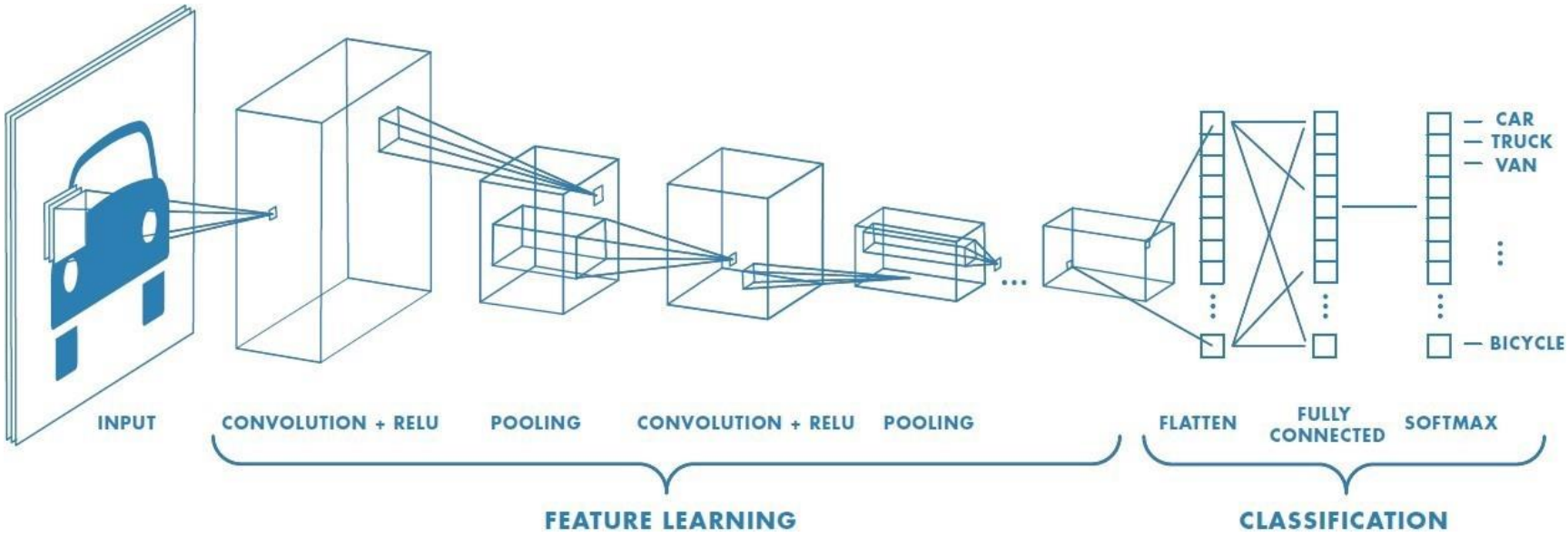
Convolutional Neural Networks

Data Mining

Ester Vidaña Vila

laSalle ENG
Universitat Ramon Llull

Redes neuronales convolucionales (CNN)



Redes neuronales convolucionales (CNN)

Redes neuronales convolucionales (CNN)

Ampliamente utilizadas en el ámbito de análisis y clasificación de imágenes.

Ejemplo: cada capa se entrena a identificar partes de rostros: Bordes, zonas pequeñas de la cara, zonas más grandes…

First Layer Representation | Second Layer Representation | Third Layer Representation

Capas convolucionales

Análisis de píxeles – imagen en escala de grises

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

Imagen: 6x6

x

1	0	-1
1	0	-1
1	0	-1

Filtro (kernel): 3x3

=

-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

Resultado: 4x4

Capas convolucionales

Análisis de píxeles – imagen en escala de grises

Se tiene que aplicar el filtro a cada iteración de la imagen

Luego se realiza la suma de todas las multiplicaciones

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

Imagen: 6x6

x

1	0	-1
1	0	-1
1	0	-1

Filtro (kernel): 3x3

=

-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

Resultado: 4x4

Capas convolucionales

Análisis de píxeles – imagen en escala de grises

Se tiene que aplicar el filtro a cada iteración de la imagen

Luego se realiza la suma de todas las multiplicaciones

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

x

1	0	-1
1	0	-1
1	0	-1

=

-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

Imagen: 6x6 Filtro (kernel): 3x3 Resultado: 4x4

(3 x 1) + (0 x 0) + (1 x -1) + (1 x 1) + (5 x 0) + (8 x -1) + (2 x 1) + (7 x 0) + (2 x -1)

