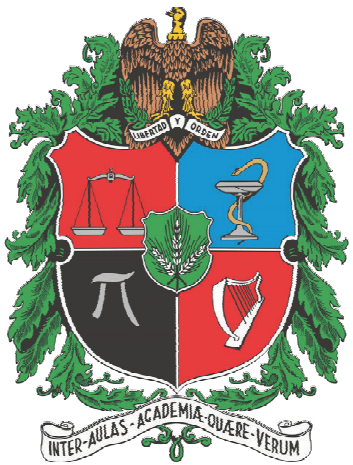
Visión por Computador usando Redes neuronales Convolucionales (CNNs)

A. M. Alvarez-Meza, Ph.D.
D. F. Collazos-Huertas, Ph.D(c)
amalvarezme@unal.edu.co, dfcollazos@unal.edu.co

Minería de datos
Departamento de Matemáticas y Estadística
Universidad Nacional de Colombia-sede Manizales



Contenido

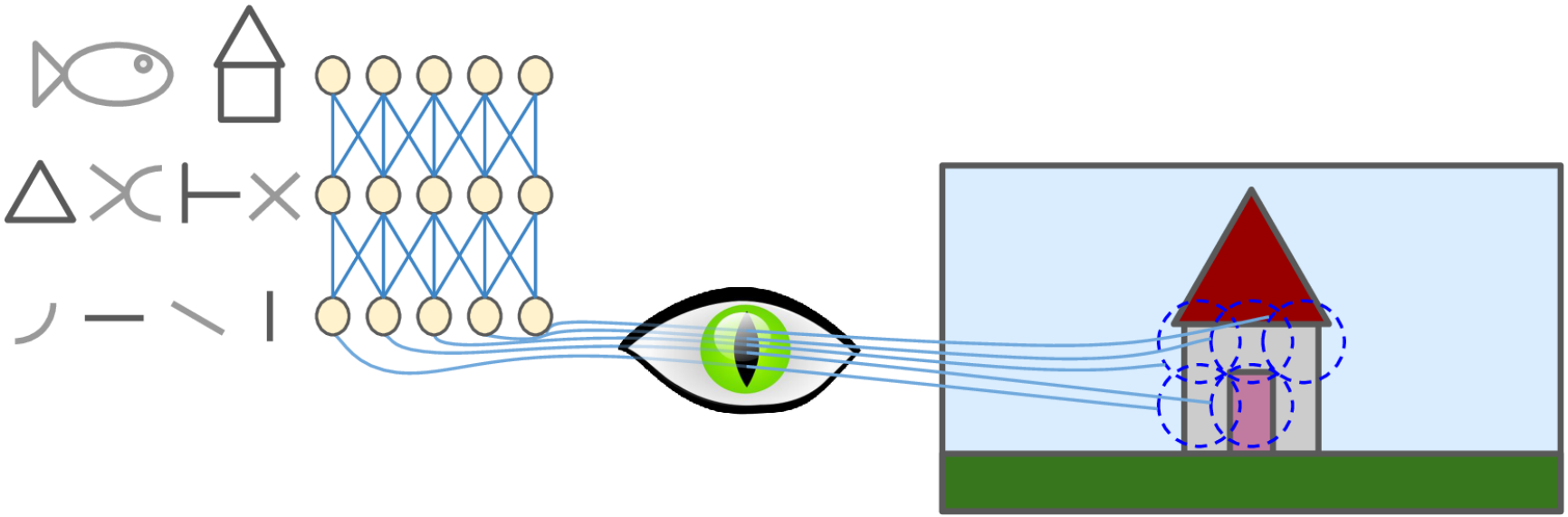
Contenido

La arquitectura de la corteza visual I

- *David H. Hubel* y *Torsten Wiesel* realizaron una serie de experimentos, brindando información crucial sobre la estructura de la corteza visual.
- Mostraron que **muchas neuronas en la corteza visual tienen un pequeño campo receptivo local**, lo que significa que reaccionan solo a estímulos visuales ubicados en una región del campo visual.
- Los campos receptivos de diferentes neuronas pueden superponerse, y juntos unen **todo el campo visual**.
- Mostraron que algunas neuronas reaccionan solo a imágenes de líneas horizontales, mientras que otras reaccionan solo a líneas con diferentes orientaciones.
- También notaron que algunas neuronas **tienen campos receptivos más grandes** y reaccionan a **patrones más complejos**.

