

Redes Neuronales Convolucionales

Una introducción...

Imágenes

Imágenes B/N

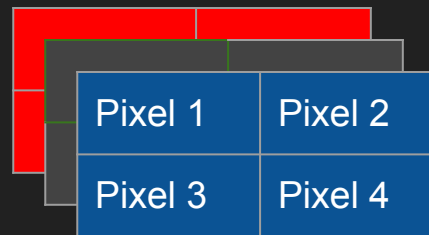
Píxel 1	Píxel 2
Píxel 3	Píxel 4

Un arreglo bidimensional

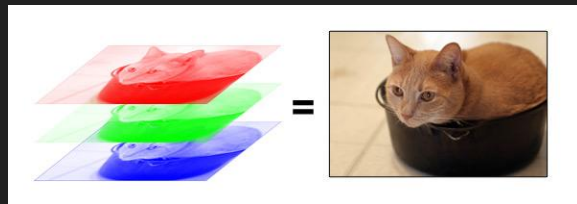
Intensidad

Cada píxel contiene un número **entero** comprendido entre 0 y 255. Se puede escalar a valores **reales** entre 0 y 1

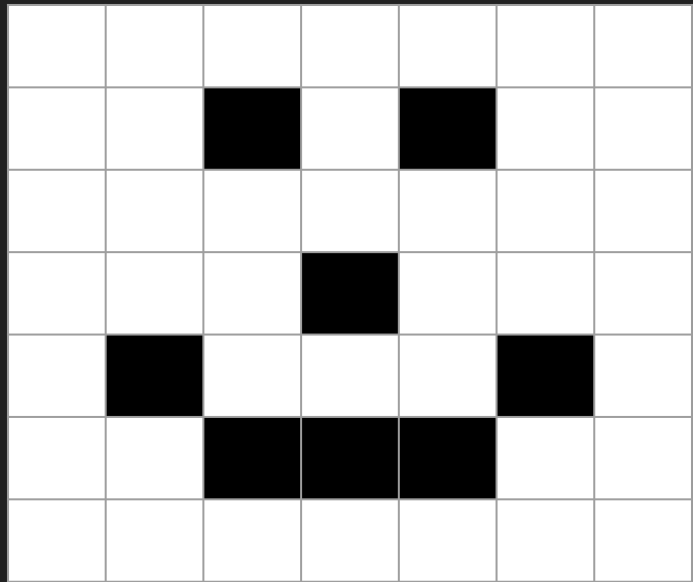
Imágenes a color



Tres arreglos bidimensionales



Ejemplo de representación



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

Cuatro fases de la convolución

1. Convolución
2. Max Pooling
3. Flattening
4. Capa de Full Connection

Convolución

Operación de convolución

$$(f * g)(x) = \int f(y)g(x - y)dy$$

Ejemplo de convolución

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

Imagen de entrada

0	1	0
0	0	1
1	1	0

Detector de características

Ejemplo de convolución

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

Imagen de entrada



0	1	0
0	0	1
1	1	0

Detector de características

=

Mapa de características

Ejemplo de convolución

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

Imagen de entrada



0	1	0
0	0	1
1	1	0

Detector de características

=

1				

Mapa de características

Ejemplo de convolución

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

Imagen de entrada

\otimes

0	1	0
0	0	1
1	1	0

Detector de características

=

1	0			

Mapa de características

Ejemplo de convolución

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

Imagen de entrada



0	1	0
0	0	1
1	1	0

Detector de características

=

1	0	1		