3 + (-1) + 1 + (-8) + 2 + (-2) = -5

Capas convolucionales

Análisis de píxeles – imagen en escala de grises

Se tiene que aplicar el filtro a cada iteración de la imagen

Luego se realiza la suma de todas las multiplicaciones

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

Imagen: 6x6

x

1	0	-1
1	0	-1
1	0	-1

Filtro (kernel): 3x3

=

-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

Resultado: 4x4

Capas convolucionales

Análisis de píxeles – imagen en escala de grises

Se tiene que aplicar el filtro a cada iteración de la imagen

Luego se realiza la suma de todas las multiplicaciones

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

Imagen: 6x6

x

1	0	-1
1	0	-1
1	0	-1

Filtro (kernel): 3x3

=

-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

Resultado: 4x4

Capas convolucionales

Análisis de píxeles – imagen en escala de grises

Se tiene que aplicar el filtro a cada iteración de la imagen

Luego se realiza la suma de todas las multiplicaciones

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

Imagen: 6x6

x

1	0	-1
1	0	-1
1	0	-1

Filtro (kernel): 3x3

=

-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

Resultado: 4x4

Capas convolucionales

Análisis de píxeles – imagen en escala de grises

Se tiene que aplicar el filtro a cada iteración de la imagen

Luego se realiza la suma de todas las multiplicaciones

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

Imagen: 6x6

x

1	0	-1
1	0	-1
1	0	-1

Filtro (kernel): 3x3

=

-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

Resultado: 4x4

Capas convolucionales

Análisis de píxeles – imagen en escala de grises

Se tiene que aplicar el filtro a cada iteración de la imagen

Luego se realiza la suma de todas las multiplicaciones

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

Imagen: 6x6

x

1	0	-1
1	0	-1
1	0	-1

Filtro (kernel): 3x3

=

-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

Resultado: 4x4

Capas convolucionales

Análisis de píxeles – imagen en escala de grises

Se tiene que aplicar el filtro a cada iteración de la imagen

Luego se realiza la suma de todas las multiplicaciones

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

Imagen: 6x6

x

1	0	-1
1	0	-1
1	0	-1

Filtro (kernel): 3x3

=

-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

Resultado: 4x4

Capas convolucionales

Análisis de píxeles – imagen en escala de grises

Se tiene que aplicar el filtro a cada iteración de la imagen

Luego se realiza la suma de todas las multiplicaciones

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

Imagen: 6x6

x

1	0	-1
1	0	-1
1	0	-1

Filtro (kernel): 3x3

=

-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

Resultado: 4x4

Capas convolucionales

Análisis de píxeles – imagen en escala de grises

Se tiene que aplicar el filtro a cada iteración de la imagen

Luego se realiza la suma de todas las multiplicaciones

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

Imagen: 6x6

x

1	0	-1
1	0	-1
1	0	-1

Filtro (kernel): 3x3

=

-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

Resultado: 4x4

Capas convolucionales

Análisis de píxeles – imagen en escala de grises

Se tiene que aplicar el filtro a cada iteración de la imagen

Luego se realiza la suma de todas las multiplicaciones

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

Imagen: 6x6

x

1	0	-1
1	0	-1
1	0	-1

Filtro (kernel): 3x3

=

-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

Resultado: 4x4

Capas convolucionales

Análisis de píxeles – imagen en escala de grises

Se tiene que aplicar el filtro a cada iteración de la imagen

Luego se realiza la suma de todas las multiplicaciones

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

Imagen: 6x6

x

1	0	-1
1	0	-1
1	0	-1

Filtro (kernel): 3x3

=

-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

Resultado: 4x4

Capas convolucionales

Análisis de píxeles – imagen en escala de grises

Se tiene que aplicar el filtro a cada iteración de la imagen

Luego se realiza la suma de todas las multiplicaciones

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

Imagen: 6x6

x

1	0	-1
1	0	-1
1	0	-1

Filtro (kernel): 3x3

=

-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

Resultado: 4x4

Capas convolucionales

Análisis de píxeles – imagen en escala de grises

Se tiene que aplicar el filtro a cada iteración de la imagen

Luego se realiza la suma de todas las multiplicaciones

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

Imagen: 6x6

x

1	0	-1
1	0	-1
1	0	-1

Filtro (kernel): 3x3

=

-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

Resultado: 4x4

Capas convolucionales

Análisis de píxeles – imagen en escala de grises

Se tiene que aplicar el filtro a cada iteración de la imagen

Luego se realiza la suma de todas las multiplicaciones

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

Imagen: 6x6

x

1	0	-1
1	0	-1
1	0	-1

Filtro (kernel): 3x3

=

-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

Resultado: 4x4

Capas convolucionales

Análisis de píxeles – imagen en escala de grises

Se tiene que aplicar el filtro a cada iteración de la imagen

Luego se realiza la suma de todas las multiplicaciones

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

Imagen: 6x6

x

1	0	-1
1	0	-1
1	0	-1

Filtro (kernel): 3x3

=

-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

Resultado: 4x4

Capas convolucionales

Análisis de píxeles – imagen en escala de grises

Se tiene que aplicar el filtro a cada iteración de la imagen

Luego se realiza la suma de todas las multiplicaciones

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

Imagen: 6x6

x

1	0	-1
1	0	-1
1	0	-1

Filtro (kernel): 3x3

=

-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

Resultado: 4x4

Detección de bordes

Hay ciertos Kernels que nos permiten hacer funciones específicas.

Por ejemplo, con el siguiente Kernel podemos encontrar **bordes** verticales.

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

Imagen: 6x6

x

1	0	-1
1	0	-1
1	0	-1

Filtro (kernel): 3x3

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

Resultado: 4x4

Detección de bordes

Hay ciertos Kernels que nos permiten hacer funciones específicas.

Por ejemplo, con el siguiente Kernel podemos encontrar **bordes** verticales.

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

Imagen: 6x6

x

1	0	-1
1	0	-1
1	0	-1

Filtro (kernel): 3x3

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

Resultado: 4x4

Detección de bordes

Hay ciertos Kernels que nos permiten hacer funciones específicas.

Por ejemplo, con el siguiente Kernel podemos encontrar **bordes** verticales.

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

Imagen: 6x6

x

1	0	-1
1	0	-1
1	0	-1

Filtro (kernel): 3x3

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

Resultado: 4x4

Detección de bordes

Hay ciertos Kernels que nos permiten hacer funciones específicas.

Por ejemplo, con el siguiente Kernel podemos encontrar **bordes** verticales.

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

Imagen: 6x6

x

1	0	-1
1	0	-1
1	0	-1

Filtro (kernel): 3x3

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

Resultado: 4x4

Detección de bordes

Hay ciertos Kernels que nos permiten hacer funciones específicas.

Por ejemplo, con el siguiente Kernel podemos encontrar **bordes** verticales.

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

Imagen: 6x6

x

1	0	-1
1	0	-1
1	0	-1

Filtro (kernel): 3x3

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

Resultado: 4x4

Detección de bordes

Hay ciertos Kernels que nos permiten hacer funciones específicas.

Por ejemplo, con el siguiente Kernel podemos encontrar **bordes** verticales.

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

Imagen: 6x6

x

1	0	-1
1	0	-1
1	0	-1

Filtro (kernel): 3x3

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

Resultado: 4x4

Detección de bordes

Hay ciertos Kernels que nos permiten hacer funciones específicas.

Por ejemplo, con el siguiente Kernel podemos encontrar **bordes** verticales.

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

Imagen: 6x6

x

1	0	-1
1	0	-1
1	0	-1

Filtro (kernel): 3x3

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

Resultado: 4x4

Detección de bordes

Hay ciertos Kernels que nos permiten hacer funciones específicas.

Por ejemplo, con el siguiente Kernel podemos encontrar **bordes** verticales.

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

Imagen: 6x6

x

1	0	-1
1	0	-1
1	0	-1

Filtro (kernel): 3x3

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

Resultado: 4x4

Detección de bordes

Hay ciertos Kernels que nos permiten hacer funciones específicas.

Por ejemplo, con el siguiente Kernel podemos encontrar **bordes** verticales.

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

Imagen: 6x6

x

1	0	-1
1	0	-1
1	0	-1

Filtro (kernel): 3x3

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

Resultado: 4x4

Detección de bordes

Hay ciertos Kernels que nos permiten hacer funciones específicas.

Por ejemplo, con el siguiente Kernel podemos encontrar **bordes** verticales.

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

Imagen: 6x6

x

1	0	-1
1	0	-1
1	0	-1

Filtro (kernel): 3x3

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

Resultado: 4x4

Detección de bordes

Hay ciertos Kernels que nos permiten hacer funciones específicas.

Por ejemplo, con el siguiente Kernel podemos encontrar **bordes** verticales.

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

Imagen: 6x6

x

1	0	-1
1	0	-1
1	0	-1

Filtro (kernel): 3x3

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

Resultado: 4x4

Detección de bordes

Hay ciertos Kernels que nos permiten hacer funciones específicas.

Por ejemplo, con el siguiente Kernel podemos encontrar **bordes** verticales.

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

Imagen: 6x6

x

1	0	-1
1	0	-1
1	0	-1

Filtro (kernel): 3x3

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

Resultado: 4x4

Detección de bordes

Hay ciertos Kernels que nos permiten hacer funciones específicas.

