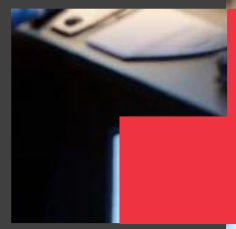
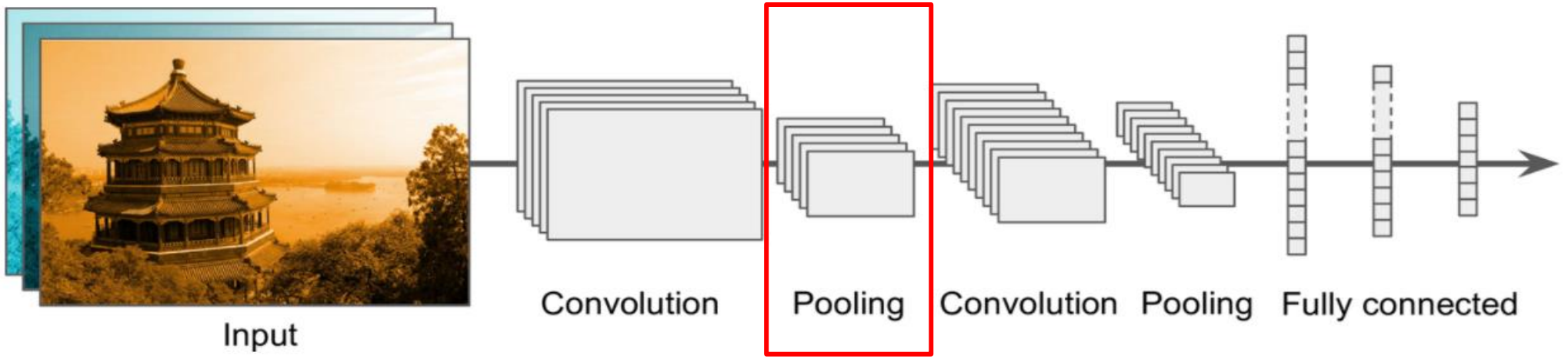
THE BRIDGE

DL en Imágenes

Capas de Pooling y Densas




Input
Convolution
Pooling
Convolution
Pooling
Fully connected

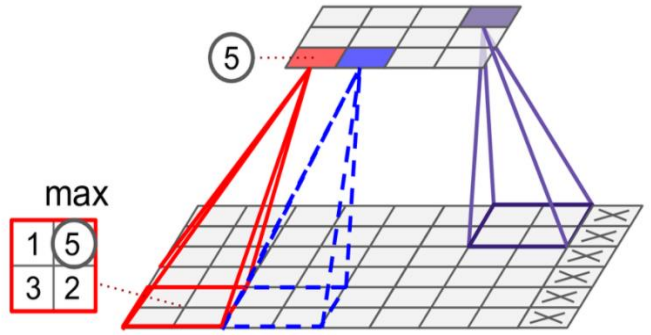
Pooling layer

Las pooling layers se encargan de realizar un submuestreo de sus inputs, de cara a reducir la dimensionalidad, el uso de memoria y el número de parámetros.

Pooling layer

Las pooling layers se encargan de realizar un submuestreo de sus inputs, de cara a reducir la dimensionalidad, el uso de memoria y el número de parámetros.

Las capas de pooling tiene también un kernel y un stride para decidir la cantidad de pixels que se desplaza el kernel.

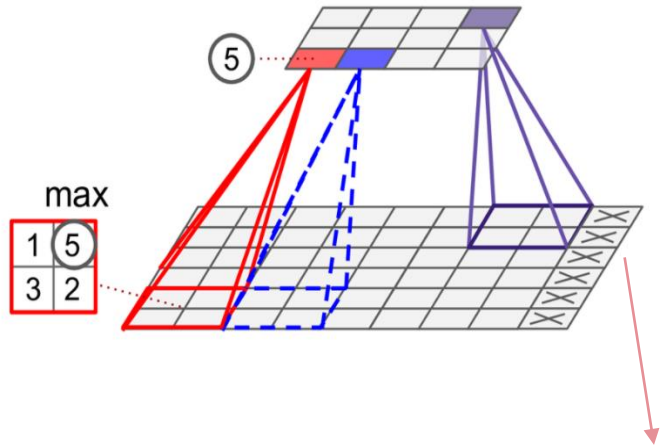


Pooling layer

Las pooling layers se encargan de realizar un submuestreo de sus inputs, de cara a reducir la dimensionalidad, el uso de memoria y el número de parámetros.

Las capas de pooling tiene también un kernel y un stride para decidir la cantidad de pixels que se desplaza el kernel.

Pero ahora no se aplica una "regresión neuronal", a los datos contenidos en el kernel se le aplican típicamente dos posibles funciones: max() o mean(), es decir se obtiene el máximo o la media.



