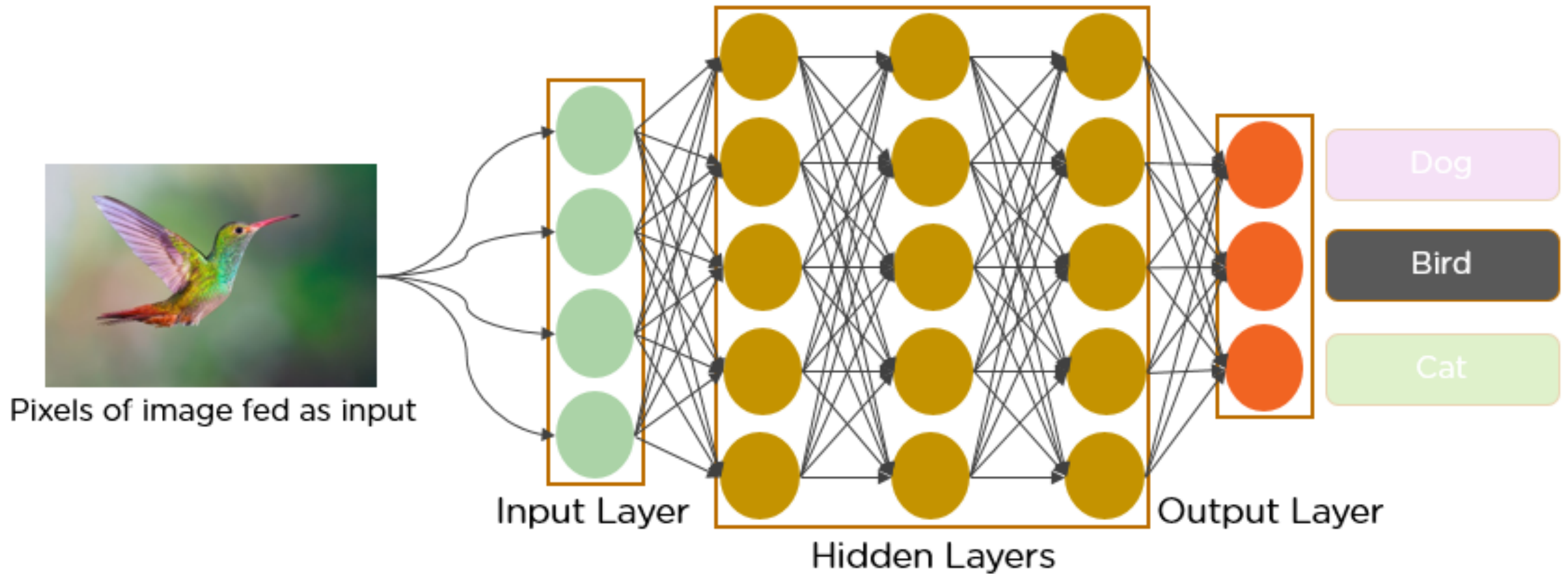


Clasificación de imágenes



Red Neuronal Convolutcional

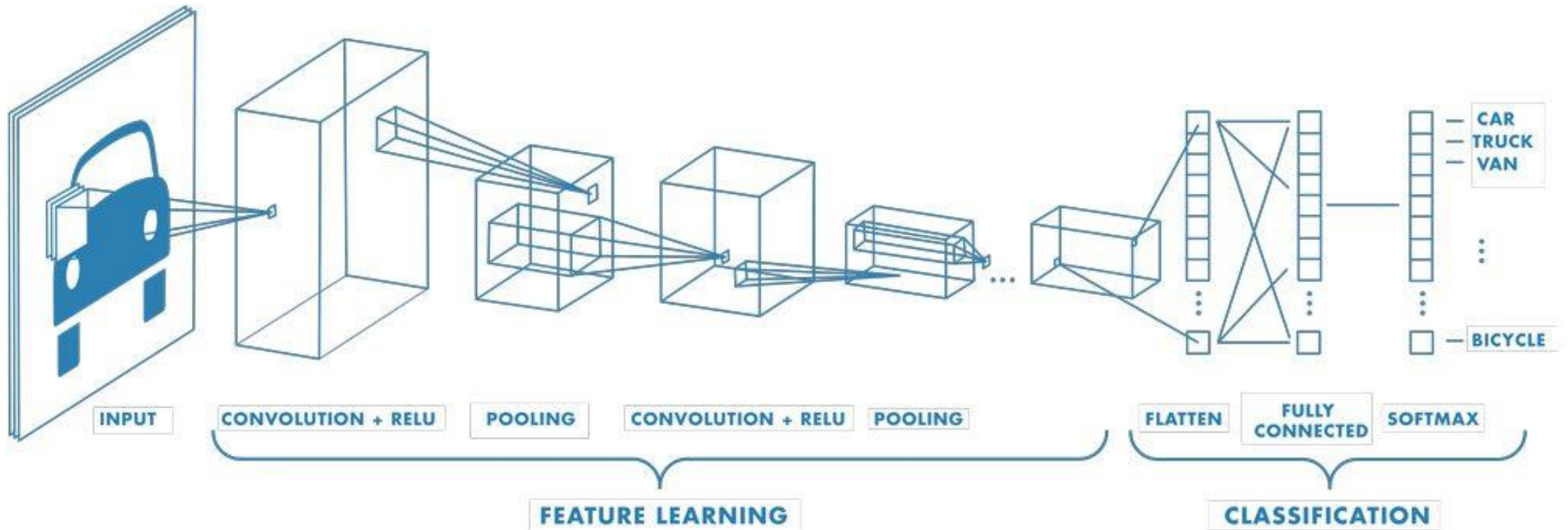


Imagen digital

- Está representada por una matriz de $M \times N$ pixels.
- Cada pixel representa la intensidad de luz en ese punto.
- Su valor dependerá del tipo de imagen

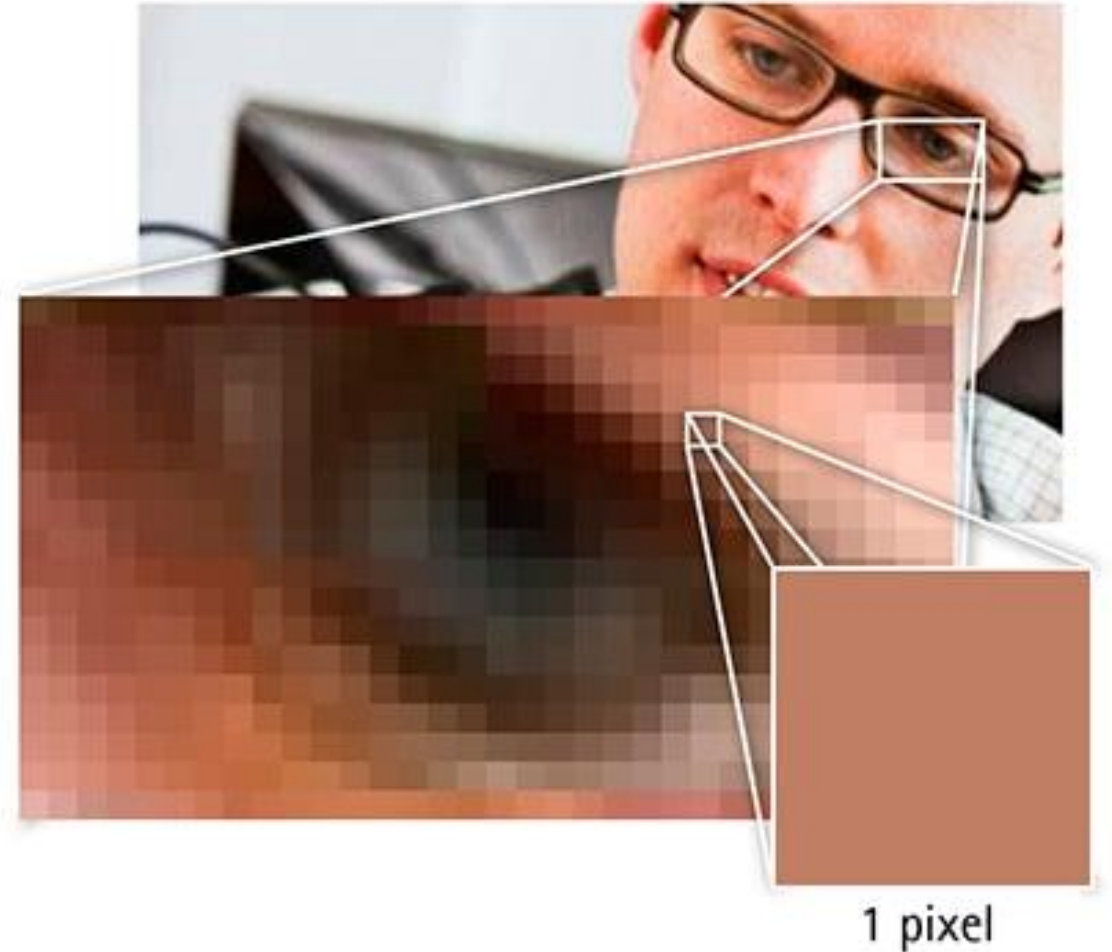


Imagen en tonos de grises



Greyscale image

		112		
	112	105	106	
105	105	105	105	112
105	112	112	105	112
112	106	106	112	105
105	112	112	112	106
106	112	116	117	105
120	123	105	105	10
	127	127	106	
	127	127	10	

