

Redes neuronales convolucionales

**Licenciatura en Inteligencia Artificial y Ciencia de Datos, CUGDL,
Universidad de Guadalajara.**

M. en C. Iván Toledano, Octubre 2025

CNN

Contexto

- CNN's (Convolutional Neural Networks) son un tipo específico de Red Neuronal (Feed Forward).
- Referencia
- No dejan de ser redes neuronales, y por tanto incluyen una capa de **entrada**, capa de **salida**, y diferentes capas **ocultas**.
- Su especialización radica en las capas ocultas, las cuales se relacionan con operaciones de **convolución**, **max pooling**, y capas totalmente conectadas.
- Su principal ventaja se encuentra a la hora de procesar datos que pueden representarse mediante **arrays**, como lo son las imágenes (2 dimensiones en blanco y negro, tres dimensiones para rgb) o series temporales con una única dimensión.

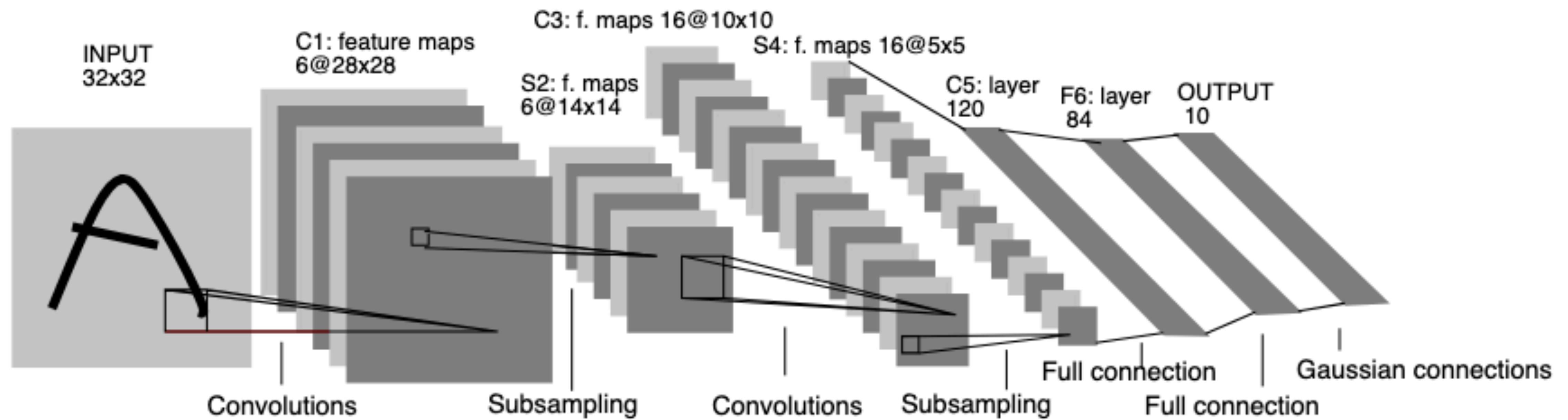


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

CNN

Convolución

What is Convolution?

Understand how this equation models and processes signals

$$g(\mathbf{r}) = \int f(\mathbf{r}') h(\mathbf{r} - \mathbf{r}') d\mathbf{r}'$$

Sound

Vision

$$(f * g)(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau)g(t - \tau) d\tau$$

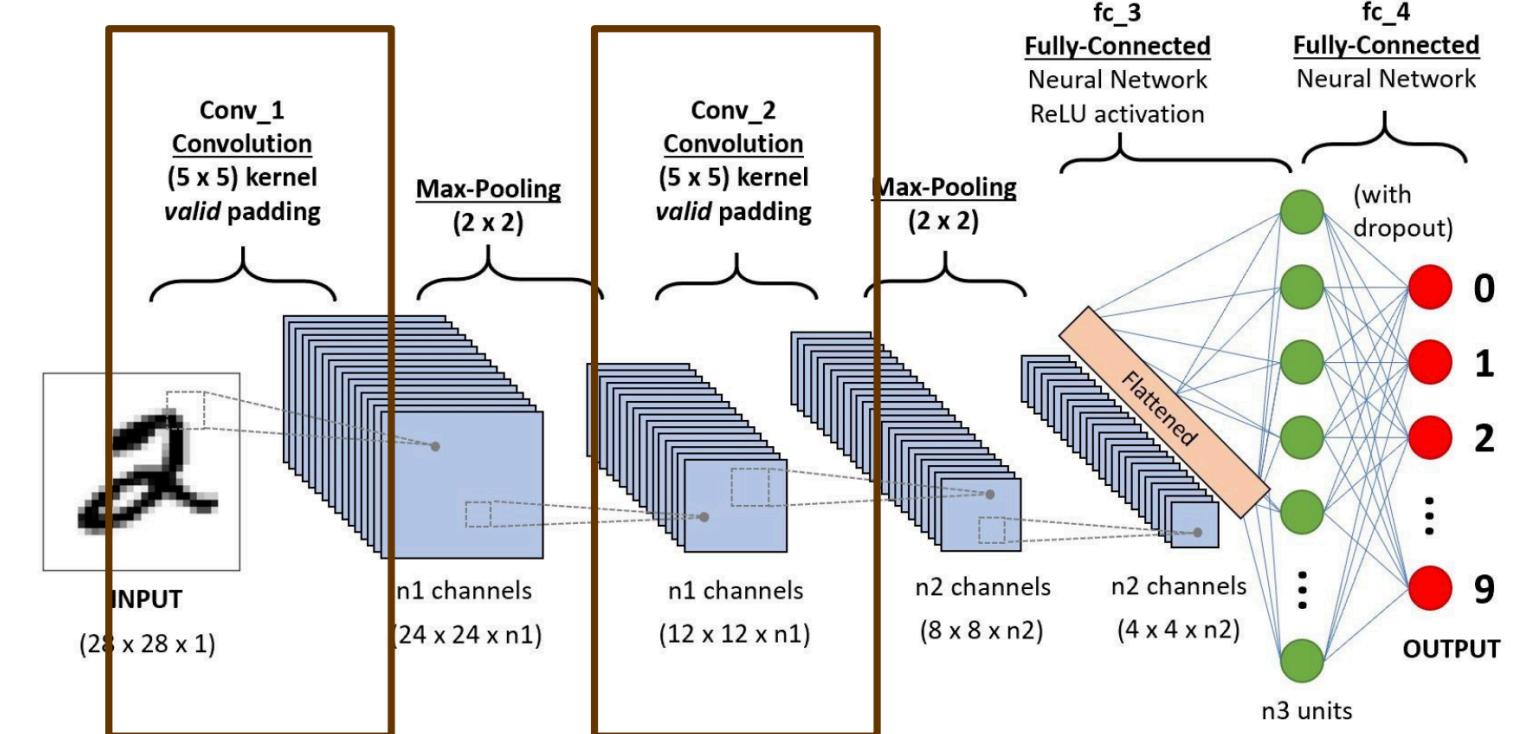
To convolve a kernel with an input signal:
flip the signal, move to the desired time,
and accumulate every interaction with the kernel

[Ejemplo 1](#)

[Ejemplo 2](#)

CNN

Convolución



- La primera capa oculta de una CNN suele ser una **capa convolucional**, la cual codifica la información para la aplicación de filtros (conocido como funciones kernel).
- Por ejemplo, un kernel podría ser utilizado para **detección de bordes**. Esto se basaría, por ejemplo, en buscar grandes diferencias entre un pixel y otro pixel que lo rodea.
- Esta operación se realiza sobre toda la imagen completa, por lo que aparece una primera característica: se tienen **pesos compartidos**, i. e., los parámetros para detectar un borde de la imagen deben de ser iguales en todas las otras zonas de la imagen.