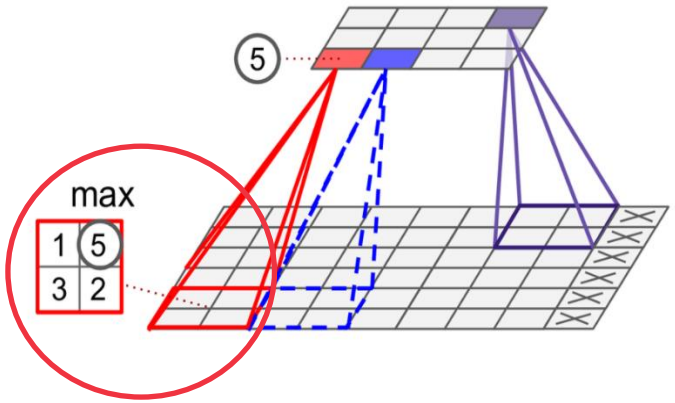
La "imagen" inferior probablemente será un mapa de características (la salida de un filtro de una capa convolucional)

Pooling layer

Las pooling layers se encargan de realizar un submuestreo de sus inputs, de cara a reducir la dimensionalidad, el uso de memoria y el número de parámetros.

Las capas de pooling tiene también un kernel y un stride para decidir la cantidad de pixels que se desplaza el kernel.

Pero ahora no se aplica una "regresión neuronal", a los datos contenidos en el kernel se le aplican típicamente dos posibles funciones: max() o mean(), es decir se obtiene el máximo o la media.

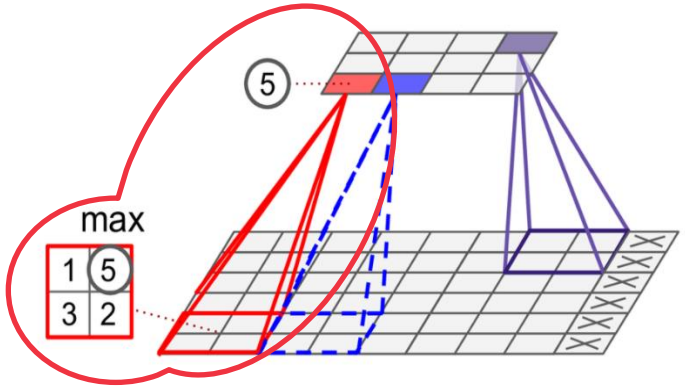


Pooling layer

Las pooling layers se encargan de realizar un submuestreo de sus inputs, de cara a reducir la dimensionalidad, el uso de memoria y el número de parámetros.

Las capas de pooling tiene también un kernel y un stride para decidir la cantidad de pixels que se desplaza el kernel.

Pero ahora no se aplica una "regresión neuronal", a los datos contenidos en el kernel se le aplican típicamente dos posibles funciones: max() o mean(), es decir se obtiene el máximo o la media.

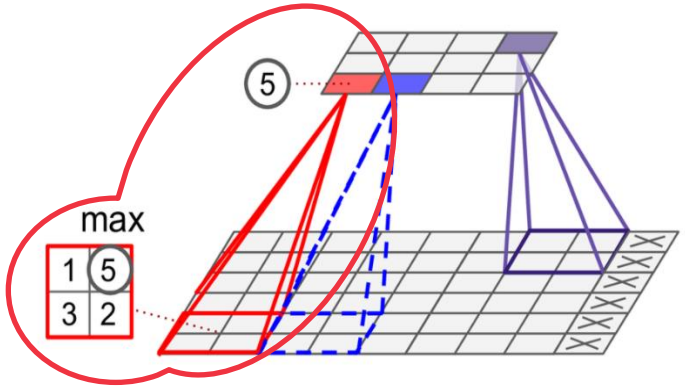


Pooling layer

Las pooling layers se encargan de realizar un submuestreo de sus inputs, de cara a reducir la dimensionalidad, el uso de memoria y el número de parámetros.

Las capas de pooling tiene también un kernel y un stride para decidir la cantidad de pixels que se desplaza el kernel.

Pero ahora no se aplica una "regresión neuronal", a los datos contenidos en el kernel se le aplican típicamente dos posibles funciones: max() o mean(), es decir se obtiene el máximo o la media.



