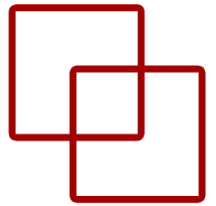
Redes Neurnoales Convolucionales / Recurrentes

Dr. Manuel Castillo-Cara
www.manuelcastillo.eu
Departamento de Inteligencia Artificial
Escuela Técnica Superior de Ingeniería Informática
Universidad Nacional de Educación a Distancia (UNED)

Preliminar

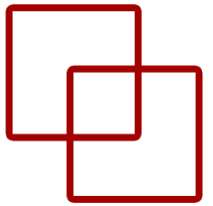


-

UNED

ETS de Ingeniería Informática

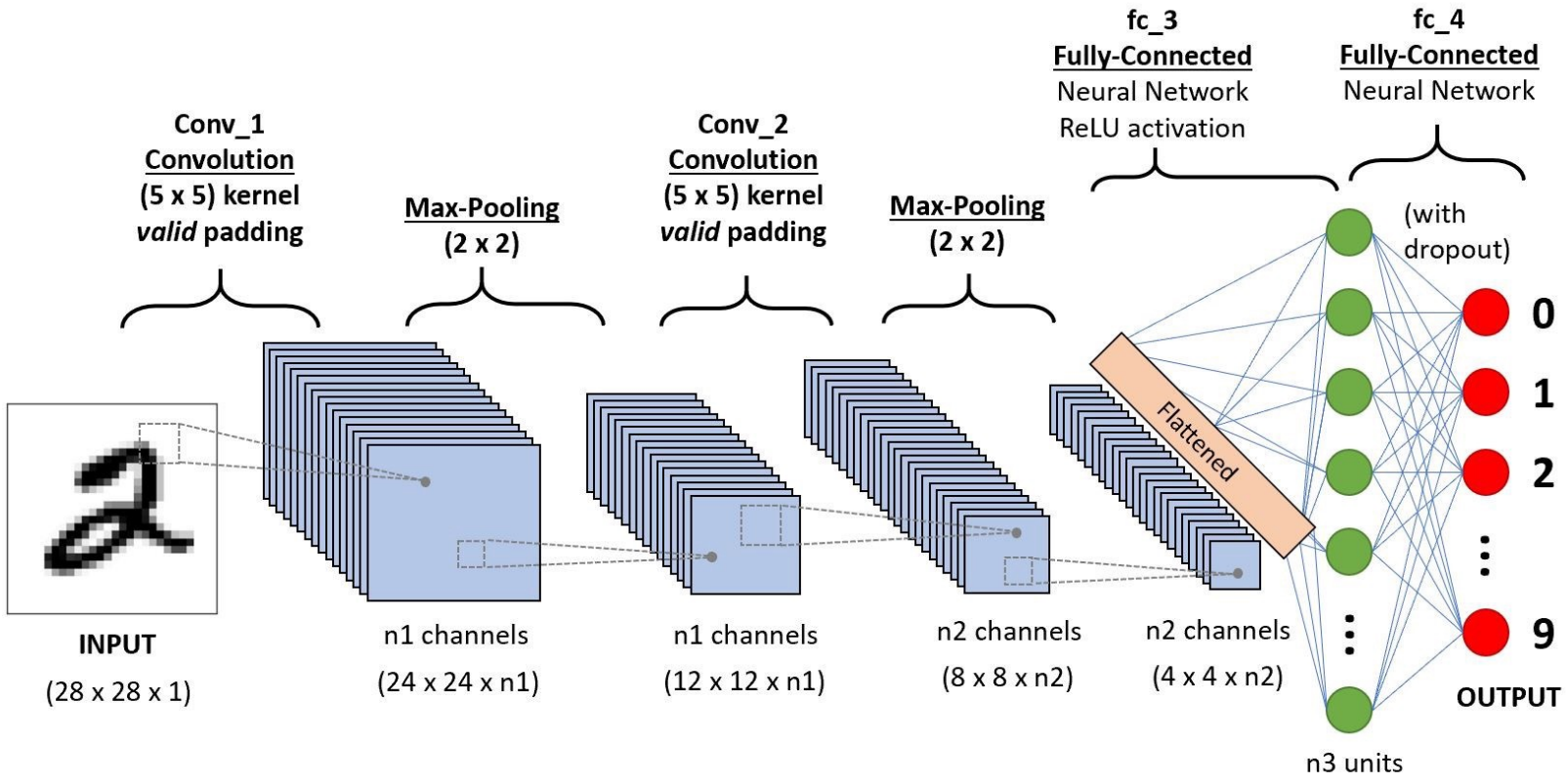
Índice



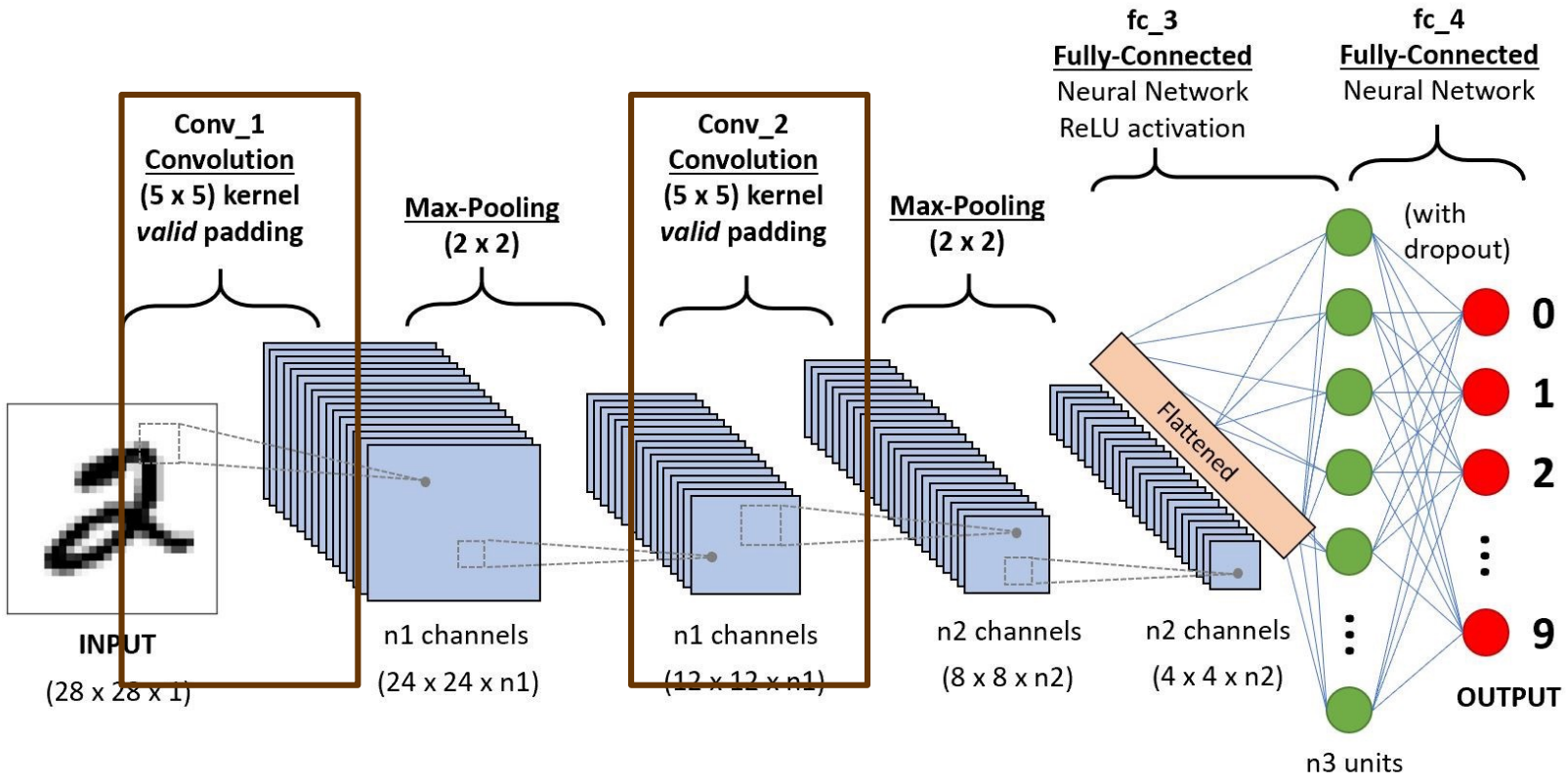
- Redes Neuronales Convolucionales
- Operación Convolución
- Operación Pooling
- Hacia la estandarización
- Transfer learning and finetunning
- CNN Vs RNN

Redes Neuronales Convolucionales

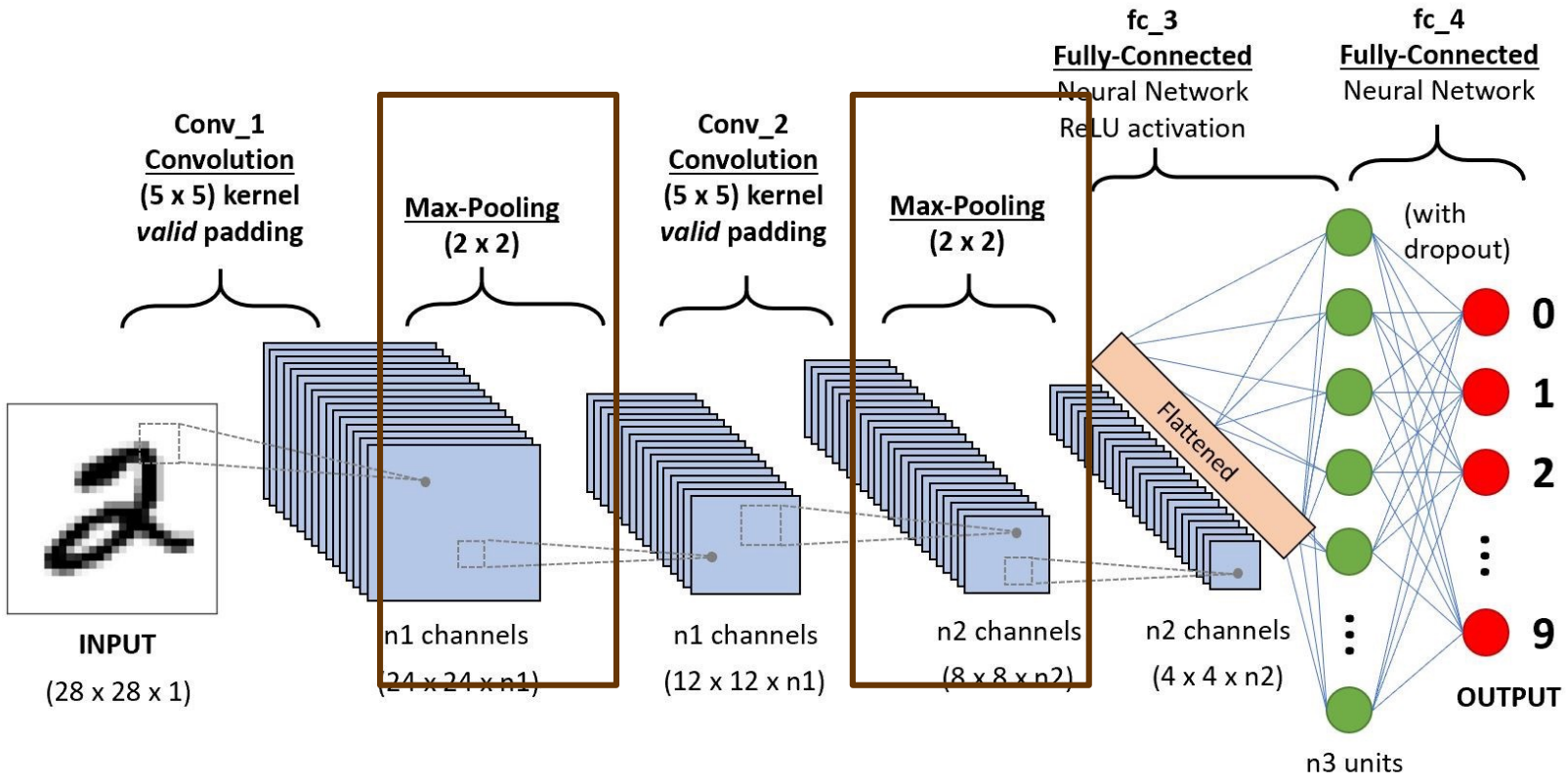
Background


fc_3
Fully-Connected
Neural Network
ReLU activation
fc_4
Fully-Connected
Neural Network
Conv_1
Convolution
(5 x 5) kernel
valid padding
Max-Pooling
(2 x 2)
Conv_2
Convolution
(5 x 5) kernel
valid padding
Max-Pooling
(2 x 2)
(with
dropout)
Flattened
0
1
2
9
INPUT
(28 x 28 x 1)
n1 channels
(24 x 24 x n1)
n1 channels
(12 x 12 x n1)
n2 channels
(8 x 8 x n2)
n2 channels
(4 x 4 x n2)
n3 units
OUTPUT