Mapa de características

Ejemplo de convolución

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

Imagen de entrada



0	1	0
0	0	1
1	1	0

Detector de características

=

1	0	1	0	

Mapa de características

Ejemplo de convolución

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

Imagen de entrada



0	1	0
0	0	1
1	1	0

Detector de características

=

1	0	1	0	0

Mapa de características

Ejemplo de convolución

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

Imagen de entrada

\otimes

0	1	0
0	0	1
1	1	0

Detector de características

=

1	0	1	0	0
0				

Mapa de características

Ejemplo de convolución

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

Imagen de entrada



0	1	0
0	0	1
1	1	0

Detector de características

=

1	0	1	0	0
0	1			

Mapa de características

Ejemplo de convolución

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

Imagen de entrada

\otimes

0	1	0
0	0	1
1	1	0

Detector de características

=

1	0	1	0	0
0	1	1		

Mapa de características

Ejemplo de convolución

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

Imagen de entrada

\otimes

0	1	0
0	0	1
1	1	0

Detector de características

=

1	0	1	0	0
0	1	1	2	

Mapa de características

Ejemplo de convolución

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

Imagen de entrada

\otimes

0	1	0
0	0	1
1	1	0

Detector de características

=

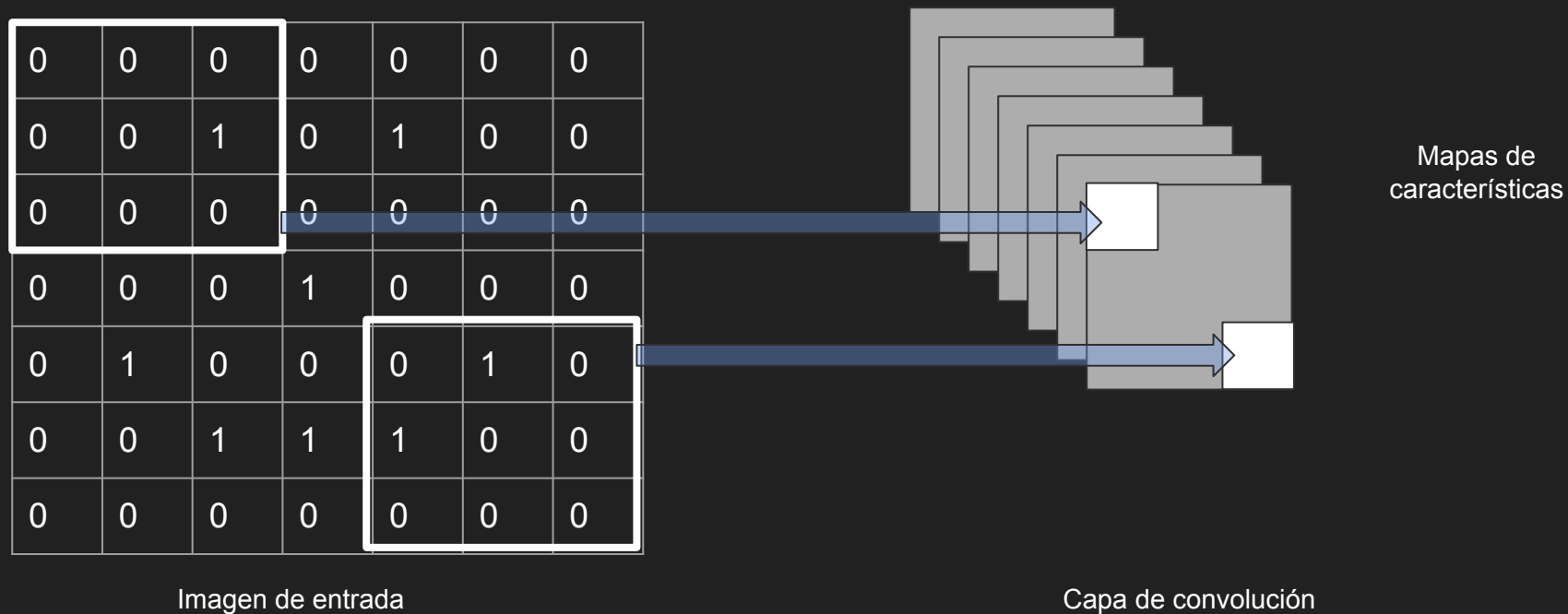
1	0	1	0	0
0	1	1	2	1
1	2	0	0	1
0	1	3	3	1
2	1	1	0	1

Mapa de características

Nuestro cerebro



En resumen



Para probar un poco...

Descargar herramienta (gimp).