La arquitectura de la corteza visual II

Estas observaciones llevaron a la idea de que las neuronas de nivel superior se basan en los resultados de las neuronas vecinas de nivel inferior.



Esta poderosa arquitectura es capaz de detectar todo tipo de patrones complejos en cualquier área del campo visual.

La arquitectura de la corteza visual III

- Estos estudios de la corteza visual inspiraron el **neocognitrón**, introducido en 1980, que evolucionó gradualmente hacia lo que ahora llamamos redes neuronales convolucionales.

- Un hito importante fue un artículo de **1998** de *Yann LeCun*, *Léon Bottou*, *Yoshua Bengio* y *Patrick Haffner*, que presentó la famosa arquitectura LeNet-5, ampliamente utilizada para reconocer números de cheques escritos a mano.

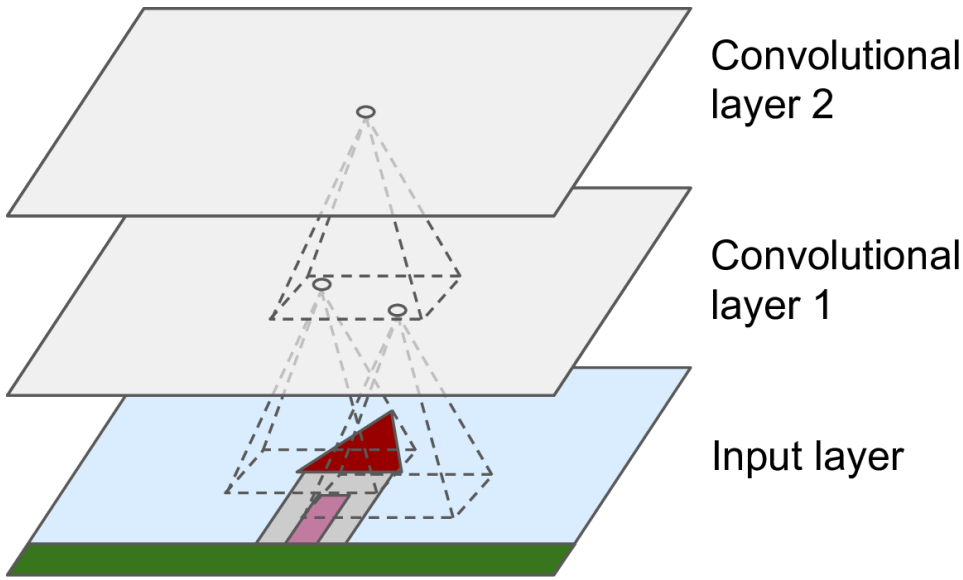
- Esta arquitectura tiene algunos bloques de construcción que ya conoce, pero también presenta dos nuevos bloques de construcción: capas convolucionales y capas de agrupación.

Contenido

Capa convolucional I

"Las neuronas en la primera capa convolucional no están conectadas a cada píxel en la imagen de entrada, sino solo a los píxeles en sus campos receptivos".