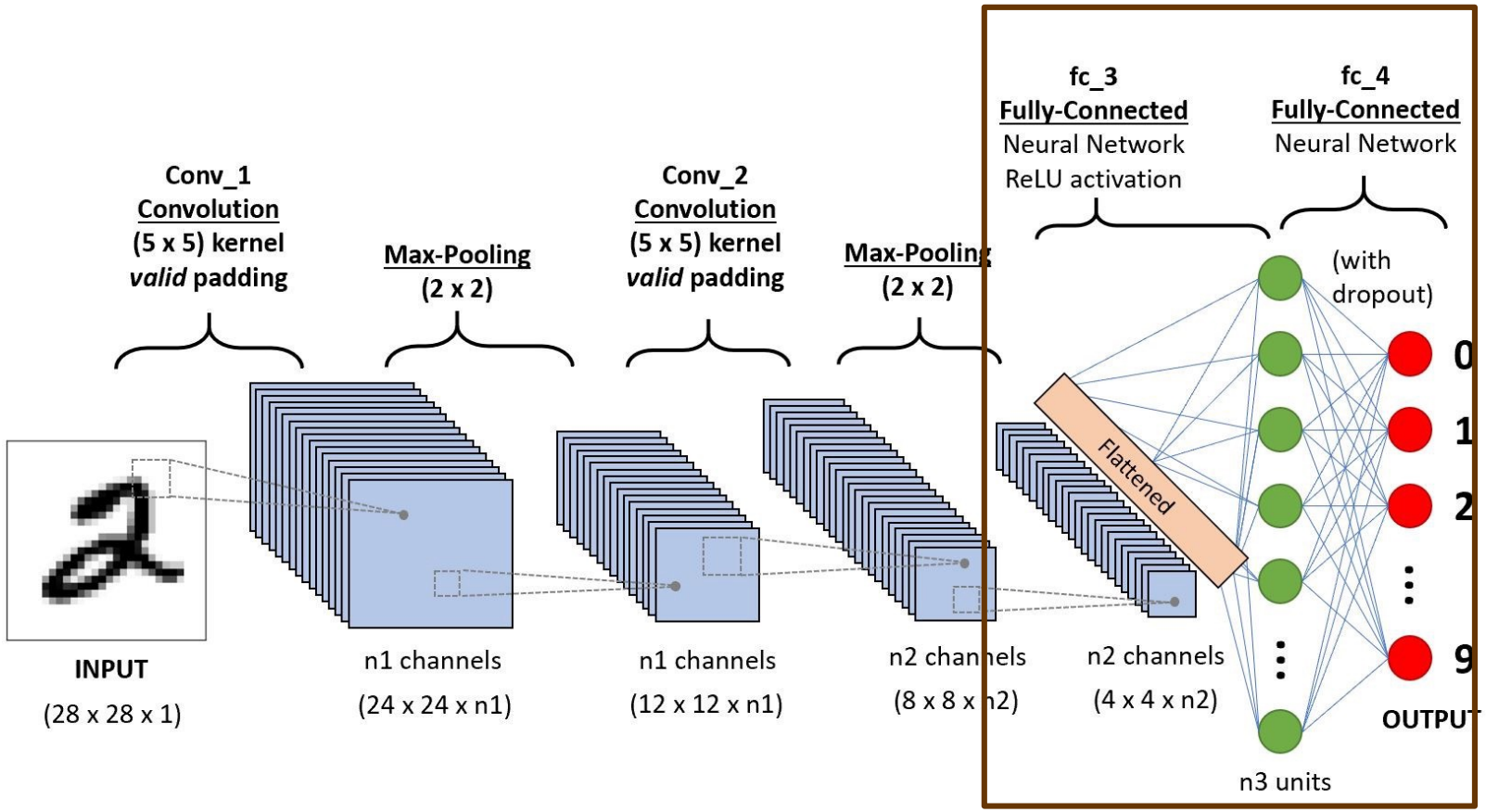
Capas convolucionales



Capas de agrupación *(pooling)*



Capas completamente conectadas



Ejemplo

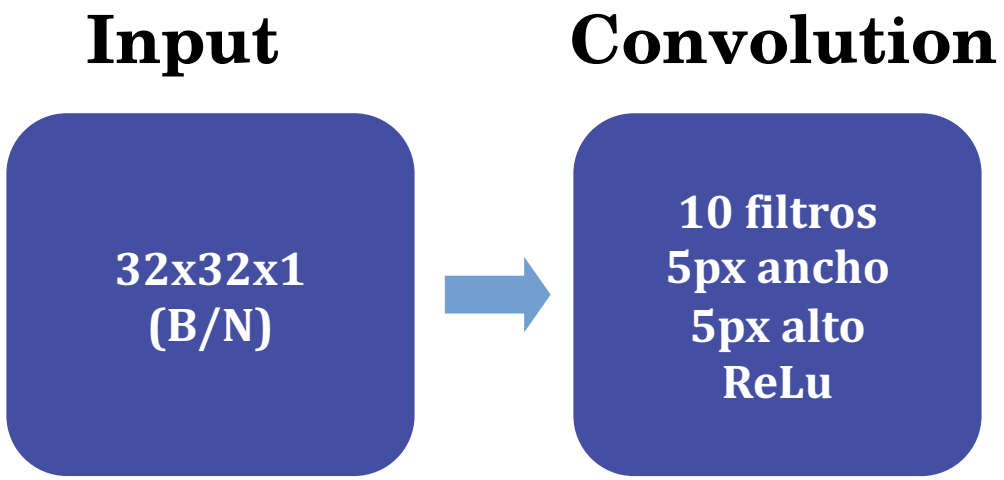
Imagen de entrada (32x32)

Input

32x32x1
(B/N)

Ejemplo

Imagen de entrada (32x32)

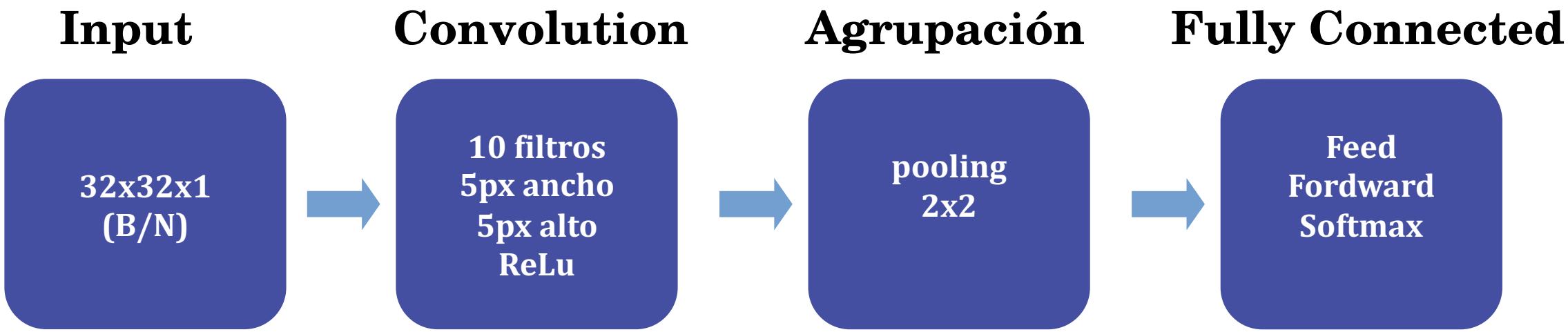


Ejemplo

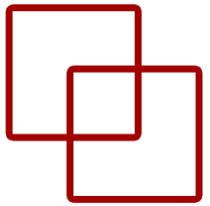
Imagen de entrada (32x32)

Ejemplo

Imagen de entrada (32x32)



Consejos

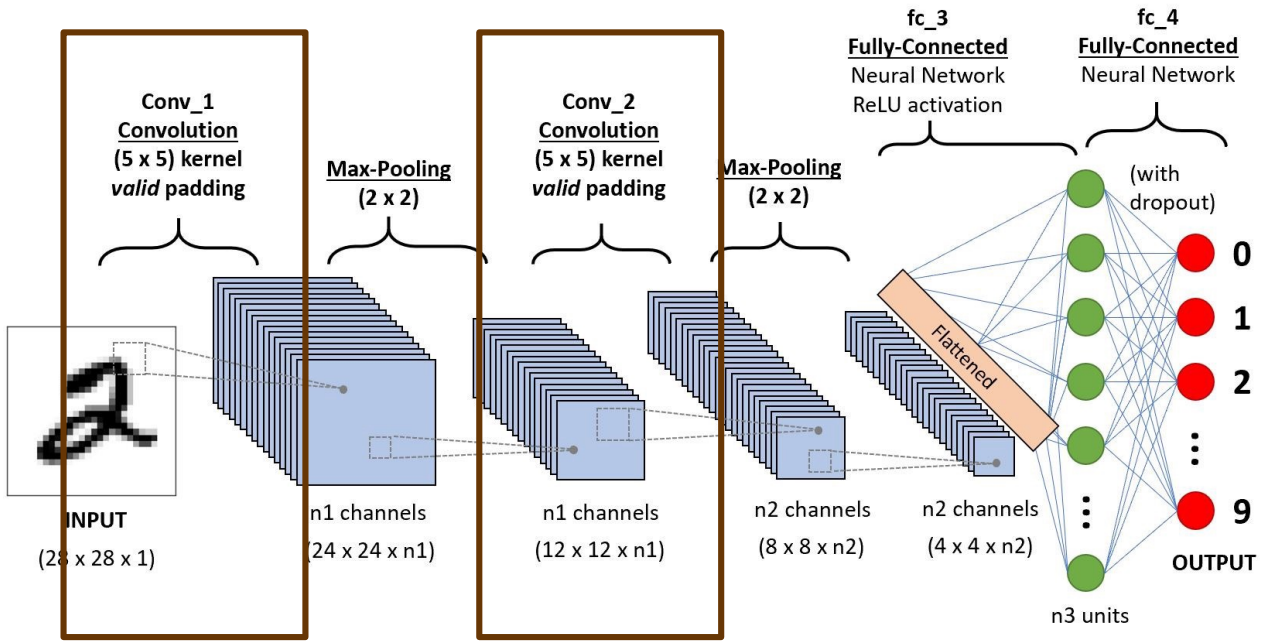


- Dimensiones del campo receptivo de entrada:
- Tamaño del campo receptivo
- Ancho de stride
- Número de filtros
- Relleno
- Pooling
- Preparación de datos
- Arquitectura de patrones
- Dropout