CNN

Convolución (Ejemplo)

Input Image

0	0	0	0	0	0	0
0	1	1	1	1	1	0
0	1	0	0	1	0	0
0	1	0	1	0	0	0
0	1	1	0	0	0	0
0	1	0	0	0	0	0
0	0	0	0	0	0	0

11
10
01
00

Convolution

0	0	0
0	1	1
0	1	0

Convolved Image

1	1	1	1	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	0	0	0	0

- Asumiendo una imagen de 7X7 píxeles, la detección de un borde necesita trabajar con los 8 píxeles que le rodean. Trasladando el problema a resolver sobre la topología de una CNN, incluiríamos una capa inicial con 49 neuronas (puede mostrarse visualmente como una matriz de 7X7).
- La primera capa oculta contará con **una neurona por cada resultado del filtro (5X5)**.
- Cada una de estas 25 neuronas estará conectada con 9 neuronas de la capa inicial, pero los **pesos** utilizados para esta interacción w_1, w_2, \dots, w_9 serán **compartidos** por todas las neuronas de la capa oculta.
- Una vez aprendida, la función de activación (ReLU normalmente) nos permitirá saber qué píxel de la imagen se corresponde o no con un borde.

CNN

Convolución (Ejemplo)

Input Image

0	0	0	0	0	0	0
0	1	1	1	1	1	0
0	1	0	0	1	0	0
0	1	0	1	0	0	0
0	1	1	0	0	0	0
0	1	0	0	0	0	0
0	0	0	0	0	0	0

Convolution

0	0	0
0	1	1
0	1	0

Convolved Image

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

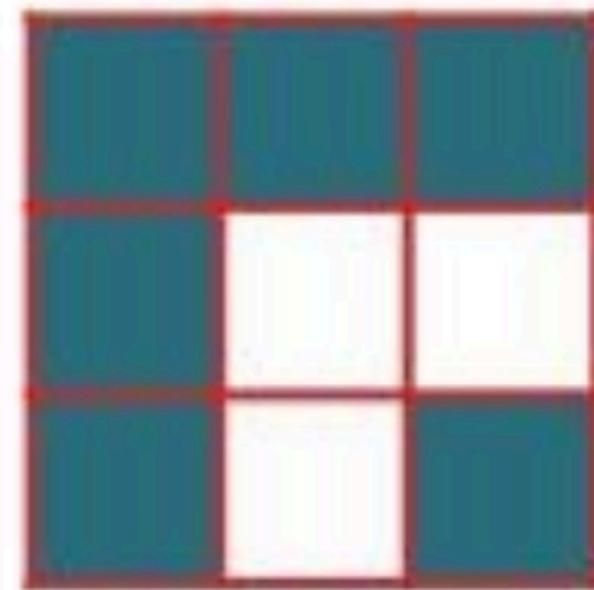
CNN

Convolución (Ejemplo)

Input Image

0	0	0	0	0	0	0
1	1	1	1	1	1	0
1	1	0	0	1	0	0
0	1	0	1	0	0	0
0	1	1	0	0	0	0
0	1	0	0	0	0	0
0	0	0	0	0	0	0

Convolution



Convolved Image

3			

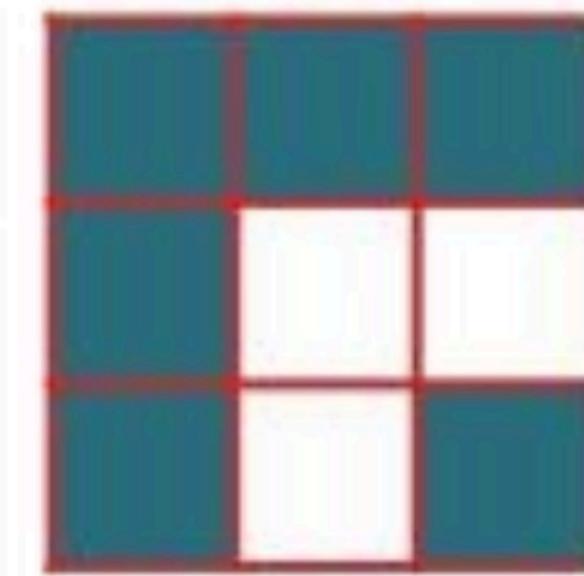
CNN

Convolución (Ejemplo)

Input Image

0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	0	1	0	0	0	0
0	1	0	1	0	0	0
0	1	1	0	0	0	0
0	1	0	0	0	0	0
0	0	0	0	0	0	0

Convolution



Convolved Image

3	2			

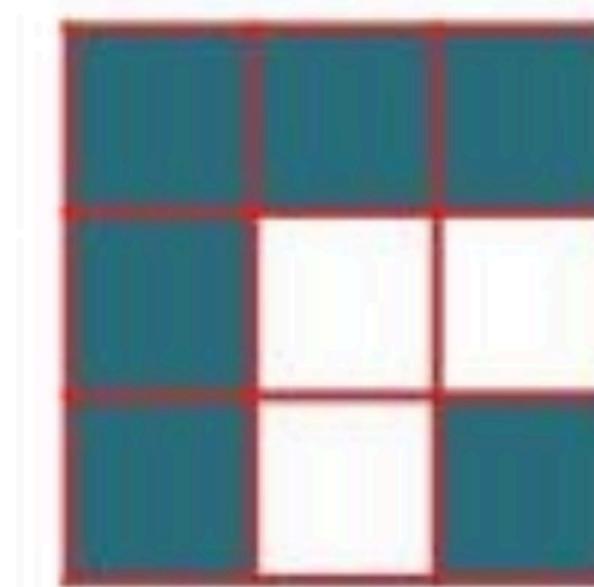
CNN

Convolución (Ejemplo)