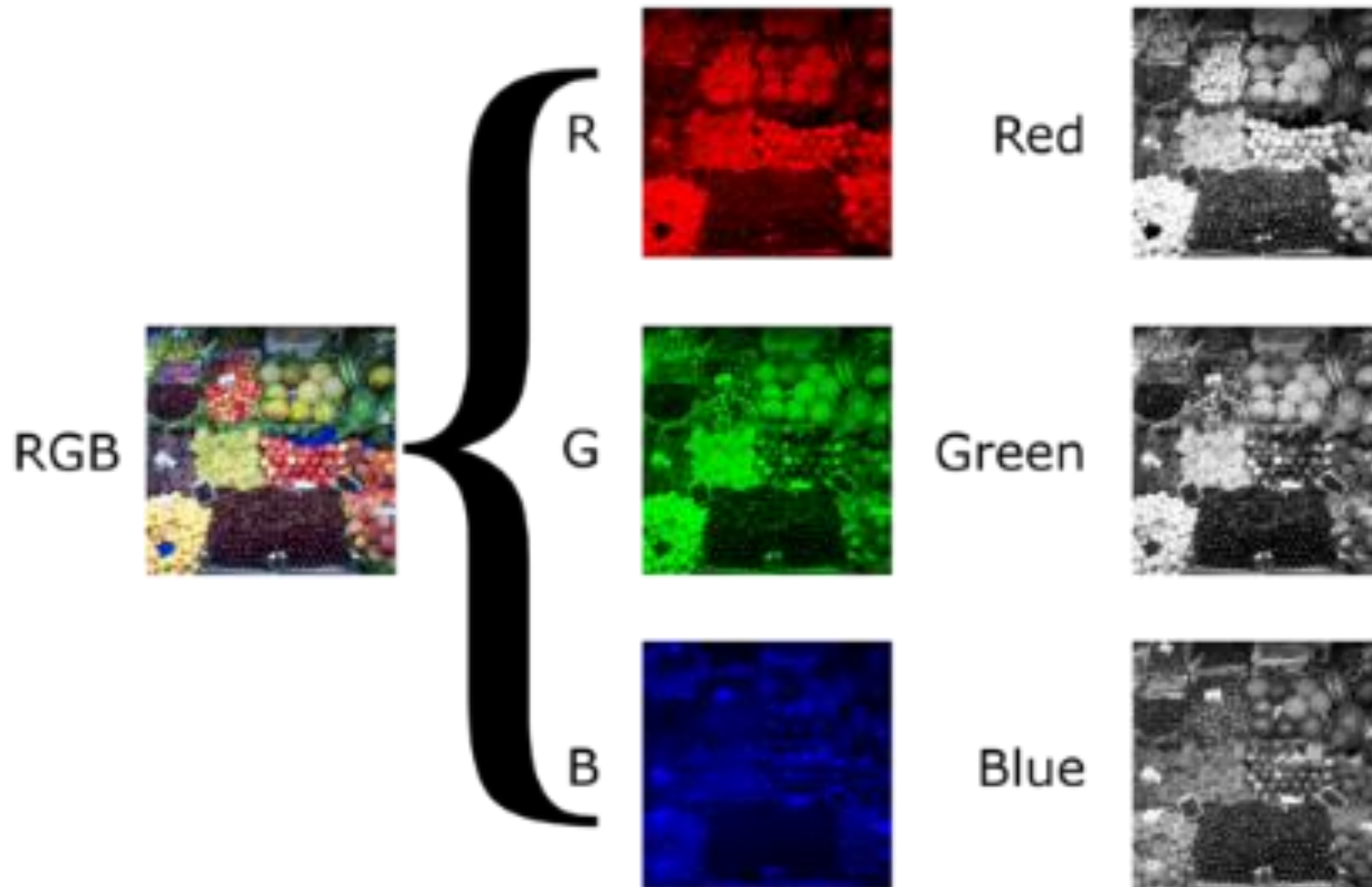
Pixel/intensity value

Imagen en tonos de grises



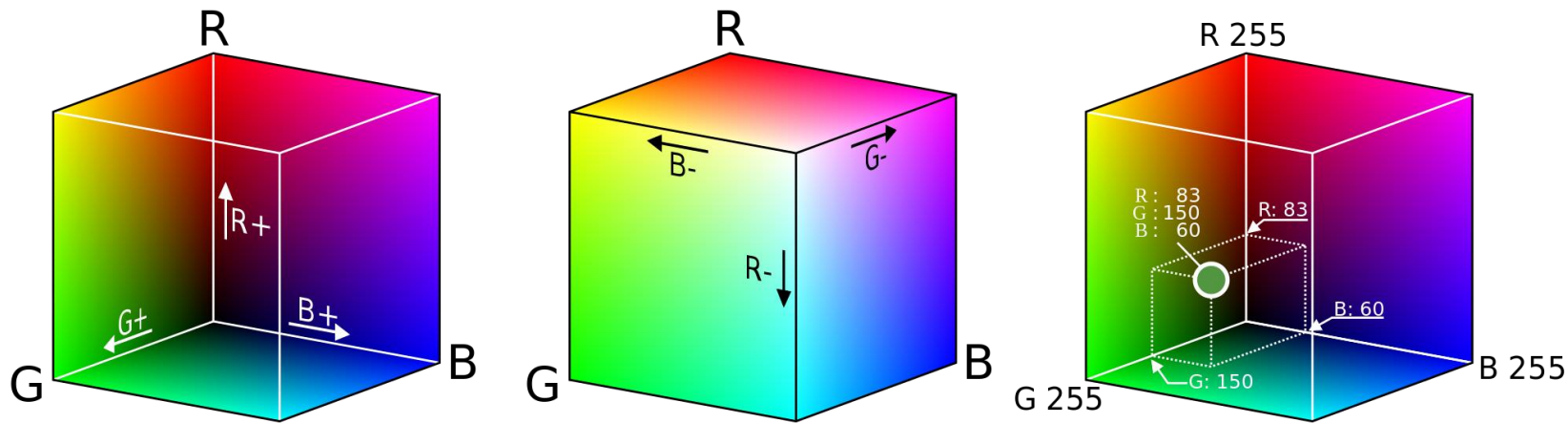
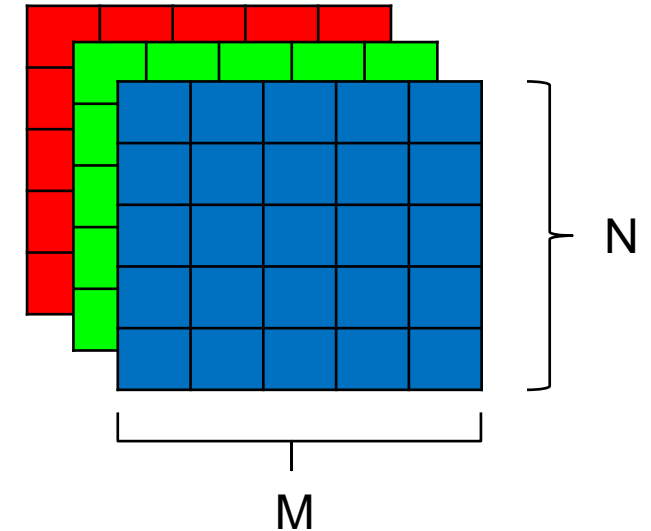
0	2	15	0	0	11	10	0	0	0	0	9	9	0	0	0
0	0	0	4	60	157	236	255	255	177	95	61	32	0	0	29
0	10	16	119	238	255	244	245	243	250	249	255	222	103	10	0
0	14	170	255	255	244	254	255	253	245	255	249	253	251	124	1
2	98	255	228	255	251	254	211	141	116	122	215	251	238	255	49
13	217	243	255	155	33	226	52	2	0	10	13	232	255	255	36
16	229	252	254	49	12	0	0	7	7	0	70	237	252	235	62
6	141	245	255	212	25	11	9	3	0	115	236	243	255	137	0
0	87	252	250	248	215	60	0	1	121	252	255	248	144	6	0
0	13	113	255	255	245	255	182	181	248	252	242	208	36	0	19
1	0	5	117	251	255	241	255	247	255	241	162	17	0	7	0
0	0	0	4	58	251	255	246	254	253	255	120	11	0	1	0
0	0	4	97	255	255	255	248	252	255	244	255	182	10	0	4
0	22	205	252	246	251	241	100	24	113	255	245	255	194	9	0
0	111	255	242	255	158	24	0	0	6	39	255	232	230	56	0
0	218	251	250	137	7	11	0	0	0	2	62	255	250	125	3
0	173	255	255	101	9	20	0	13	3	13	182	251	245	61	0
0	107	251	241	255	230	98	55	19	118	217	248	253	255	52	4
0	18	146	250	255	247	255	255	255	249	255	240	255	129	0	5
0	0	23	113	215	255	250	248	255	255	248	248	118	14	12	0
0	0	6	1	0	52	153	233	255	252	147	37	0	0	4	1
0	0	5	5	0	0	0	0	0	14	1	0	6	6	0	0

Imagen color - RGB



Modelo RGB

- Una imagen se representa mediante 3 matrices de $N \times M$, una para cada canal.
- A mayor valor en los 3 canales los colores se aclaran.



Visualización

```
from skimage import io
imgColor = io.imread("hoja.jpg")
plt.figure()
plt.imshow (imgColor)
plt.show()
```

```
C = ['Reds', 'Greens', 'Blues']
for p in [0,1,2]:
    plt.figure()
    plt.imshow (imgColor[:, :, p], cmap=C[p])
    plt.show()
```

verImagen_RGB.ipynb



RGB a escala de grises

- Puede combinarse la información de los 3 canales para producir una imagen en tonos de grises

$$\text{GRIS} = (R + G + B) / 3$$



RGB a escala de grises

- Hay otras conversiones que reflejan mejor la percepción del ojo humano

$$\text{GRIS} = (0.3 R + 0.59 G + 0.11 B)$$



Cargando una imagen

```
from skimage import io  
imgColor = io.imread("tigre.jpg")
```



(150, 134, 3)

```
from skimage import io  
imgGris = io.imread("tigre.jpg",  
                    as_gray=True)
```



(150, 134)

Filtros en el dominio espacial

- En este proceso se relaciona un conjunto de píxeles próximos al píxel objetivo con la finalidad de obtener una información útil



*Usaremos máscaras o kernels de **convolución***

Convolución 2D

- La operación de convolución de una imagen con un filtro o kernel permite destacar ciertas características de dicha imagen.