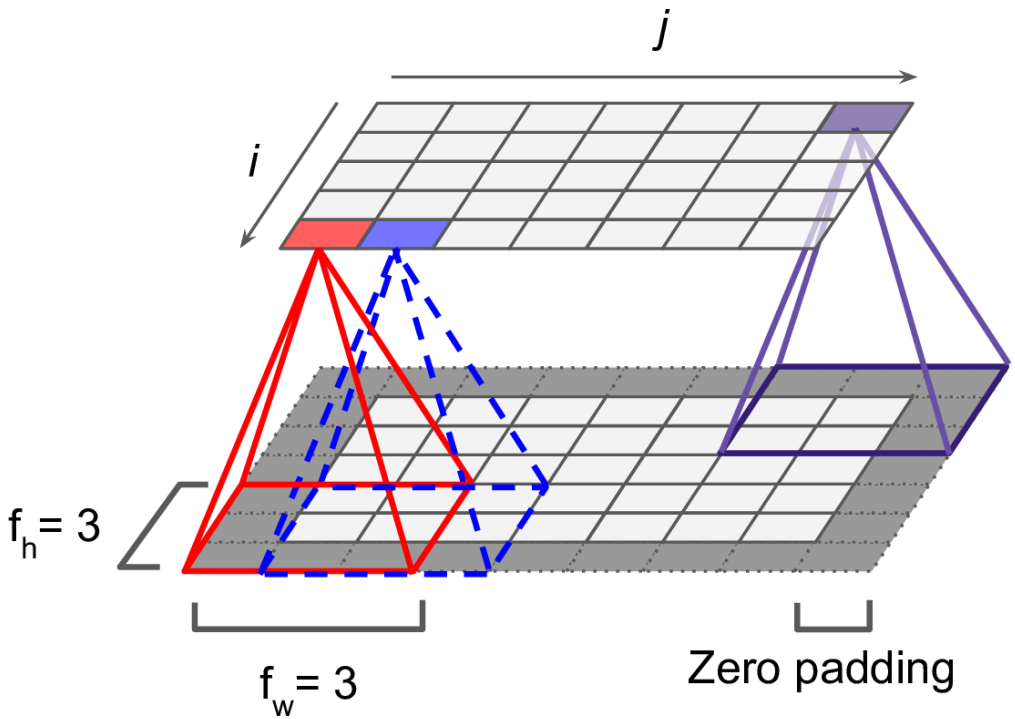
- A su vez, cada neurona en la segunda capa convolucional **está conectada solo a las neuronas ubicadas dentro de un pequeño rectángulo en la primera capa**.



Esta estructura jerárquica es común en las imágenes del mundo real, razón por la que las CNN funcionan tan bien para el reconocimiento de imágenes.

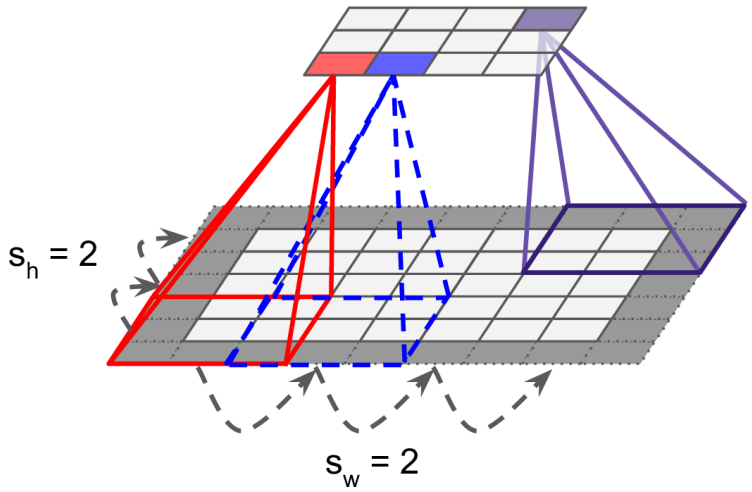
Capa convolucional II

- Una neurona ubicada en la fila i, la columna j de una capa dada está conectada a las salidas de las neuronas en la capa anterior ubicada en las filas i a $i + f_h - 1$, las columnas j a $j + f_w - 1$, donde f_h y f_w son la **altura** y el **ancho** del campo receptivo.
- Para que una capa tenga la misma altura y anchura que la capa anterior, es común agregar ceros alrededor de las entradas. Esto se llama *Zero padding*.



Capa convolucional III

Es posible conectar una capa de entrada grande a una capa mucho más pequeña espaciando los campos receptivos (*stride*).