Input Image

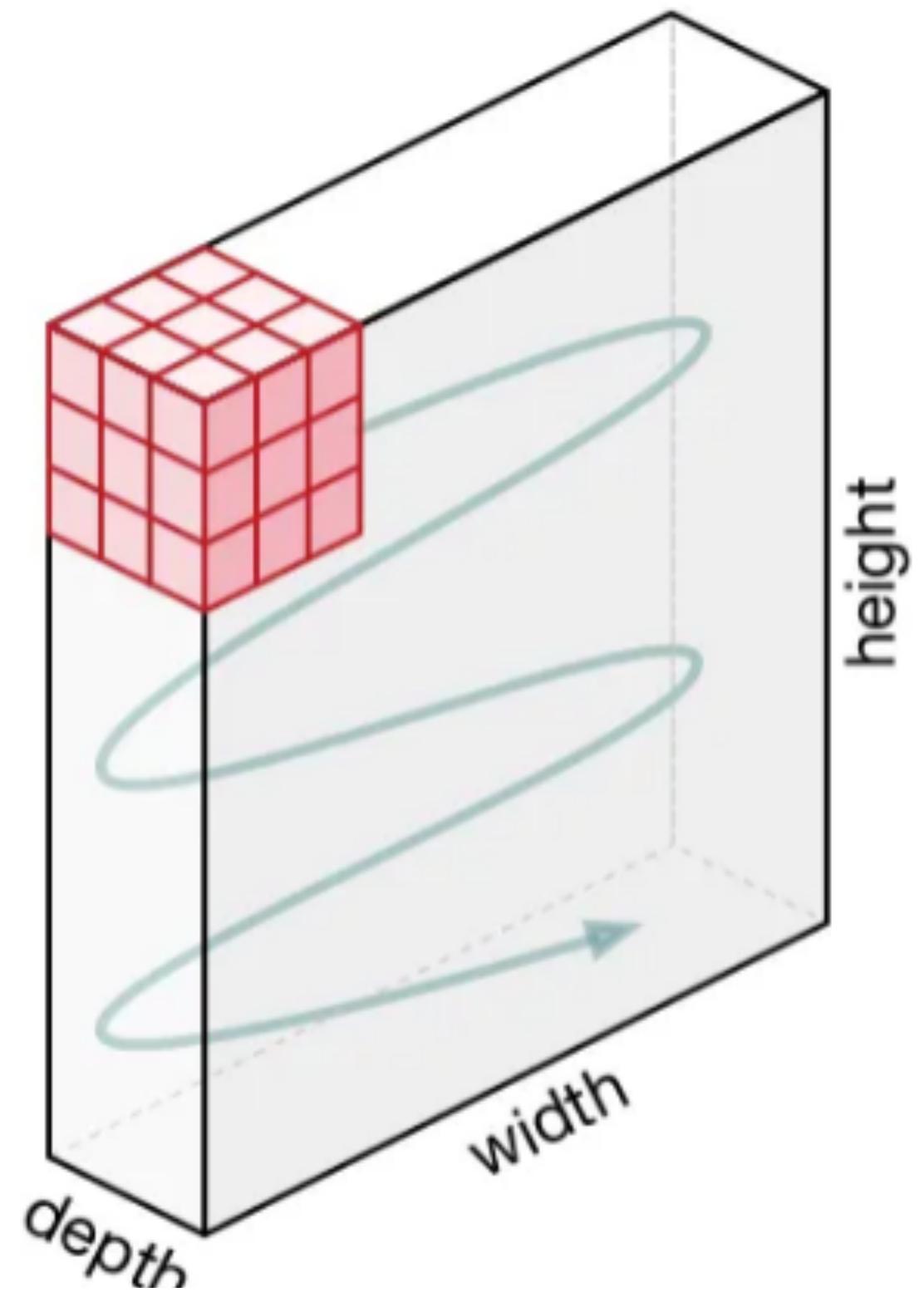
0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	0
0	1	0	0	1	0	0	0
0	1	0	1	0	0	0	0
0	1	1	0	0	0	0	0
0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Convolution



Convolved Image

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



7	91	9	164
87	221	66	13
29	34	1	81
4	87	72	45

Matrix red

0	1
1	0

Filter red

$$0 * 7 + 1 * 91 + 1 * 87 + 0 * 221$$

→ 178

99	15	3	47
51	114	19	82
189	74	255	2
48	10	147	79

Matrix green

0	1
1	1

Filter green

$$0 * 99 + 1 * 15 + 1 * 51 + 1 * 114$$

→ 180 →

475		

187	234	168	2
88	29	172	71
92	58	9	110
62	47	69	5

Matrix blue

0	0
1	1

Filter blue

$$0 * 187 + 0 * 234 + 1 * 88 + 1 * 29$$

→ 117

CNN

Convolución (Kernel típicos)

Input Image

0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	0
0	1	0	0	1	0	0	0
0	1	0	1	0	0	0	0
0	1	1	0	0	0	0	0
0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Identity
Convolution

1	1	1	1	1	1
0	0	0	0	0	0
0	1	0	0	0	0
1	0	1	0	0	0
0	0	0	0	0	0

Convolved Image

0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	0
0	1	0	0	1	0	0	0
0	1	0	1	0	0	0	0
0	1	1	0	0	0	0	0
0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Blurring
Convolution

.4	.5	.5	.5	.4
.4	.2	.3	.6	.3
.5	.4	.4	.2	.1
.5	.6	.2	.1	0
.4	.3	.1	0	0

Convolved Image

0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	0
0	1	0	0	1	0	0	0
0	1	1	0	0	0	0	0
0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0

CNN key idea:
Treat convolution matrix as parameters and learn them!