Por ejemplo, con el siguiente Kernel podemos encontrar **bordes** verticales.

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

Imagen: 6x6

x

1	0	-1
1	0	-1
1	0	-1

Filtro (kernel): 3x3

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

Resultado: 4x4

Detección de bordes

Hay ciertos Kernels que nos permiten hacer funciones específicas.

Por ejemplo, con el siguiente Kernel podemos encontrar **bordes** verticales.

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

Imagen: 6x6

x

1	0	-1
1	0	-1
1	0	-1

Filtro (kernel): 3x3

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

Resultado: 4x4

Detección de bordes

Hay ciertos Kernels que nos permiten hacer funciones específicas.

Por ejemplo, con el siguiente Kernel podemos encontrar **bordes** verticales.

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

Imagen: 6x6

x

1	0	-1
1	0	-1
1	0	-1

Filtro (kernel): 3x3

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

Resultado: 4x4

Detección de bordes

Hay ciertos Kernels que nos permiten hacer funciones específicas.

Por ejemplo, con el siguiente Kernel podemos encontrar **bordes** verticales.

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

Imagen: 6x6

x

1	0	-1
1	0	-1
1	0	-1

Filtro (kernel): 3x3

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

Resultado: 4x4

Detección de bordes

Análisis de píxels – imagen en escala de grises

Píxels de numeración alta = tonalidad baja

Píxels de numeración bajar = tonalidad oscura

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

Imagen: 6x6

x

1	0	-1
1	0	-1
1	0	-1

Filtro (kernel): 3x3

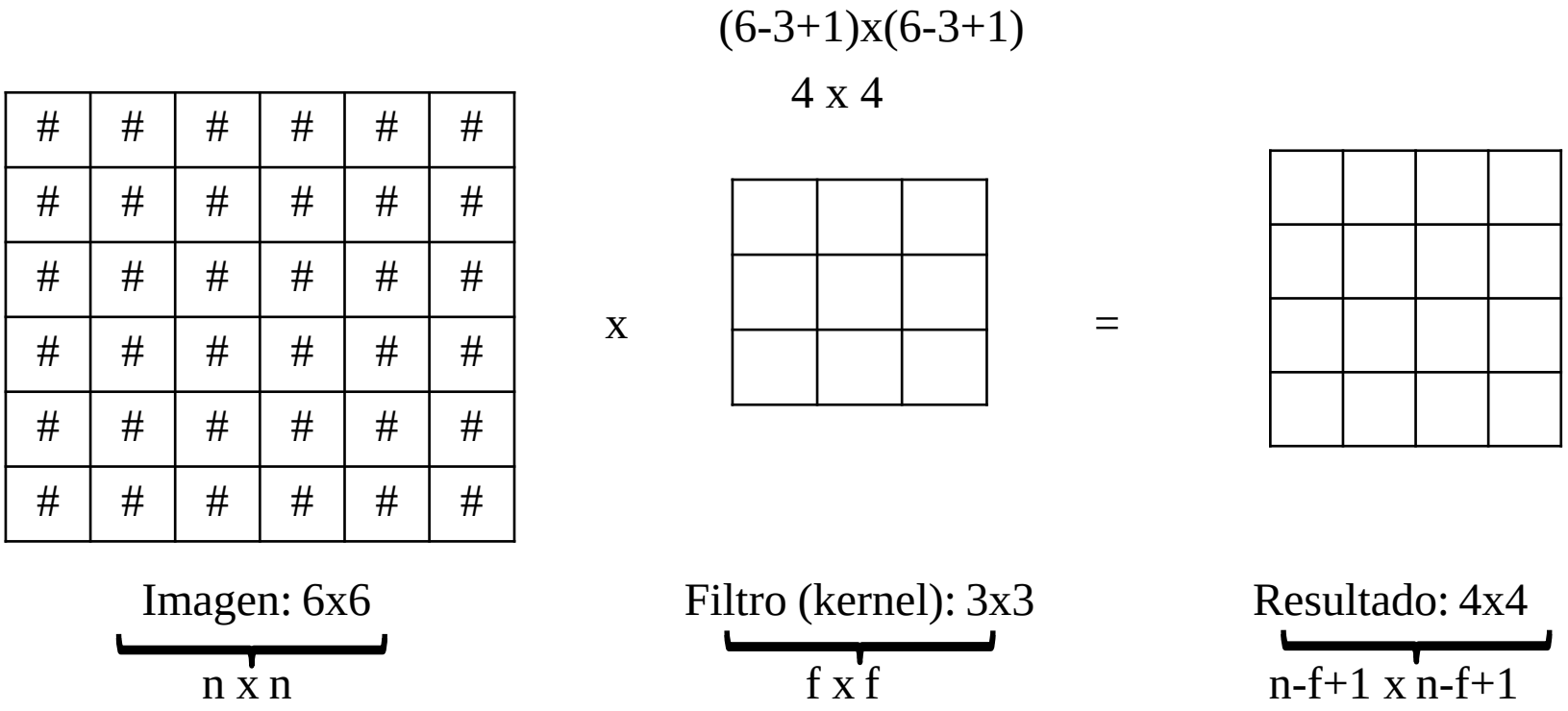
=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