-1	-1	-1
-1	8	-1
-1	-1	-1



Convolución 2D

- Filtro de detección de bordes horizontal



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



Convolución 2D

- Filtro de detección de bordes diagonal



-1	0
0	1



Convolución 2D

- Filtro de detección de bordes diagonal



0	-1
1	0

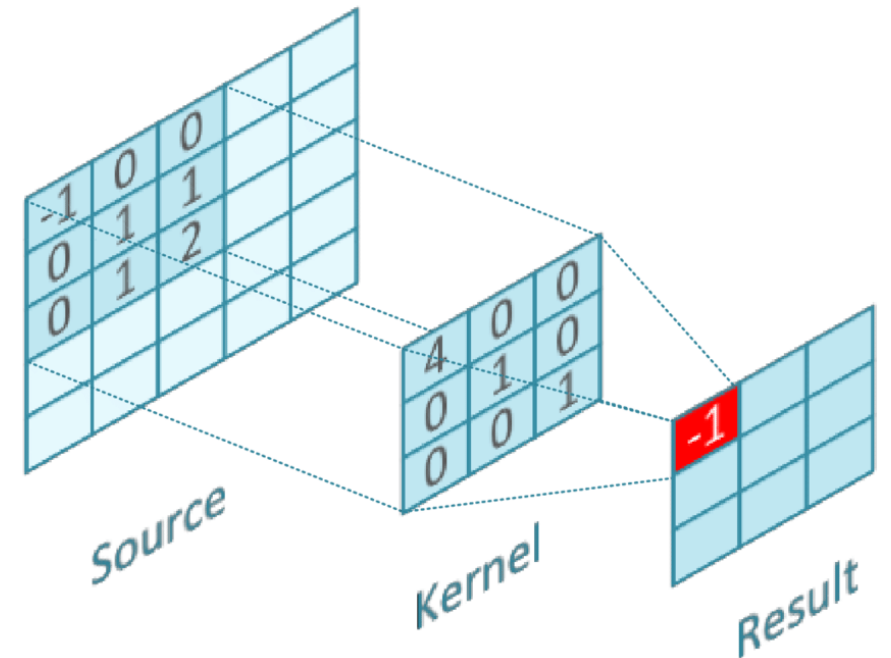


Convolución 2D

- La convolución discreta de dos funciones f y g se define como

$$(f * g)[x, y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1, n_2] \cdot g[x - n_1, y - n_2]$$

- La función g se desplaza antes de multiplicar.



Convolución 2D

Entrada

60	113	56	139	85
73	121	54	84	128
131	99	70	129	127
80	57	115	69	134
104	126	123	95	130

Kernel

0	-1	0
-1	5	-1
0	-1	0

Salida

	266			

$$\begin{aligned} &60 * 0 + 113 * (-1) + 56 * 0 + \\ &73 * (-1) + 121 * 5 + 54 * (-1) + \\ &131 * 0 + 99 * (-1) + 70 * 0 = 266 \end{aligned}$$

Convolución 2D

Entrada

60	113	56	139	85
73	121	54	84	128
131	99	70	129	127
80	57	115	69	134
104	126	123	95	130

Kernel

0	-1	0
-1	5	-1
0	-1	0

Salida

	266	-61		

$$\begin{aligned} &113 * 0 + 56 * (-1) + 139 * 0 + \\ &121 * (-1) + 54 * 5 + 84 * (-1) + \\ &99 * 0 + 70 * (-1) + 129 * 0 = -61 \end{aligned}$$

Convolución 2D

Entrada

60	113	56	139	85
73	121	54	84	128
131	99	70	129	127
80	57	115	69	134
104	126	123	95	130

Kernel

0	-1	0
-1	5	-1
0	-1	0

Salida

	266	-61	-30	

$$\begin{aligned} &56 * 0 + 139 * (-1) + 85 * 0 + \\ &54 * (-1) + 84 * 5 + 128 * (-1) + \\ &70 * 0 + 129 * (-1) + 127 * 0 = -30 \end{aligned}$$

Convolución 2D

Entrada

60	113	56	139	85
73	121	54	84	128
131	99	70	129	127
80	57	115	69	134
104	126	123	95	130

Kernel

0	-1	0
-1	5	-1
0	-1	0

Salida

	266	-61	-30	
	116			

$$\begin{aligned} &73 * 0 + 121 * (-1) + 54 * 0 + \\ &131 * (-1) + 99 * 5 + 70 * (-1) + \\ &80 * 0 + 57 * (-1) + 115 * 0 = 116 \end{aligned}$$

Convolución 2D

Entrada

60	113	56	139	85
73	121	54	84	128
131	99	70	129	127
80	57	115	69	134
104	126	123	95	130

Kernel

0	-1	0
-1	5	-1
0	-1	0

Salida

	266	-61	-30	
	116	-47		

$$\begin{aligned} &121 * 0 + 54 * (-1) + 84 * 0 + \\ &99 * (-1) + 70 * 5 + 129 * (-1) + \\ &57 * 0 + 115 * (-1) + 69 * 0 = -47 \end{aligned}$$

Convolución 2D

Entrada

60	113	56	139	85
73	121	54	84	128
131	99	70	129	127
80	57	115	69	134
104	126	123	95	130

Kernel

0	-1	0
-1	5	-1
0	-1	0

Salida

	266	-61	-30	
	116	-47	295	

$$\begin{aligned} & 54 * 0 + 84 * (-1) + 128 * 0 + \\ & 70 * (-1) + 129 * 5 + 127 * (-1) + \\ & 115 * 0 + 69 * (-1) + 134 * 0 = 295 \end{aligned}$$

Convolución 2D

Entrada

60	113	56	139	85
73	121	54	84	128
131	99	70	129	127
80	57	115	69	134
104	126	123	95	130

Kernel

0	-1	0
-1	5	-1
0	-1	0

Salida

	266	-61	-30	
	116	-47	295	
	-135			

$$\begin{aligned} &131 * 0 + 99 * (-1) + 70 * 0 + \\ &80 * (-1) + 57 * 5 + 115 * (-1) + \\ &104 * 0 + 126 * (-1) + 123 * 0 = -135 \end{aligned}$$

Convolución 2D

Entrada

60	113	56	139	85
73	121	54	84	128
131	99	70	129	127
80	57	115	69	134
104	126	123	95	130

Kernel

0	-1	0
-1	5	-1
0	-1	0

Salida

	266	-61	-30	
	116	-47	295	
	-135	256		

$$\begin{aligned} & 99 * 0 + 70 * (-1) + 129 * 0 + \\ & 57 * (-1) + 115 * 5 + 69 * (-1) + \\ & 126 * 0 + 123 * (-1) + 95 * 0 = 256 \end{aligned}$$

Convolución 2D

Entrada

60	113	56	139	85
73	121	54	84	128
131	99	70	129	127
80	57	115	69	134
104	126	123	95	130

Kernel

0	-1	0
-1	5	-1
0	-1	0

Salida

	266	-61	-30	
	116	-47	295	
	-135	256	-128	

$$\begin{aligned} &70 * 0 + 129 * (-1) + 127 * 0 + \\ &115 * (-1) + 69 * 5 + 134 * (-1) + \\ &123 * 0 + 95 * (-1) + 130 * 0 = -128 \end{aligned}$$

Convolución 2D

Entrada

60	113	56	139	85
73	121	54	84	128
131	99	70	129	127
80	57	115	69	134
104	126	123	95	130

Kernel

0	-1	0
-1	5	-1
0	-1	0

Salida

	266	-61	-30	
	116	-47	295	
	-135	256	-128	

□ Parámetros

▣ **Kernel_size:** tamaño del filtro o kernel. En este caso = 3

▣ **Stride:** desplazamiento del filtro cada vez que se aplica. En este caso = 1

Convolución 2D

Entrada

60	113	56	139	85
73	121	54	84	128
131	99	70	129	127
80	57	115	69	134
104	126	123	95	130

Kernel

0	-1	0
-1	5	-1
0	-1	0

Salida

	266	-61	-30	
	116	-47	295	
	-135	256	-128	

¿Por qué el resultado de la convolución tiene un tamaño menor al de la entrada?

Convolución 2D - Padding

- Usando padding conservamos el tamaño de la imagen original

0	0	0	0	0	0	0
0	60	113	56	139	85	0
0	73	121	54	84	128	0
0	131	99	70	129	127	0
0	80	57	115	69	134	0
0	104	126	123	95	130	0
0	0	0	0	0	0	0