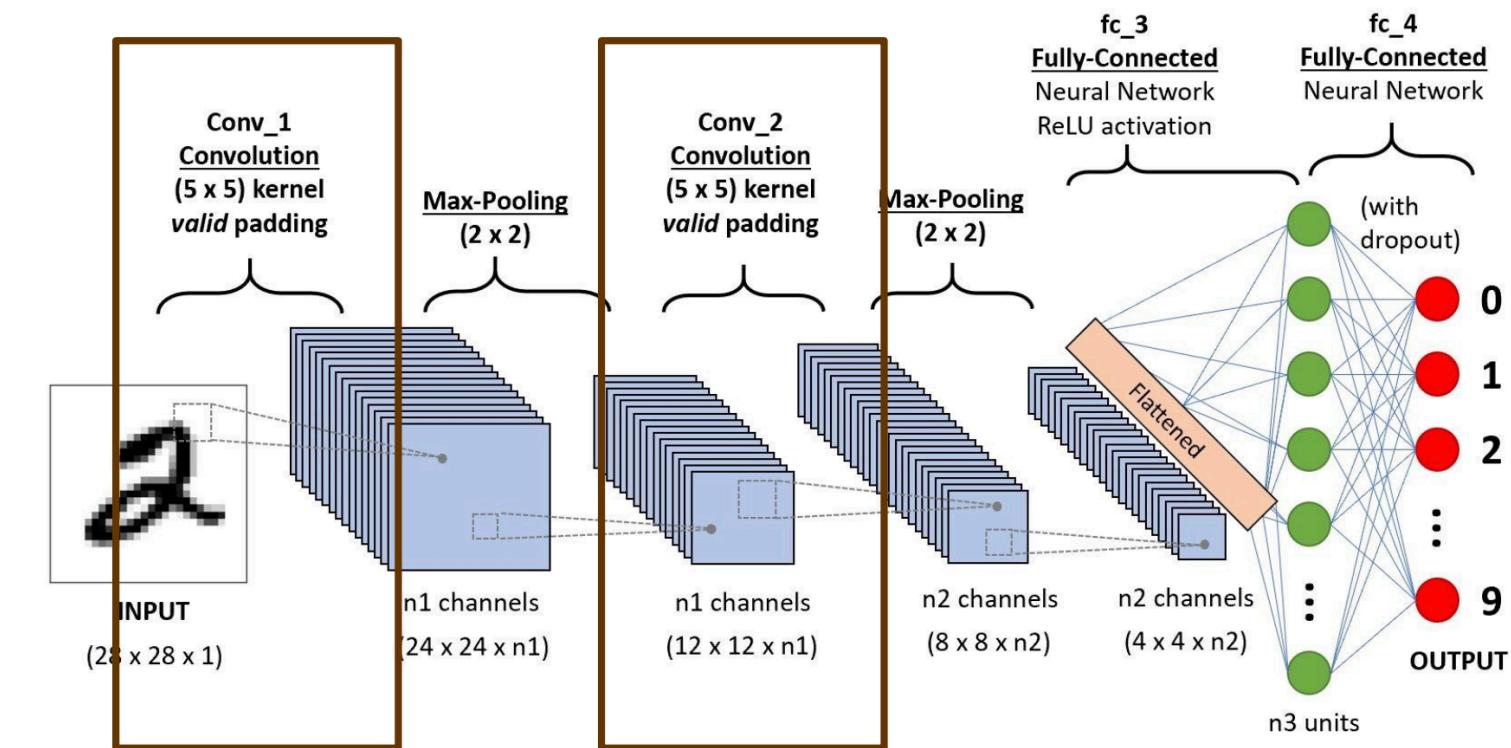
Learned Convolution

.4	.5	.5	.5	.4
.4	.2	.3	.6	.3
.5	.4	.4	.2	.1
.5	.6	.2	.1	0
.4	.3	.1	0	0

Convolved Image

Ejemplo

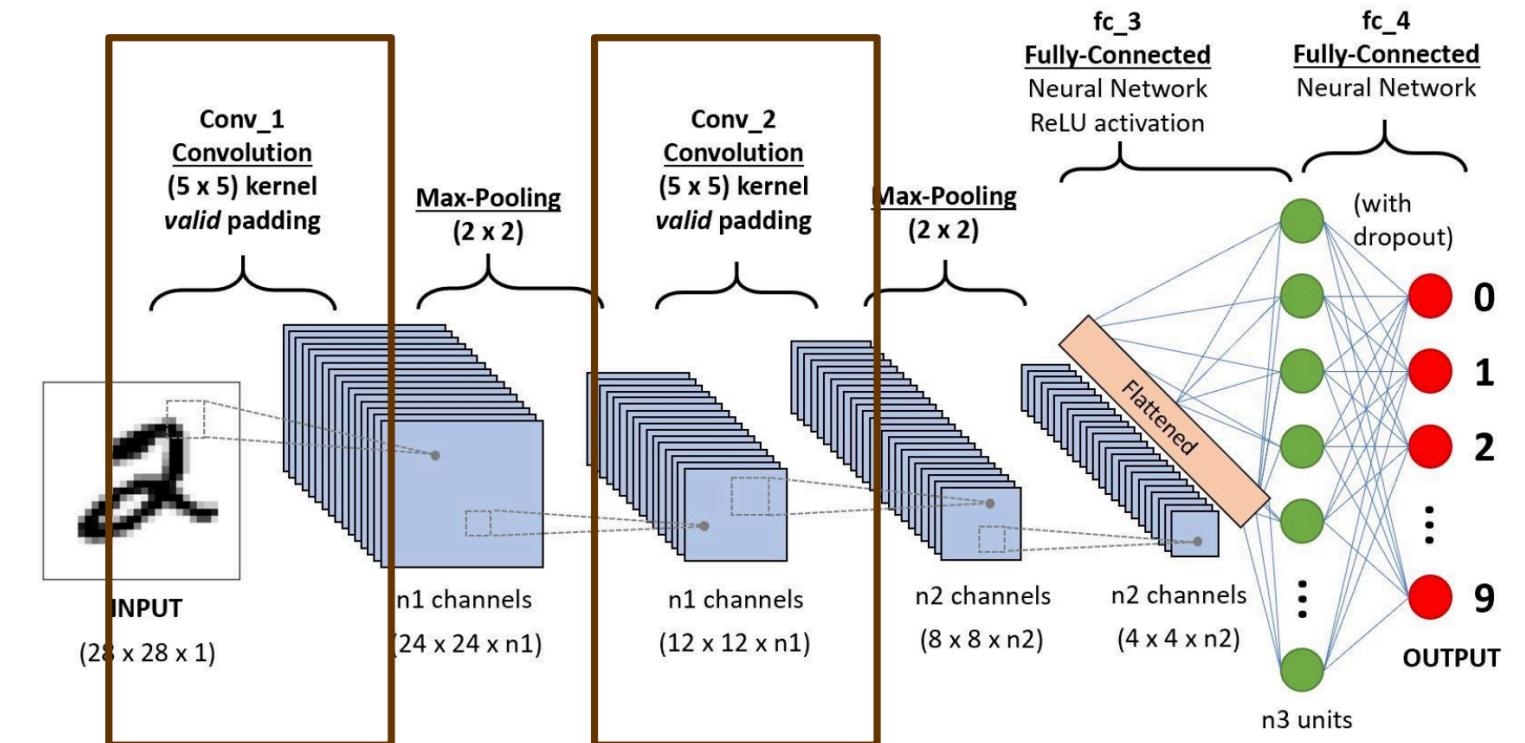
CNN Padding



- Se observa que se tiende a perder información en la localización de los píxeles en el perímetro de la imagen cuando aplicamos capas convolucionales.
- Una solución a esto es añadir píxeles (**neuronas sintéticas**) de relleno extra alrededor del borde de la imagen, para incrementar el tamaño efectivo de la misma.
- Si realizamos una operación de convolución y nos quedamos con una matriz con la misma dimensión, se refiere a **Same Padding**.
- Si la matriz de salida tiene una dimensión menor a la original, se refiere a **Valid Padding**.

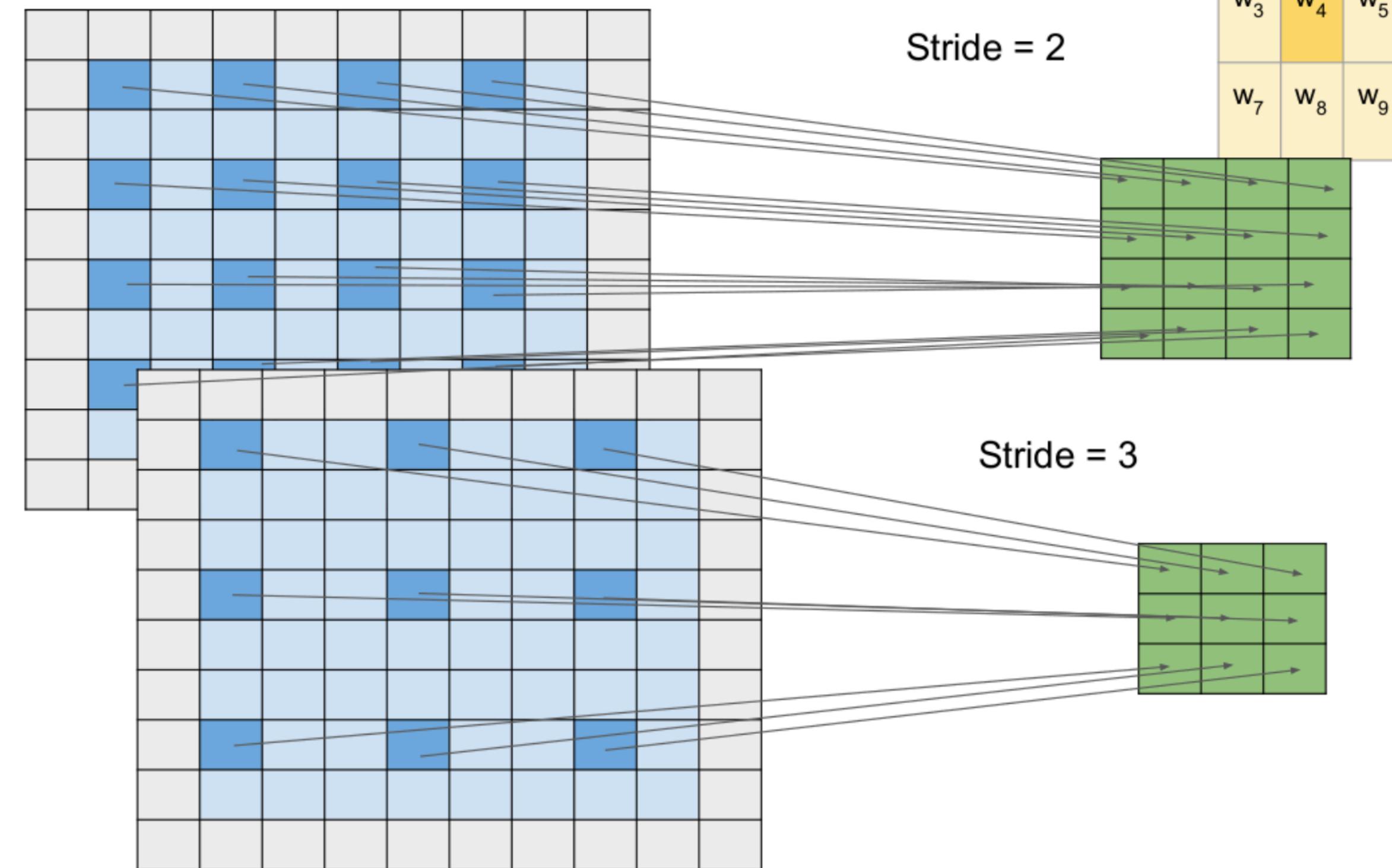
CNN

Padding



- El parámetro **stride** nos indica el número de neuronas a avanzar tras la aplicación de una convolución.