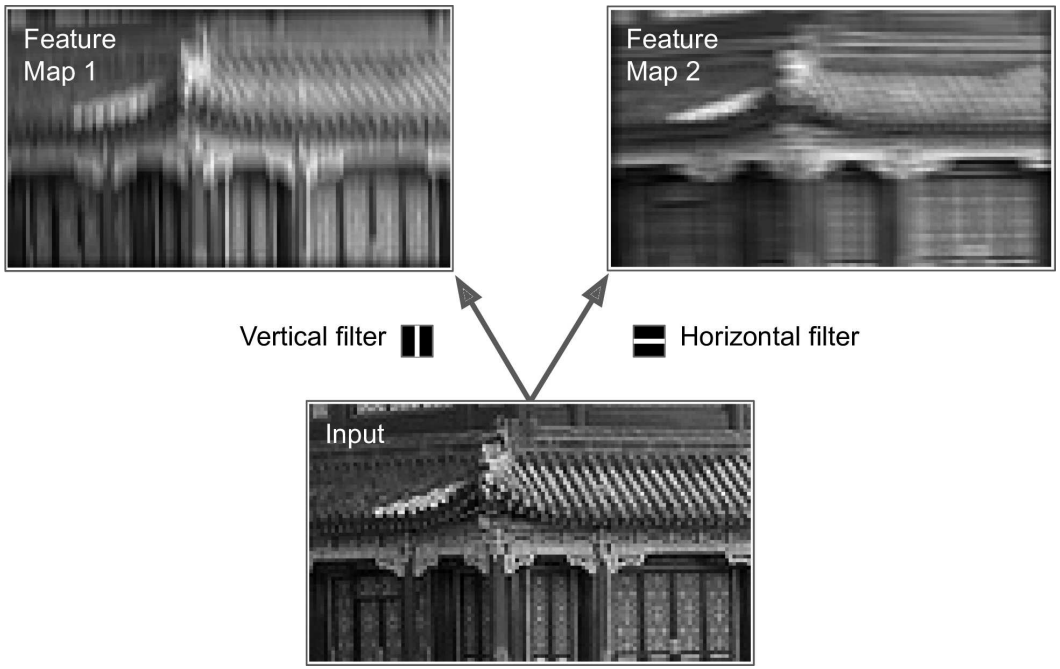


- Una capa de entrada de 5×7 (más relleno de cero) está conectada a una capa de 3×4, utilizando campos receptivos de 3×3 y un stride de 2.
- Una neurona en la fila i, la columna j está conectada a las salidas de las neuronas en la capa anterior ubicada en las filas $i \cdot s_h$ a $i \cdot s_h + f_h - 1$, las columnas $j \cdot s_w$ a $j \cdot s_w + f_w - 1$, donde s_h y s_w son los strides verticales y horizontales.

Contenido

Filtros I

Los pesos de una neurona se pueden representar **como una imagen pequeña del tamaño del campo receptivo** (filtros o núcleos de convolución).



1. Un cuadrado negro con una línea blanca vertical en el medio (es una matriz de 7×7 llena de 0 a excepción de la columna central, que está llena de 1).
2. Un cuadrado negro con una línea blanca horizontal en el medio.

Filtros II

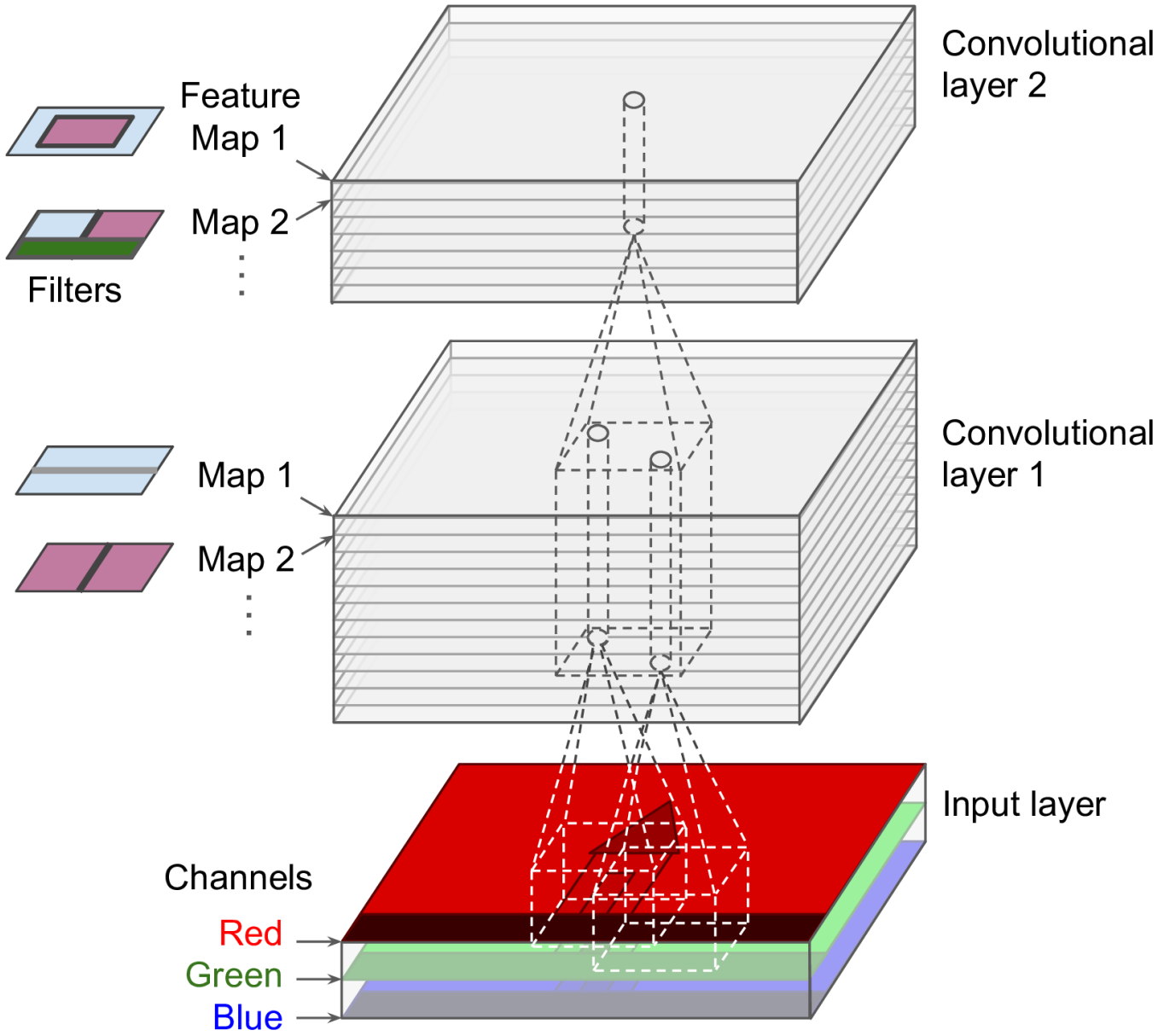
- Si todas las neuronas de una capa usan el mismo **filtro de línea vertical**, las líneas blancas verticales se mejoran mientras que el resto se vuelve borroso.