Resultado: 4x4

Padding

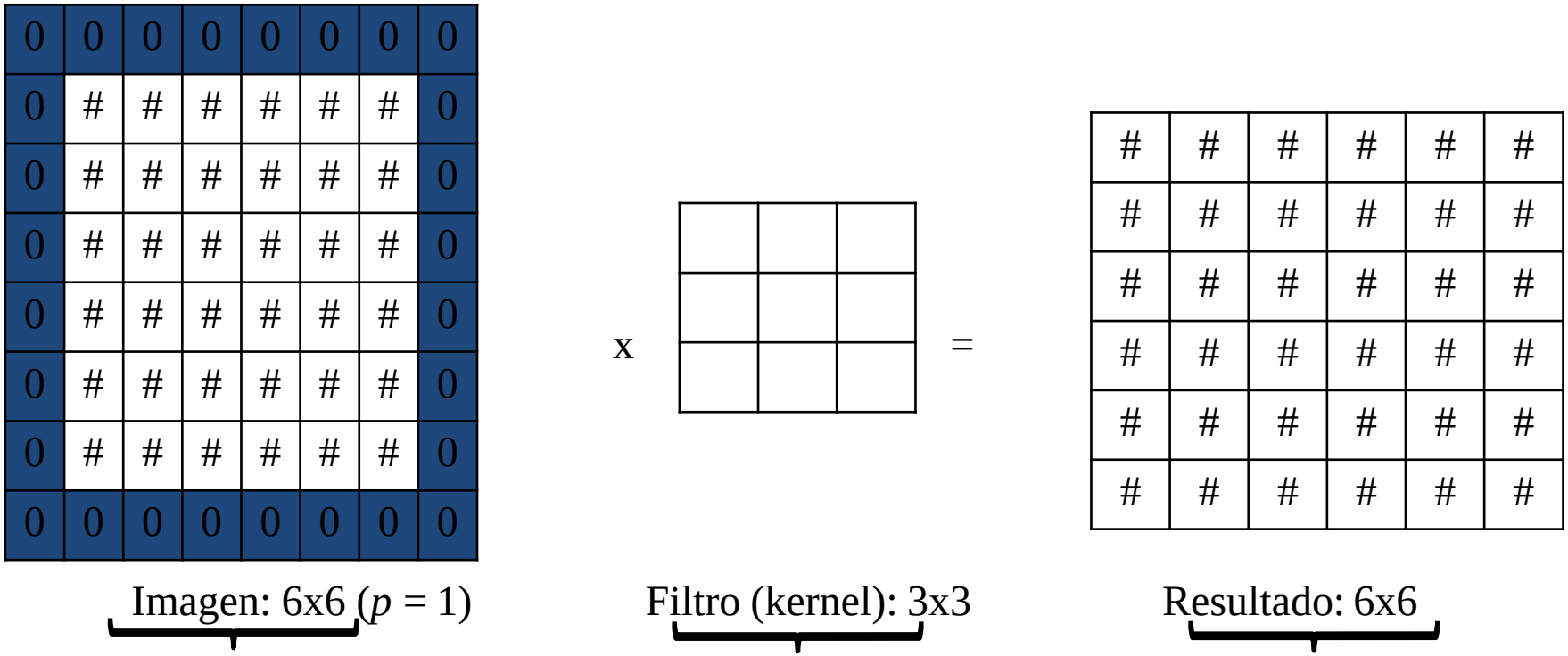
- Concepto de padding: obtener el mismo numero de píxels que en la imagen original.



Padding

Consiste en rellenar con *ceros* el contorno de la imagen original.

De esta forma, obtendremos como salida el mismo tamaño que la imagen inicial.



Stride

Strides - Saltos o zancadas

En vez de desplazarse de elemento en elemento a la derecha se dará un salto de un número dado por el valor del stride.

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

Imagen: 7x7

x

1	2	1
1	2	0
0	1	2

Filtro (kernel): 3x3

=

18	21	24
14	26	14
12	23	17

Resultado: 3x3

Stride

Strides - Saltos o zancadas

En vez de desplazarse de un elemento a la derecha se dará un salto de un numero dado por el valor del stride.

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

Imagen: 7x7

x

1	2	1
1	2	0
0	1	2

Filtro (kernel): 3x3

=

18	21	24
14	26	14
12	23	17

Resultado: 3x3

Stride

Strides - Saltos o zancadas

En vez de desplazarse de un elemento a la derecha se dará un salto de un numero dado por el valor del stride.

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

Imagen: 7x7

x

1	2	1
1	2	0
0	1	2

Filtro (kernel): 3x3

=

18	21	24
14	26	14
12	23	17

Resultado: 3x3

Stride

Strides - Saltos o zancadas

En vez de desplazarse de un elemento a la derecha se dará un salto de un numero dado por el valor del stride.

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

Imagen: 7x7

x

1	2	1
1	2	0
0	1	2

Filtro (kernel): 3x3

=

18	21	24
14	26	14
12	23	17

Resultado: 3x3

Stride

Strides - Saltos o zancadas

En vez de desplazarse de un elemento a la derecha se dará un salto de un numero dado por el valor del stride.

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

Imagen: 7x7

x

1	2	1
1	2	0
0	1	2

Filtro (kernel): 3x3

=

18	21	24
14	26	14
12	23	17

Resultado: 3x3

Stride

Strides - Saltos o zancadas

En vez de desplazarse de un elemento a la derecha se dará un salto de un numero dado por el valor del stride.

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

Imagen: 7x7

x

1	2	1
1	2	0
0	1	2

Filtro (kernel): 3x3

=

18	21	24
14	26	14
12	23	17

Resultado: 3x3

Stride

Strides - Saltos o zancadas

En vez de desplazarse de un elemento a la derecha se dará un salto de un numero dado por el valor del stride.

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

Imagen: 7x7

x

1	2	1
1	2	0
0	1	2

Filtro (kernel): 3x3

=

18	21	24
14	26	14
12	23	17

Resultado: 3x3

Stride

Strides - Saltos o zancadas

En vez de desplazarse de un elemento a la derecha se dará un salto de un numero dado por el valor del stride.

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

Imagen: 7x7

x

1	2	1
1	2	0
0	1	2

Filtro (kernel): 3x3

=

18	21	24
14	26	14
12	23	17

Resultado: 3x3

Stride

Strides - Saltos o zancadas

En vez de desplazarse de un elemento a la derecha se dará un salto de un numero dado por el valor del stride.

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

Imagen: 7x7

x

1	2	1
1	2	0
0	1	2

Filtro (kernel): 3x3

=

18	21	24
14	26	14
12	23	17

Resultado: 3x3

Stride

Strides - Saltos o zancadas

En vez de desplazarse de un elemento a la derecha se dará un salto de un numero dado por el valor del stride.

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

Imagen: 7x7

x

1	2	1
1	2	0
0	1	2

Filtro (kernel): 3x3

=

18	21	24
14	26	14
12	23	17

Resultado: 3x3

Imágenes RGB

Análisis de píxels – imagen en RGB (0 a 255)

filas x columnas x canales (planos) – mismo numero de canales entre imagen y filtro