Operación Convolución

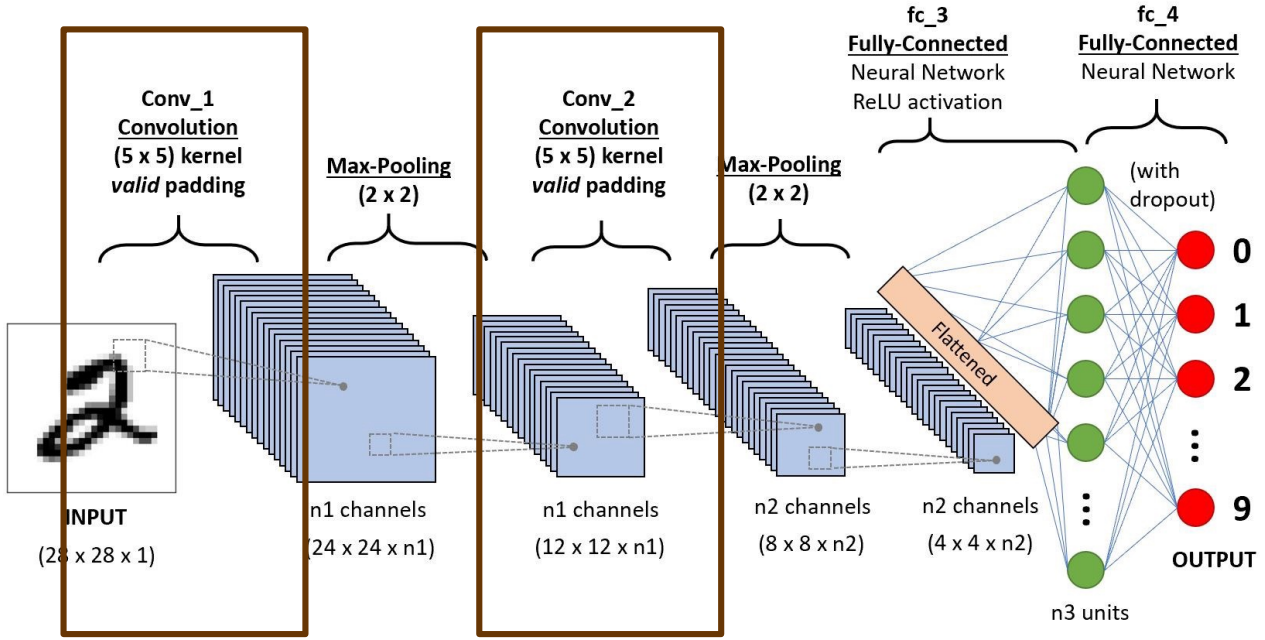
Operación convolución (I)

- La primera capa oculta de una CNN se suele corresponder con una **capa convolucional**, la cual puede entenderse como la codificación necesaria para la aplicación de filtros (conocidos como funciones *kernel*)
- Para una mejor comprensión de la operación convolución, pensemos en una capa de entrada que **codifica** una imagen binaria (B/N) y una convolución con la que queremos representar un filtro de **detección de bordes**.



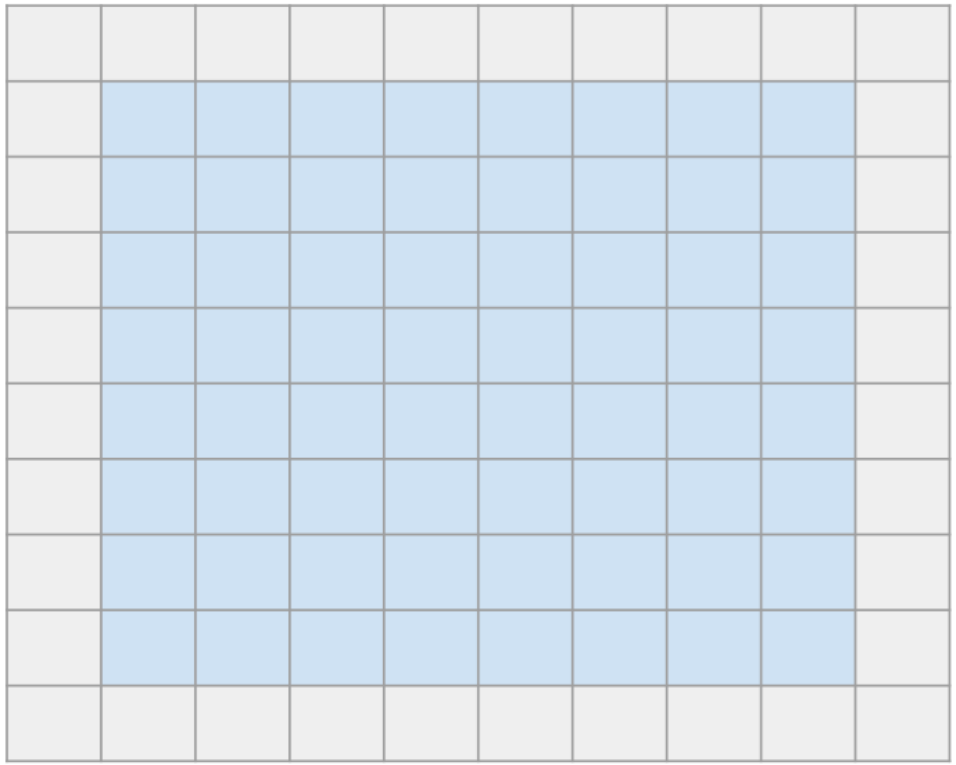
Operación convolución (II)

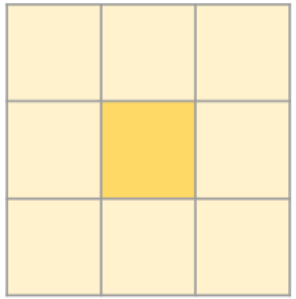
- Sin entrar en más detalles, la búsqueda de bordes se basa en buscar grandes diferencias entre un píxel y los píxeles que le rodean
- El **mismo tipo de operación** la queremos realizar sobre la imagen completa, por lo que aparece una primera cualidad: el uso de **pesos compartidos**
  - Esto significa que los parámetros / condiciones para detectar un borde en un pixel de la imagen deben ser iguales en todas las otras zonas de la imagen



Ejemplo

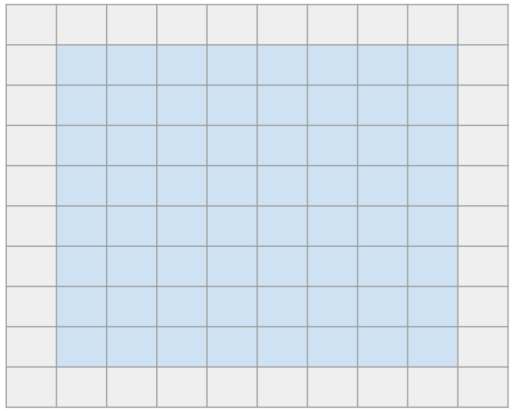
- Asumiendo una imagen de 10x10 píxeles, si la detección de un borde necesita trabajar con los 8 píxeles que le rodean tendríamos lo siguiente

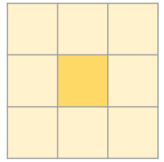




Ejemplo
Operación convolución (I)

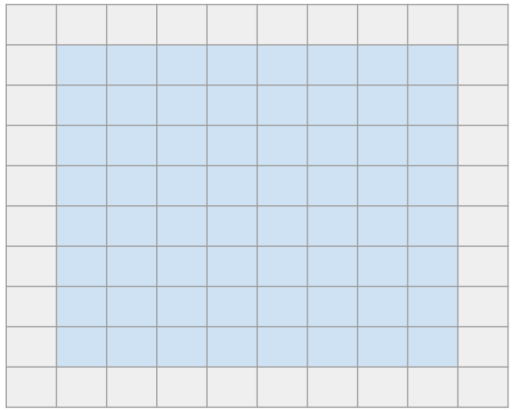
- En la imagen mostramos un filtro que se puede aplicar únicamente sobre las zonas resaltadas

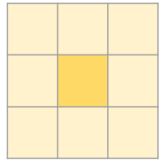




Ejemplo
Operación convolución (I)

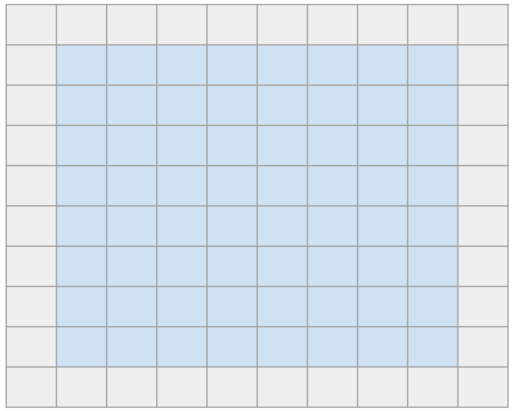
- En la imagen mostramos un filtro que se puede aplicar únicamente sobre las zonas resaltadas
- Trasladando el problema a resolver sobre la topología de una CNN, incluiríamos una capa inicial con 100 neuronas (puede mostrarse visualmente como una matriz de 10x10)

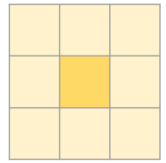




Ejemplo
Operación convolución (I)

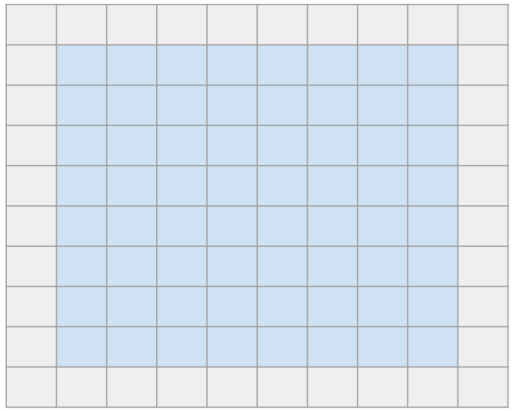
- En la imagen mostramos un filtro que se puede aplicar únicamente sobre las zonas resaltadas
- Trasladando el problema a resolver sobre la topología de una CNN, incluiríamos una capa inicial con 100 neuronas (puede mostrarse visualmente como una matriz de 10x10)
- La primera capa oculta contará con **una neurona por cada resultado** del filtro → tendrá 8x8 neuronas

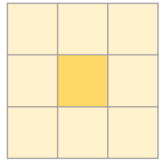




Ejemplo
Operación convolución (II)

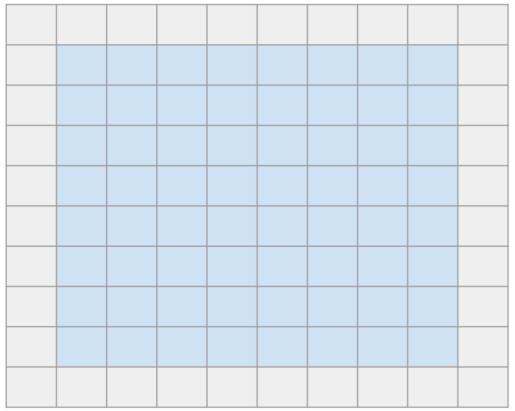
- Cada una de estas 64 neuronas estará conectada con 9 neuronas de la capa inicial, pero los **pesos** utilizados en esta interacción $w_1 ... w_9$ serán **compartidos** por todas las neuronas de esta capa oculta

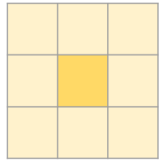




Ejemplo
Operación convolución (II)

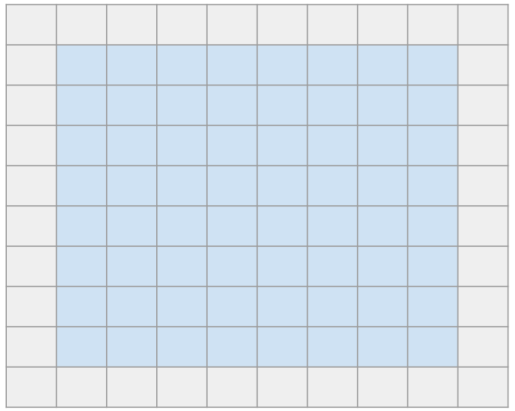
- Cada una de estas 64 neuronas estará conectada con 9 neuronas de la capa inicial, pero los **pesos** utilizados en esta interacción $w_1 ... w_9$ serán **compartidos** por todas las neuronas de esta capa oculta
- Esto **simplifica** el aprendizaje enormemente, además de dar **coherencia** al filtro a realizar