Kernel

0	-1	0
-1	5	-1
0	-1	0

114				

Convolución 2D - Padding

- Usando padding conservamos el tamaño de la imagen original

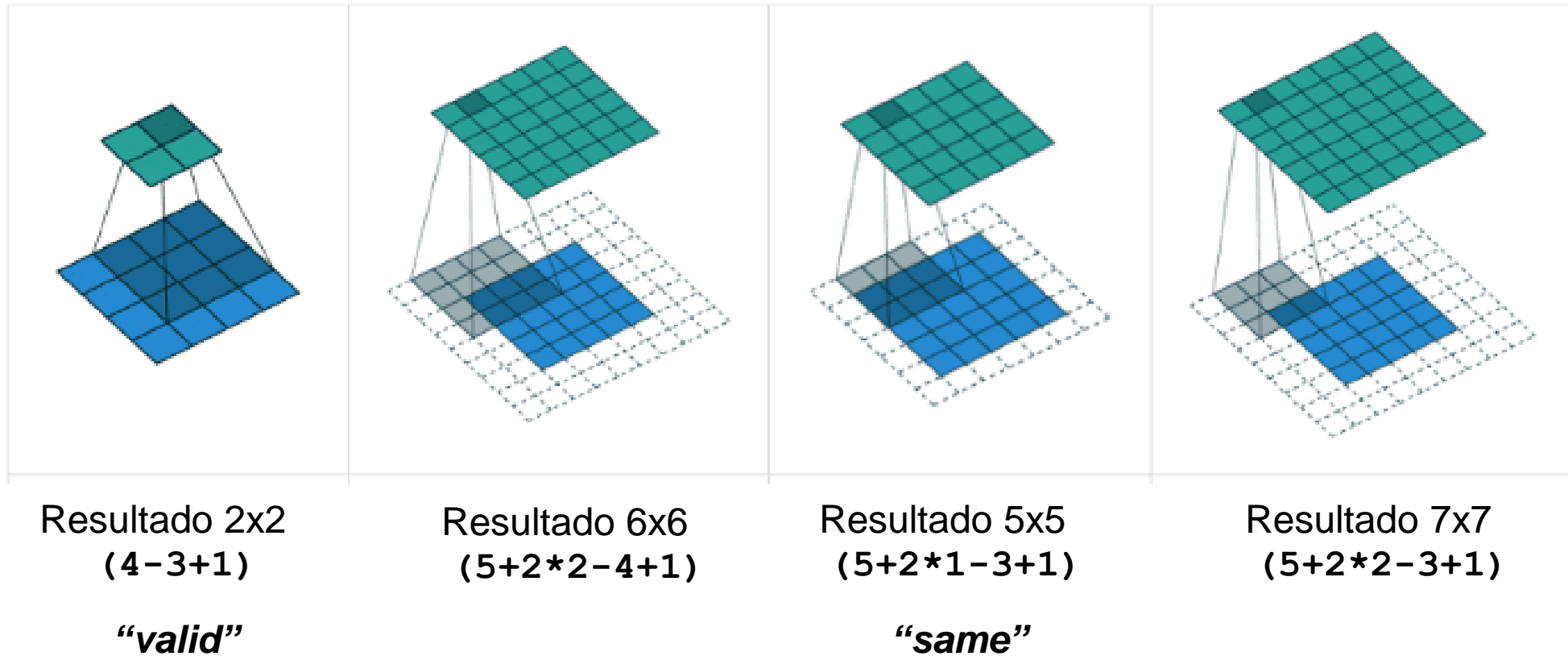
0	0	0	0	0	0	0
0	60	113	56	139	85	0
0	73	121	54	84	128	0
0	131	99	70	129	127	0
0	80	57	115	69	134	0
0	104	126	123	95	130	0
0	0	0	0	0	0	0

Kernel

0	-1	0
-1	5	-1
0	-1	0

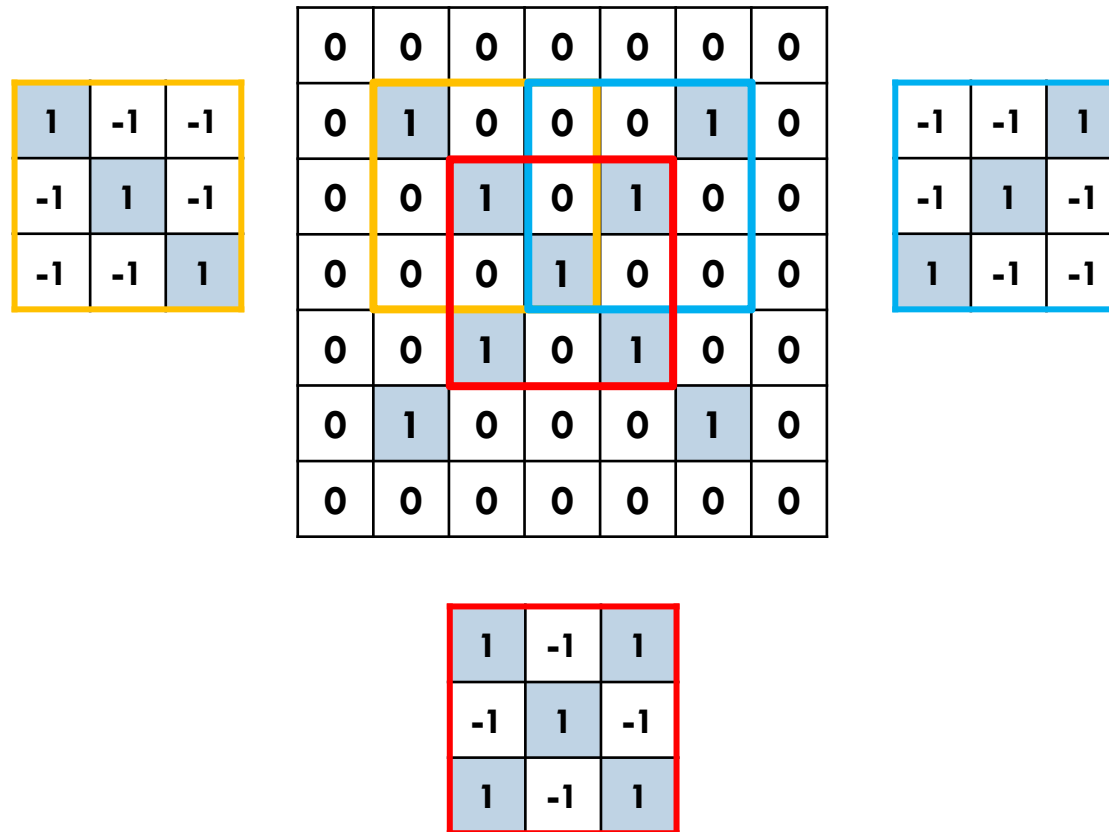
114	328	-26	470	158
53	266	-61	-30	344
403	116	-47	295	244
108	-135	256	-128	344
314	346	279	153	421

Convolución 2D - Padding



Filtros y Extracción de Características

Problema simple: definir filtros para “detectar” la X de ejemplo



Filtros y Extracción de Características

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



1	-1	1
-1	1	-1
1	-1	1

0	0	0	0	0	0	0
0	2	-2	2	-2	2	0
0	-2	3	-3	3	-2	0
0	2	-3	5	-3	2	0
0	-2	3	-3	3	-2	0
0	2	-2	2	-2	2	0
0	0	0	0	0	0	0



RELU

2	0	2	0	2
0	3	0	3	0
2	0	5	0	2
0	3	0	3	0
2	0	2	0	2

Filtros y Extracción de Características

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



1	-1	-1
-1	1	-1
-1	-1	1

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



RELU

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

Filtros y Extracción de Características

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



-1	-1	1
-1	1	-1
1	-1	-1

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



RELU

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

Filtros y Extracción de Características

KERNELS

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

2	0	2	0	2
0	3	0	3	0
2	0	5	0	2
0	3	0	3	0
2	0	2	0	2

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

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

SALIDAS

Filtros y Extracción de Características

2	0	2	0	2
0	3	0	3	0
2	0	5	0	2
0	3	0	3	0
2	0	2	0	2

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

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

□ ¿Que representa cada nuevo valor?

Cada nuevo valor representa el grado de coincidencia entre el filtro y la sección correspondiente de la imagen original

Filtros y Extracción de Características

2	0	2	0	2
0	3	0	3	0
2	0	5	0	2
0	3	0	3	0
2	0	2	0	2

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

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

□ ¿El resultado del filtro es una imagen?

Si, pero el nuevo valor es una intensidad relacionada con la coincidencia del filtro

Filtros y Extracción de Características

2	0	2	0	2
0	3	0	3	0
2	0	5	0	2
0	3	0	3	0
2	0	2	0	2

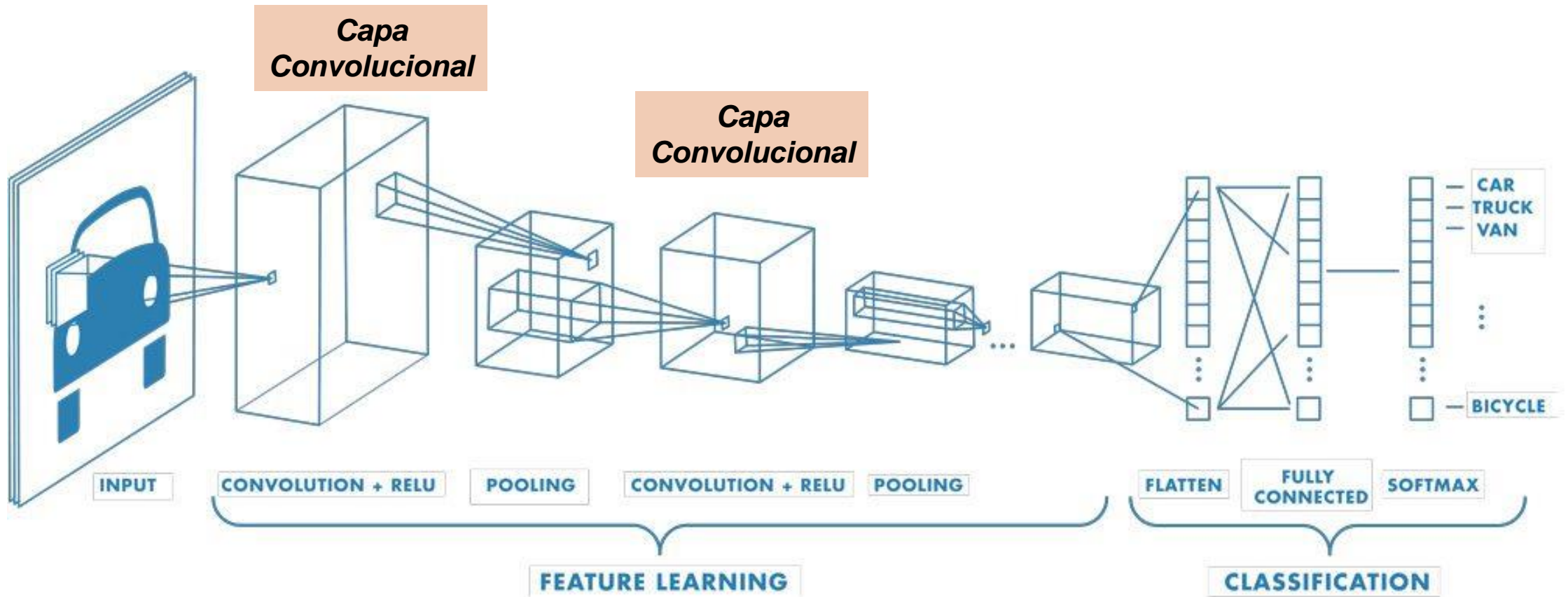
2	0	0	0	0
0	3	0	0	0
0	0	1	0	0
0	0	0	3	0
0	0	0	0	2

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

- Si aplicamos un nuevo filtro al resultado, ¿qué sucede?