Operación Convolución

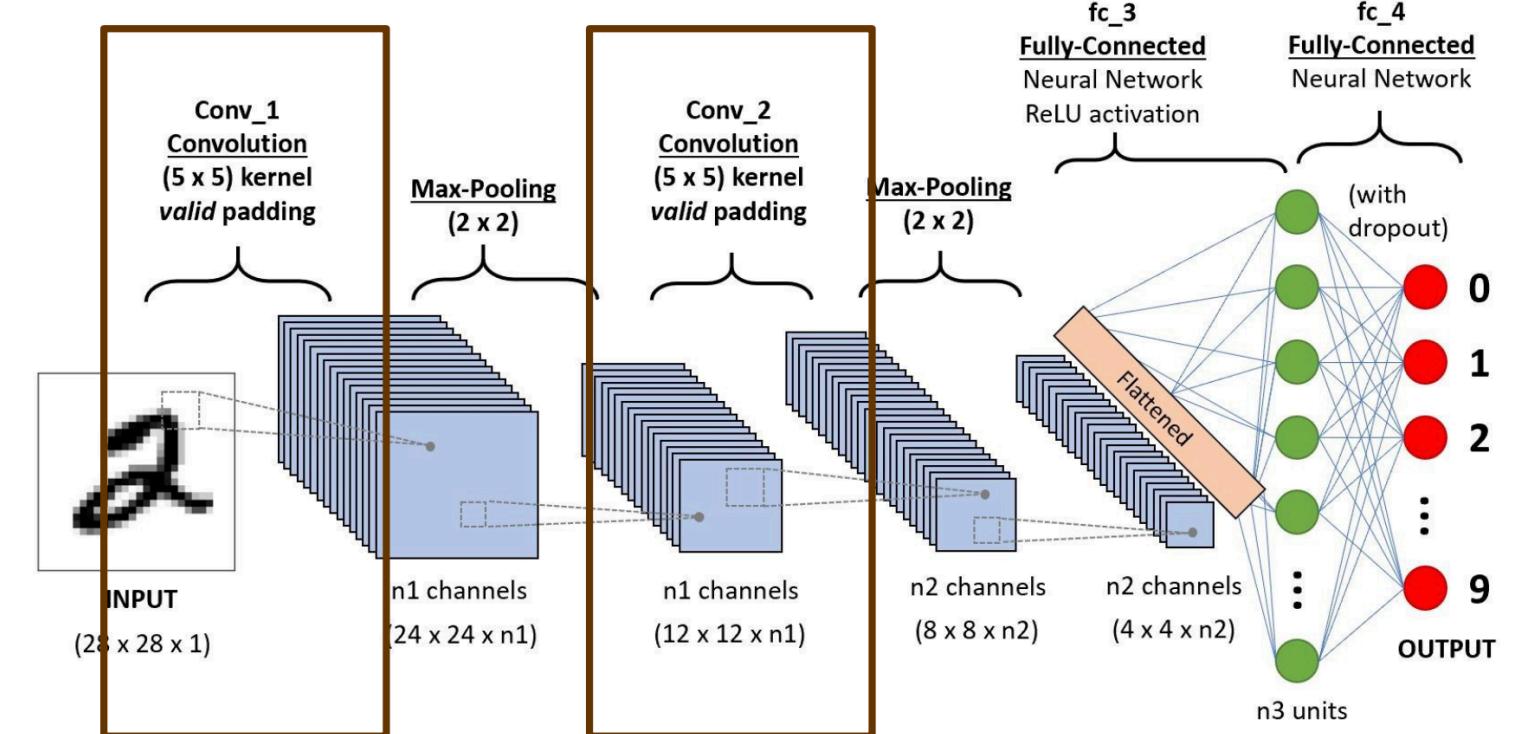


CNN

Características

- El objetivo de las CNN's no suele ser el aprendizaje de filtros básicos (e.g. bordes), aunque puede ser un paso necesario (pero no suficiente) para tener más adelante objetos más complejos.
- Los filtros **no** se introducen explícitamente en la definición del problema, sino que las características de la red harán que se aprendan de forma automática.
- Las primeras capas tienden a aprender características **simples**. Esto pueden ser los bordes, por ejemplo.
- Las características **más complejas** (formas básicas) se aprenden a medida que pasamos a capas más profundas.