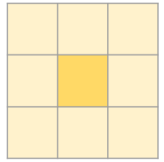




Ejemplo
Operación convolución (II)

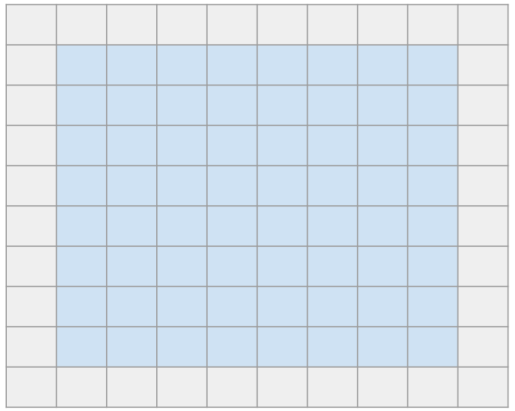
- Cada una de estas 64 neuronas estará conectada con 9 neuronas de la capa inicial, pero los **pesos** utilizados en esta interacción $w_1 ... w_9$ serán **compartidos** por todas las neuronas de esta capa oculta
- Esto **simplifica** el aprendizaje enormemente, además de dar **coherencia** al filtro a realizar
- Una vez aprendida, la función de activación (ReLU normalmente) nos permitirá saber qué píxel de la imagen se **corresponde o no con un borde**

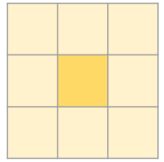




Ejemplo
Operación convolución (III)

- En el ejemplo propuesto, se plantean convoluciones que involucran tanto al píxel bajo estudio, como los **8 píxeles a su alrededor** (3x3)

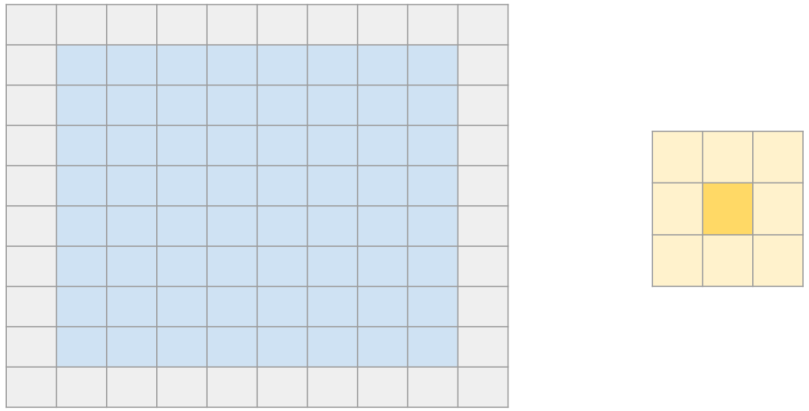




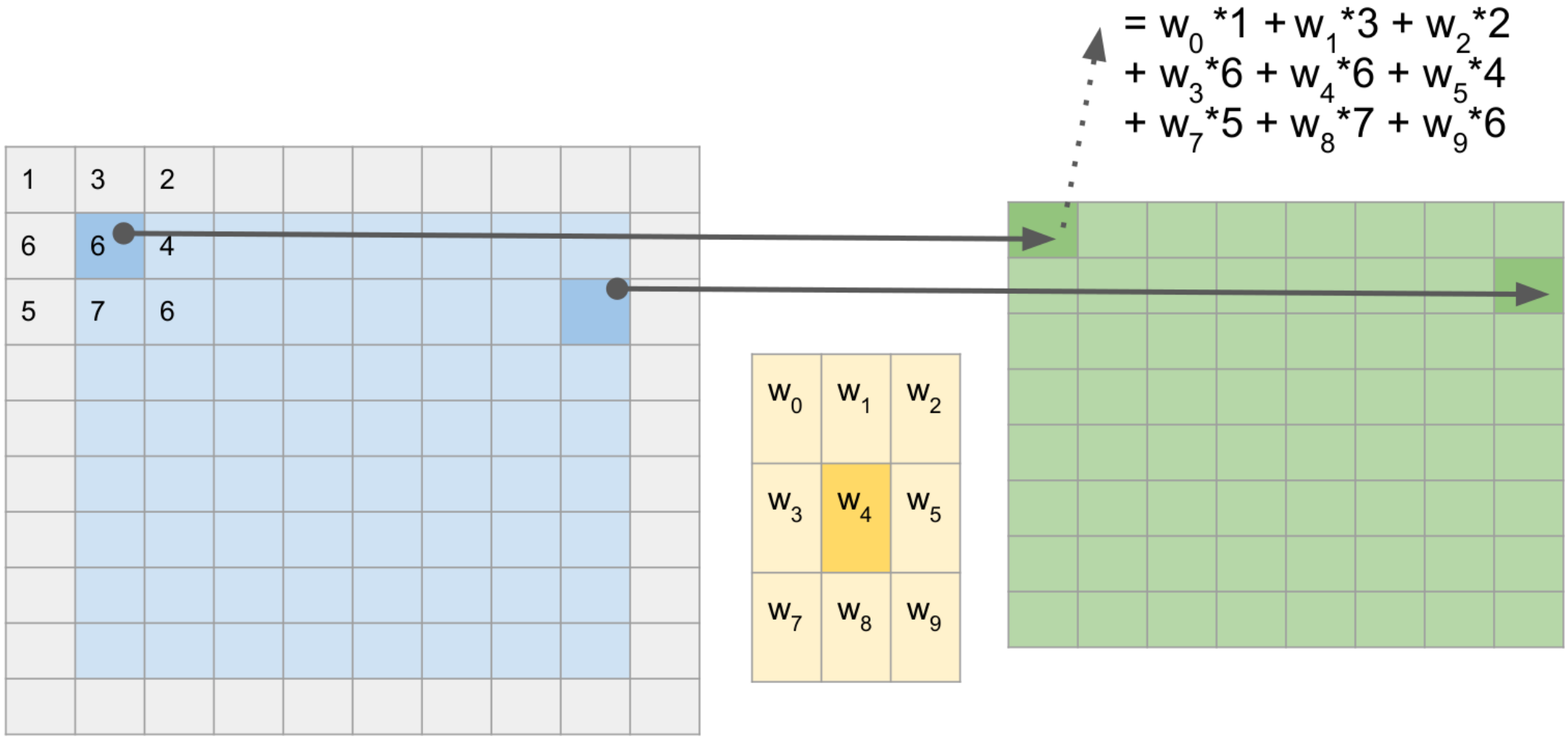
Ejemplo
Operación convolución (III)

- En el ejemplo propuesto, se plantean convoluciones que involucran tanto al píxel bajo estudio, como los **8 píxeles a su alrededor** (3x3)
- Además, se asume que esta operación se realiza uno a uno, pero sólo sobre los píxeles de la imagen que es posible (salvo las filas / columnas 0 y 9), obteniendo como resultado una **capa de tamaño inferior** (10x10 → 8x8)

  → 2 parámetros definen esta configuración: ***padding*** y ***stride***

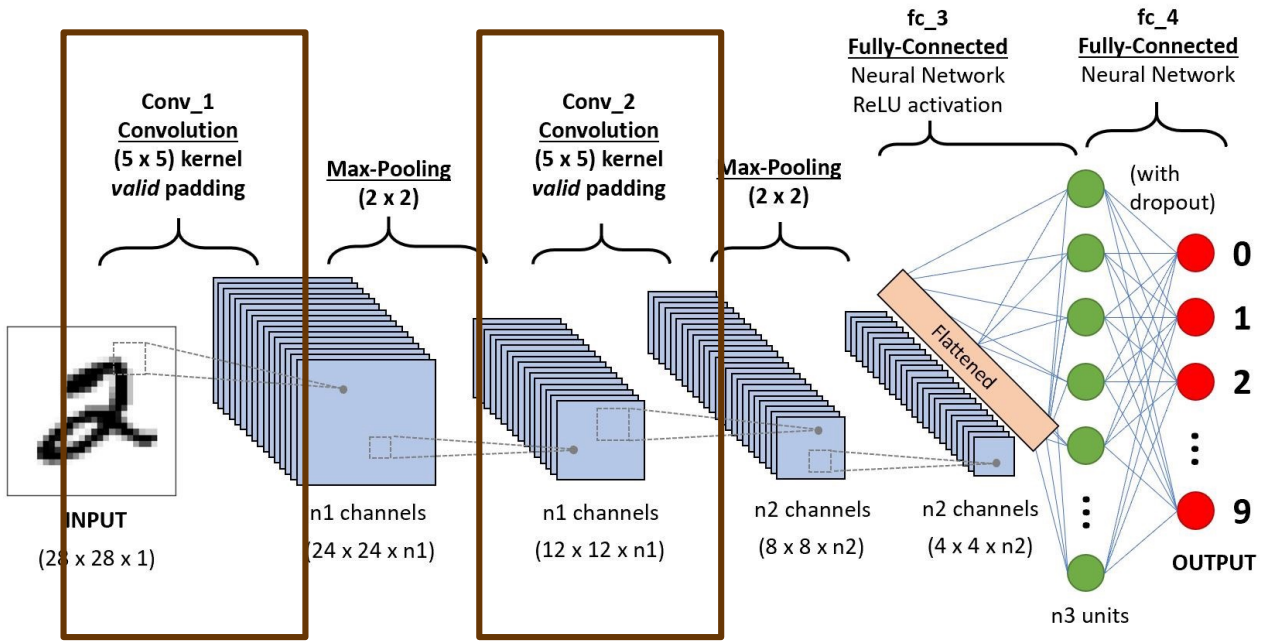


Ejemplo
Operación convolución (IV)