Tiene un plugin para hacer convolución

- Aquí solo veremos la documentación de [este](#) enlace:

La capa ReLU

1. Será el paso final de la convolución.
2. No es un paso separado sino un pequeño paso que se aplica al final.
3. Veamos...

¿Dónde está la capa ReLU?

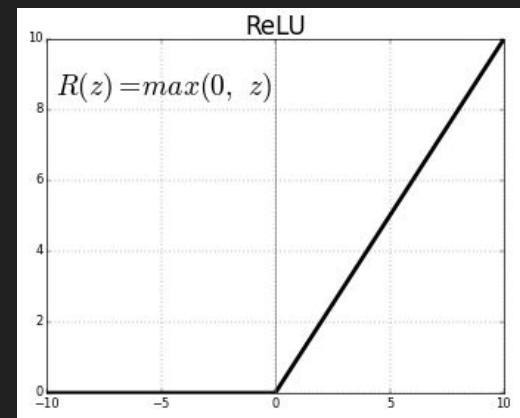
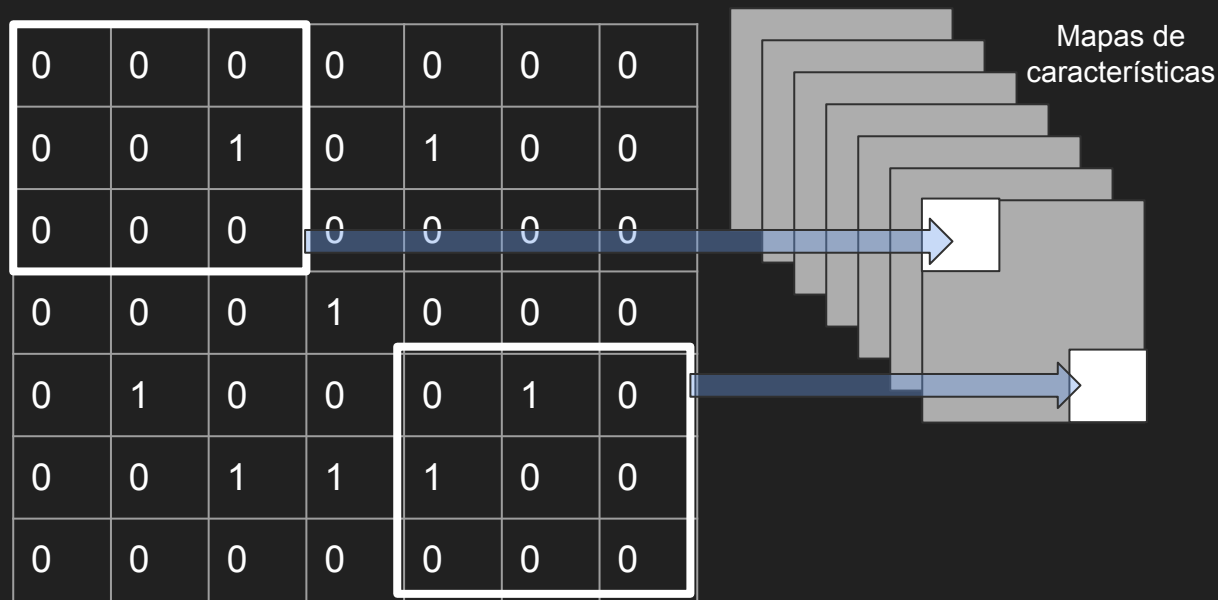


Imagen de entrada

Capa de convolución

Función rectificadora

Max pooling

Max-pooling

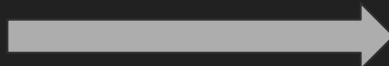
- Transformaciones afines.
- La idea es que asegurar que la red neuronal sea capaz de reconocer una figura (forma), independientemente de la forma en que se presente en la imagen.