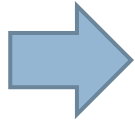
Un nuevo filtro relaciona las características de filtros anteriores, agregando un nivel de abstracción en la interpretación de la imagen

Red Neuronal Convolutacional



Capa convolucional

- Está formada por uno o varios filtros.
- Su función es hacer la convolución de cada filtro sobre cada ejemplo de entrada.



0	-1	0			
	-				
-1		-1	1	1	
0	2	-1	-1	-1	
	-	-1	8	-1	
	-		-1	-1	-1

```
Conv2D(cant_filtros,  
       kernel_size=k,  
       strides=(n,m)  
       activation='relu',  
       padding='same')
```

Capa convolucional

Aplicar_filtro.ipynb

- Utilice un modelo secuencial formado por una única capa para aplicar este filtro que detecta bordes en diagonal



-1	0
0	1



Capa convolucional

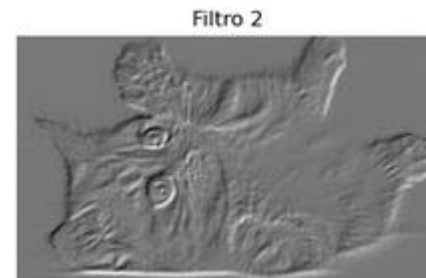
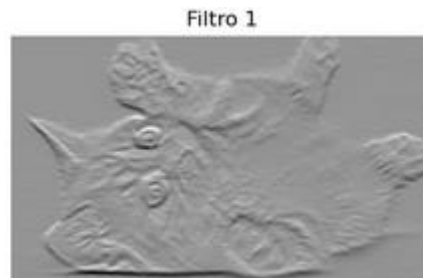
Aplicar_filtros_Conv2.ipynb

- Utilice un modelo secuencial formado por una única capa convolucional para aplicar tres filtros de 3x3

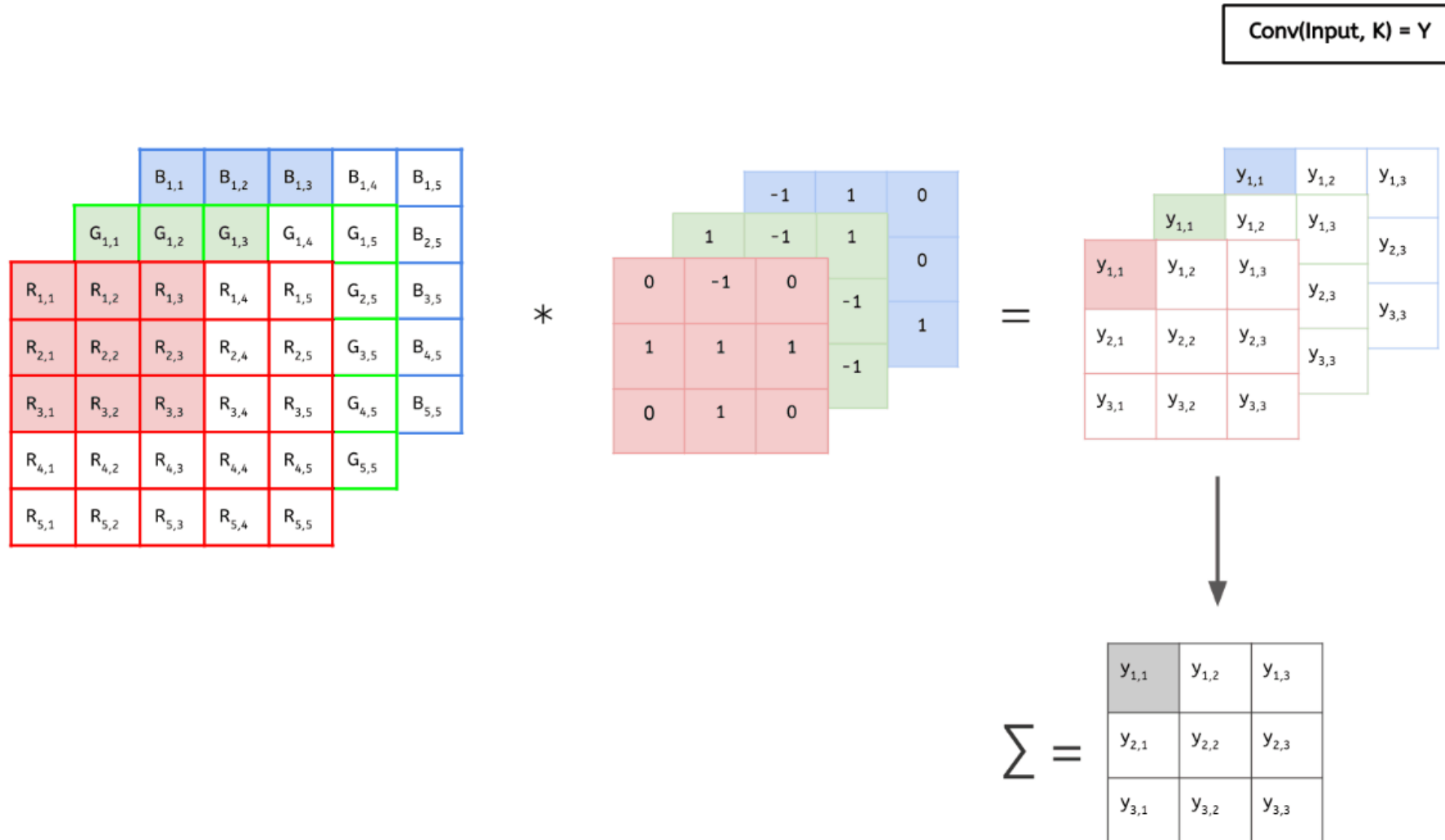
1	1	1
0	0	0
-1	-1	-1

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

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



Convolución 2D con 3 canales



Input Volume (+pad 1) (7x7x3)

$x[:, :, 0]$						
0	0	0	0	0	0	0
0	0	0	1	0	2	0
0	1	0	2	0	1	0
0	1	0	2	2	0	0
0	2	0	0	2	0	0
0	2	1	2	2	0	0
0	0	0	0	0	0	0
$x[:, :, 1]$						
0	0	0	0	0	0	0
0	2	1	2	1	1	0
0	2	1	2	0	1	0
0	0	2	1	0	1	0
0	1	2	2	2	2	0
0	0	1	2	0	1	0
0	0	0	0	0	0	0
$x[:, :, 2]$						
0	0	0	0	0	0	0
0	2	1	1	2	0	0
0	1	0	0	1	0	0
0	0	1	0	0	0	0
0	1	0	2	1	0	0
0	2	2	1	1	1	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

$w0[:, :, 0]$		
-1	0	1
0	0	1
1	-1	1
$w0[:, :, 1]$		
-1	0	1
1	-1	1
0	1	0
$w0[:, :, 2]$		
-1	1	1
1	1	0
0	-1	0
Bias b0 (1x1x1)		
$b0[:, :, 0]$		
1		

Filter W1 (3x3x3)

$w1[:, :, 0]$		
0	1	-1
0	-1	0
0	-1	1
$w1[:, :, 1]$		
-1	0	0
1	-1	0
1	-1	0
$w1[:, :, 2]$		
-1	1	-1
0	-1	-1
1	0	0
Bias b1 (1x1x1)		
$b1[:, :, 0]$		
0		

Output Volume (3x3x2)

$o[:, :, 0]$		
2	3	3
3	7	3
8	10	-3
$o[:, :, 1]$		
-8	-8	-3
-3	1	0
-3	-8	-5

Input : $N \times M \times N_c = 5 \times 5 \times 3$
 Kernel_size= $K \times K \times N_c = 3 \times 3 \times 3$
 Cant_filtros = $N_f = 2$
 Padding = $P = 1$
 Stride = $S = 2$

Input Volume (+pad 1) (7x7x3)

$x[:, :, 0]$						
0	0	0	0	0	0	0
0	0	0	1	0	2	0
0	1	0	2	0	1	0
0	1	0	2	2	0	0
0	2	0	0	2	0	0
0	2	1	2	2	0	0
0	0	0	0	0	0	0
$x[:, :, 1]$						
0	0	0	0	0	0	0
0	2	1	2	1	1	0
0	2	1	2	0	1	0
0	0	2	1	0	1	0
0	1	2	2	2	2	0
0	0	1	2	0	1	0
0	0	0	0	0	0	0
$x[:, :, 2]$						
0	0	0	0	0	0	0
0	2	1	1	2	0	0
0	1	0	0	1	0	0
0	0	1	0	0	0	0
0	1	0	2	1	0	0
0	2	2	1	1	1	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

$w0[:, :, 0]$		
-1	0	1
0	0	1
1	-1	1
$w0[:, :, 1]$		
-1	0	1
1	-1	1
0	1	0
$w0[:, :, 2]$		
-1	1	1
1	1	0
0	-1	0
Bias b0 (1x1x1)		
$b0[:, :, 0]$		
1		

Filter W1 (3x3x3)

$w1[:, :, 0]$		
0	1	-1
0	-1	0
0	-1	1
$w1[:, :, 1]$		
-1	0	0
1	-1	0
1	-1	0
$w1[:, :, 2]$		
-1	1	-1
0	-1	-1
1	0	0
Bias b1 (1x1x1)		
$b1[:, :, 0]$		
0		

Output Volume (3x3x2)

$o[:, :, 0]$		
2	3	3
3	7	3
8	10	-3
$o[:, :, 1]$		
-8	-8	-3
-3	1	0
-3	-8	-5

Input : $N \times M \times N_c = 5 \times 5 \times 3$
 Kernel_size= $K \times K \times N_c = 3 \times 3 \times 3$
 Cant_filtros = $N_f = 2$
 Padding = $P = 1$
 Stride = $S = 2$

Input Volume (+pad 1) (7x7x3)

x[:, :, 0]						
0	0	0	0	0	0	0
0	0	0	1	0	2	0
0	1	0	2	0	1	0
0	1	0	2	2	0	0
0	2	0	0	2	0	0
0	2	1	2	2	0	0
0	0	0	0	0	0	0
x[:, :, 1]						
0	0	0	0	0	0	0
0	2	1	2	1	1	0
0	2	1	2	0	1	0
0	0	2	1	0	1	0
0	1	2	2	2	2	0
0	0	1	2	0	1	0
0	0	0	0	0	0	0
x[:, :, 2]						
0	0	0	0	0	0	0
0	2	1	1	2	0	0
0	1	0	0	1	0	0
0	0	1	0	0	0	0
0	1	0	2	1	0	0
0	2	2	1	1	1	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

w0[:, :, 0]		
-1	0	1
0	0	1
1	-1	1
w0[:, :, 1]		
-1	0	1
1	-1	1
0	1	0
w0[:, :, 2]		
-1	1	1
1	1	0
0	-1	0
Bias b0 (1x1x1)		
b0[:, :, 0]	1	

Filter W1 (3x3x3)

w1[:, :, 0]		
0	1	-1
0	-1	0
0	-1	1
w1[:, :, 1]		
-1	0	0
1	-1	0
1	-1	0

Output Volume (3x3x2)

o[:, :, 0]		
2	3	3
3	7	3
8	10	-3
o[:, :, 1]		
-8	-8	-3
-3	1	0
-3	-8	-5



Output : $T \times U \times N_f = 3 \times 3 \times 2$

$$T = (N + 2 \cdot P - K) / S + 1$$

$$= (5 + 2 \cdot 1 - 3) / 2 + 1 = 3$$

$$U = (M + 2 \cdot P - K) / S + 1 = 3$$

Input : $N \times M \times N_c = 5 \times 5 \times 3$

Kernel_size = $K \times K \times N_c = 3 \times 3 \times 3$


Cant_filtros = $N_f = 2$

Padding = $P = 1$

Stride = $S = 2$

MNIST

```
model = Sequential()  
model.add(Conv2D(64, kernel_size=3, activation="relu",  
                 input_shape=input_shape))  
model.add(Flatten())  
model.add(Dense(10, activation='softmax'))  
model.summary()
```



A diagram consisting of a red line that starts from a light orange box containing the text `(28, 28, 1)` on the right. The line moves left, then turns downwards, then left again, ending with an arrowhead pointing to the `input_shape` parameter in the `Conv2D` layer of the code above.

MNIST

```
model = Sequential()
model.add(Conv2D(64, kernel_size=3, activation="relu",
                 input_shape=input_shape))
model.add(Flatten())
model.add(Dense(10, activation='softmax'))
model.summary()
```

Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 26, 26, 64)	640
flatten_10 (Flatten)	(None, 43264)	0
dense_10 (Dense)	(None, 10)	432650

=====
Total params: 433,290

¿Por qué la salida es de 26x26 si las imágenes son de 28x28?

$$(N + 2 * P - K) / S + 1$$

$$(28 + 0 - 3) / 1 + 1 = 26$$

MNIST

```
model = Sequential()
model.add(Conv2D(64, kernel_size=3, activation="relu",
                 input_shape=input_shape))
model.add(Flatten())
model.add(Dense(10, activation='softmax'))
model.summary()
```

¿Por qué la capa
convolucional tiene 640
parámetros?

Layer (type)	Output Shape	Param #
=====		
conv2d_4 (Conv2D)	(None, 26, 26, 64)	640

flatten_10 (Flatten)	(None, 43264)	0

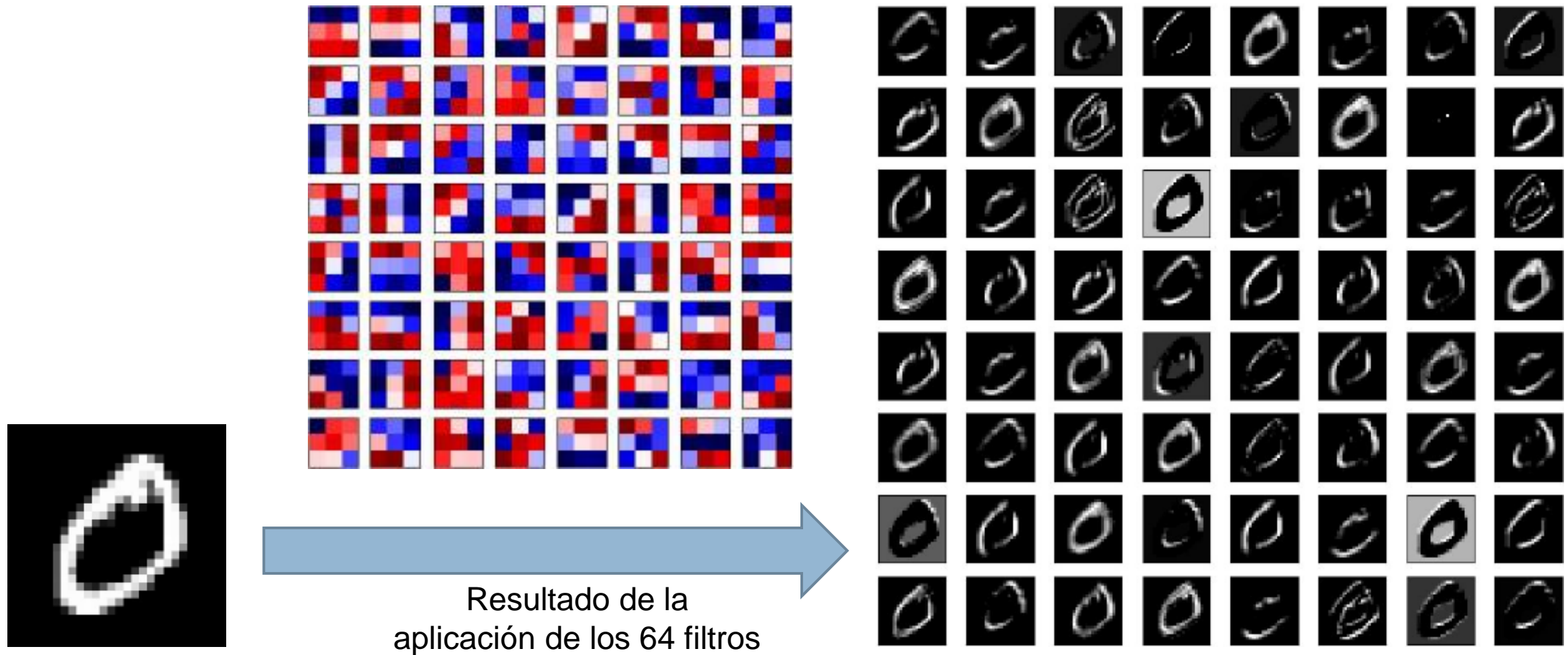
dense_10 (Dense)	(None, 10)	432650
=====		
Total params:		433,290

$$N_f * K * K * N_c + N_f$$

64 * 3 * 3 + 64

↑ ↑ ↑
Cantidad de Tamaño del Bias
filtros filtro

Capa Conv2D de MNIST



MNIST

```
model = Sequential()
model.add(Conv2D(64, kernel_size=3, activation="relu",
                 input_shape=input_shape))
model.add(Flatten())
model.add(Dense(10, activation='softmax'))
model.summary()
```

Cantidad de parámetros de
la capa Flatten

Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 26, 26, 64)	640
flatten_10 (Flatten)	(None, 43264)	0
dense_10 (Dense)	(None, 10)	432650
Total params: 433,290		

$$N_f * T * U$$

64 * 26 * 26

↑
Cantidad de
filtros

└──────────┘
Tamaño de la
imagen filtrada

MNIST

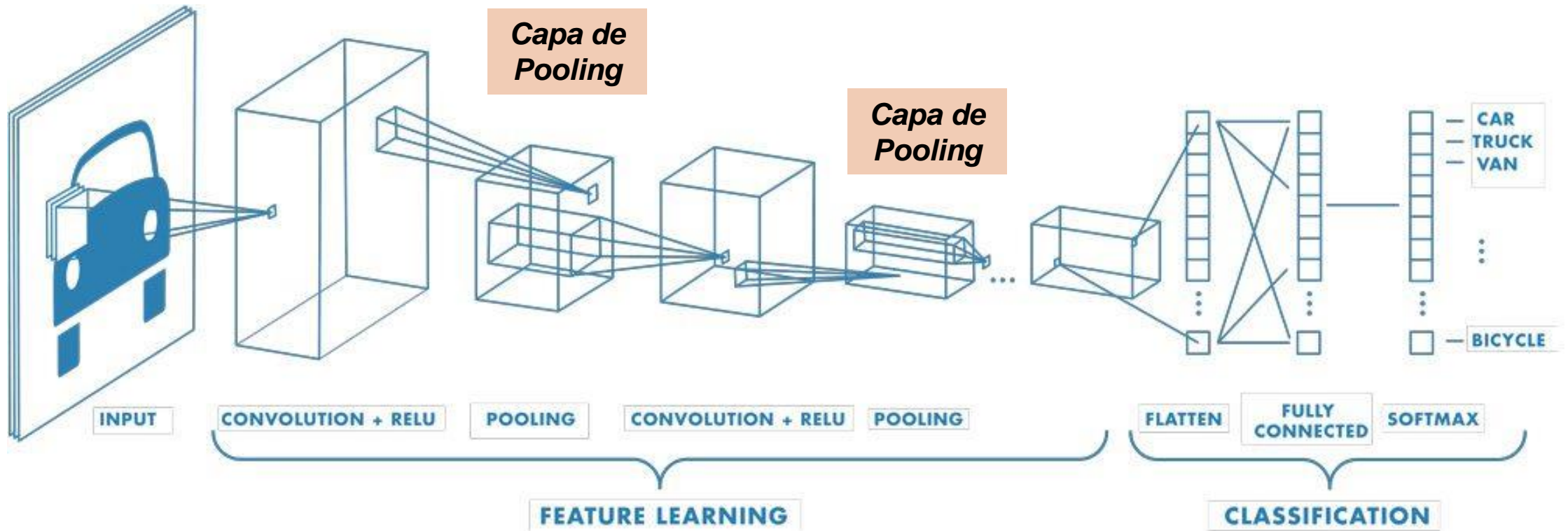
```
model = Sequential()
model.add(Conv2D(64, kernel_size=3, activation="relu", input_shape=input_shape))
model.add(Flatten())
model.add(Dense(15, activation='tanh'))
model.add(Dense(10, activation='softmax'))
model.summary()
```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 26, 26, 64)	640
flatten (Flatten)	(None, 43264)	0
dense (Dense)	(None, 15)	648975
dense_1 (Dense)	(None, 10)	160
Total params: 649,775		

Cantidad de parámetros o pesos de la capa oculta del multiperceptrón

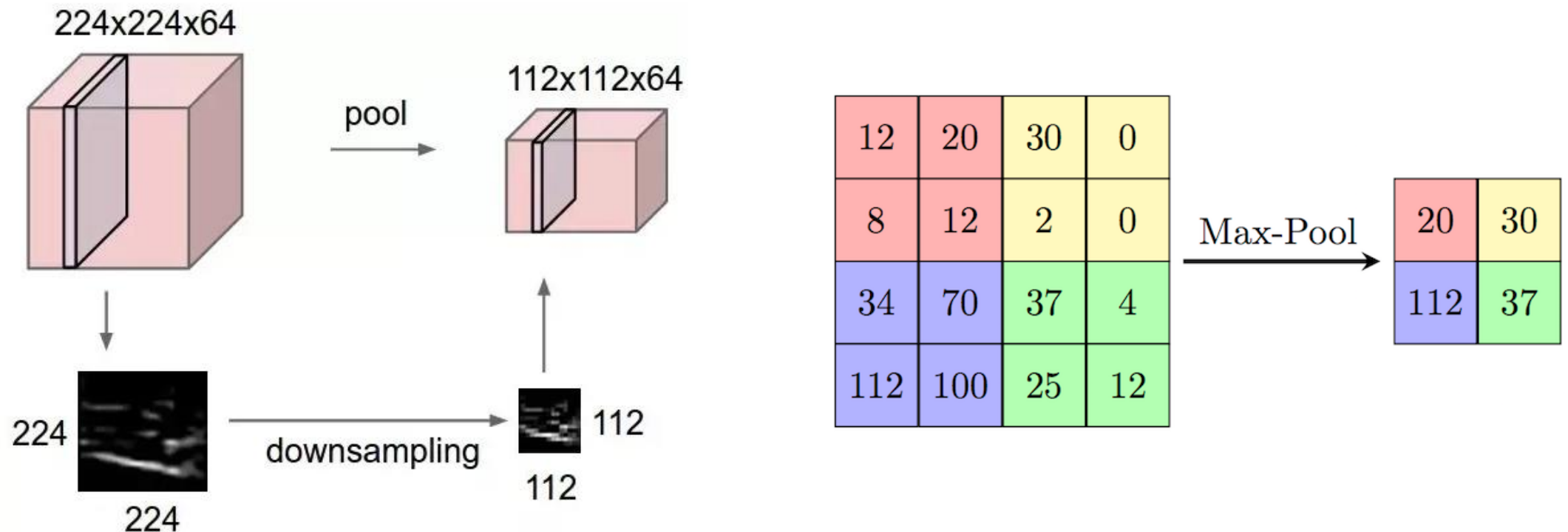
$$43264 * 15 + 15$$

Red Neuronal Convolucional

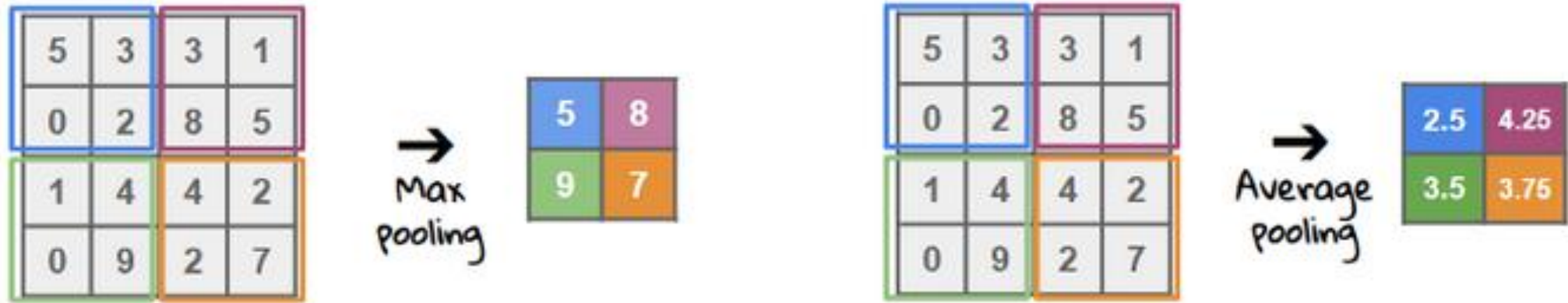


Capa de pooling

- La capa de pooling reduce el tamaño de la salida de la capa convolucional. Se trata de una convolución con un stride igual al tamaño del kernel que calcula la función sobre todos los pixels.



Pooling



- La reducción de tamaño permite eliminar parte del ruido y extraer datos más significativos.
- Reduce el exceso de ajuste y acelera el cálculo

Capa MaxPooling2D

```
model = Sequential()
model.add(Conv2D(64, kernel_size=3, activation="relu", input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
model.add(Dense(15, activation='tanh'))
model.add(Dense(10, activation='softmax'))
model.summary()
```

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 26, 26, 64)	640
max_pooling2d (MaxPooling2D)	(None, 13, 13, 64)	0
flatten_1 (Flatten)	(None, 10816)	0
dense_2 (Dense)	(None, 15)	162255
dense_3 (Dense)	(None, 10)	160
Total params: 163,055		

Luego de aplanar la cantidad de entradas se redujo un 75%

Antes eran 43264 y ahora quedaron 10816

MNIST

```
model = Sequential()
model.add(Input(shape=(28, 28, 1)))
model.add(Conv2D(32, kernel_size=3, strides=(1,1), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2), name='max_pool_1'))
model.add(Conv2D(16, kernel_size=3, strides=(1,1), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2), name='max_pool_2'))
model.add(Flatten())
model.add(Dense(10, activation='softmax'))
model.summary()
```

¿Por qué la salida es de 26x26 si las imágenes son de 28x28?

Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 26, 26, 32)	320
max_pool_1 (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_3 (Conv2D)	(None, 11, 11, 16)	4624
max_pool_2 (MaxPooling2D)	(None, 5, 5, 16)	0
flatten_1 (Flatten)	(None, 400)	0
dense_1 (Dense)	(None, 10)	4010

Total params: 8,954

$$(N + 2 * P - K) / S + 1$$

$$(28 + 0 - 3) / 1 + 1 = 26$$

MNIST

```
model = Sequential()
model.add(Input(shape=(28, 28, 1)))
model.add(Conv2D(32, kernel_size=3, strides=(1,1), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2), name='max_pool_1'))
model.add(Conv2D(16, kernel_size=3, strides=(1,1), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2), name='max_pool_2'))
model.add(Flatten())
model.add(Dense(10, activation='softmax'))
model.summary()
```

**¿Por qué la capa
convolucional tiene 320
parámetros?**

Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 26, 26, 32)	320
max_pool_1 (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_3 (Conv2D)	(None, 11, 11, 16)	4624
max_pool_2 (MaxPooling2D)	(None, 5, 5, 16)	0
flatten_1 (Flatten)	(None, 400)	0
dense_1 (Dense)	(None, 10)	4010

Total params: 8,954

$$N_f * K * K * N_c + N_f$$
$$32 * 3 * 3 * 1 + 32$$

Cantidad de filtros Tamaño del filtro Bias

MNIST

```
model = Sequential()
model.add(Input(shape=(28, 28, 1)))
model.add(Conv2D(32, kernel_size=3, strides=(1,1), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2), name='max_pool_1'))
model.add(Conv2D(16, kernel_size=3, strides=(1,1), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2), name='max_pool_2'))
model.add(Flatten())
model.add(Dense(10, activation='softmax'))
model.summary()
```

**¿Por qué la 2da.capa
convolucional tiene 4624
parámetros?**

Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 26, 26, 32)	320
max_pool_1 (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_3 (Conv2D)	(None, 11, 11, 16)	4624
max_pool_2 (MaxPooling2D)	(None, 5, 5, 16)	0
flatten_1 (Flatten)	(None, 400)	0
dense_1 (Dense)	(None, 10)	4010

Total params: 8,954

$$N_f * K * K * N_c + N_f$$
$$16 * 3 * 3 * 32 + 16$$

↑ Cantidad de filtros Tamaño del filtro ↑ Bias

MNIST

```
model = Sequential()
model.add(Input(shape=(28, 28, 1)))
model.add(Conv2D(32, kernel_size=3, strides=(1,1), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2), name='max_pool_1'))
model.add(Conv2D(16, kernel_size=3, strides=(1,1), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2), name='max_pool_2'))
model.add(Flatten())
model.add(Dense(10, activation='softmax'))
model.summary()
```

Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 26, 26, 32)	320
max_pool_1 (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_3 (Conv2D)	(None, 11, 11, 16)	4624
max_pool_2 (MaxPooling2D)	(None, 5, 5, 16)	0
flatten_1 (Flatten)	(None, 400)	0
dense_1 (Dense)	(None, 10)	4010