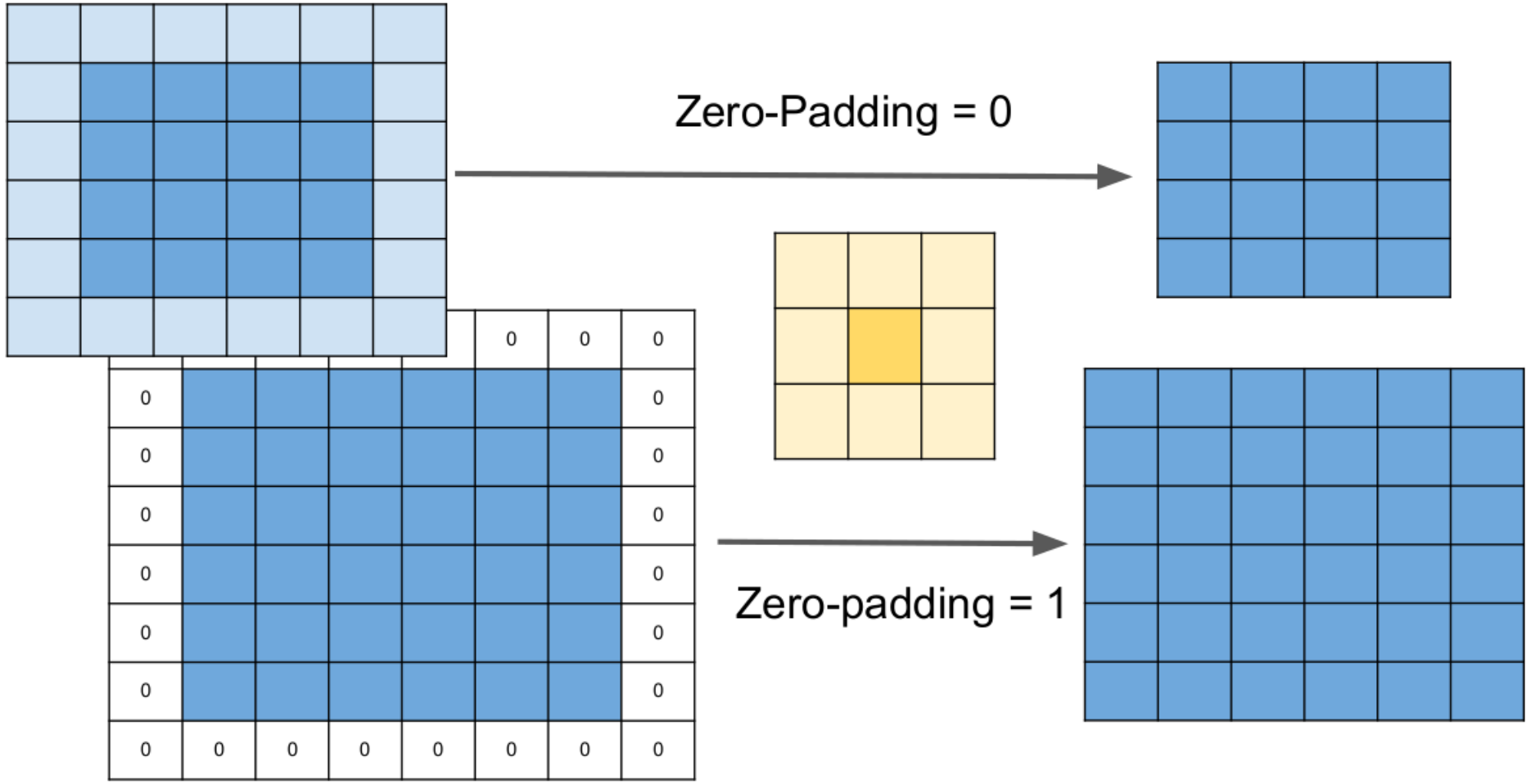
Padding
Definición

- El parámetro ***padding*** (opcional) define una serie de neuronas sintéticas usadas a modo de relleno, de forma que se permite realizar convoluciones sin que la capa resultado sea de un tamaño menor a la de entrada
- Estas neuronas deben tomar un **valor** para la realización de las operaciones, siendo cero el valor más usado.
  - En este caso nos referimos a este parámetro como *zero-padding N* (N = número de neuronas adicionales)



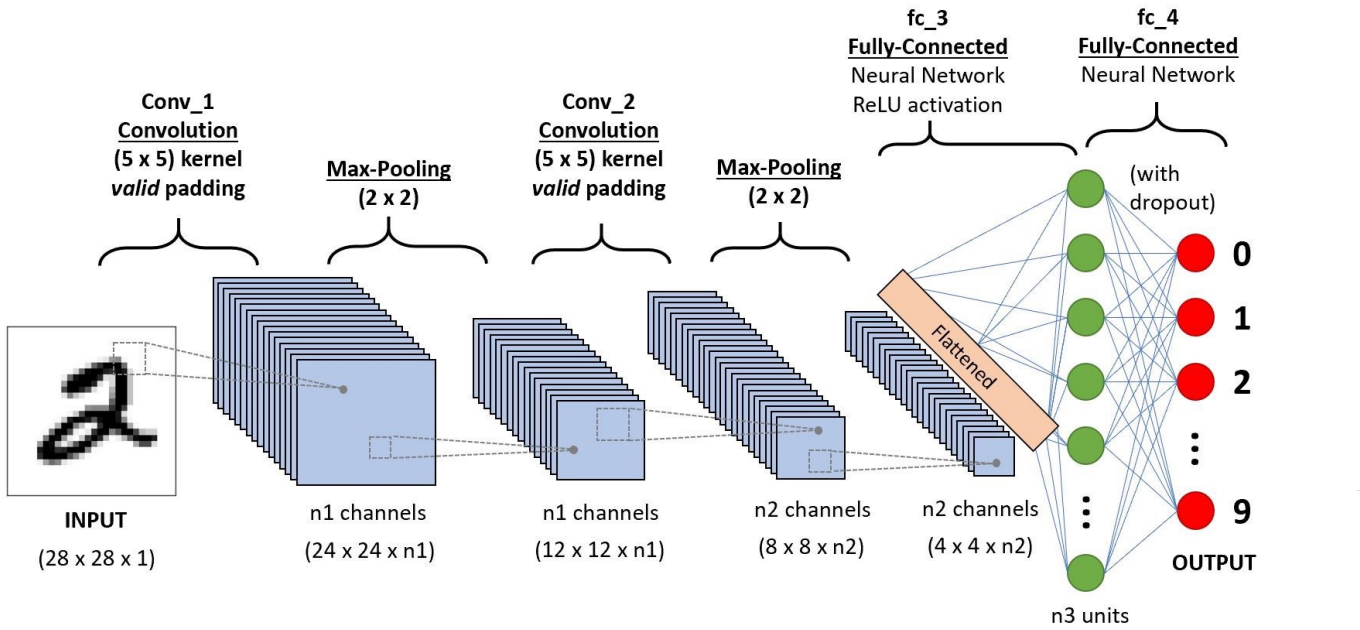
Padding
Ejemplo

- Veamos un ejemplo con imágenes de 6x6 y un filtro de 3x3



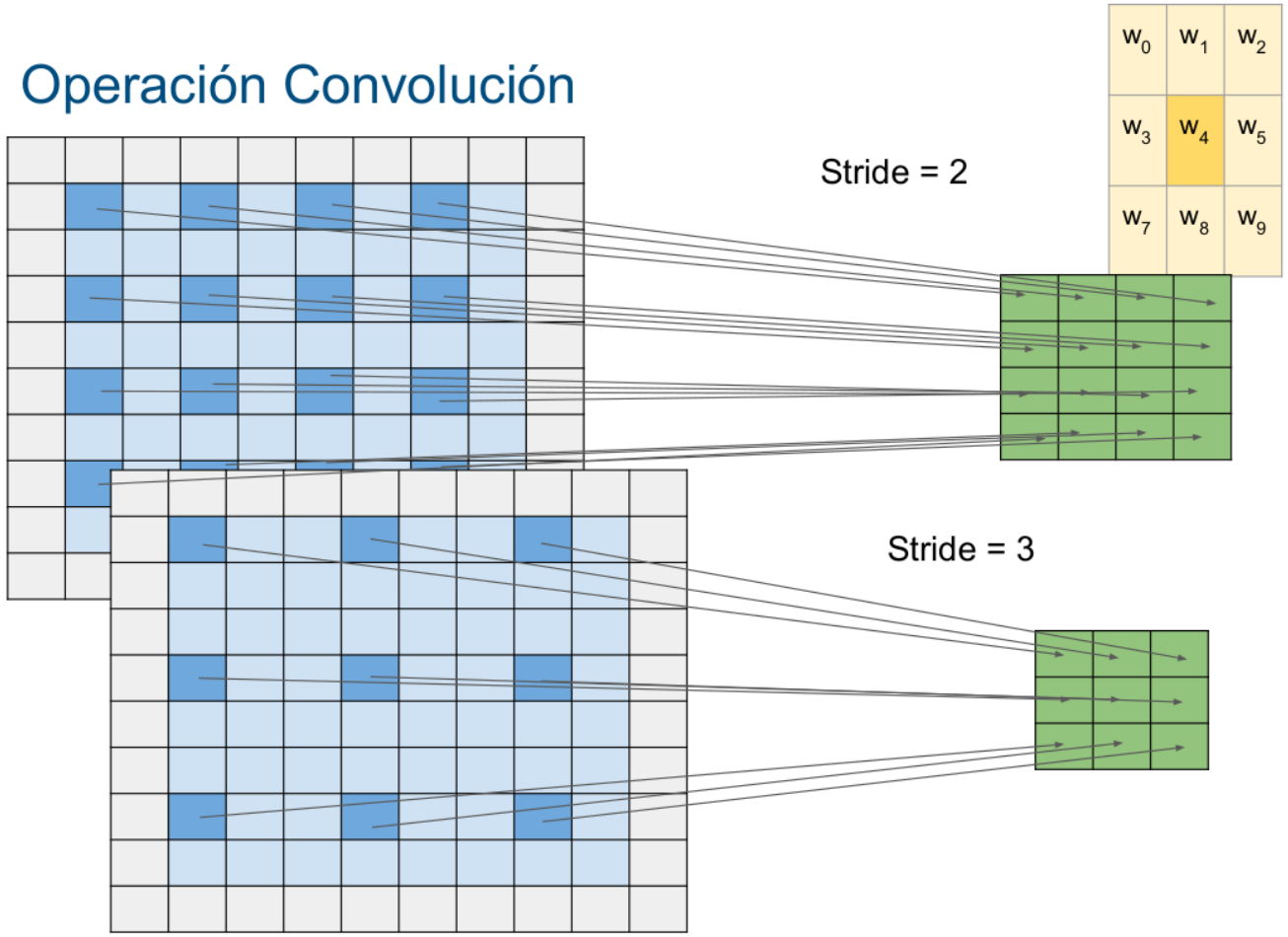
Stride
Definición

- El parámetro ***stride*** nos indica el número de neuronas a avanzar tras la aplicación de una convolución
- Para entender este parámetro recurrimos a la organización espacial: Usando una matriz de dos dimensiones, un valor de ***stride 2*** nos indicará que tras aplicar una convolución:
  - La siguiente se realizará **dos columnas a la derecha**, y
  - tras terminar la fila actual, la siguiente a procesar se seleccionará tras **bajar dos filas**



Stride
Ejemplo

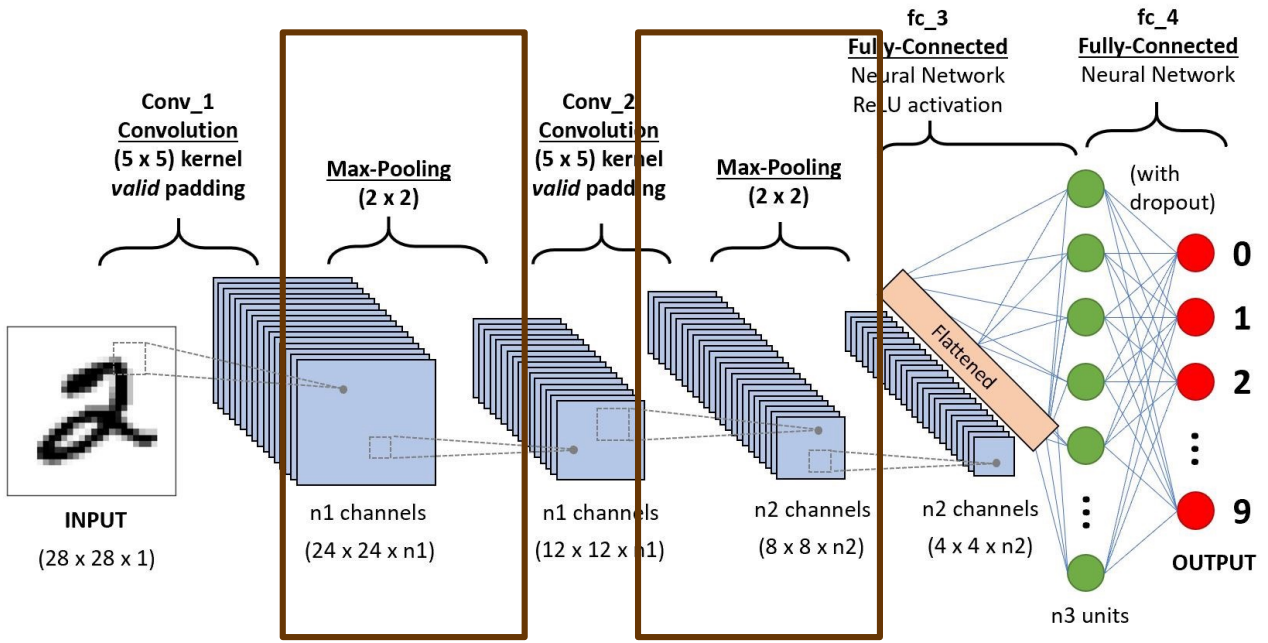
- Veamos un ejemplo con imágenes de 6x6 y un filtro de 3x3