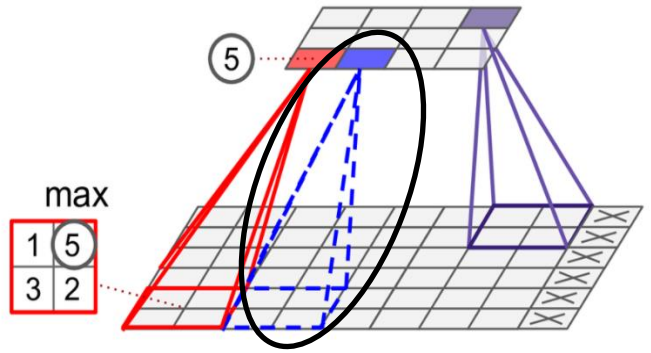
Si hubiera sido un pooling average, el resultado habría sido 11/4 = 2,75

Pooling layer

Las pooling layers se encargan de realizar un submuestreo de sus inputs, de cara a reducir la dimensionalidad, el uso de memoria y el número de parámetros.

Las capas de pooling tiene también un kernel y un stride para decidir la cantidad de pixels que se desplaza el kernel.

Pero ahora no se aplica una "regresión neuronal", a los datos contenidos en el kernel se le aplican típicamente dos posibles funciones: max() o mean(), es decir se obtiene el máximo o la media.



El kernel se desplaza "stride" pixels y hace un nuevo pooling

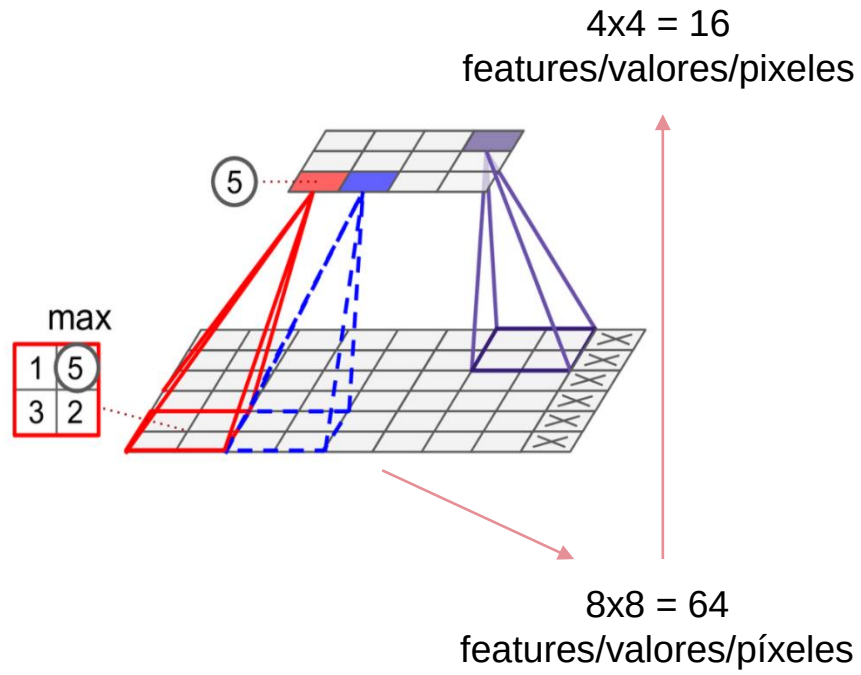
Pooling layer

Las pooling layers se encargan de realizar un submuestreo de sus inputs, de cara a reducir la dimensionalidad, el uso de memoria y el número de parámetros.

Las capas de pooling tiene también un kernel y un stride para decidir la cantidad de pixels que se desplaza el kernel.

Pero ahora no se aplica una "regresión neuronal", a los datos contenidos en el kernel se le aplican típicamente dos posibles funciones: max() o mean(), es decir se obtiene el máximo o la media.

En general se pasa de un mapa de características de $m \times n$ features a una salida de $\frac{m}{a} \times \frac{n}{b}$ features donde a y b son las dimensiones del kernel/stride… Por eso pasamos de 8x8 a $\frac{8}{2} \times \frac{8}{2} \rightarrow 4x4$



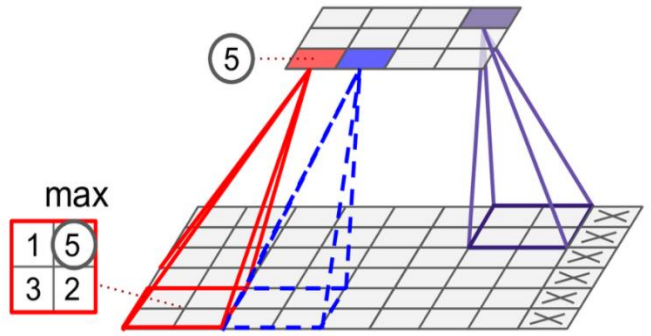
Pooling layer

Las pooling layers se encargan de realizar un submuestreo de sus inputs, de cara a reducir la dimensionalidad, el uso de memoria y el número de parámetros.

Las capas de pooling tiene también un kernel y un stride para decidir la cantidad de pixels que se desplaza el kernel.