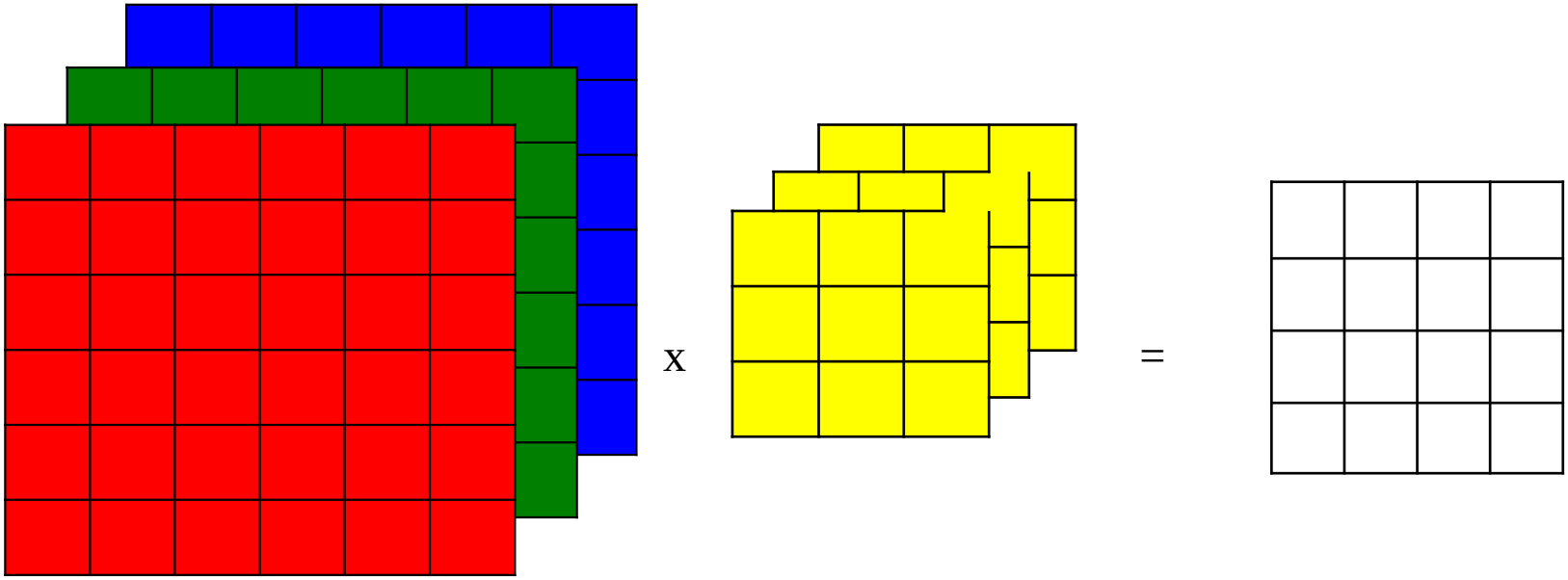


Imagen: 6x6x3 Filtro (kernel): 3x3x3 Resultado: 4x4

Imágenes RGB

Análisis de píxels – imagen en RGB (0 a 255)

imagen resultante con un solo canal

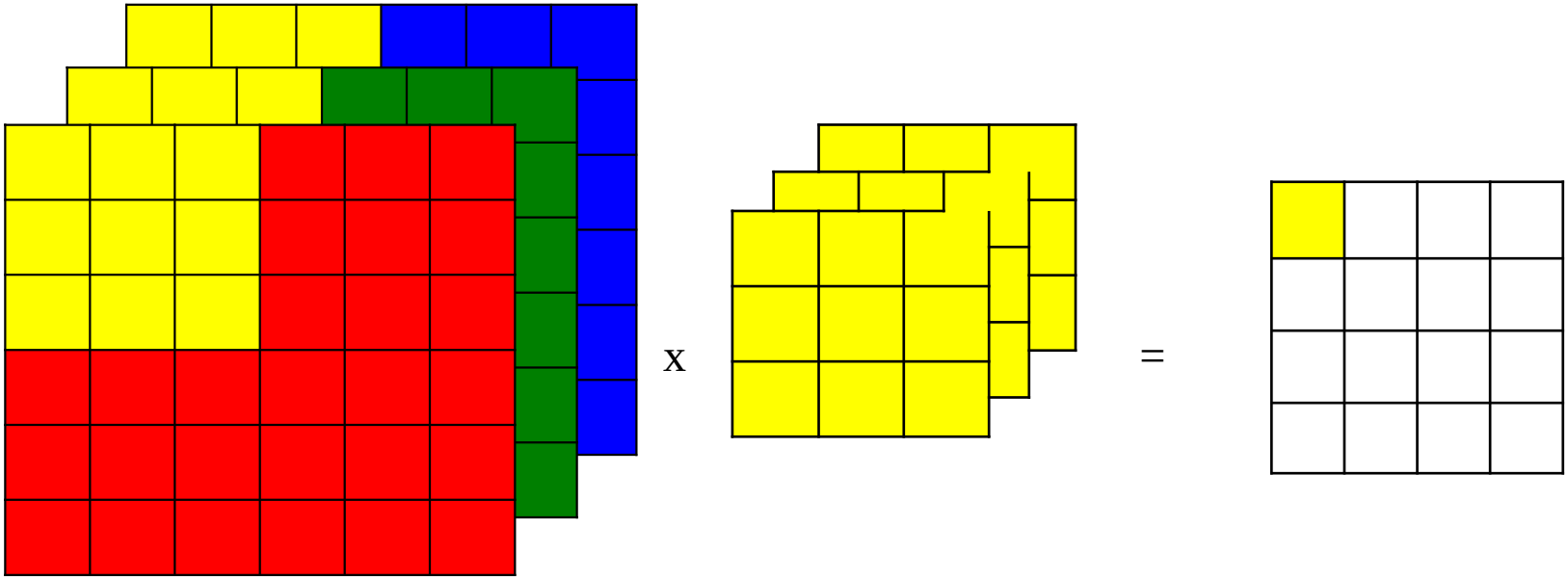


Imagen: 6x6x3 Filtro (kernel): 3x3x3 Resultado: 4x4

Imágenes RGB

Análisis de píxels – imagen en RGB (0 a 255)

imagen resultante con un solo canal – primero resultado: suma de las 27 multiplicaciones