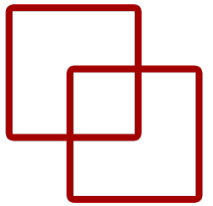
Operación *Pooling*

Definición

- La operación *pooling* tiene por objetivo **reducir la dimensionalidad** de las capas generadas tras la aplicación de operaciones de convolución
- Recordando que las capas disponen de tres dimensiones (*ancho x alto x profundidad*), la reducción se aplicaría únicamente sobre las **dimensiones de anchura y altura**, sin afectar a la profundidad de estas capas

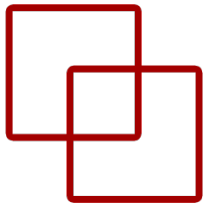


Ejemplo



- Sobre un ejemplo simple, asumamos que trabajamos con imágenes **640x480** (B/N) y que la primera convolución aplica **32 filtros de 3x3** (*stride 1, zero-padding 1*)
  - El resultado será una capa intermedia de **640x480x32**

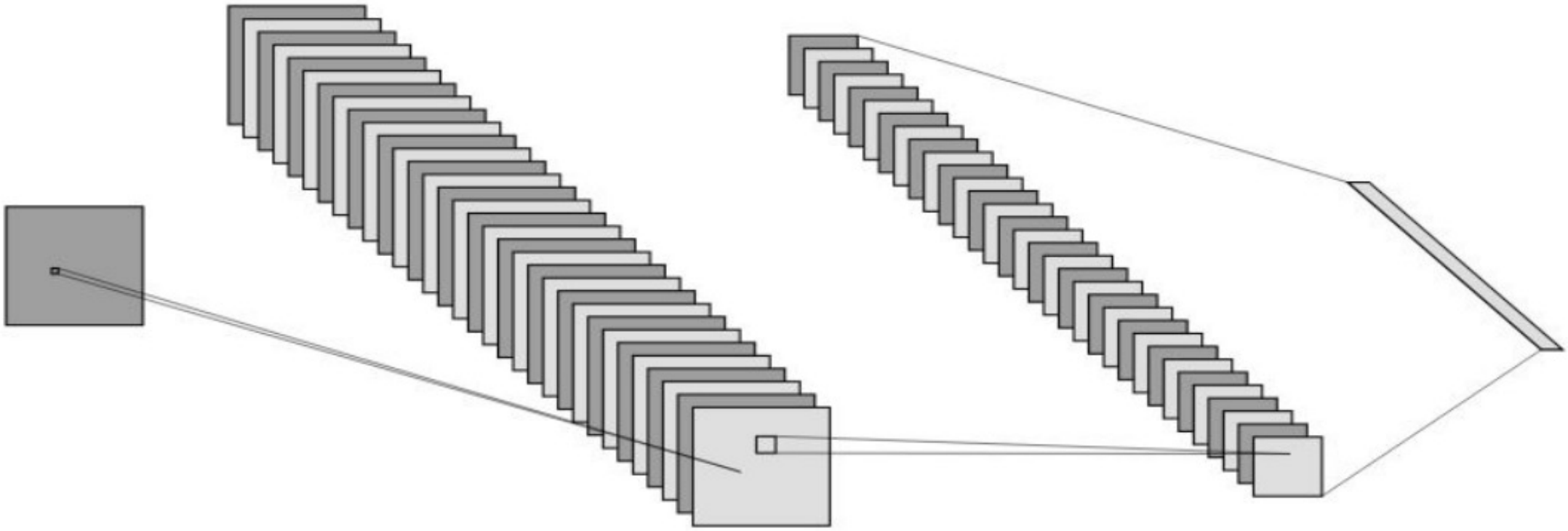
Ejemplo



- Sobre un ejemplo simple, asumamos que trabajamos con imágenes **640x480** (B/N) y que la primera convolución aplica **32 filtros de 3x3** (*stride 1, zero-padding 1*)
  - El resultado será una capa intermedia de **640x480x32**

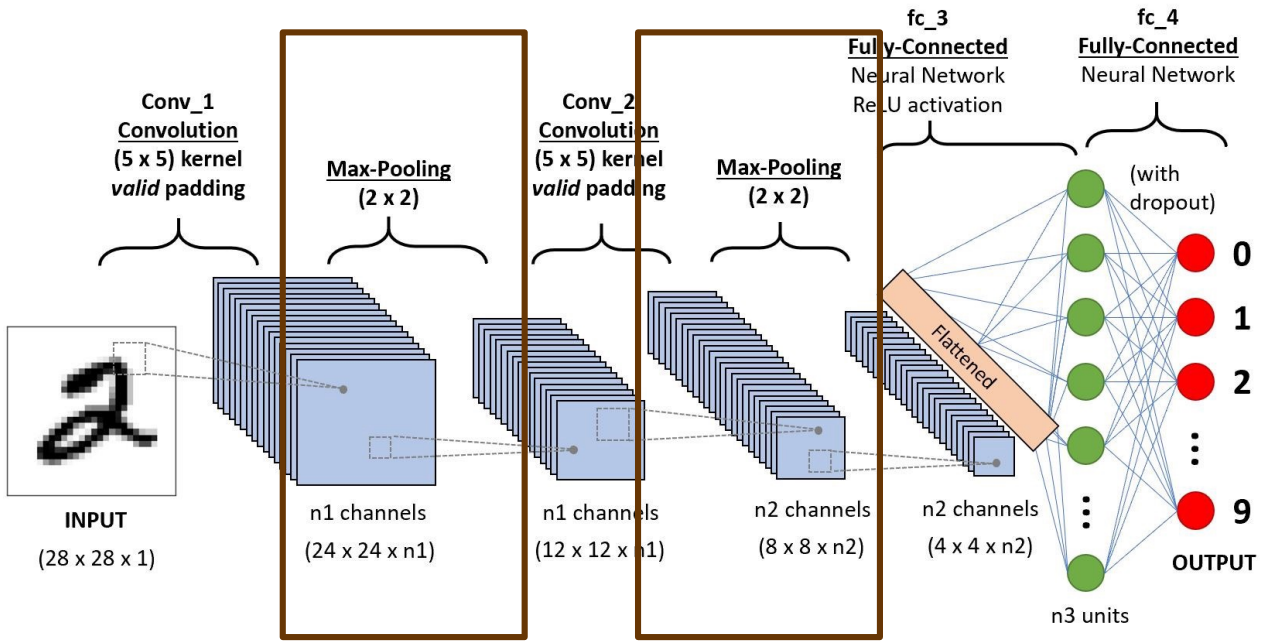
- Una operación *pooling* **reduciendo la dimensionalidad** a un 25% de la original obtendría una nueva capa intermedia de 320x240x32

Ejemplo



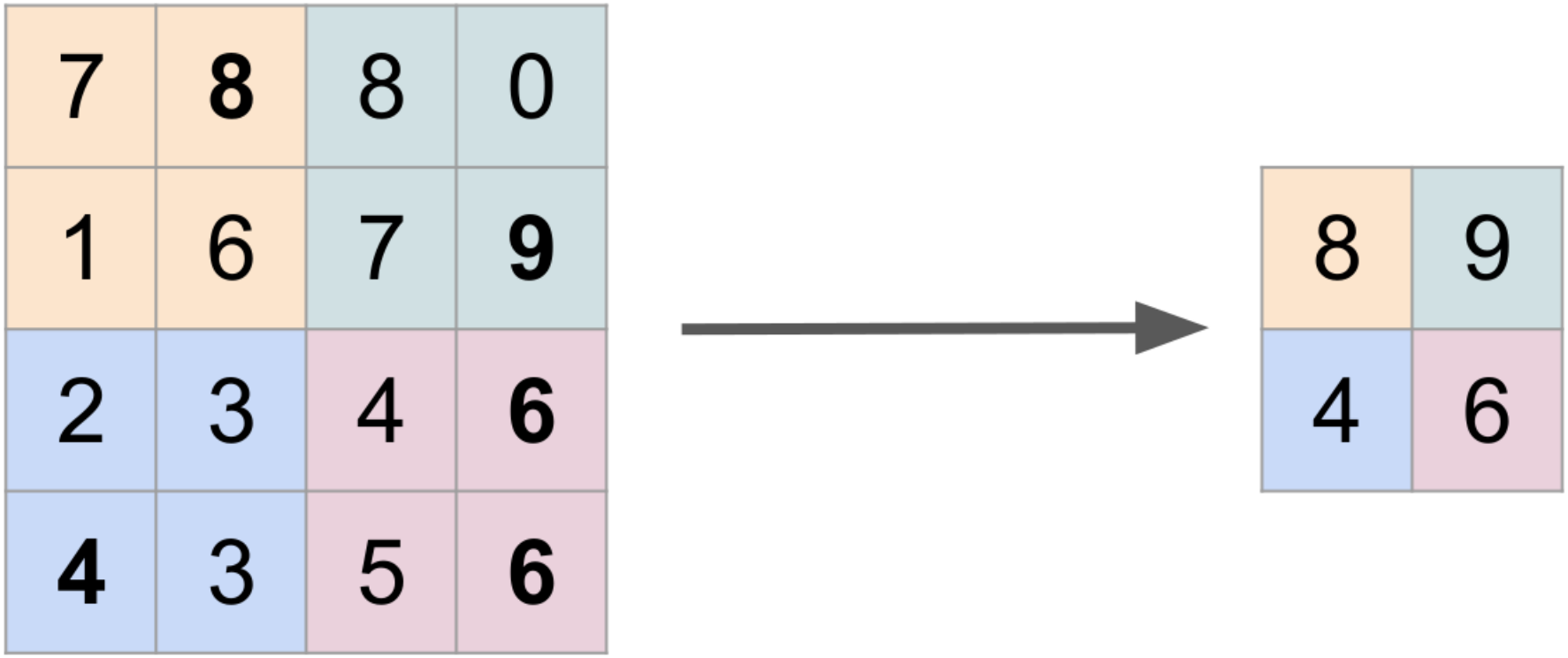
Max-pooling

- La operación de *pooling* más común es *max-pooling*, donde cada operación computa el **máximo** para una serie de neuronas **conectadas espacialmente**
- Su aplicación se realiza a través de un filtro de tamaño NxM, normalmente M=N (se define como extensión espacial).
  - También se usa el **parámetro *stride*** que determinará el factor de reducción de dimensionalidad