Pero ahora no se aplica una "regresión neuronal", a los datos contenidos en el kernel se le aplican típicamente dos posibles funciones: max() o mean(), es decir se obtiene el máximo o la media.

Los hiperparámetros de una capa de pooling son su función de pooling (max, mean, etc) y los tamaños de su kernel y su stride, que suelen ser iguales. Una capa de pooling no es entrenable, no tiene parámetros

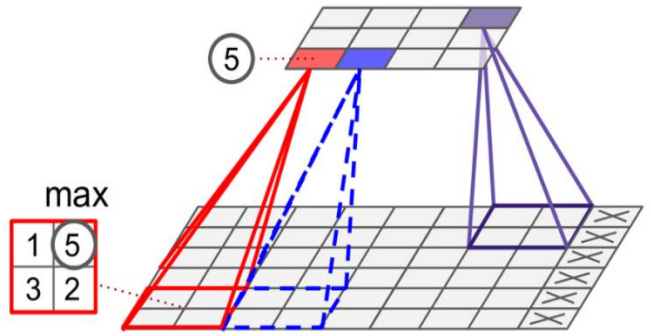


Pooling layer

Las pooling layers se encargan de realizar un submuestreo de sus inputs, de cara a reducir la dimensionalidad, el uso de memoria y el número de parámetros.

Las capas de pooling tiene también un kernel y un stride para decidir la cantidad de pixels que se desplaza el kernel.

Pero ahora no se aplica una "regresión neuronal", a los datos contenidos en el kernel se le aplican típicamente dos posibles funciones: max() o mean(), es decir se obtiene el máximo o la media.



Los hiperparámetros de una capa de pooling son su función de pooling (max, mean, etc) y los tamaños de su kernel y su stride, que suelen ser iguales. Una capa de pooling no es entrenable, no tiene parámetros