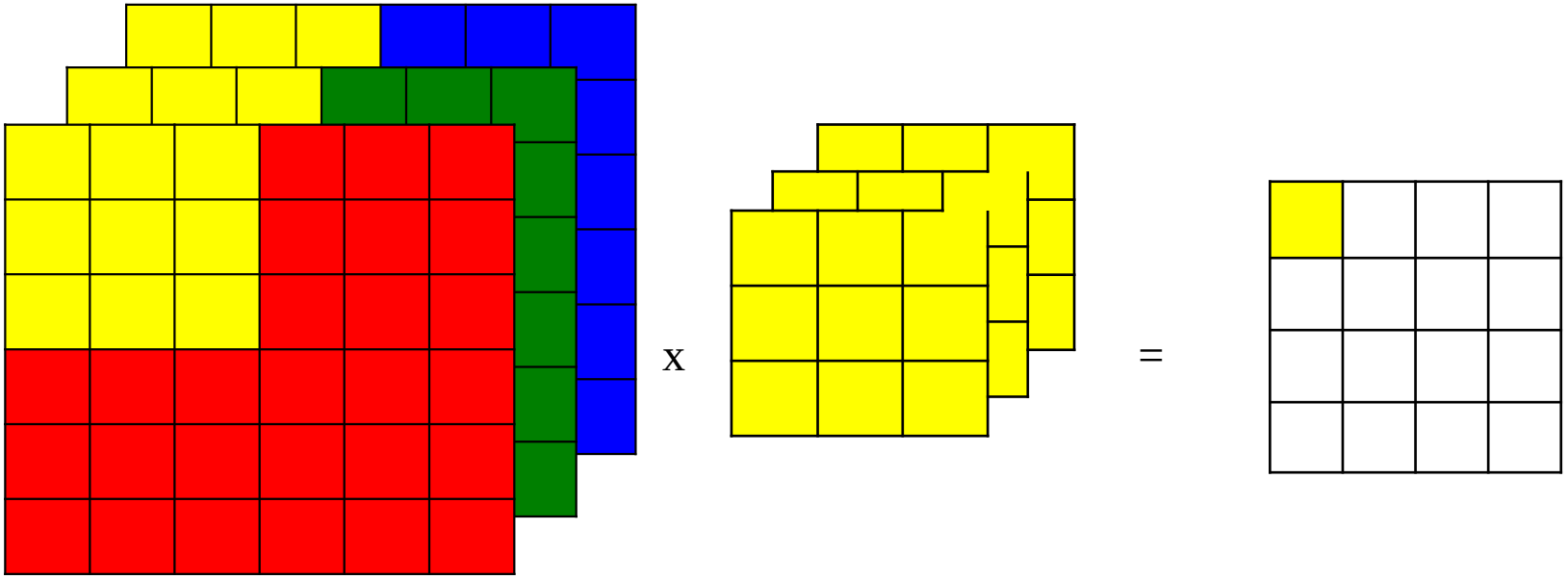


Imagen: 6x6x3 Filtro (kernel): 3x3x3 Resultado: 4x4

Max pooling

Análisis de píxels – imagen en escala de grises – Filtro Max pooling

5	3	4	1
2	7	9	2
8	1	5	3
9	7	2	1

Imagen: 4x4

Max pooling

Análisis de píxels – imagen en escala de grises – Filtro Max pooling

5	3	4	1
2	7	9	2
8	1	5	3
9	7	2	1

Imagen: 4x4

hiperparámetros

Filtro = 2x2
Stride : 2

Max pooling

Análisis de píxels – imagen en escala de grises – Filtro Max pooling.

De los valores que están dentro del filtro, se escoge el valor más grande (7).

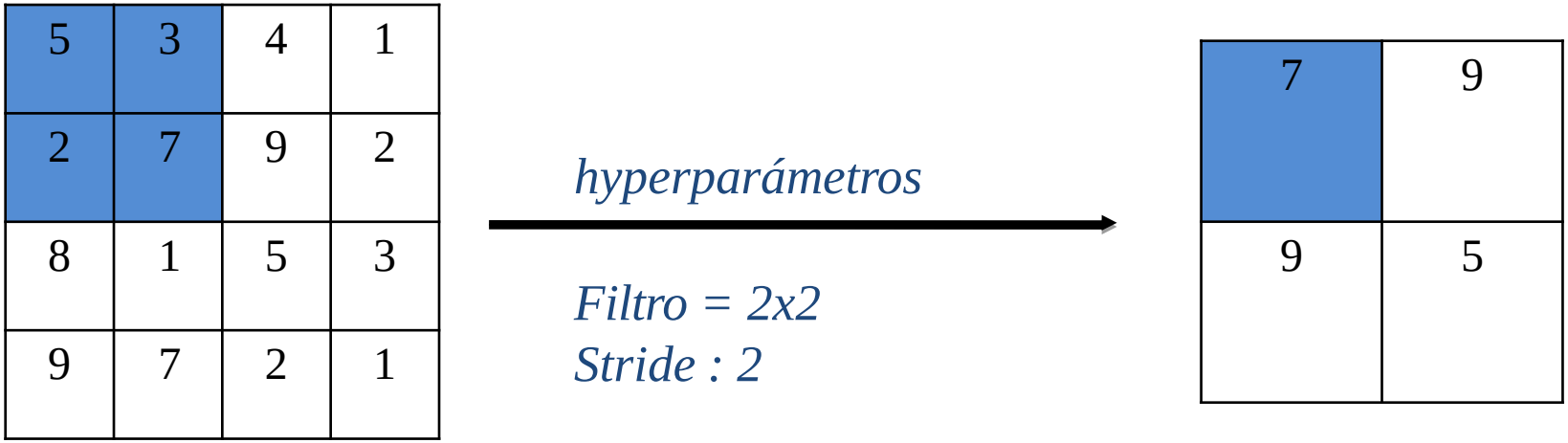


Imagen: 4x4

Resultado: 2x2

Max pooling

Análisis de píxels – imagen en escala de grises – Filtro Max pooling.

De los valores que están dentro del filtro, se escoge el valor más grande (9).

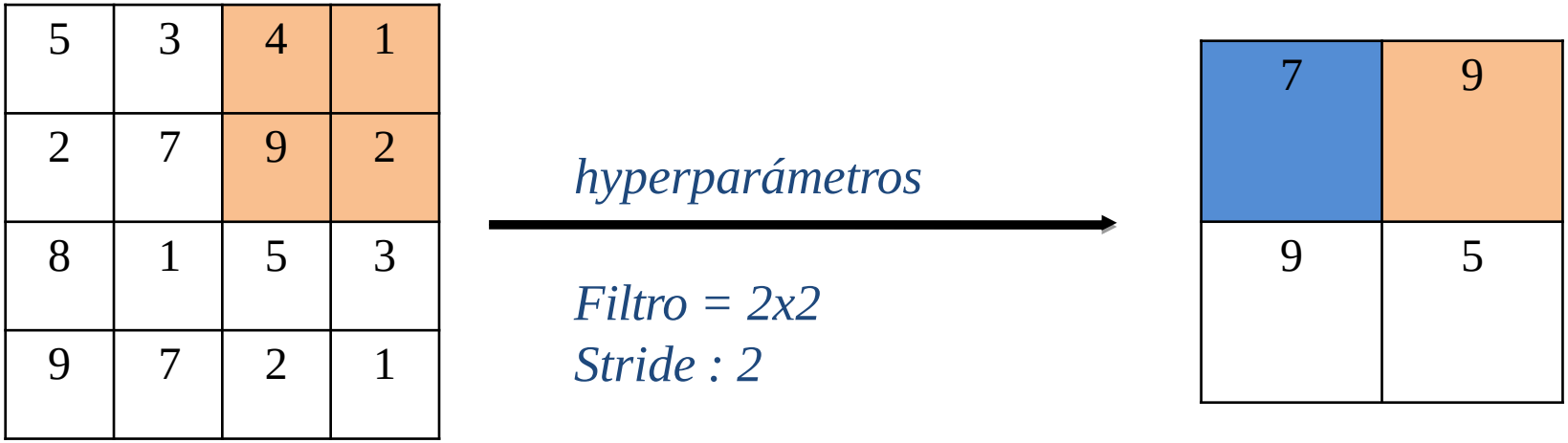


Imagen: 4x4

Resultado: 2x2

Max pooling

Análisis de píxels – imagen en escala de grises – Filtro Max pooling.

De los valores que están dentro del filtro, se escoge el valor más grande (9).