Max-pooling

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

Mapa de características

Operación de
Max-pooling



1		

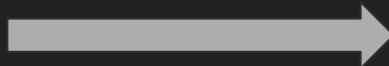
Mapa de características Pooled

Max-pooling

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

Mapa de características

Operación de
Max-pooling



1	2	

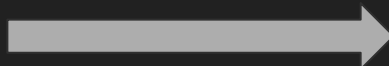
Mapa de características Pooled

Max-pooling

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

Mapa de características

Operación de
Max-pooling



1	2	1

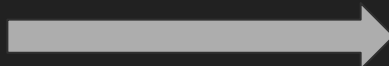
Mapa de características Pooled

Max-pooling

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

Mapa de características

Operación de
Max-pooling



1	2	1
2		

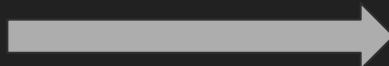
Mapa de características Pooled

Max-pooling

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

Mapa de características

Operación de
Max-pooling



1	2	1
2	3	

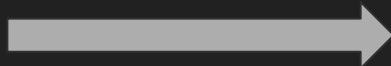
Mapa de características Pooled

Max-pooling

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

Mapa de características

Operación de
Max-pooling



1	2	1
2	3	1

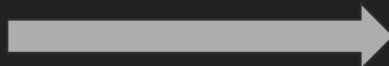
Mapa de características Pooled

Max-pooling

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

Mapa de características

Operación de
Max-pooling

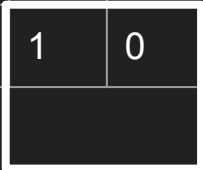


1	2	1
2	3	1
2		

Mapa de características Pooled

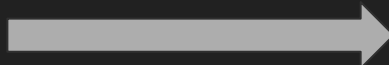
Max-pooling

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



Mapa de características

Operación de
Max-pooling