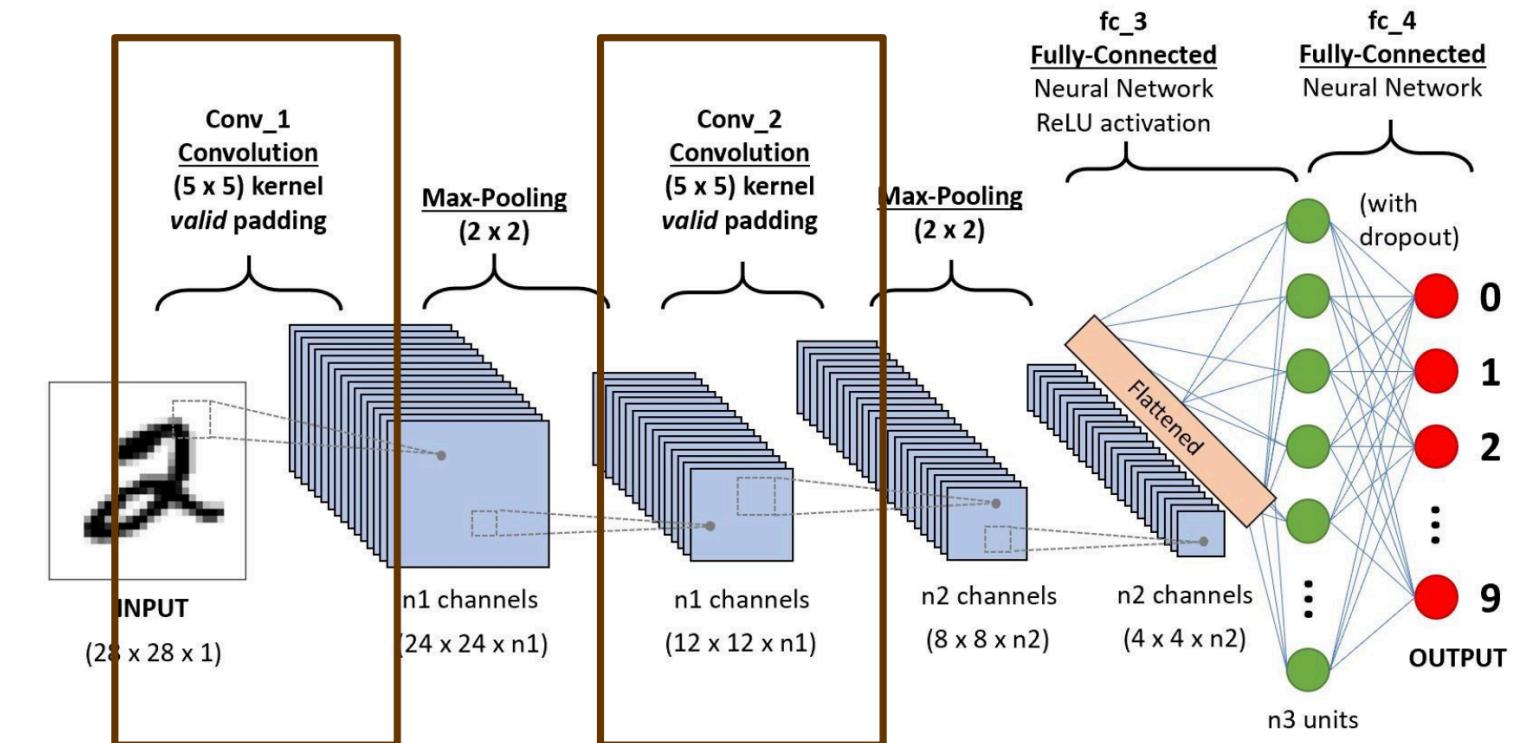
CNN Pooling



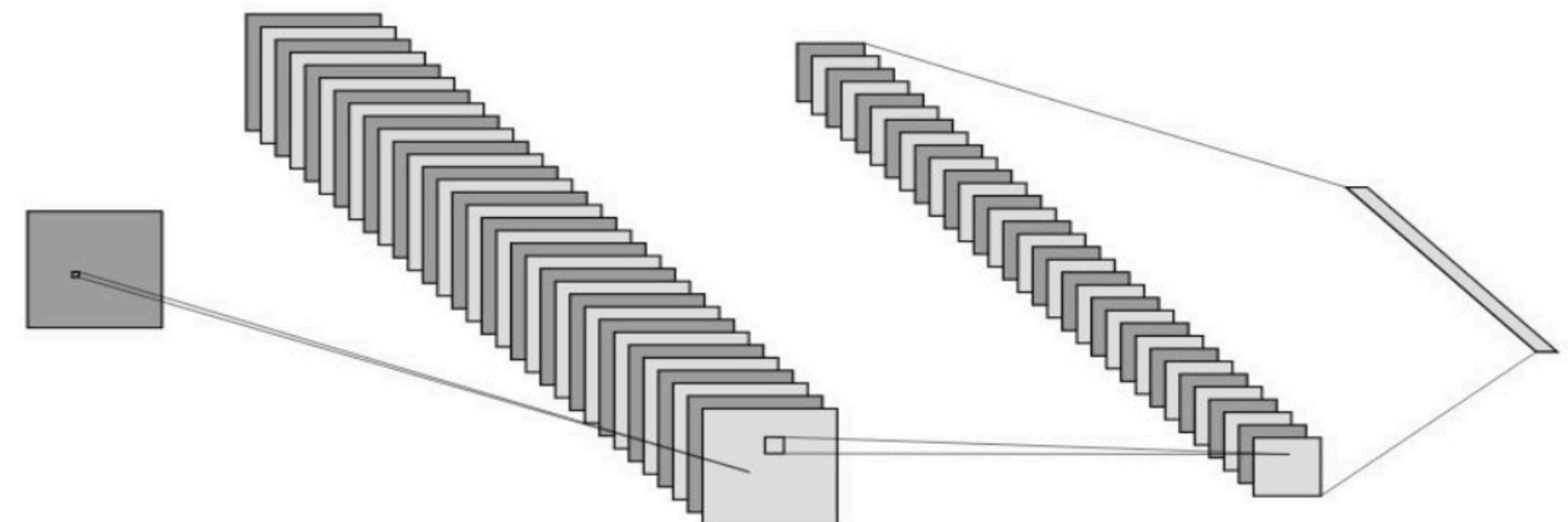
- La operación **pooling** tiene el objetivo de **reducir la dimensionalidad** de las capas generadas tras la aplicación de operaciones de convolución (sólo se aplicarán al alto y ancho de la imagen).
- Puede ser categorizada en dos tipos:
- **Max Pooling.** Regresa el valor máximo del segmento cubierto por el kernel de la imagen. Tiene una función adicional como supresor de ruido, pues elimina activaciones ruidosas.
- **Average Pooling.** Regresa la media de los valores cubiertos por el kernel. Aquí solamente se reduce la dimensionalidad.
- La que suele ser más útil es **Max Pooling**.

CNN

Pooling (Ejemplo)

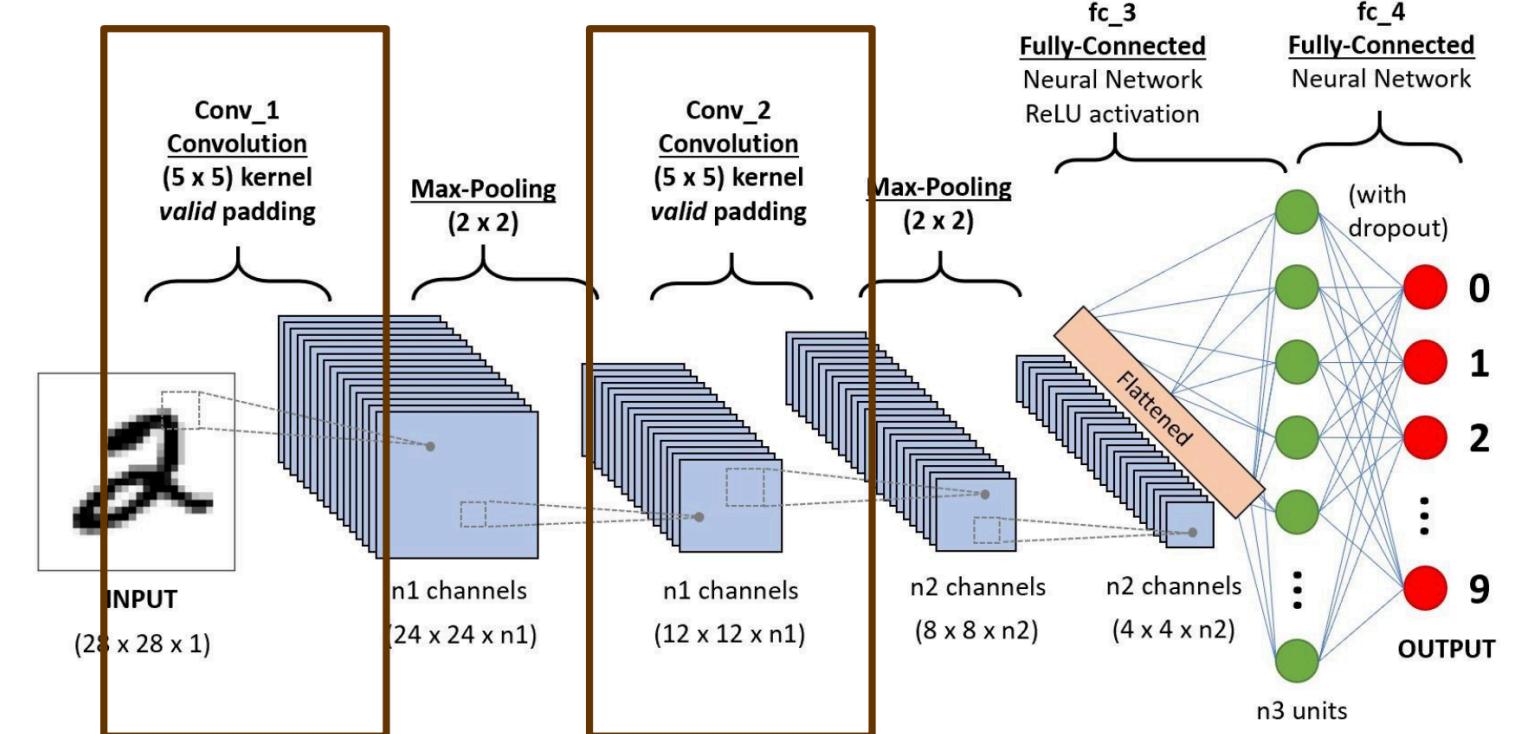


- Suponiendo que se trabajan con imágenes **640X640** (B/N) y que la primera convolución aplica 32 filtros de 3X3. El resultado será una capa intermedia de **640X640X32**.
- Una operación pooling reduciría a un 25% de la original y se tendría una capa intermedia de **320X320X32**.