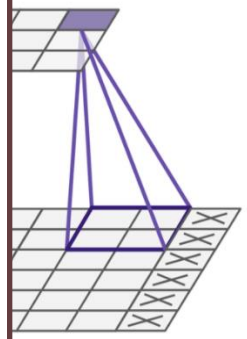
Pooling layer

Las pool educir la
dimensio

Las capa pixels que se
desplaza

Pero aho
datos co
posibles
máximo

Los hipe
función
kernel y
pooling no es entrenable, no tiene parámetros



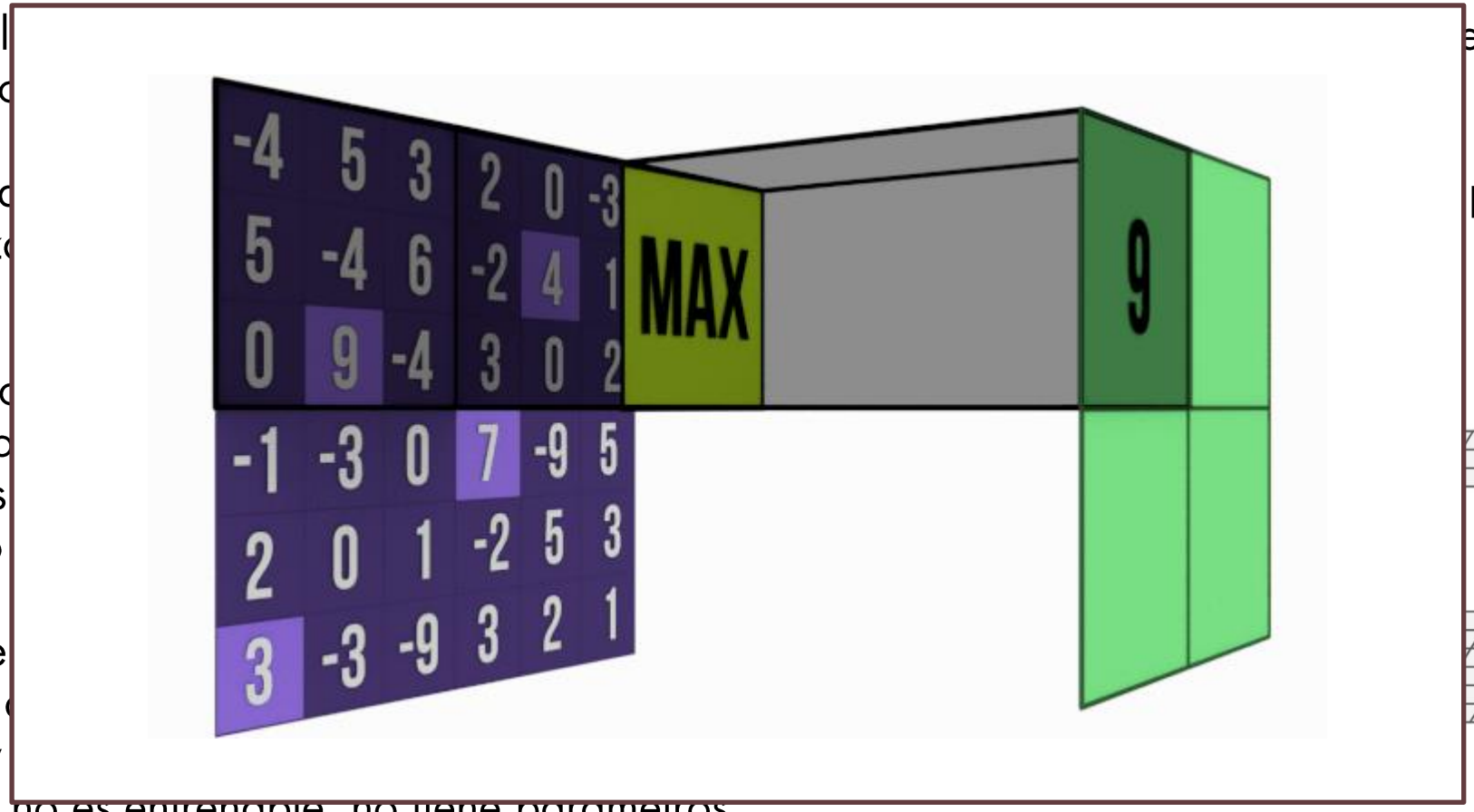
Pooling layer

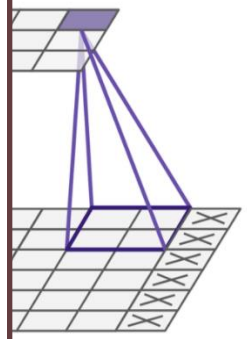
Las pool educir la
dimensio

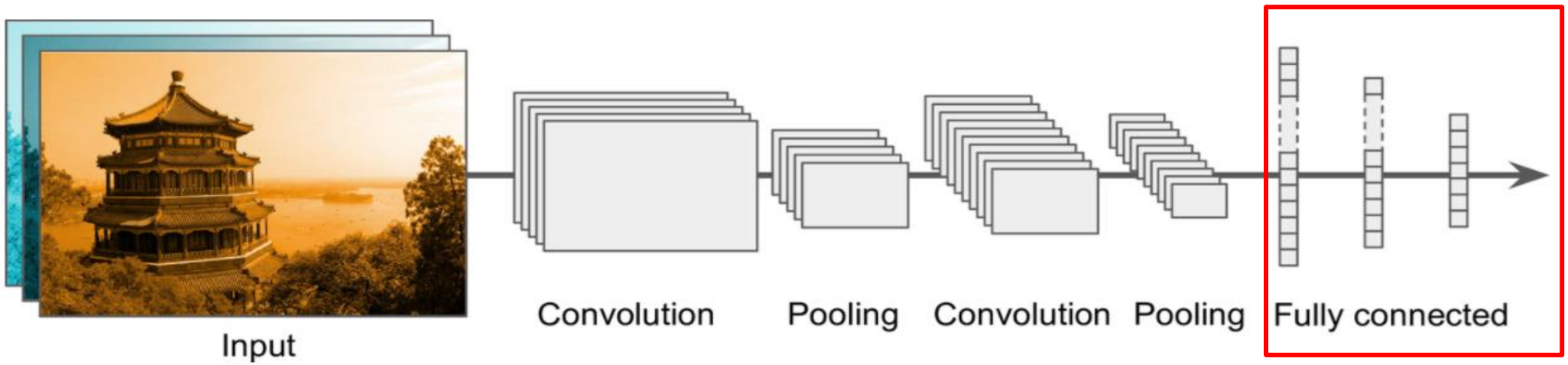
Las capa pixels que se
desplaza

Pero aho
datos co
posibles
máximo

Los hipe
función
kernel y
pooling no es entrenable, no tiene parámetros





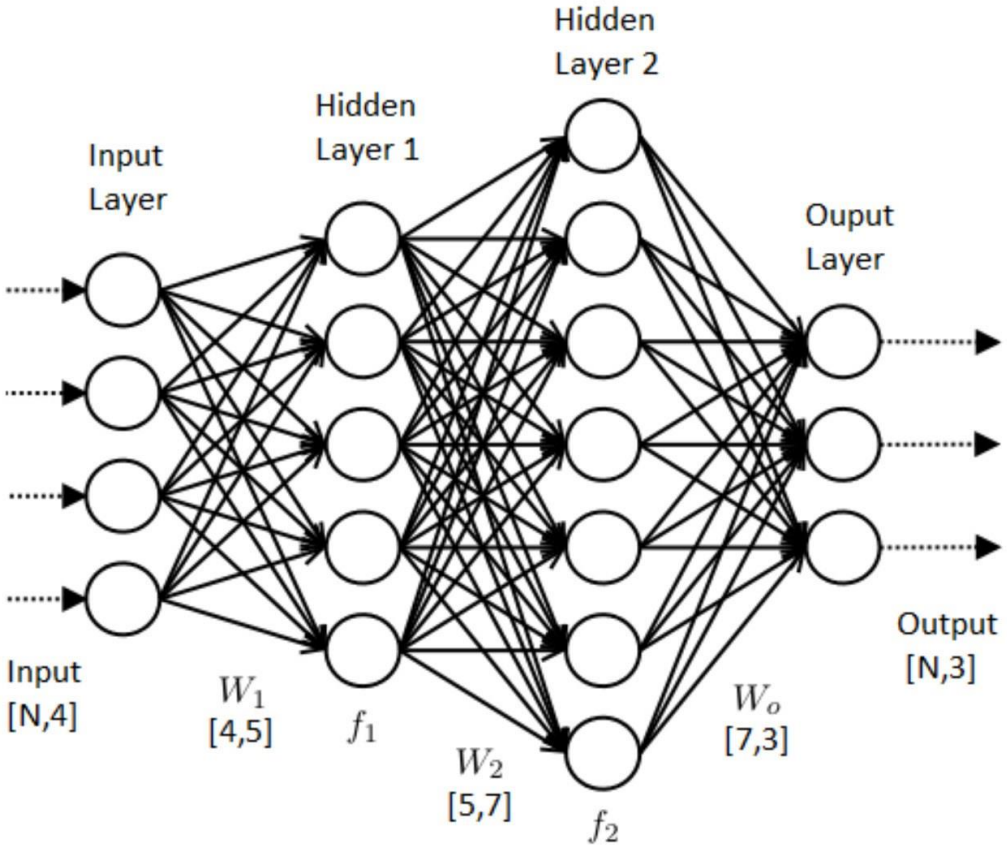

Input
Convolution
Pooling
Convolution
Pooling
Fully connected

Fully connected layers

Tras **una serie de capas convolucionales-pooling**, las features se aplanan y entran como un vector 1D a una **fully connected layers (un MLP)**, que son las capas de neuronas que ya conocemos.

Fully connected layers

Tras **una serie de capas convolucionales-pooling**, las features se aplanan y entran como un vector 1D a una **fully connected layers (un MLP)**, que son las capas de neuronas que ya conocemos.



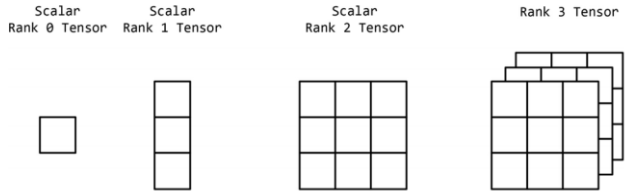
Consideraciones finales

Lo normal es trabajar con imágenes en escalas de grises (un único valor por pixel entre 0 y 255) o en con RGB. En el primer caso decimos que trabajamos con un único canal y en el segundo con 3 canales (y todo se multiplica por 3, porque habrá 3 juegos de filtros por convolucional, 3 juegos de capas pooling, etc), al final antes de las capas densas "aplanaremos" trodo