- Del mismo modo, la imagen superior derecha es lo que obtienes si todas las neuronas usan el mismo **filtro de línea horizontal**; observe que las líneas blancas horizontales se mejoran mientras que el resto se ve borroso.

Por lo tanto, una capa llena de neuronas que usan el mismo filtro genera un **feature map**, que resalta las áreas en una imagen que activan más el filtro.

Contenido

Apilamiento de múltiples feature maps I



Apilamiento de múltiples feature maps II

* En realidad una capa convolucional **tiene múltiples filtros**, y genera un mapa de características por filtro, por lo que es representado con mayor precisión en *3D*.

* Para hacerlo, tiene una neurona por píxel en cada mapa de características, y todas las neuronas dentro de un mapa de características dado comparten los mismos parámetros.

* Sin embargo, las neuronas en diferentes mapas de características utilizan diferentes parámetros.

En resumen, una capa convolucional **aplica simultáneamente múltiples filtros entrenables a sus entradas**, por lo que es capaz de **detectar múltiples características** en cualquier parte de sus entradas.

Apilamiento de múltiples feature maps III

- Una **neurona** ubicada en la fila i, columna j del mapa de características k en una capa convolucional dada l está conectada a las salidas de las neuronas en la capa anterior $l-1$, ubicada en las filas $i \cdot s_h$ a $i \cdot s_h + f_h - 1$ y las columnas $j \cdot s_w$ a $j \cdot s_w + f_w - 1$, en todos los mapas de entidades (en la capa $l-1$).

- Tenga en cuenta que todas las neuronas ubicadas en la misma fila i y columna j pero **en diferentes mapas de características** están conectadas a las salidas de las mismas neuronas exactas en la capa anterior.

- Cómo calcular la salida de una neurona dada en una capa convolucional?.
  - **R:** calcular la suma ponderada de todas las entradas, más el término *bias*.

Apilamiento de múltiples feature maps IV

$$z_{i,j,k} = b_k + \sum_{u=0}^{f_h-1} \sum_{v=0}^{f_w-1} \sum_{k'=0}^{f_{n'}-1} x_{i',j',k'} \cdot w_{u,v,k',k} \tag{1}$$

$$\begin{cases} i' = i \cdot s_h + u \\ j' = j \cdot s_w + v \end{cases}$$

- $z_{i,j,k}$ es la salida de la neurona ubicada en la fila i, columna j en el feature map k de la capa convolucional l.
- s_h y s_w son los strides verticales y horizontales, f_h y f_w son la altura y el ancho del campo receptivo, y $f_{n'}$ es el número de feature maps en la capa anterior $l-1$.