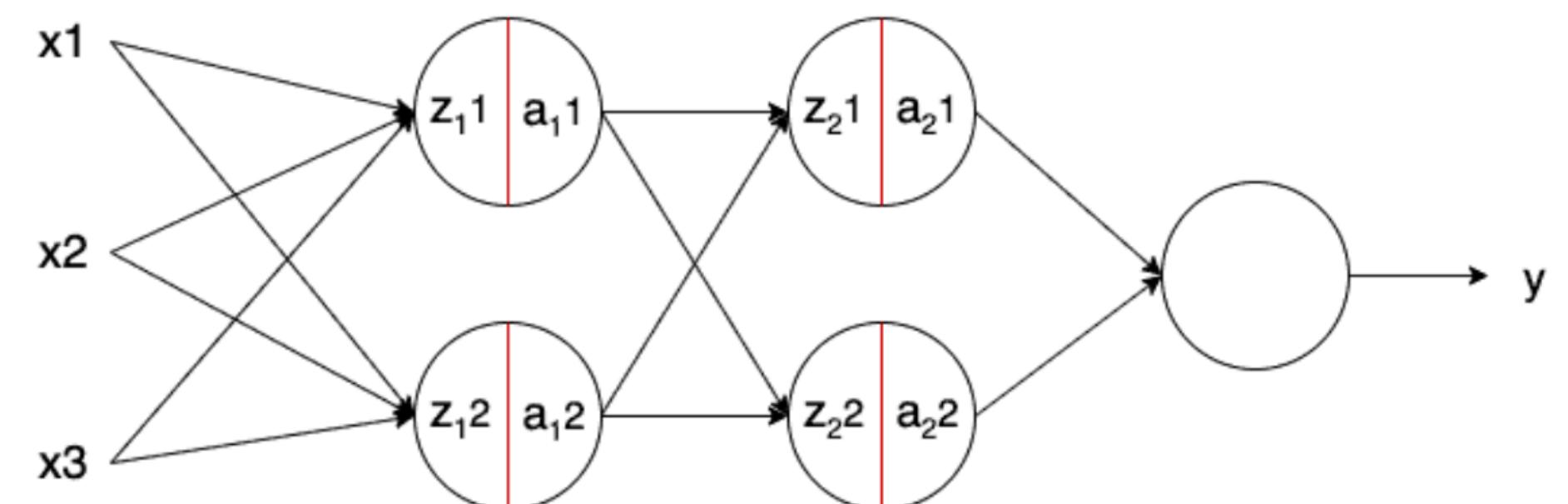


CNN

Capa totalmente conectada



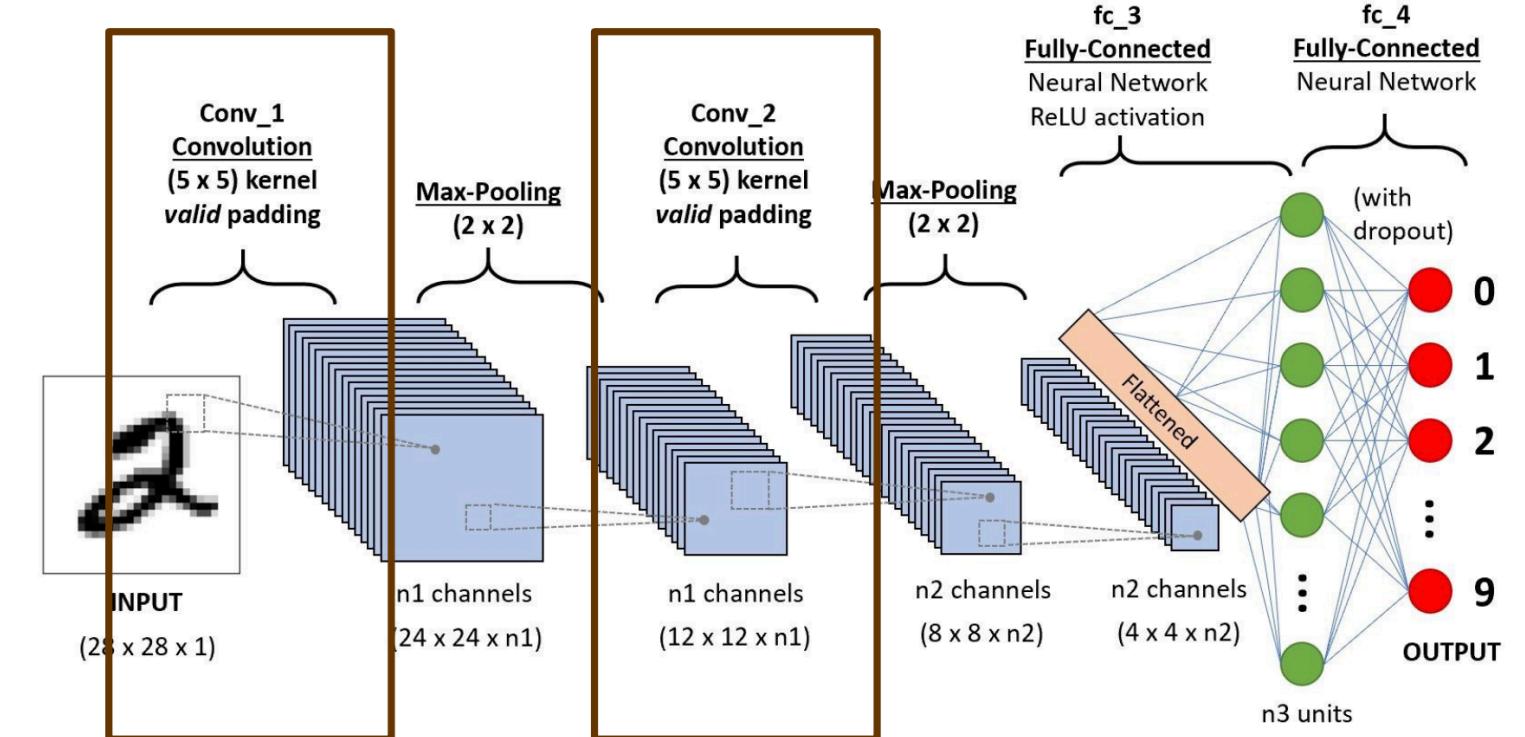
- Tras la combinación de capas de tipo convolución y pooling, las etapas finales de una CNN las forman capas de neuronas **totalmente conectadas** (fully-connected), similares a las capas de las redes neuronales tradicionales.
- Encima de esto, algunos autores han ido proponiendo el uso de **capas de normalización**.
- Referencia



$$z = g(w, x); \quad z^N = \left(\frac{z - m_z}{s_z} \right) \cdot \gamma + \beta; \quad a = f(z^N)$$