Consideraciones finales

Lo normal es trabajar con imágenes en escalas de grises (un único valor por pixel entre 0 y 255) o en con RGB. En el primer caso decimos que trabajamos con un único canal y en el segundo con 3 canales (y todo se multiplica por 3, porque habrá 3 juegos de filtros por convolucional, 3 juegos de capas pooling, etc), al final antes de las capas densas "aplanaremos" trodo

Tendremos que poner los datos en el formato de entrada de la red, que será un tensor, un array de numpy compuesto por la altura x anchura de las imágenes x los canales.

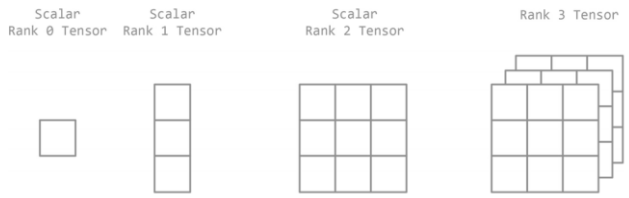


Un batch de imágenes que alimente nuestra red DL Convolucional será un rank 3 tensor con tantos valores por cuadrito como canales:
(nº de imágenes del batch, resolución vertical, resolución horizontal, canales)
Es decir, el shape de un batch (X) de 32 imágenes de 28x28 píxeles con RGB será:
(32,28,28.,3)

Consideraciones finales

Lo normal es trabajar con imágenes en escalas de grises (un único valor por pixel entre 0 y 255) o en con RGB. En el primer caso decimos que trabajamos con un único canal y en el segundo con 3 canales (y todo se multiplica por 3, porque habrá 3 juegos de filtros por convolucional, 3 juegos de capas pooling, etc), al final antes de las capas densas "aplanaremos" trodo

Tendremos que poner los datos en el formato de entrada de la red, que será un tensor, un array de numpy compuesto por la altura x anchura de las imágenes x los canales.