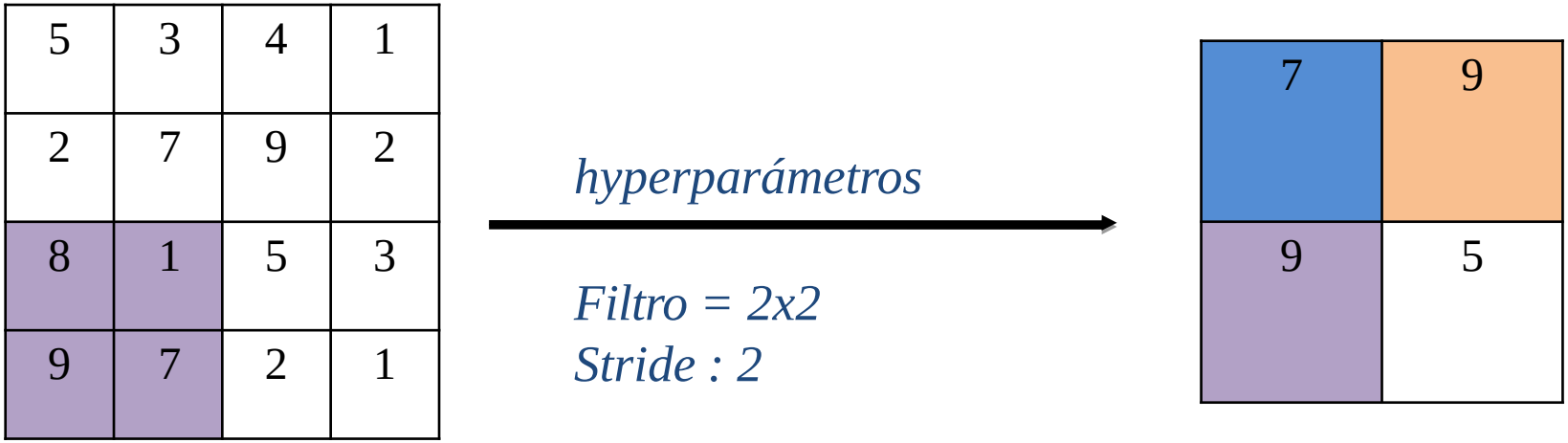


Imagen: 4x4

Resultado: 2x2

Max pooling

Análisis de píxels – imagen en escala de grises – Filtro Max pooling

De los valores que están dentro del filtro, se escoge el valor más grande (5).

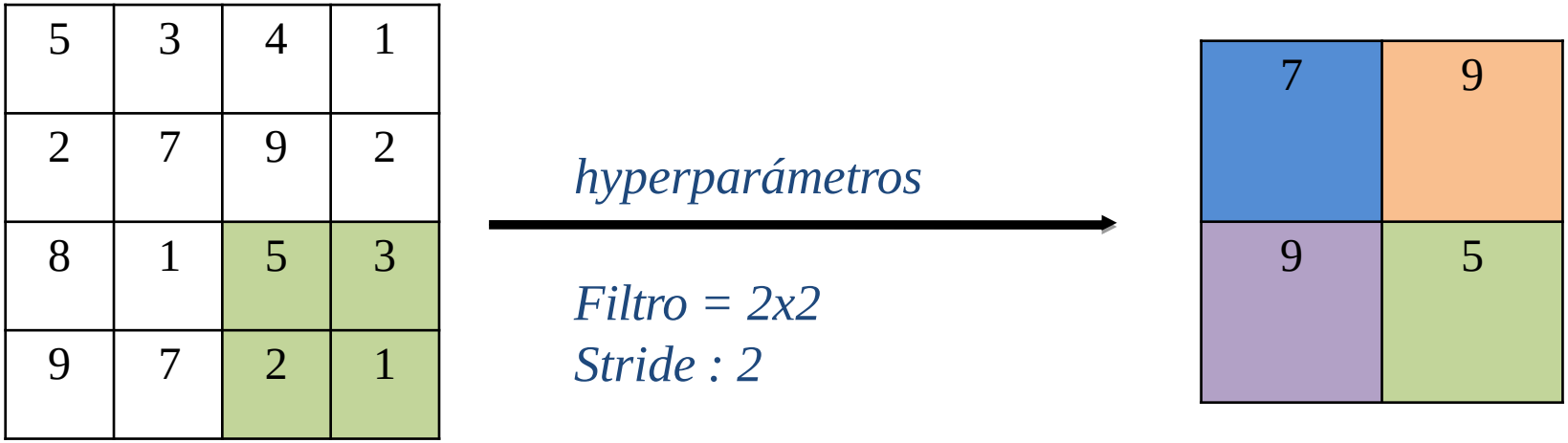


Imagen: 4x4

Resultado: 2x2

Max pooling

Análisis de pixeles – imagen en escala de grises – Filtro Max pooling

Dentro de una imagen, hablando en términos de píxels, los valores altos son los que tienen más relevancia.

El max pooling preserva estos valores.

5	3	4	1
2	7	9	2
8	1	5	3
9	7	2	1

Imagen: 4x4

hyperparámetros

Filtro = 2x2
Stride : 2

7	9
9	5

Resultado: 2x2

Otras técnicas de pooling

También existent los términos de:

- Min pooling: nos quedamos con el valor más bajo. Así, nos quedamos con los píxels más oscuros. Por contra, con Max Pool nos quedaríamos con los más brillantes.
- Average pooling: nos quedamos con el valor promedio. Efecto *smooth*.

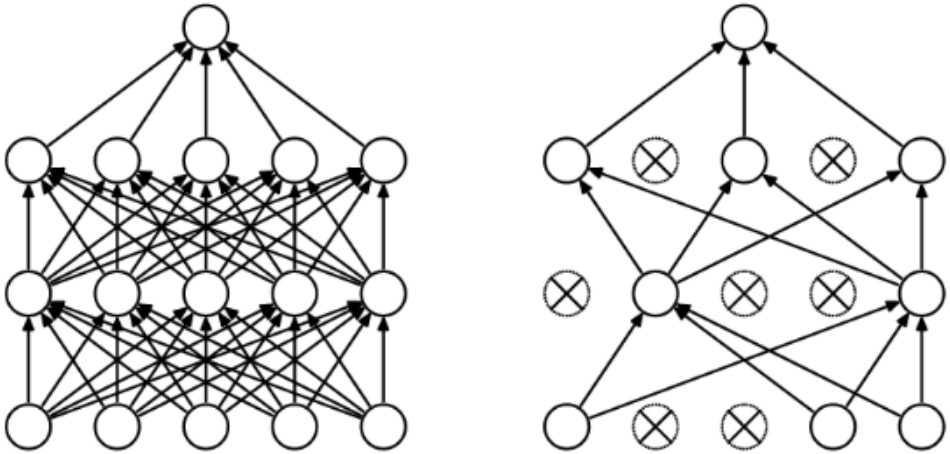
https://medium.com/@bdhuma/which-pooling-method-is-better-maxpooling-vs-minpooling-vs-average-pooling-95fb03f45a9

Regularización: dropout

El término **regularización** se refiere a la aplicación de técnicas para calibrar los modelos de machine learning y prevenir el overfitting o underfitting.