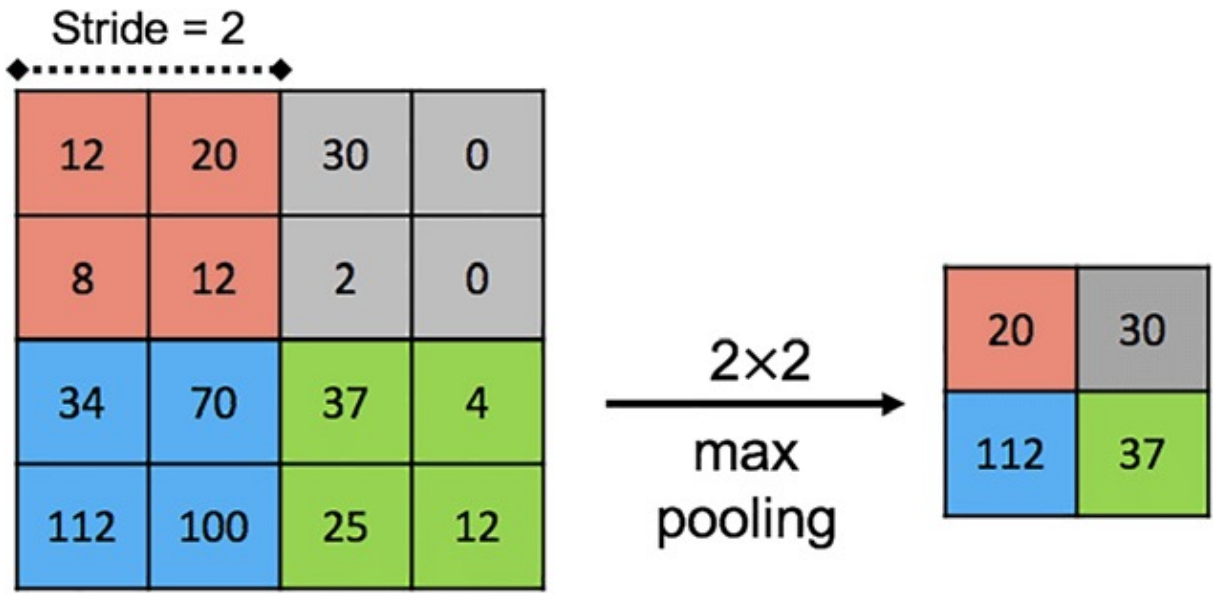
Max-pooling

Ejemplo de max-pooling 2x2 con stride=2



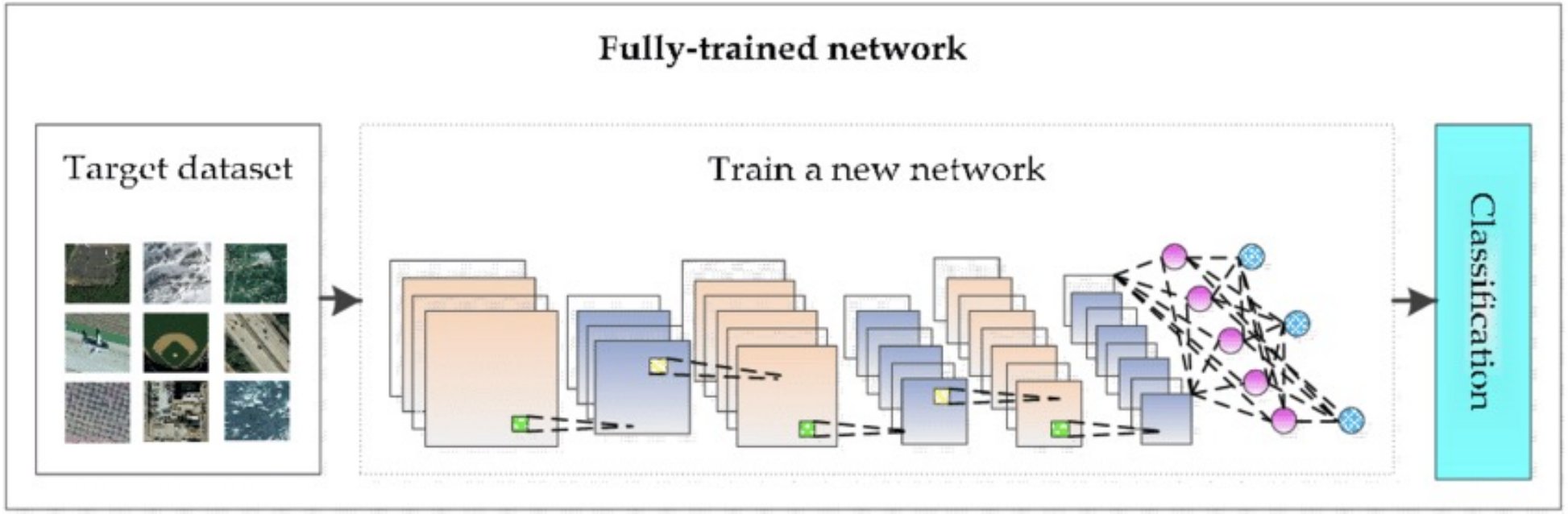
Consideraciones

- A nivel práctico, el uso de filtros mayores a 3x3, o de un valor de *stride* superior a 2, suele generar malos resultados → desaconsejamos su uso
- Como **alternativas** al *max-pooling*, podemos reemplazar la función máximo por otras funciones como la media.
  - Sin embargo, la experiencia ha mostrado el mayor poder de la función máximo