Un batch de imágenes que alimente nuestra red DL Convolucional será un rank 3 tensor con tantos valores por cuadrito como canales:
(nº de imágenes del batch, resolución vertical, resolución horizontal, canales)
Es decir, el shape de un batch (X) de 32 imágenes de 28x28 píxeles con RGB será: (32,28,28.,3)

Normalizar

Otro punto importante es normalizar los datos para mejorar los tiempos de entrenamiento y el performance del modelo. Dependerá de la codificación de la imagen, pero lo habitual es que los canales de colores vayan de 0 a 255.

Consideraciones finales

- Número de pesos que tiene una capa convolucional:

$$\text{Num\_weights} = (\text{Tam\_del\_kernel} * \text{num\_canales} + 1) * \text{numero\_de\_filtros}$$

Consideraciones finales

- Número de pesos que tiene una capa convolucional:

Num_weights = (Tam_del_kernel * num_canales + 1) * numero_de_filtros

- "Resolución" a la salida de una capa convolucional o de pooling: (es decir si entra una imagen de HxW de tamaño y n canales, cómo queda a la salida de un filtro o de una capa de pooling:

$$H_{out} = \left\lfloor \frac{H_{in} + 2 \times \text{padding}[0] - \text{dilation}[0] \times (\text{kernel\_size}[0] - 1) - 1}{\text{stride}[0]} + 1 \right\rfloor$$

$$W_{out} = \left\lfloor \frac{W_{in} + 2 \times \text{padding}[1] - \text{dilation}[1] \times (\text{kernel\_size}[1] - 1) - 1}{\text{stride}[1]} + 1 \right\rfloor$$

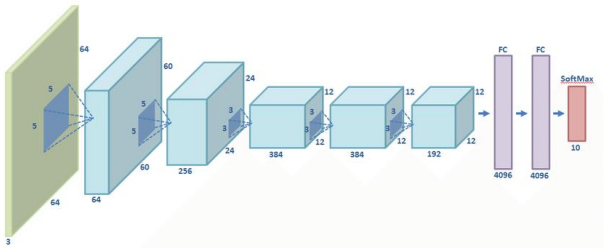
Para nosotros dilation es 1, quien quiera saber más sobre el concepto: https://tinyurl.com/4nwjc6n6

Evolución

ImageNet: https://image-net.org/

Evolución

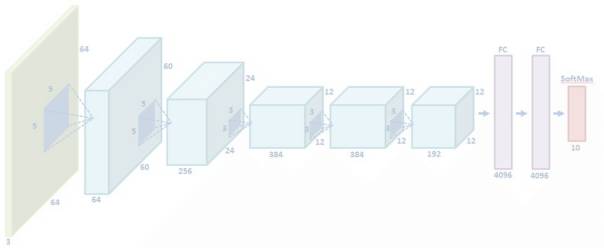
AlexNet (2012)



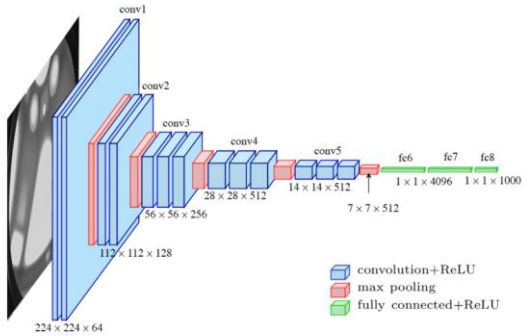
ImageNet: https://image-net.org/

Evolución

AlexNet (2012)



VGG (2014)

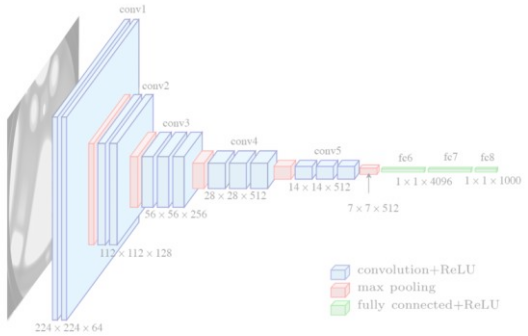


ImageNet: https://image-net.org/

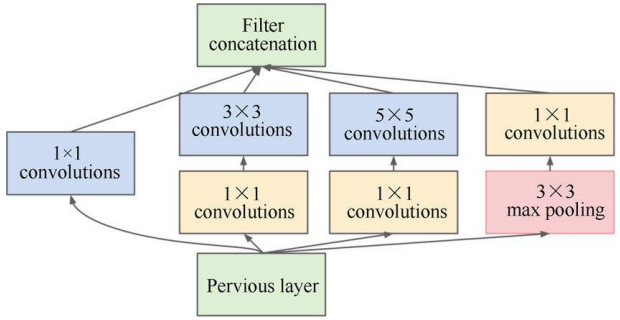
Evolución

AlexNet (2012)