Total params: 8,954

**Luego del 2do.
MaxPooling se
aplanan las 16
salidas de 5x5**

MNIST

```
model = Sequential()
model.add(Input(shape=(28, 28, 1)))
model.add(Conv2D(32, kernel_size=3, strides=(1,1), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2), name='max_pool_1'))
model.add(Conv2D(16, kernel_size=3, strides=(1,1), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2), name='max_pool_2'))
model.add(Flatten())
model.add(Dense(10, activation='softmax'))
model.summary()
```

Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 26, 26, 32)	320
max_pool_1 (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_3 (Conv2D)	(None, 11, 11, 16)	4624
max_pool_2 (MaxPooling2D)	(None, 5, 5, 16)	0
flatten_1 (Flatten)	(None, 400)	0
dense_1 (Dense)	(None, 10)	4010

Total params: 8,954

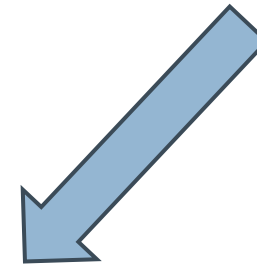
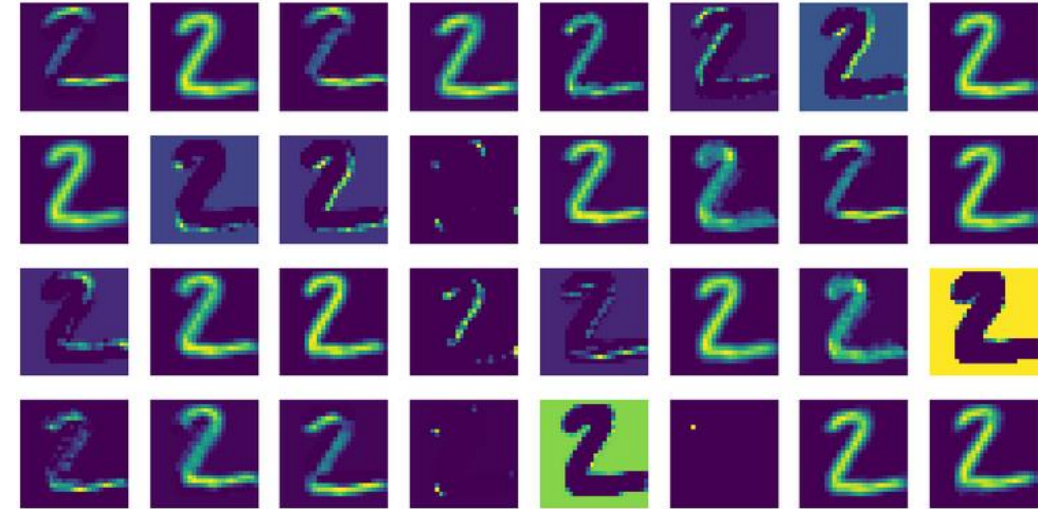
¿Por qué la capa de salida tiene 4010 parámetros?

Capa Conv2D de MNIST

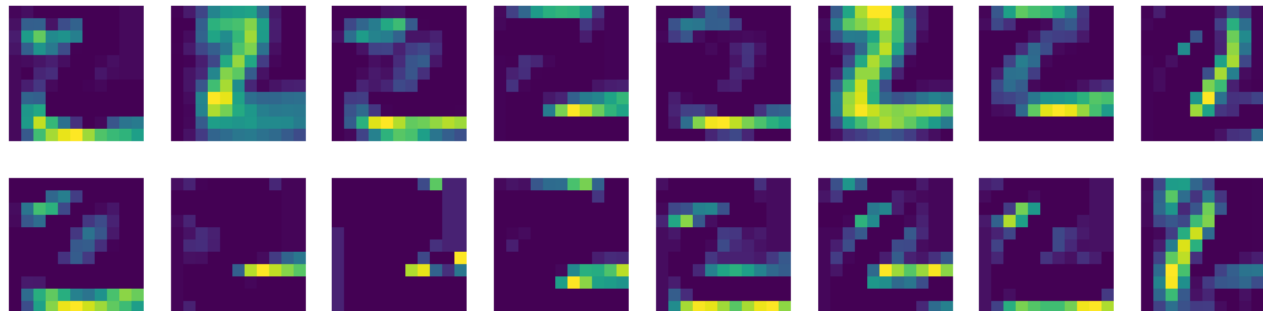
MNIST_Aplicar_modelo.ipynb



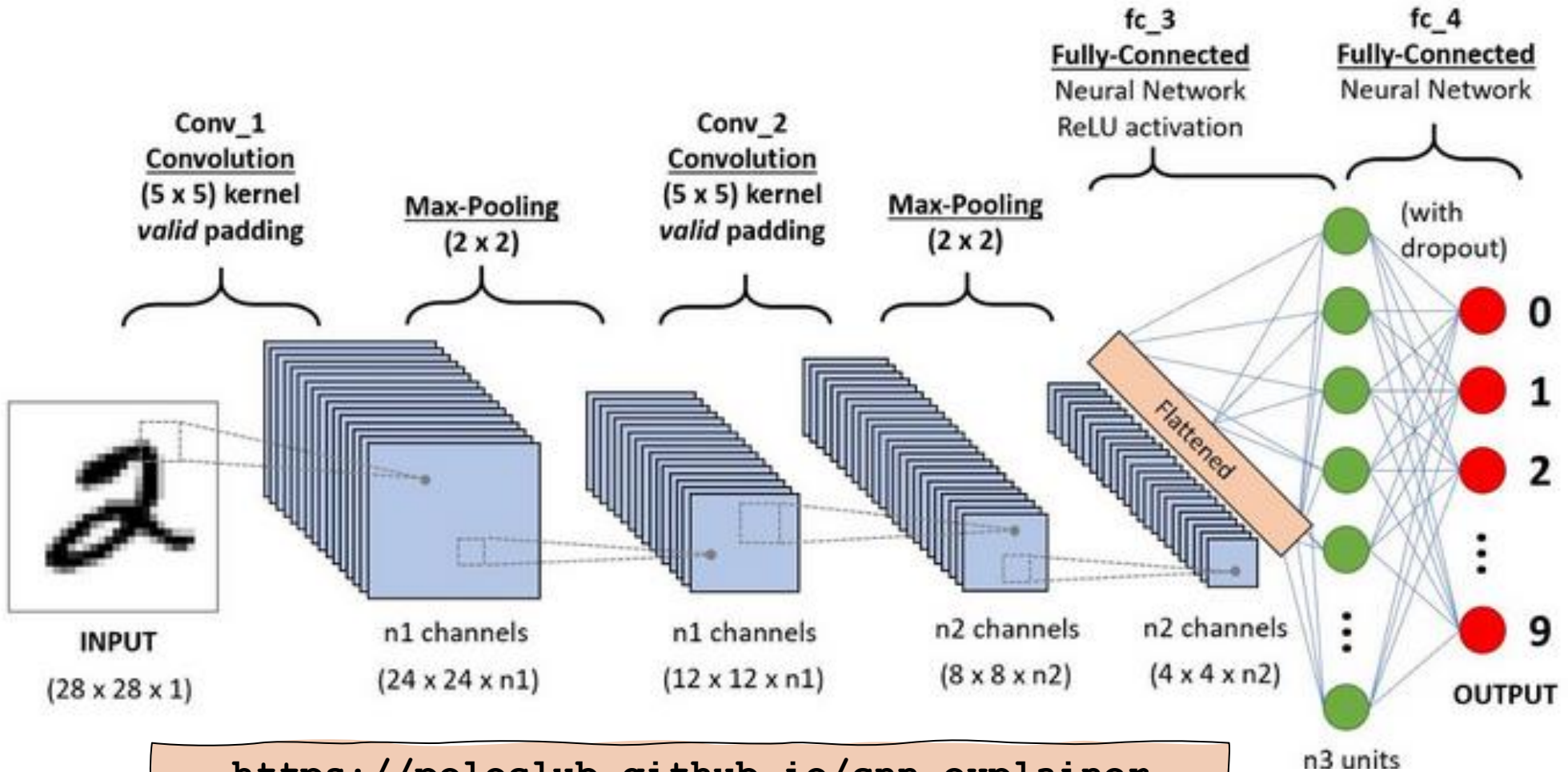
Resultado de la
aplicación de
los 32 filtros de
 $3 \times 3 \times 1$



Resultado de la
aplicación de
los 16 filtros de
 $3 \times 3 \times 32$



Reconocimiento de dígitos MNIST



<https://poloclub.github.io/cnn-explainer>

Resumen

- Las capas convolucionales 2D contienen los filtros que, se entrenan junto con los demás parámetros de la red y que permiten detectar características.
- El resultado de aplicar un filtro o máscara es un mapa de características de $T_x U_x 1$ dependiendo del padding, stride y tamaño de kernel usados.
- La salida de la capa convolucional es una nueva “imagen” de $T_x U_x F$ siendo F el número de filtros.
- Las capas de pooling reducen la dimensionalidad haciendo más rápido y eficaz el entrenamiento.
- Se suelen intercalar capas de convolución con capas de pooling hasta llegar a la parte feedforward donde para ingresar “aplanamos” las “imágenes”.

MNIST

La base de datos MNIST contiene imágenes de 28×28 en tonos de gris, de números escritos a mano.

□ Características

- Presenta poca variabilidad entre ejemplos de una misma clase.
- Las imágenes están centradas.

□ Ventajas

- Imágenes pequeñas que facilitan la rápida experimentación.
- Datos balanceados que permiten utilizar métricas de clasificación sencillas.



Ejercicio

Construir una red neuronal capaz de reconocer el dígito presente en una imagen:

- Revisar la red formada sólo por una capa softmax.
- Implementar un MLP con un 97 o 98% de accuracy.
- Implementar una CNN con un accuracy del 99% sobre los datos de testeo.

`MNIST_Conv2D_MaxPool.ipynb`