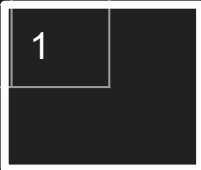


1	2	1
2	3	1
2	1	

Mapa de características Pooled

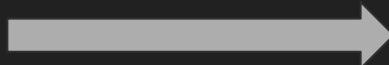
Max-pooling

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



Mapa de características

Operación de
Max-pooling



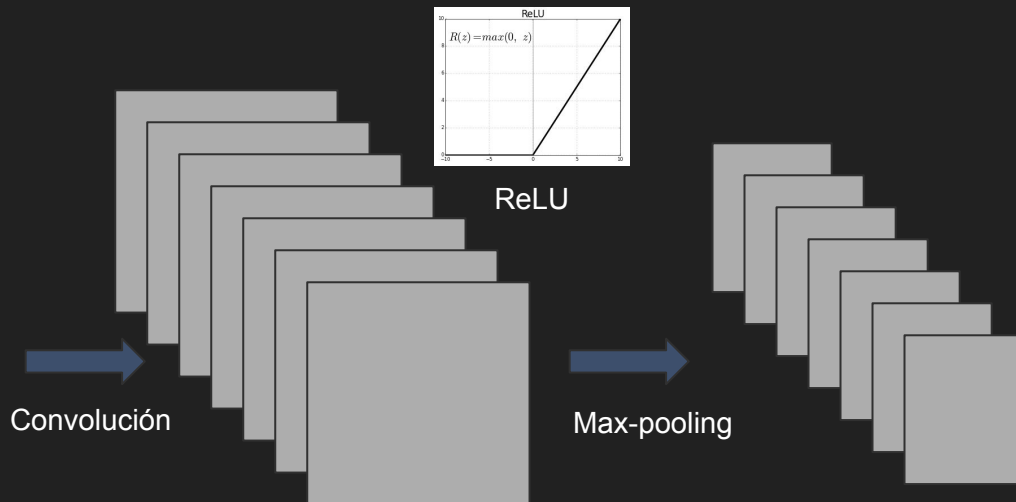
1	2	1
2	3	1
2	1	1

Mapa de características Pooled

Max-pooling

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

Imagen de entrada



Capa de convolución

Capa de max-pooling

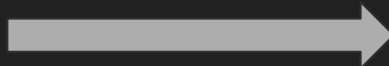
Flattening

Max-pooling

1	2	1
2	3	1
2	1	1

Mapa de características Pooled

Flattening



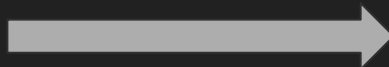
1
2
1

Max-pooling

1	2	1
2	3	1
2	1	1

Mapa de características Pooled

Flattening



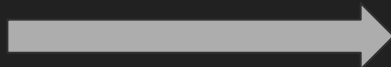
1
2
1
2
3
1

Max-pooling

1	2	1
2	3	1
2	1	1

Mapa de características Pooled

Flattening



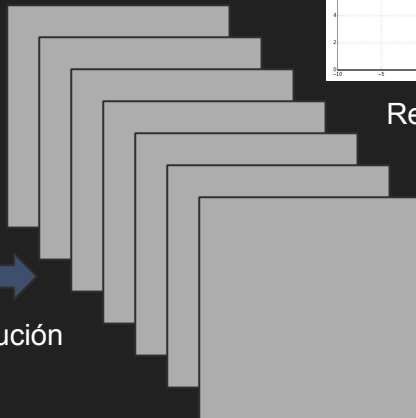
1
2
1
2
3
1
2
1
1

Max-pooling

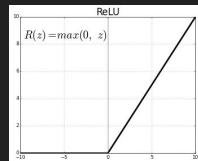
0	0	0	0	0	0	0
0	0	1	0	1	0	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Imagen de entrada

Convolución



Capa de convolución



ReLU

Max-pooling



Capa de max-pooling

Flattening



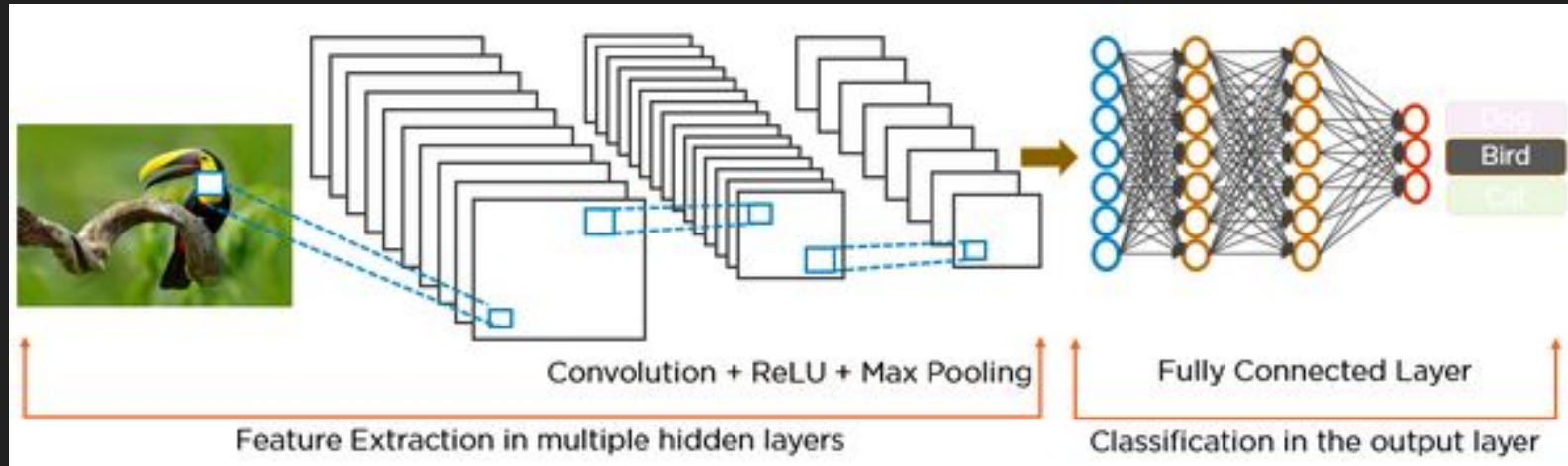
Capa de
entrada de
una RN

Capa Full Connection

Capa full connection

- Es un paso más simple.
- Se trata de una red neuronal como las que hemos visto hasta el momento.
- Las entradas serán los vectores que conseguimos en la etapa anterior (luego de hacer flattening a la imagen de características pooleada).
- La idea general es completar el siguiente proceso:

Capa full connection



¿Cómo se programa esto?