Un método para regularizar redes neuronales es la aplicación de capas de **dropout**.

Dropout: se refiere a eliminar nodos de la red de forma temporal, creando así una nueva arquitectura.



(a) Standard Neural Net (b) After applying dropout.

Paper explicativo: https://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf

Dropout

Los nodos van a cortarse temporalmente utilizando una probabilidad p.

Así pues, si en una capa interna tenemos 100 neuronas y definimos una probabilidad del 0.2, habría 20 neuronas que quedarían desactivadas (sacarían un resultado de 0).

GIF para ejemplificar:
https://mohcinemadkour.github.io/posts/2020/04/Deep%20Learning%20Regularization/

Dropout y overfitting

Recordatorio:

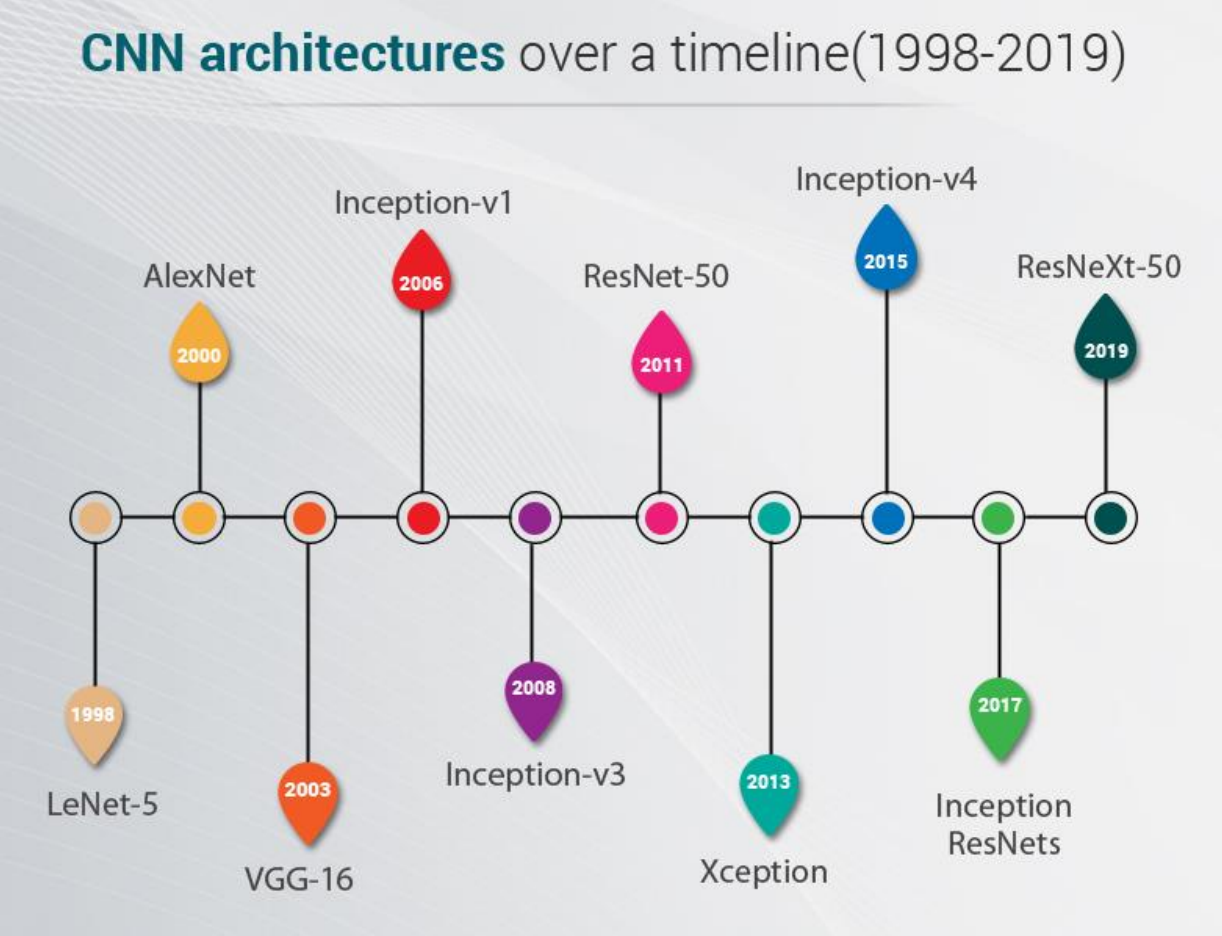
El *overfitting* ocurre cuando el modelo aprende del ruido estadístico que hay en los datos de entrada, dado que las neuronas intentan minimizar la función de pérdida global. Esto hace que haya neuronas que intenten compensar los errores de otras neuronas.

Con dropout, la red tiene que aprender a distinguir las *features* más importantes sin confiar en el resto de neuronas, ya que con los cortes aleatorios de neuronas, habrá una red nueva a entrenar en cada paso del entrenamiento.

Podemos verlo como un *ensemble*: un grupo de redes con arquitecturas distintas.

- **Importante**: dropout se utiliza únicamente para entrenar la red, pero no se utiliza para inferencia.

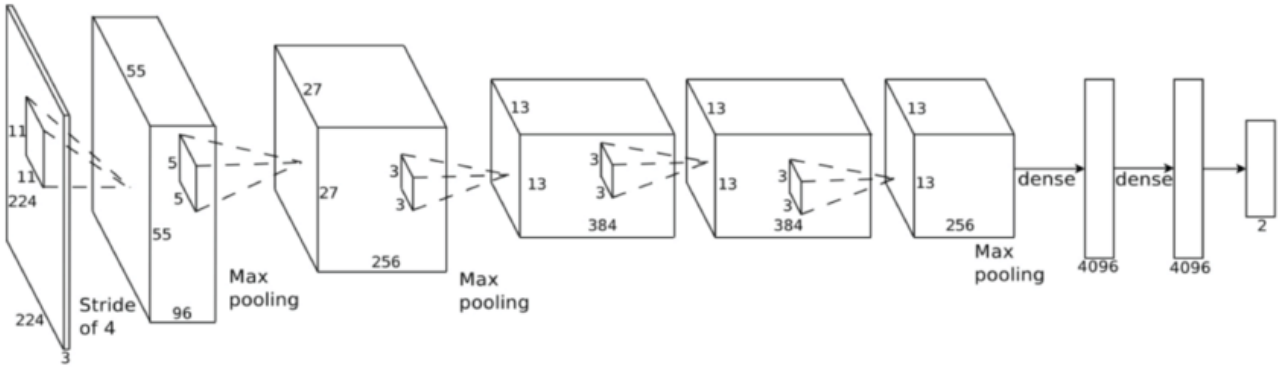
Ejemplo de CNNs



https://www.analyticsvidhya.com/blog/2020/10/what-is-the-convolutional-neural-network-architecture/

Ejemplo de CNNs

- AlexNet:



- VGG16: