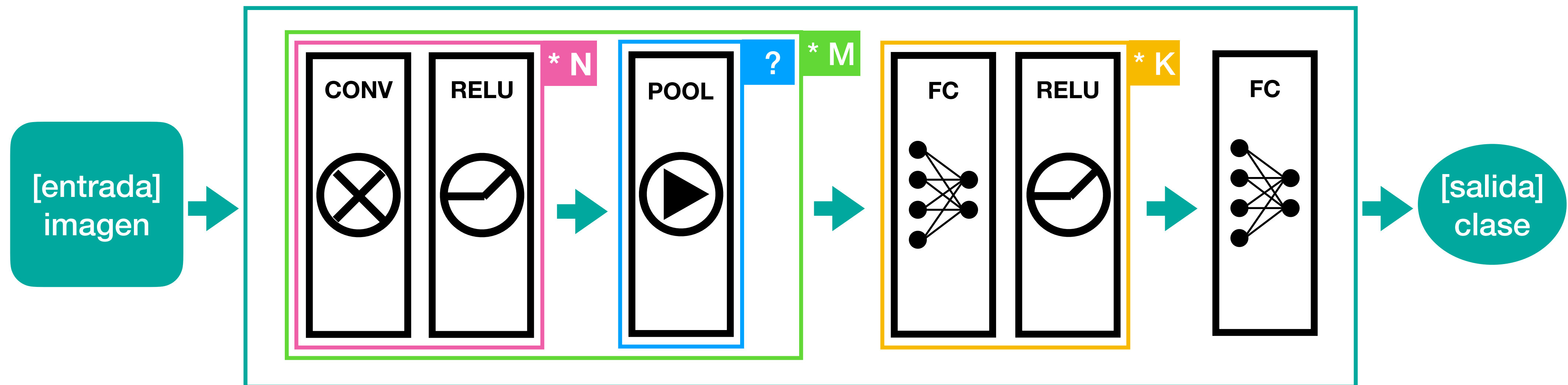
1. Introducción
2. Filtros y convoluciones
3. Redes convolucionales
4. Capas de las redes convolucionales
5. **Arquitecturas y entrenamiento**
6. Casos de estudio

Arquitectura

- La arquitectura de una CNN se define mediante una **secuencia de capas**: capa convolucional (CONV), capa ReLU (ReLU), capa pooling (POOL) y capa completamente conectada (FC).
- ¿Cómo definir una secuencia de capas “apropiada”?
 - ¿Qué capas?
 - ¿Número de capas?
 - ¿Tamaño de capas?
 -

Definición de capas

- Patrón para definir una CNN:



donde * indica repetición y ? indica opcional

- Recomendaciones: $0 \leq N \leq 3$, $M \geq 0$, $0 \leq K < 3$

Definición de capas

- Arquitecturas básicas:
 - Clasificador lineal: $N = M = K = 0$
 - Sin capas pooling: $N = 1, M = K = 0$
 - Una capa convolucional antes de cada capa pooling: $N = K = 1, M = 2$
 - Dos capas convolucionales antes de cada capa pooling: $N = K = 2, M = 3$
- Agrupar varias capas CONV+RELU con filtros pequeños
 - Más expresividad
 - Menos parámetros

Redes grandes
y profundas



Hiperparámetros de capas

- Capa de entrada:
 - Tamaño divisible por 2 (varias veces)
- Capas convolucionales:
 - Filtros pequeños ($F = 3$, $F = 5$)
 - Padding y stride: $P = (F-1)/2$, $S = 1$
- Capa pooling:
 - Max-pooling
 - Reducción del 75% ($F = 2$, $S = 2$)

En la práctica, considera modelos que funcionan bien con ImageNet

Aprendizaje

- **Objetivo:** obtener los parámetros del modelo que lo hagan óptimo para resolver su tarea predictiva.
- Modelo obtenido:
 - Parámetros: pesos de las filtros (capas CONV), y pesos de las conexiones entre neuronas (capas FC).
 - Hiperparámetros de las capas: número y tamaño de filtros, padding, stride (capas CONV y POOL); número de neuronas (capas FC).
 - Otros hiperparámetros: learning rate, factor de regularización, número de iteraciones en el entrenamiento, etc.

Entrenamiento

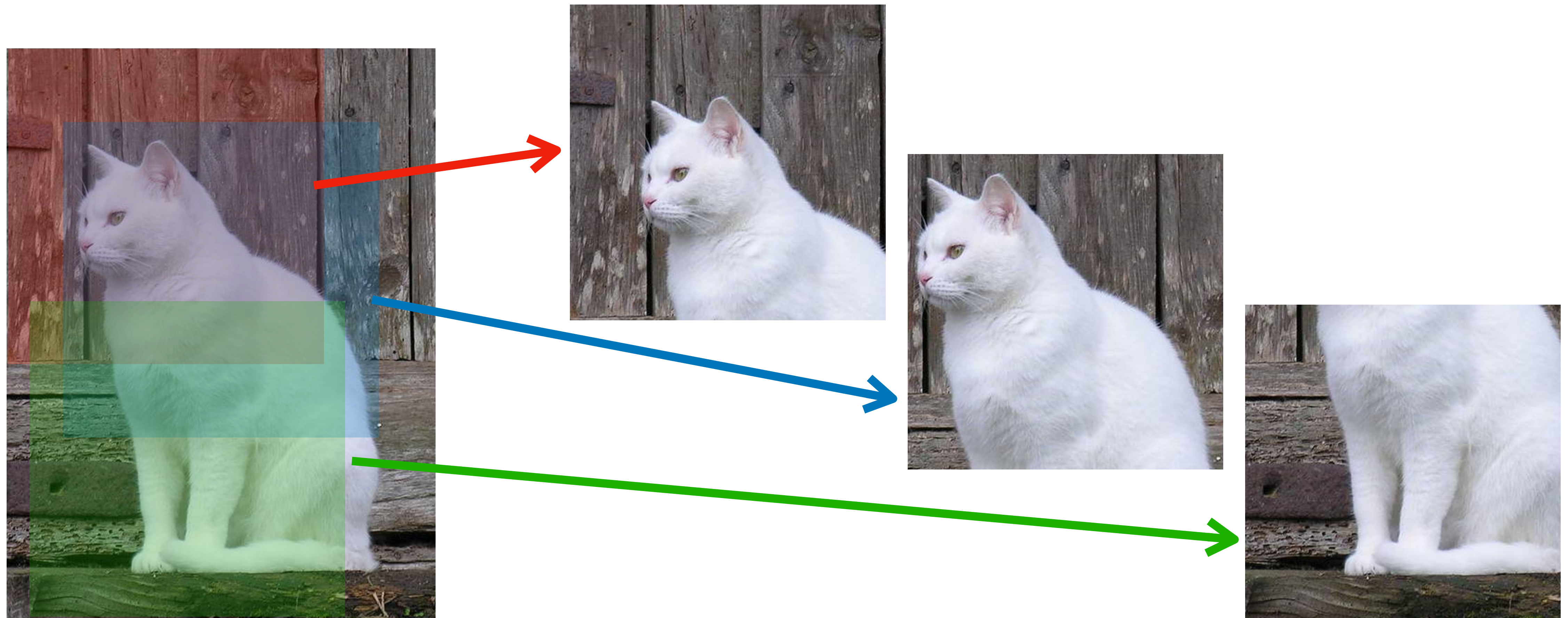
- Estimando la calidad del modelo
- Evitando el sobreajuste: regularización, dropout, early stopping, **data augmentation**
- Estrategias para acelerar el entrenamiento: inicialización de pesos, learning rate, uso de mini-batches, etc.
- Algoritmos de optimización: RMSProp, Adam
- **Transferencia de aprendizaje (transfer learning)**
 - └→ **Ajuste de parámetros (fine-tuning)**

Data augmentation

- **Objetivo:** aumentar artificialmente el número de muestras del conjunto de entrenamiento.
- Transformación de imágenes mediante una o varias operaciones:
 - Rotaciones
 - Traslaciones
 - Escala de grises
 - ...

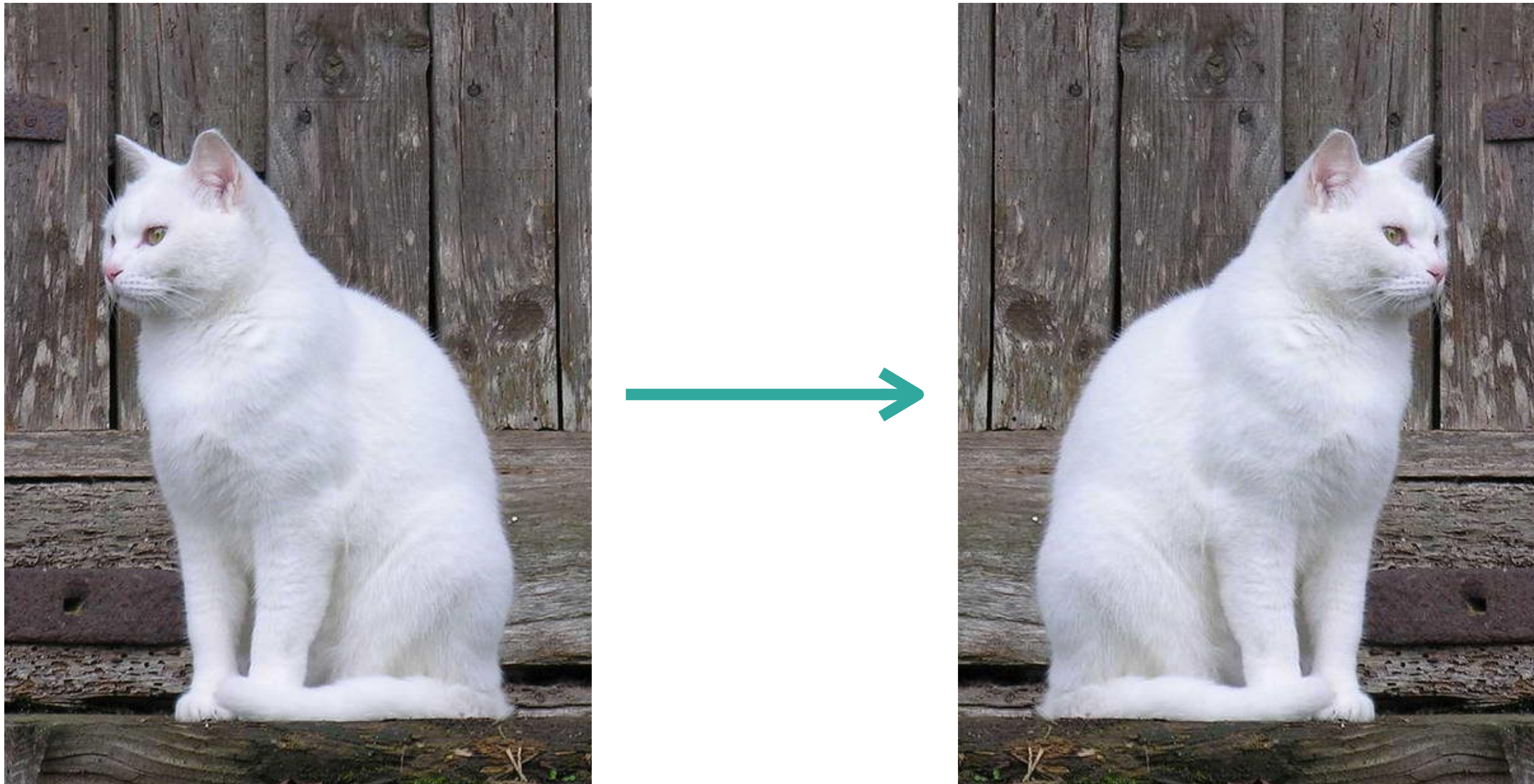
Data augmentation

- **Random cropping:** recorte aleatorio de la imagen original



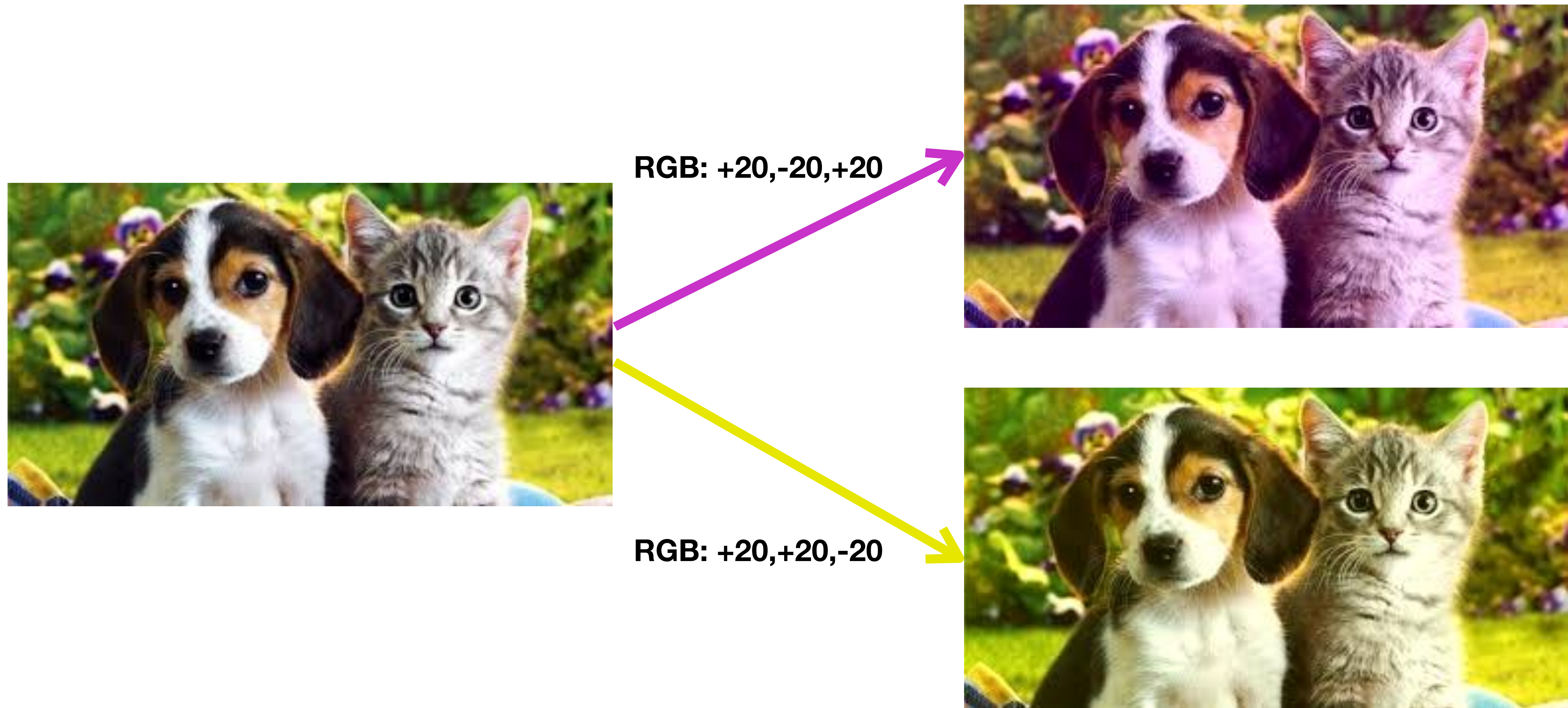
Data augmentation

- **Mirroring:** giro horizontal de la imagen original



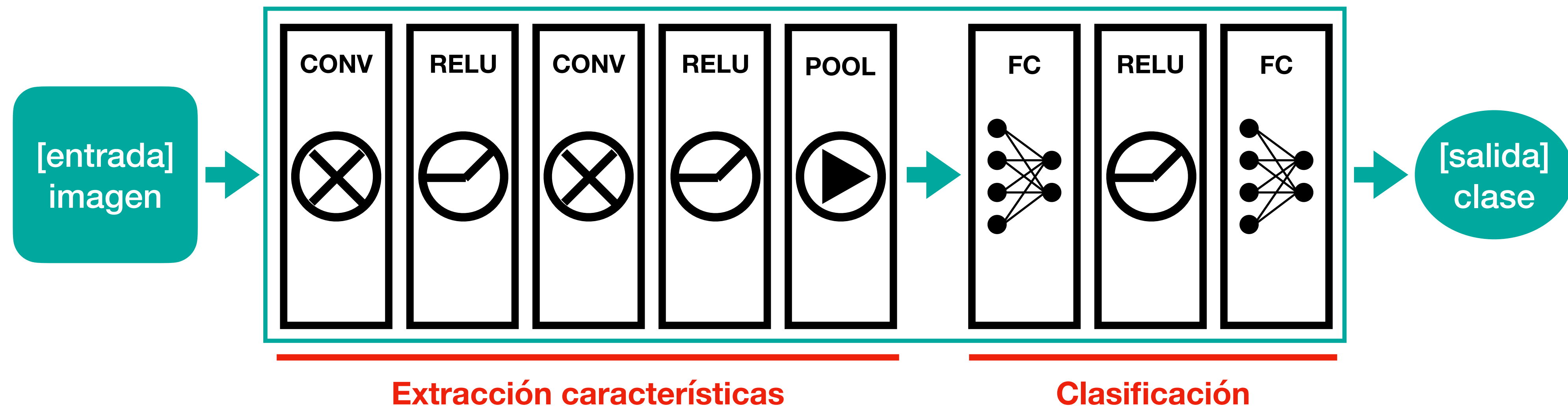
Data augmentation

- **Color shifting:** modificación de los canales RGB



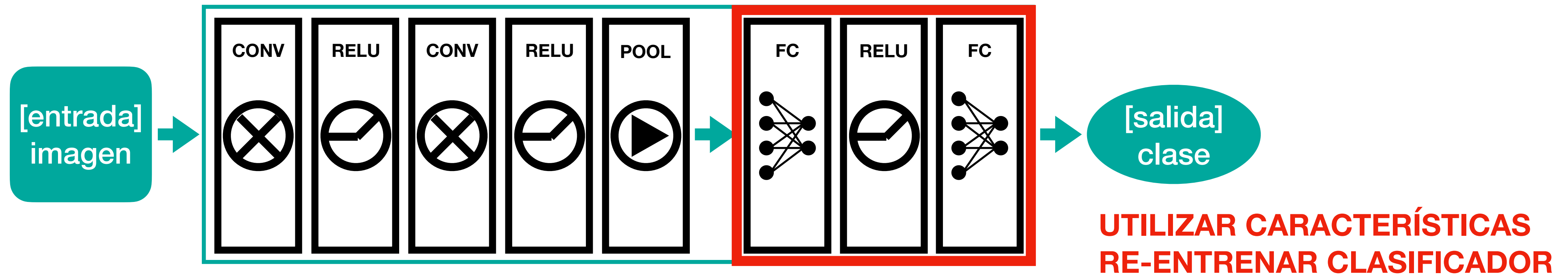
Transfer learning

- Entrenar una CNN:
 - Desde cero (inicialización aleatoria)
 - CNN pre-entrenada + transfer learning

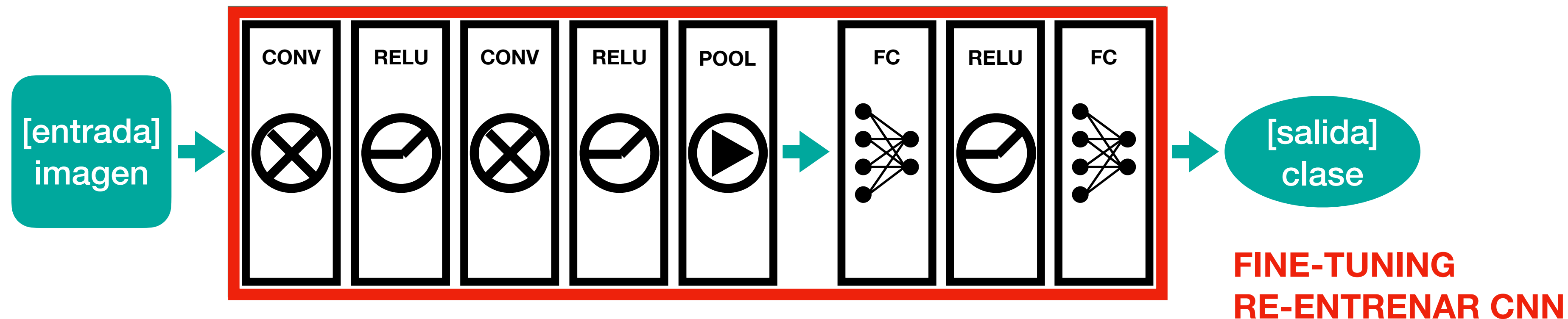


Transfer learning

- **Estrategia 1:** utilizar la CNN pre-entrenada como extractor de características



- **Estrategia 2:** re-entrenar la CNN pre-entrenada para un mejor ajuste



Transfer learning

- **¿Qué estrategia utilizar?** Diferentes escenarios:
 - Nuevo conjunto **pequeño**: estrategia 1 (CNN como extractor de características)
 - Similar al original: características de alto nivel (antes de la última FC)
 - Diferente del original: características de bajo nivel (capas intermedias)
 - Nuevo conjunto **grande**: estrategia 2 (fine-tuning)
 - Similar al original: características de alto nivel similares
 - Diferente del original: requiere más entrenamiento pero se beneficia igualmente de la inicialización de pesos a partir de un modelo pre-entrenado

ÍNDICE

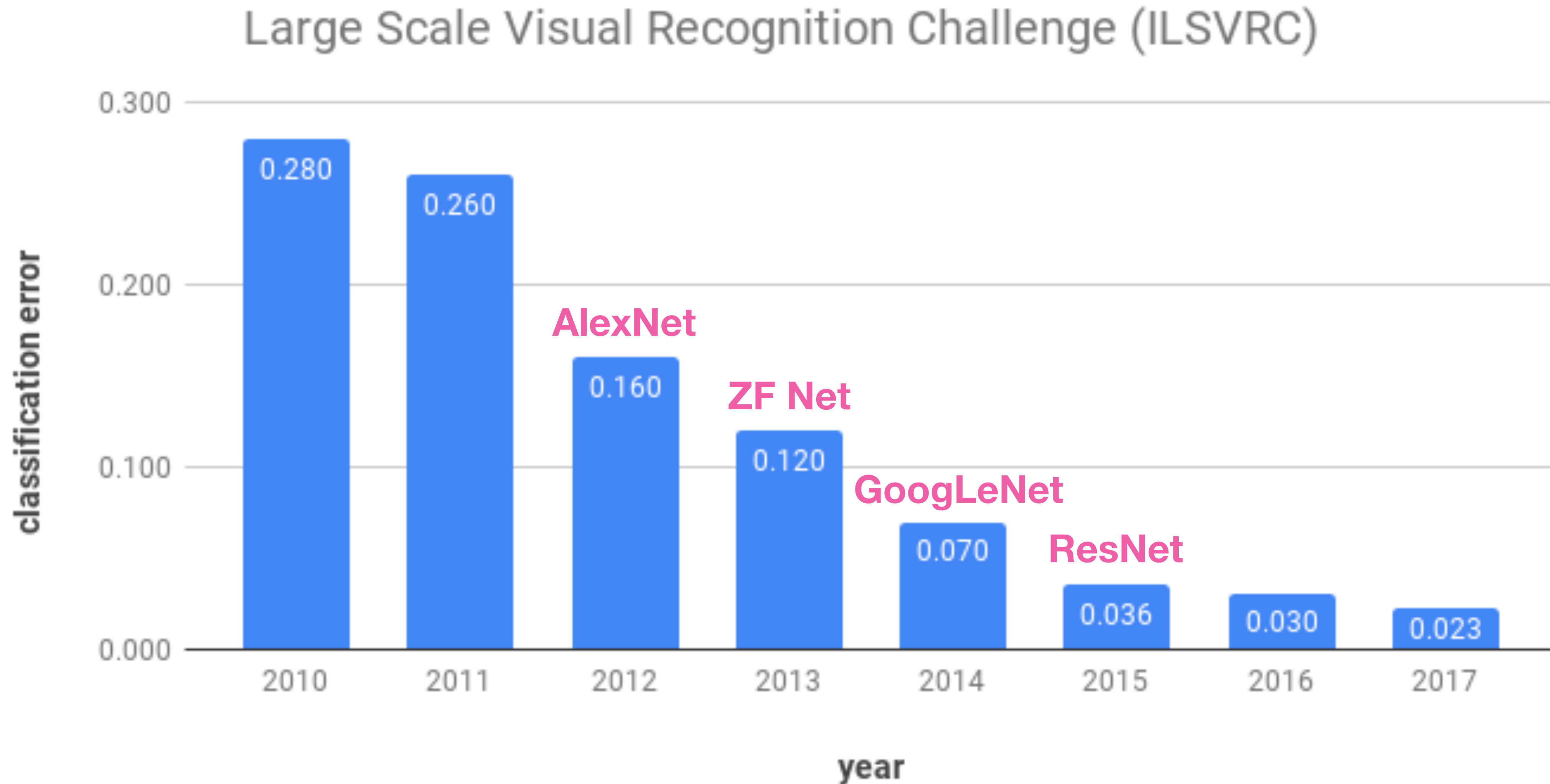
1. Introducción
2. Filtros y convoluciones
3. Redes convolucionales
4. Capas de las redes convolucionales
5. Arquitecturas y entrenamiento
- 6. Casos de estudio**

ImageNet

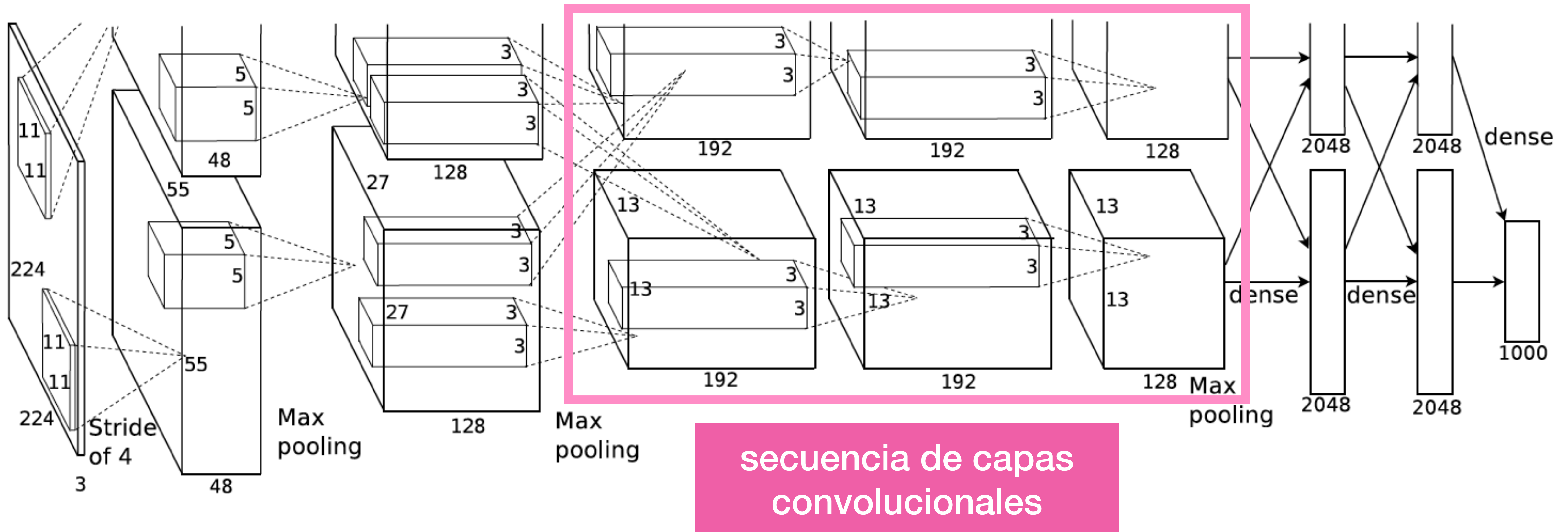
- **Base de datos de imágenes a gran escala**
 - Millones de imágenes y anotaciones
 - Clasificación, localización, etc.
 - Jerarquía **WordNet** (base de datos léxica)
 - Cada nodo en la jerarquía: cientos, miles de imágenes
- **Large Scale Visual Recognition Challenge (ILSVRC): 2010-2017**



ImageNet



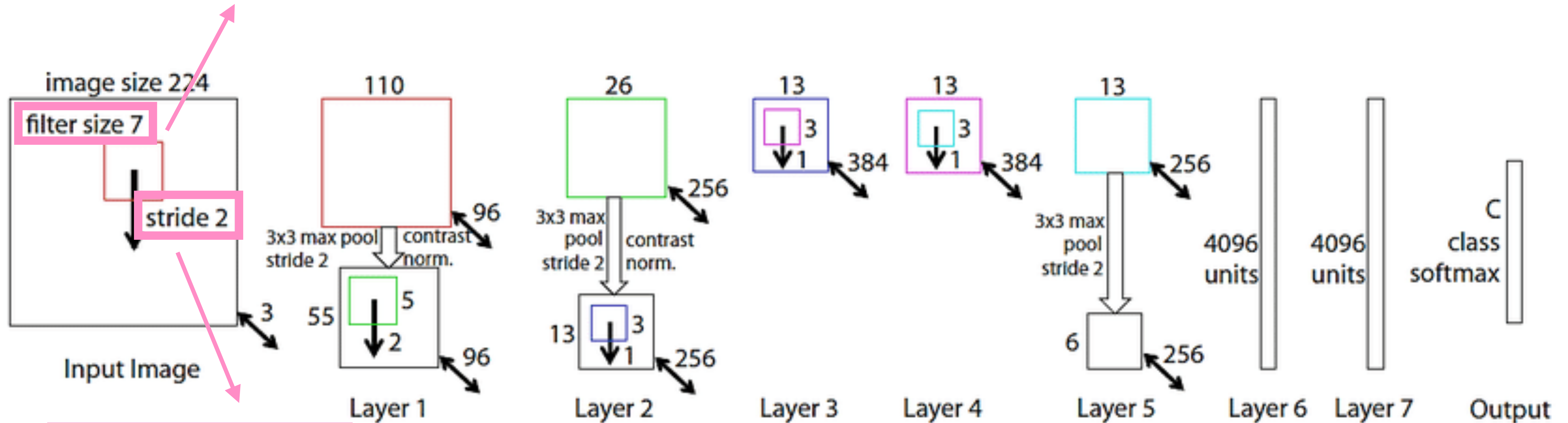
AlexNet



Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097-1105.

ZF Net

menor campo receptivo
(se reduce de 11 a 7)



menor stride
(se reduce de 4 a 2)

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pp. 818-833.

VGGNet

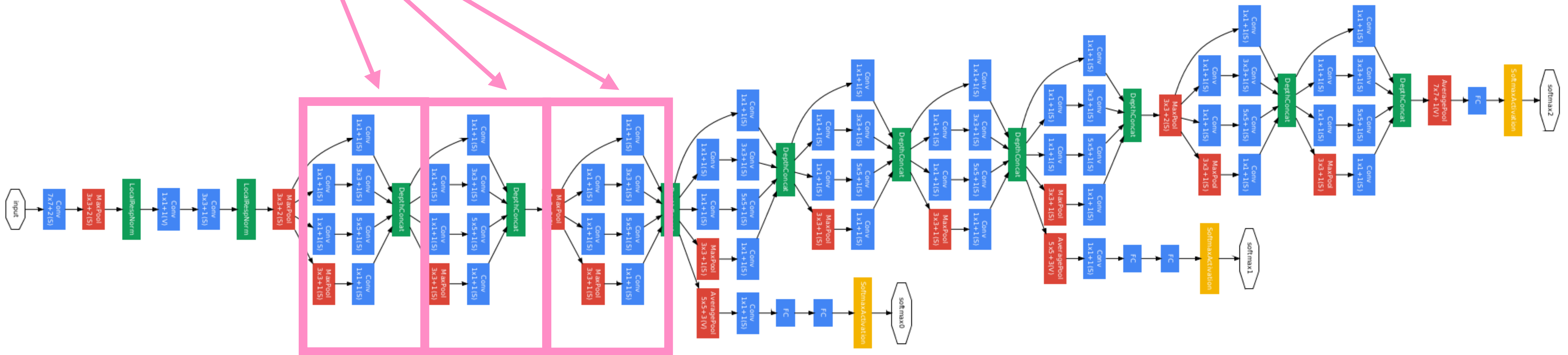
| ConvNet Configuration | | | | | |
|-----------------------------|------------------------|-------------------------------|--|--|---|
| A | A-LRN | B | C | D | E |
| 11 weight layers | 11 weight layers | 13 weight layers | 16 weight layers | 16 weight layers | 19 weight layers |
| input (224 × 224 RGB image) | | | | | |
| conv3-64 | conv3-64 LRN | conv3-64 conv3-64 | conv3-64 conv3-64 | conv3-64 conv3-64 | conv3-64 conv3-64 |
| maxpool | | | | | |
| conv3-128 | conv3-128 | conv3-128 conv3-128 | conv3-128 conv3-128 | conv3-128 conv3-128 | conv3-128 conv3-128 |
| maxpool | | | | | |
| conv3-256 conv3-256 | conv3-256 conv3-256 | conv3-256 conv3-256 | conv3-256 conv3-256 conv1-256 | conv3-256 conv3-256 conv3-256 | conv3-256 conv3-256 conv3-256 conv3-256 |
| maxpool | | | | | |
| conv3-512 conv3-512 | conv3-512 conv3-512 | conv3-512 conv3-512 | conv3-512 conv3-512 conv1-512 | conv3-512 conv3-512 conv3-512 | conv3-512 conv3-512 conv3-512 conv3-512 |
| maxpool | | | | | |
| conv3-512 conv3-512 | conv3-512 conv3-512 | conv3-512 conv3-512 | conv3-512 conv3-512 conv1-512 | conv3-512 conv3-512 conv3-512 | conv3-512 conv3-512 conv3-512 conv3-512 |
| maxpool | | | | | |
| FC-4096 | | | | | |
| FC-4096 | | | | | |
| FC-1000 | | | | | |
| soft-max | | | | | |

Keep it deep
Keep it simple

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

GoogLeNet

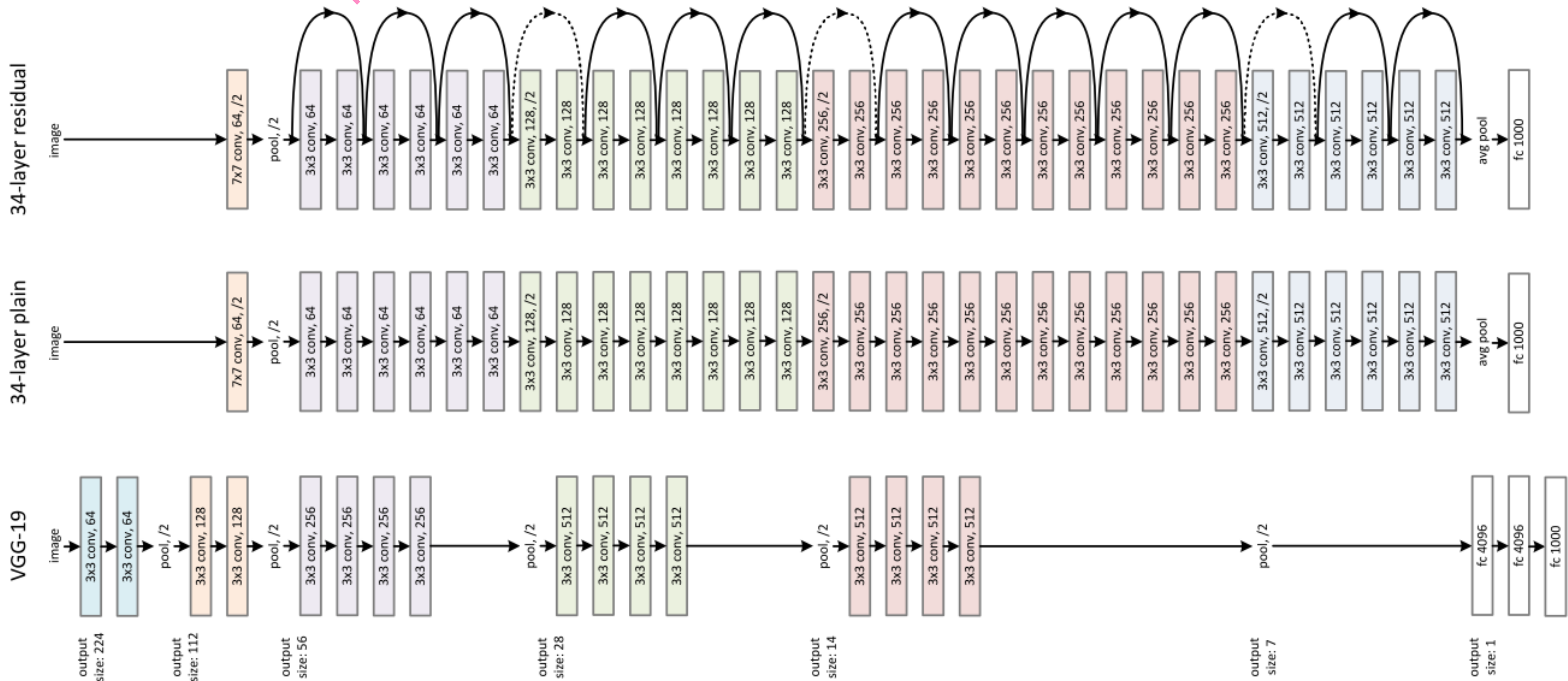
módulo Inception



Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1-9.