VGG (2014)



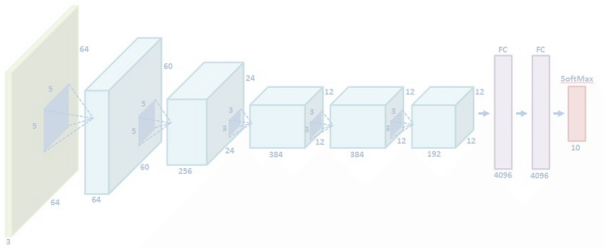
Inception (2014)



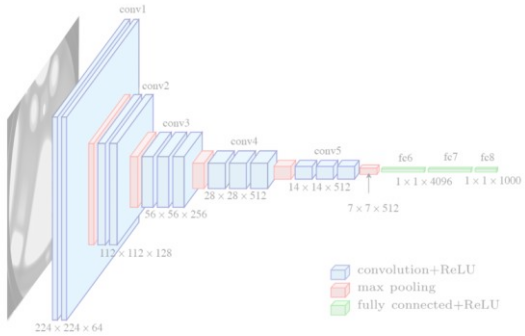
ImageNet: https://image-net.org/

Evolución

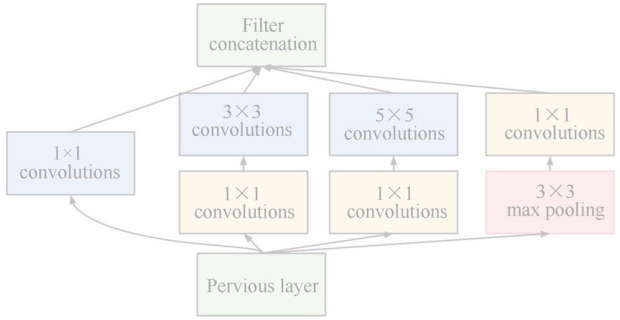
AlexNet (2012)



VGG (2014)



Inception (2014)



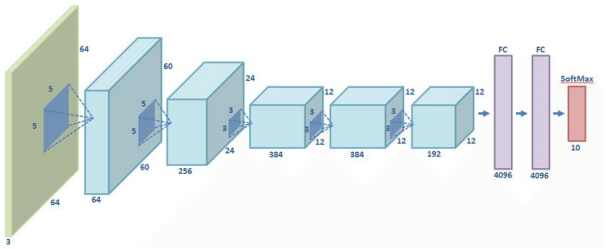
ResNet (2015)

- Muy profunda, muchas capas
- Unidades residuales con skip connections

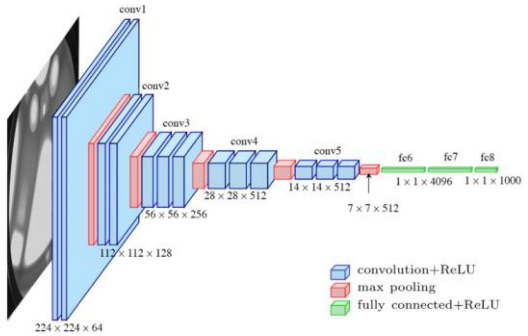
ImageNet: https://image-net.org/

Evolución

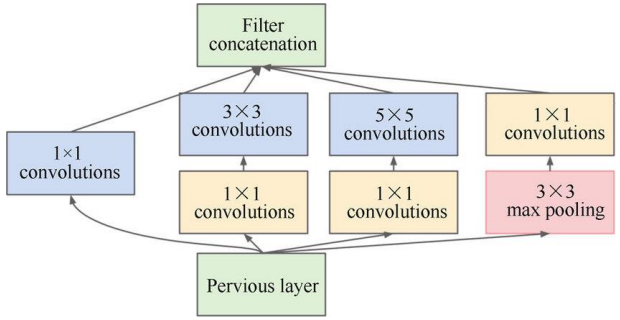
AlexNet (2012)



VGG (2014)



Inception (2014)



ResNet (2015)

- Muy profunda, muchas capas
- Unidades residuales con skip connections

ImageNet: https://image-net.org/


THE
BRIDGE