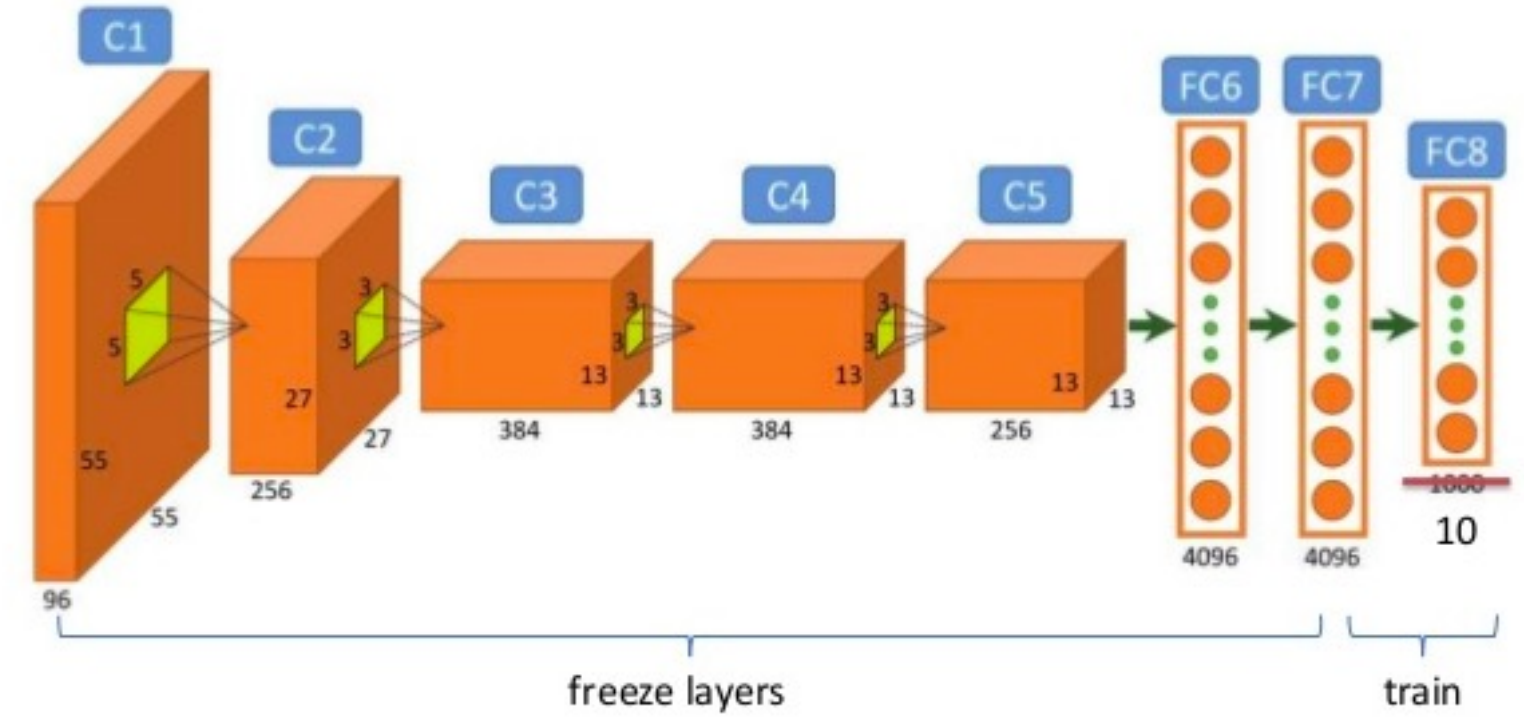


Hacia la estandarización

1. Transfer Learning

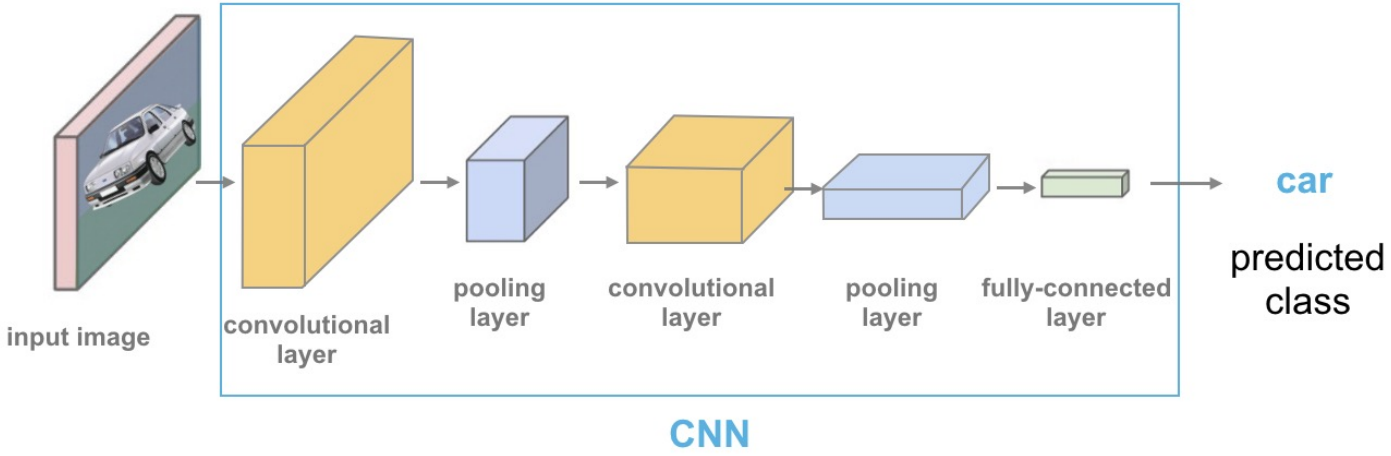


2. Finetunning



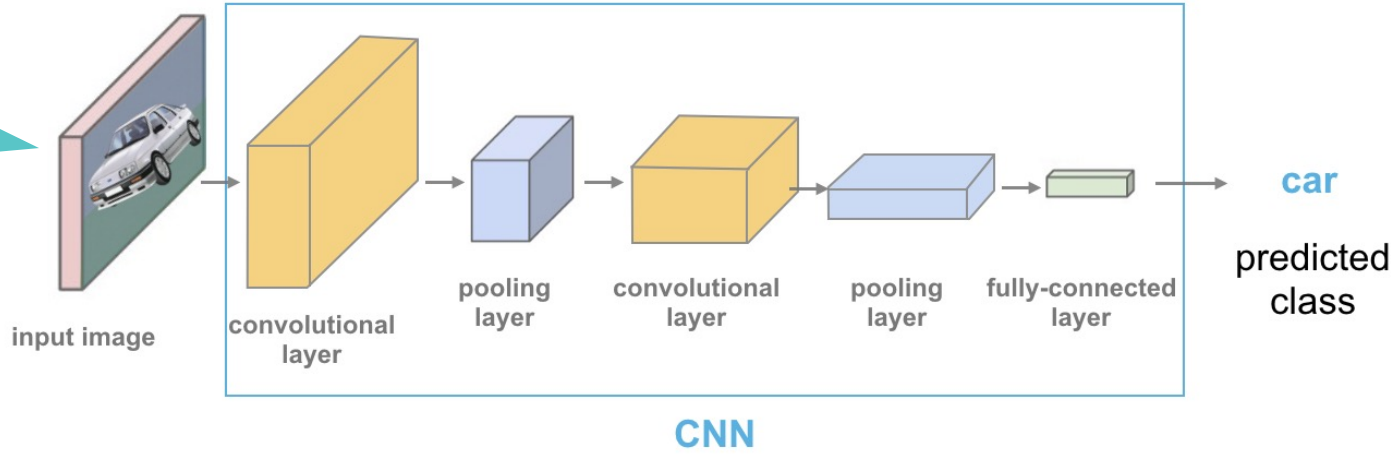
CNN Vs RNN

Background
CNN vs. RNN



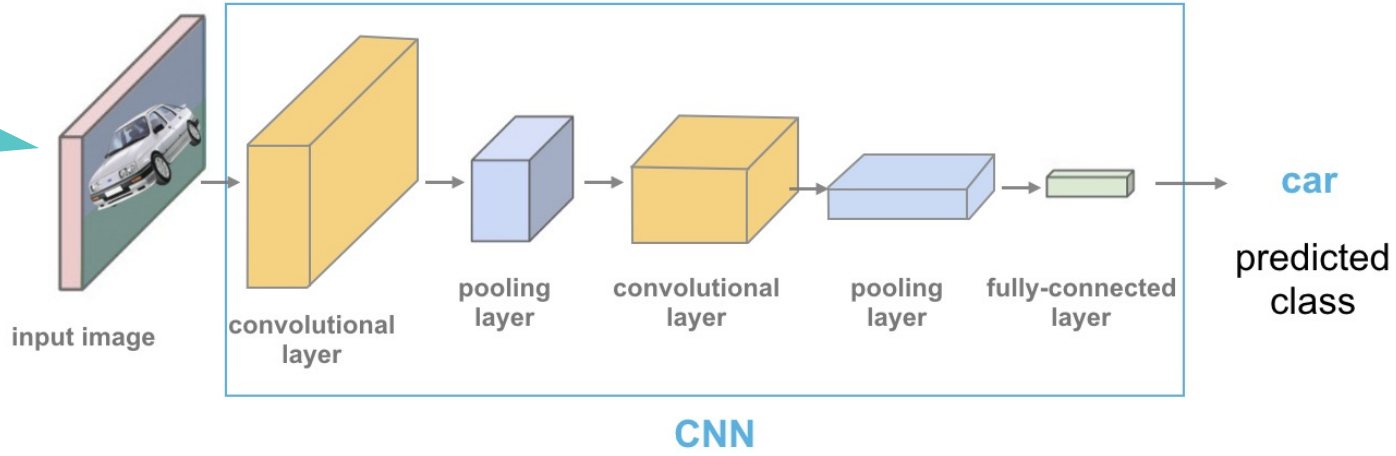
Background
CNN vs. RNN

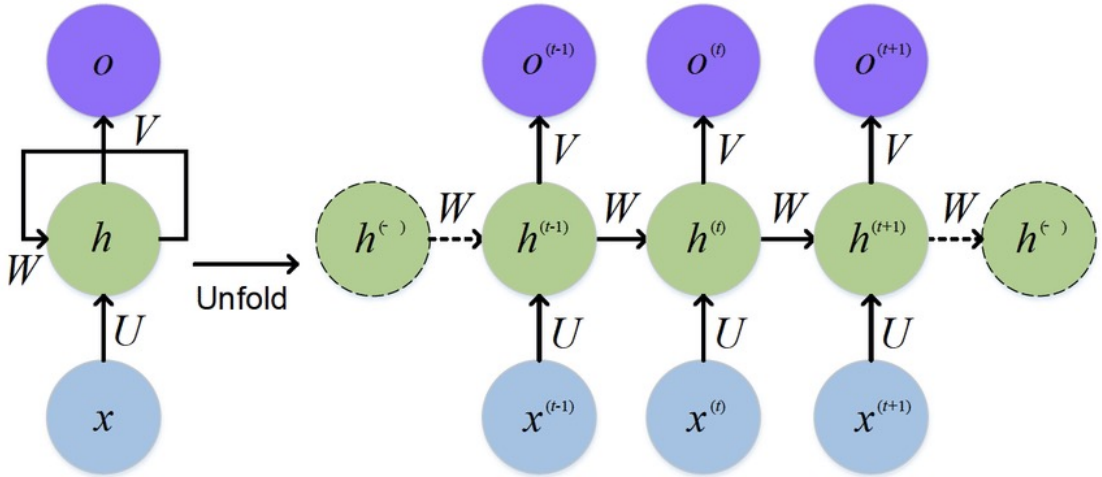
Dato (imagen, secuecnia, palabra, etc) a clasificar
¿Qué pasa si se introduce una secuencia?



Background
CNN vs. RNN

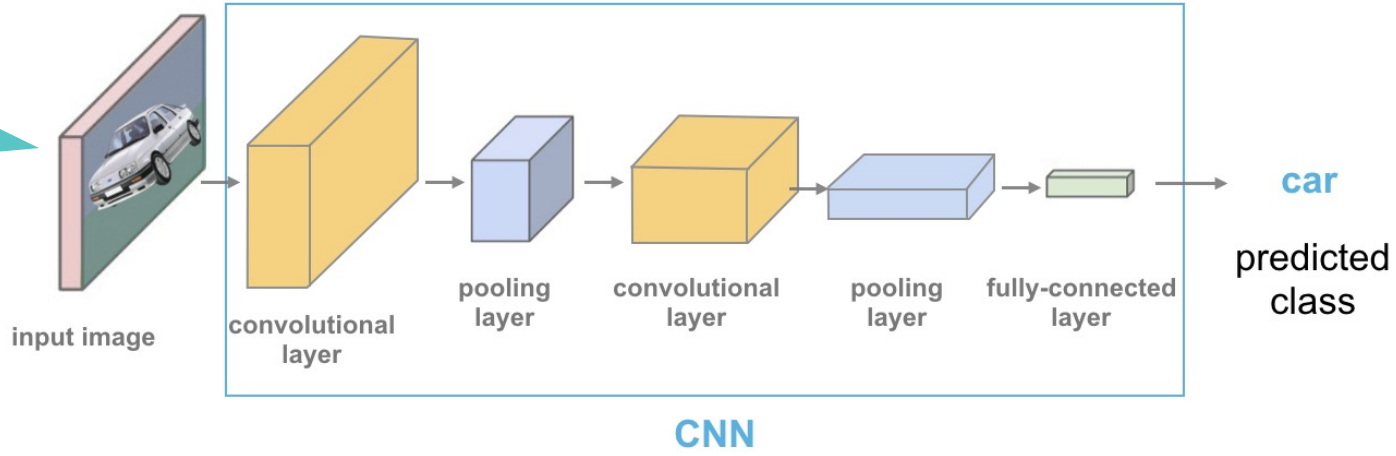
Dato (imagen, secuecnia, palabra, etc) a clasificar
¿Qué pasa si se introduce una secuencia?



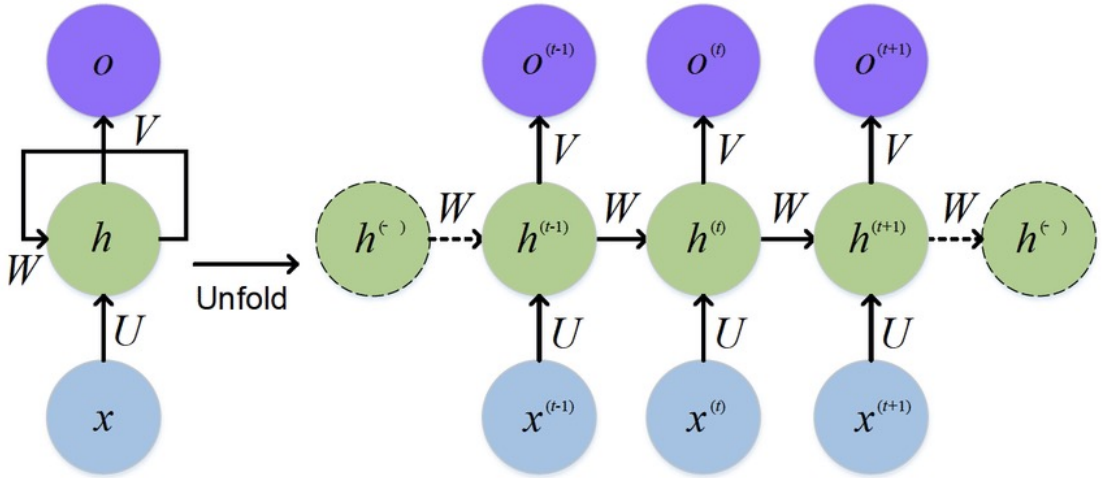


Background
CNN vs. RNN

Dato (imagen, secuecnia, palabra, etc) a clasificar
¿Qué pasa si se introduce una secuencia?

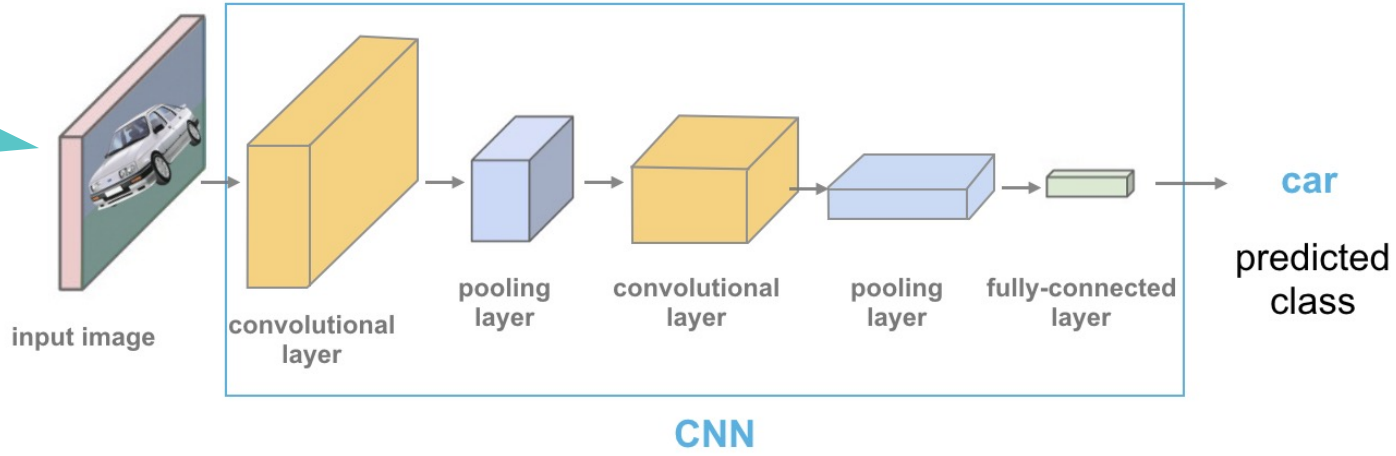


Secuencia (conversación, texto, vídeo) de datos con orden
Los datos están correlacionados dependiendo del texto anterior

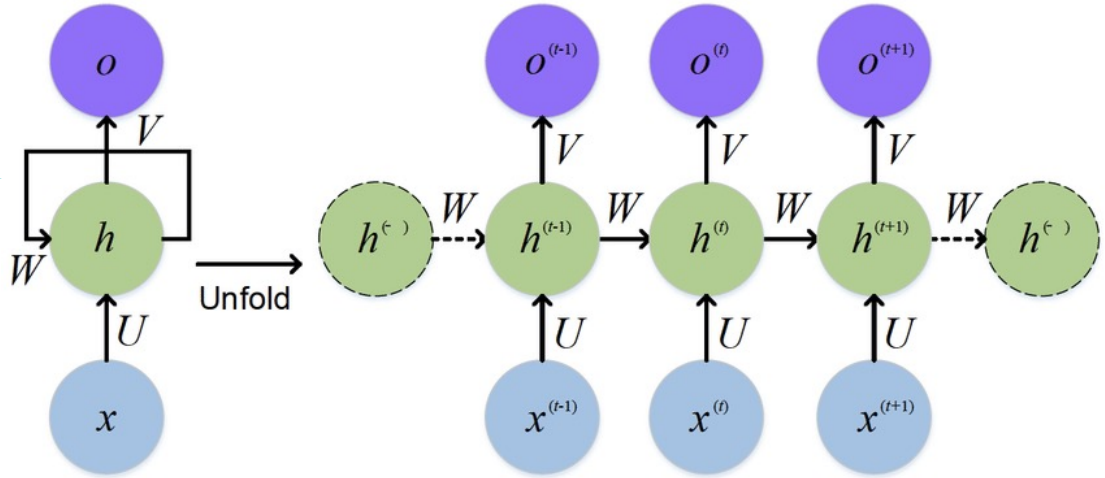


Background
CNN vs. RNN

Dato (imagen, secuecnia, palabra, etc) a clasificar
¿Qué pasa si se introduce una secuencia?



Para ello utiliza la activación de la **iteración anterior** para dar nuevos resultados, i.e., tiene memoria



¡Gracias!

Dr. Manuel Castillo-Cara
www.manuelcastillo.eu
Departamento de Inteligencia Artificial
Escuela Técnica Superior de Ingeniería Informática
Universidad Nacional de Educación a Distancia (UNED)