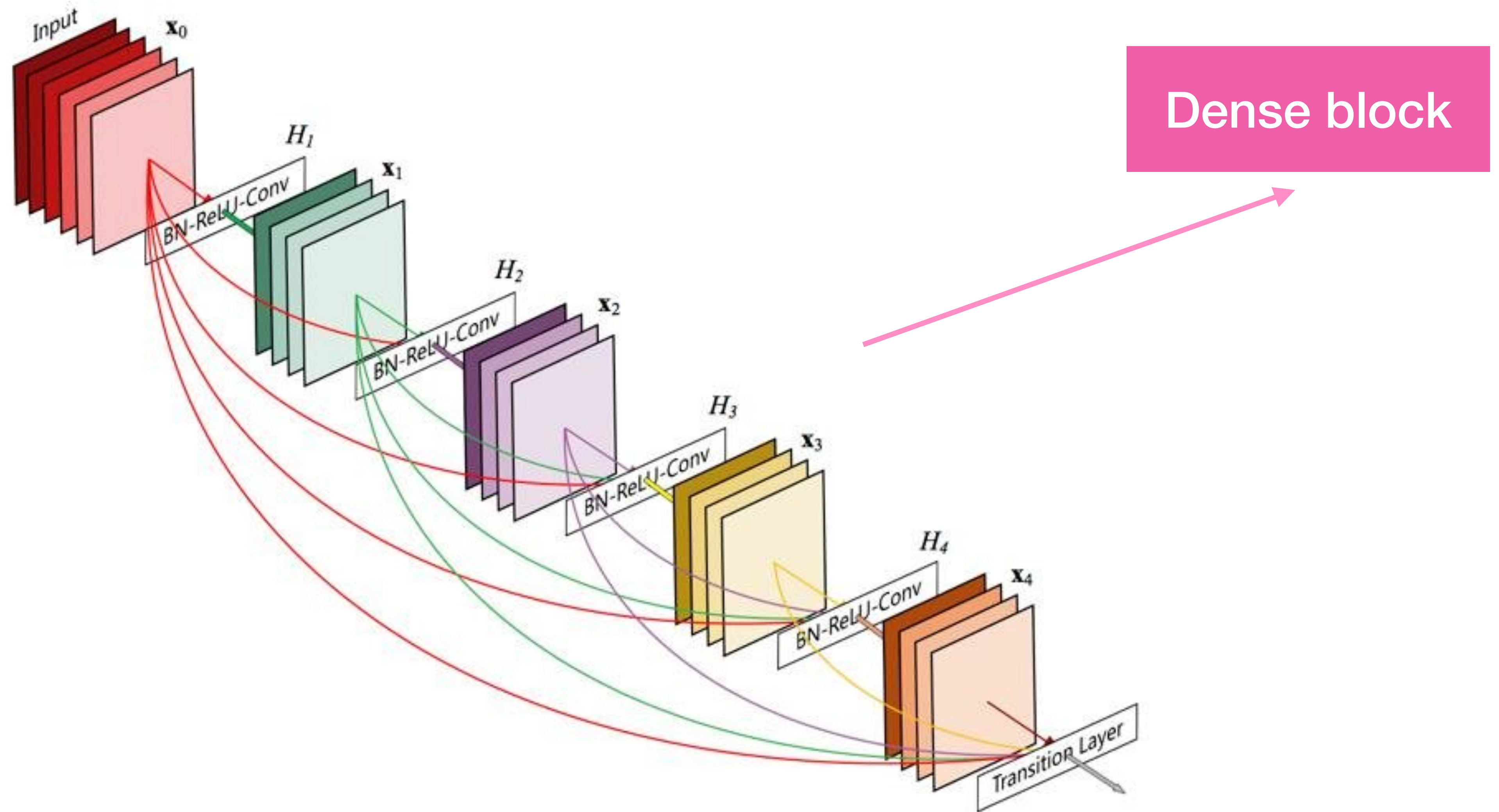
skip connections
(residual learning)

ResNet



He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770-778.

DenseNet



Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely Connected Convolutional Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4700-4708.

Comparativa

| Modelo | Precisión* Top-1 | Precisión* Top-5 | Tamaño | Parámetros | Profundidad |
|-------------------|------------------|------------------|--------|------------|-------------|
| VGG16 | 0,713 | 0,901 | 528 MB | 138,4M | 16 |
| VGG19 | 0,713 | 0,900 | 549 MB | 143,7M | 19 |
| ResNet50 | 0,749 | 0,921 | 98 MB | 25,6M | 107 |
| InceptionV3 | 0,779 | 0,937 | 92 MB | 23,9M | 189 |
| InceptionResNetV2 | 0,803 | 0,953 | 215 MB | 55,9M | 449 |
| DenseNet121 | 0,750 | 0,923 | 33 MB | 8,1M | 242 |
| DenseNet169 | 0,762 | 0,932 | 57 MB | 14,3M | 338 |
| DenseNet201 | 0,773 | 0,936 | 80 MB | 20,2M | 402 |

* Obtenidas con el conjunto de datos de validación de ImageNet, más información en:
<https://keras.io/applications/#documentation-for-individual-models>