1. Se hace igual que una red neuronal común y corriente, pero agregando capas convolucionales.
2. Existen dos maneras de crear capas convolucionales:
 - a. Tensorflow: `tf.nn.conv2d()`
 - b. Keras: `model.add(layers.conv2D())`
3. Una vez definidas las capas convolucionales, se definen las capas de pooling
 - a. Tensorflow: `tf.nn.max_pool2d()`
 - b. `tf.keras.layers.MaxPool2D()`

Ejemplo 1 (capa convolucional y pooling)

```
from keras import layers
```

```
from keras import models
```

```
model = models.Sequential()
```

```
model.add(layers.Conv2D(32,(5,5),activation='relu',input_shape=(28, 28,1)))
```

```
model.add(layers.MaxPooling2D((2, 2)))
```

Ejemplo 1 (capa convolucional y pooling)

```
model.summary()
```

Layer (type)	Output Shape	Param #
=====		
conv2d_1 (Conv2D)	(None, 24, 24, 32)	832

max_pooling2d_1 (MaxPooling2)	(None, 12, 12, 32)	0

Total params: 832		
Trainable params: 832		
Non-trainable params: 0		

Ejemplo 2 (capa convolucional y pooling)

```
from keras import layers
```

```
from keras import models
```

```
model = models.Sequential()
```

```
model.add(layers.Conv2D(32,(5,5),activation='relu',input_shape=(28, 28,1)))
```

```
model.add(layers.MaxPooling2D((2, 2)))
```

```
model.add(layers.Conv2D(64, (5, 5), activation='relu'))
```

```
model.add(layers.MaxPooling2D((2, 2)))
```

Ejemplo 1 (capa convolucional y pooling)

```
model.summary()
```

Layer (type)	Output Shape	Param #
=====		
conv2d_1 (Conv2D)	(None, 24, 24, 32)	832

max_pooling2d_1 (MaxPooling2)	(None, 12, 12, 32)	0

conv2d_2 (Conv2D)	(None, 8, 8, 64)	51264

max_pooling2d_2 (MaxPooling2)	(None, 4, 4, 64)	0
=====		
Total params: 52,096		
Trainable params: 52,096		
Non-trainable params: 0		

```
model.add(layers.Flatten())
```

```
model.add(layers.Dense(10, activation='softmax'))
```


Ejemplo 1 (capa convolucional y pooling)

model.summary()

Layer (type)	Output Shape	Param #
=====		
conv2d_1 (Conv2D)	(None, 24, 24, 32)	832

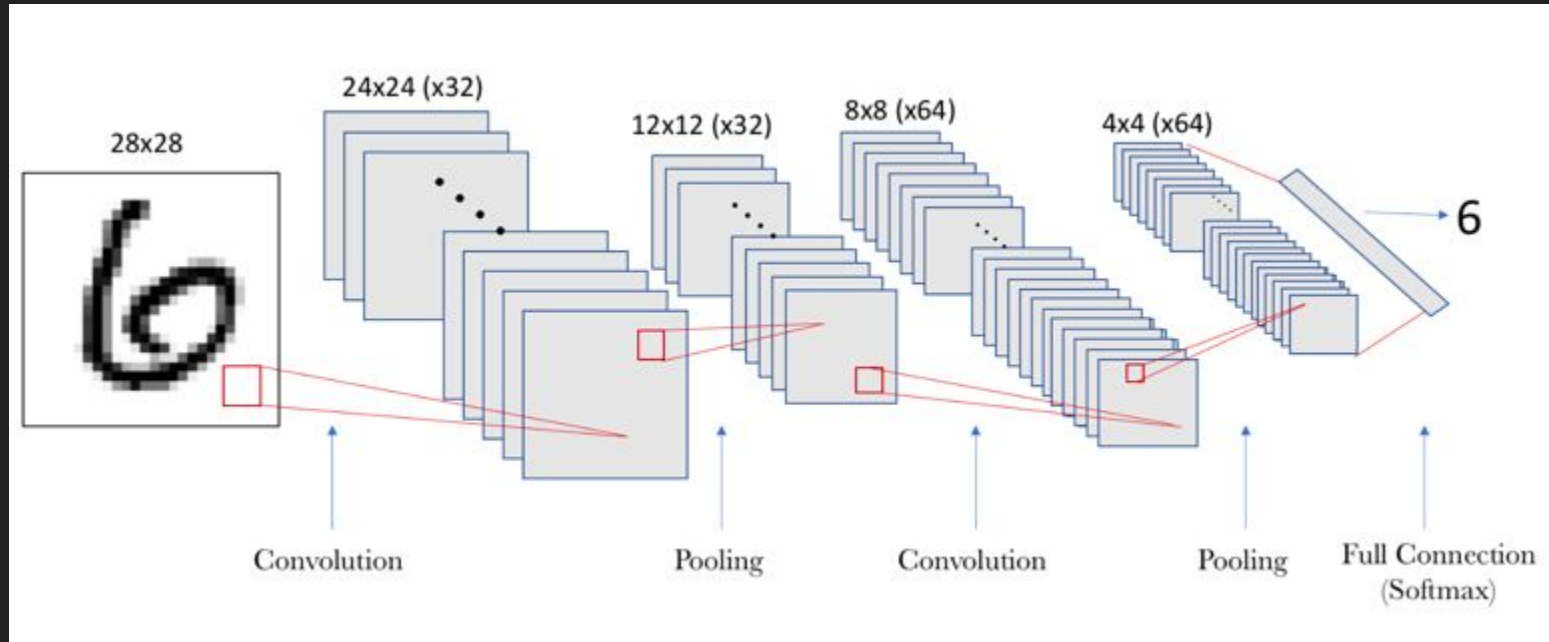
max_pooling2d_1 (MaxPooling2D)	(None, 12, 12, 32)	0

conv2d_2 (Conv2D)	(None, 8, 8, 64)	51264

max_pooling2d_2 (MaxPooling2D)	(None, 4, 4, 64)	0

dense_1 (Dense)	(None, 10)	10250
=====		
Total params: 62,346		
Trainable params: 62,346		
Non-trainable params: 0		

Ejemplo 1 (capa convolucional y pooling)



Ejemplo 2 (completa)

Crearemos una RN Convolutiva, cuya arquitectura será la siguiente:

- Una capa convolutiva 3×3 (sin paddings) seguida de una capa de MaxPooling de 2×2
- Una capa convolutiva 3×3 (sin paddings) seguida de una capa de MaxPooling de 2×2
- Aplanar el resultado para poder aplicar sobre la capa full-connected

Ejemplo 2

Crearemos una RN Convolutiva, cuya arquitectura será la siguiente:

- Una capa convolutiva 3×3 (sin paddings) seguida de una capa de MaxPooling de 2×2
- Una capa convolutiva 3×3 (sin paddings) seguida de una capa de MaxPooling de 2×2
- Aplanar el resultado para poder aplicar sobre la capa full-connected

```
import tensorflow as tf

model = tf.keras.models.Sequential()

# Añadimos la primera capa
model.add(tf.keras.layers.Conv2D(64,(3,3), activation = 'relu', input_shape = (128,128,3)))
model.add(tf.keras.layers.MaxPooling2D(pool_size = (2,2)))

# Añadimos la segunda capa
model.add(tf.keras.layers.Conv2D(64,(3,3), activation = 'relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size = (2,2)))

# Hacemos un flatten para poder usar una red fully connected
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(64, activation='relu'))

# Añadimos una capa softmax para que podamos clasificar las imágenes
model.add(tf.keras.layers.Dense(2, activation='softmax'))

model.compile(optimizer="rmsprop",
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

Ejemplo

Volvamos al ejemplo que detecta neumonía a partir de imágenes radiográficas