CNN

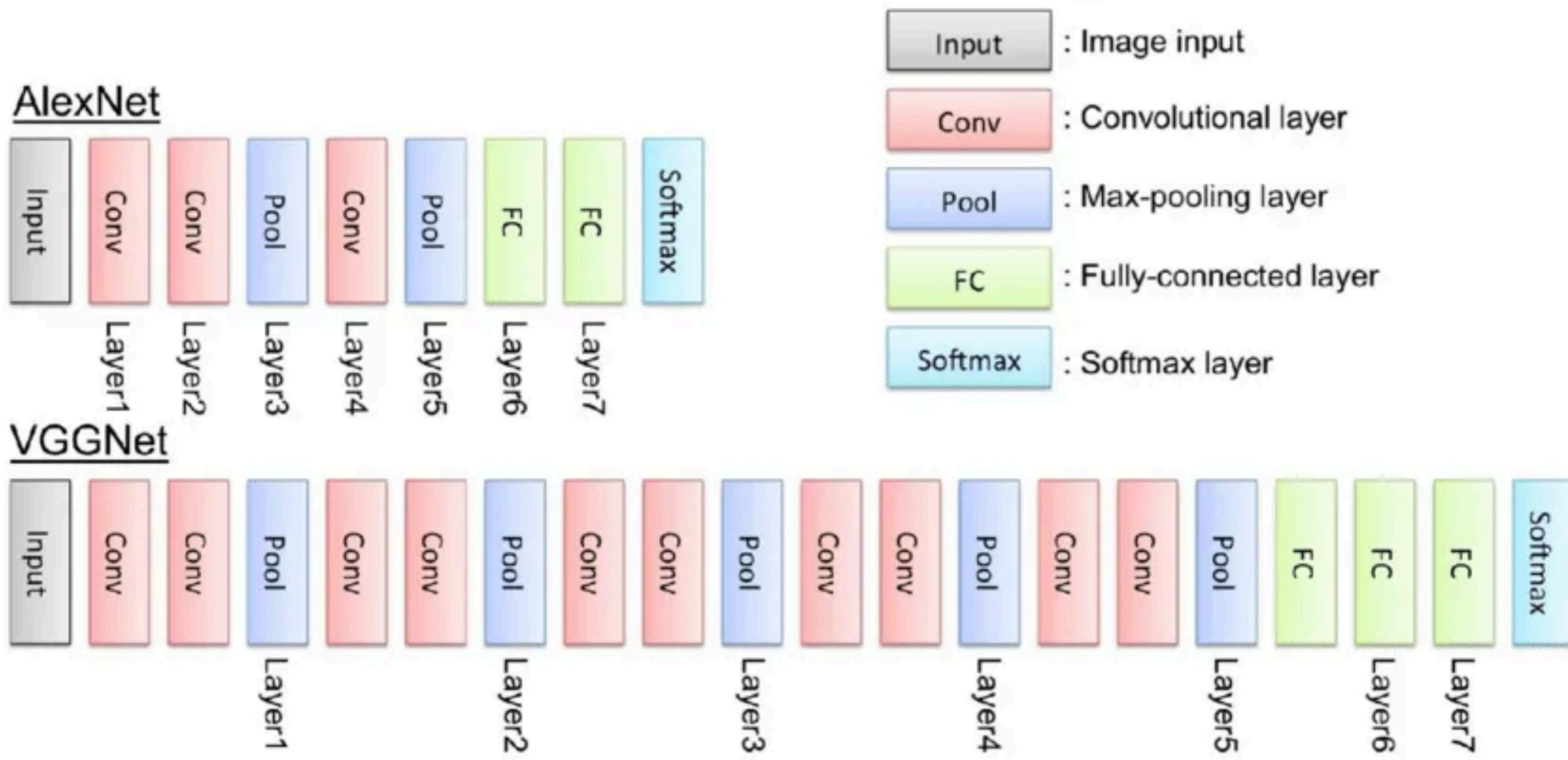
Función de activación



- Las operaciones de convolución suelen incluir una etapa de activación basada en el uso de la función **ReLU**, lo que aporta capacidad de resolución de problemas no lineales.
- También se suelen encontrar capas **SoftMax**, enfocadas en la clasificación multiclase, la cual permite interpretar las decisiones del clasificado como probabilidades.

CNN

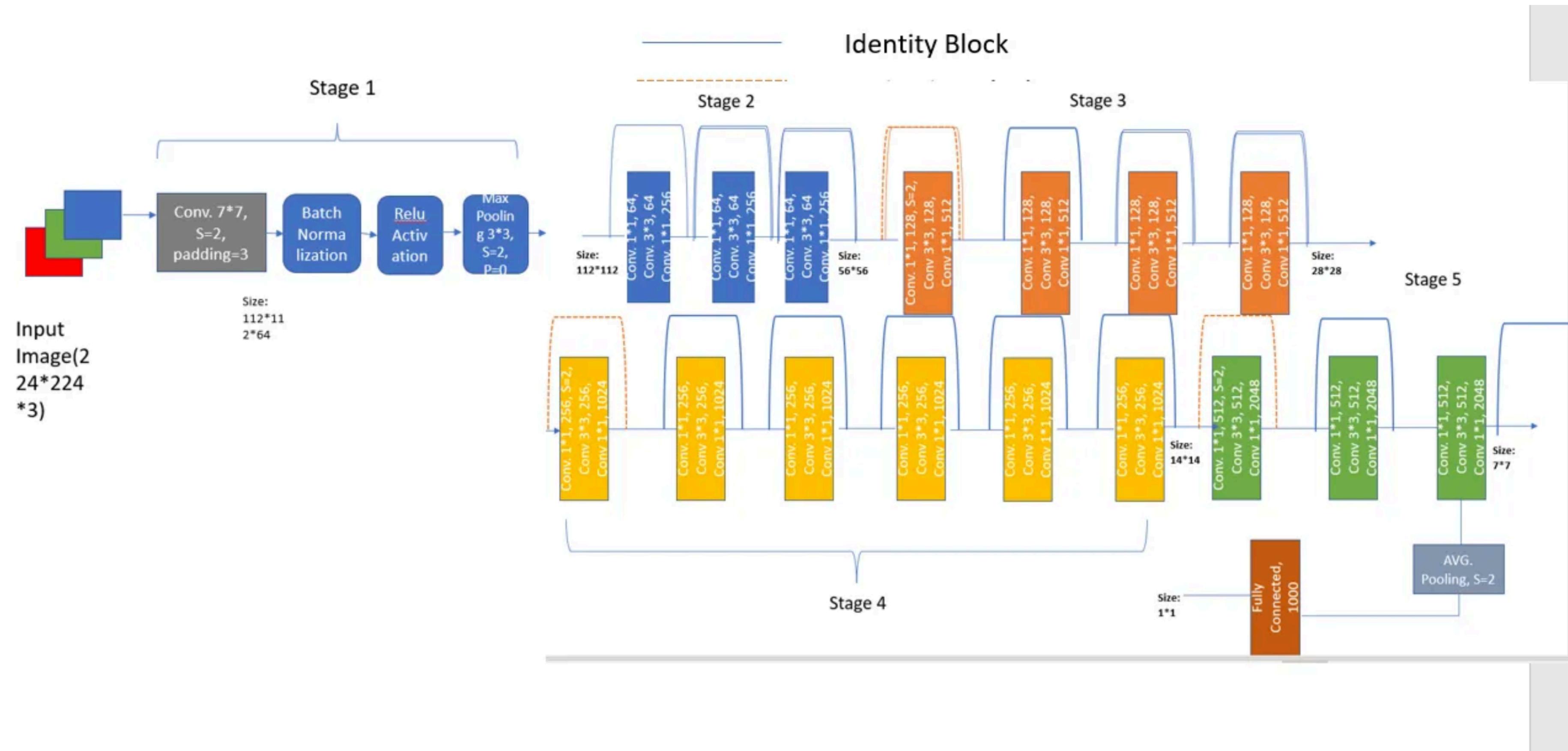
Arquitectura: AlexNet (Krizhevsky et al. 2012) y VGGNet



Referencia

CNN

Arquitectura: ResNet



Referencia