Apilamiento de múltiples feature maps V

$$z_{i,j,k} = b_k + \sum_{u=0}^{f_h-1} \sum_{v=0}^{f_w-1} \sum_{k'=0}^{f_{n'}-1} x_{i',j',k'} \cdot w_{u,v,k',k} \tag{2}$$

$$\begin{cases} i' = i \cdot s_h + u \\ j' = j \cdot s_w + v \end{cases}$$

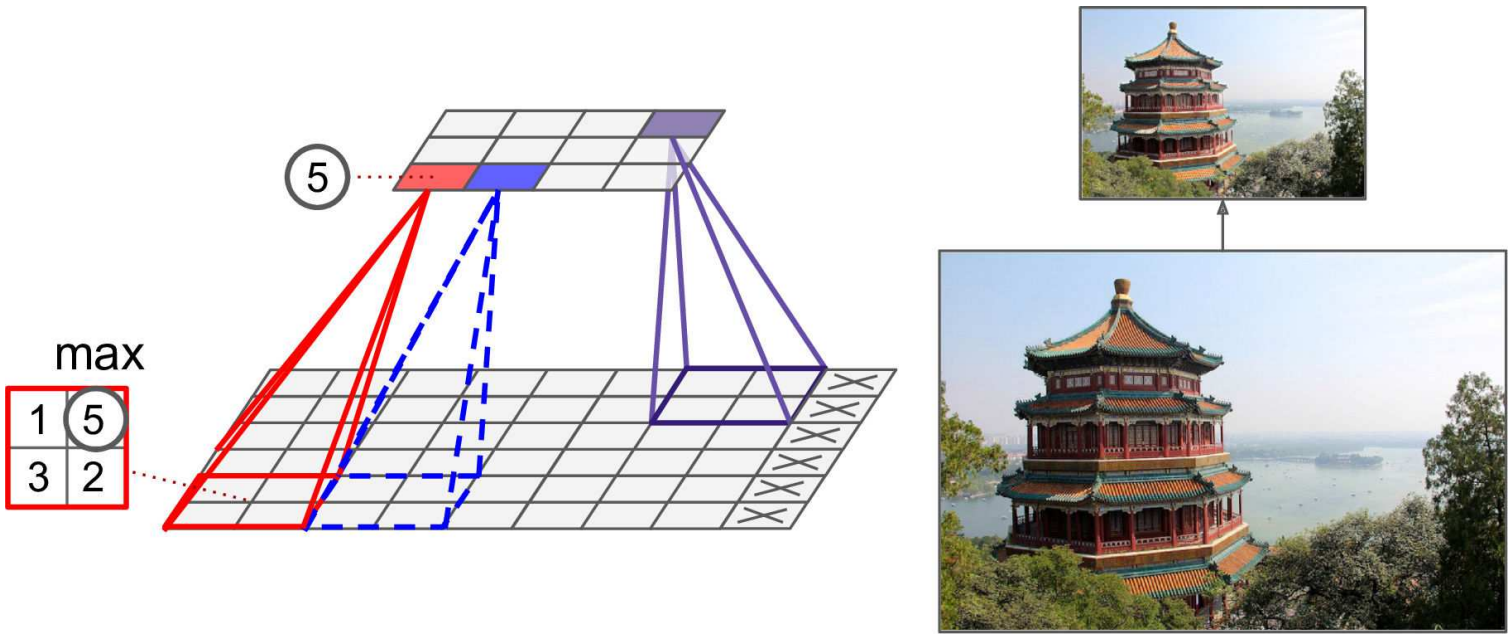
- $x_{i',j',k'}$ es la salida de la neurona ubicada en la capa $l-1$, fila i', columna j', mapa de características k' (o canal k' si la capa anterior es la capa de entrada).
- b_k es el bias para el feature map k (en la capa l).
- $w_{u,v,k',k}$ es el peso de conexión entre cualquier neurona en el feature map k de la capa l y su entrada ubicada en la fila u, columna v, y el mapa de características k'.

Implementación de convolución

Contenido

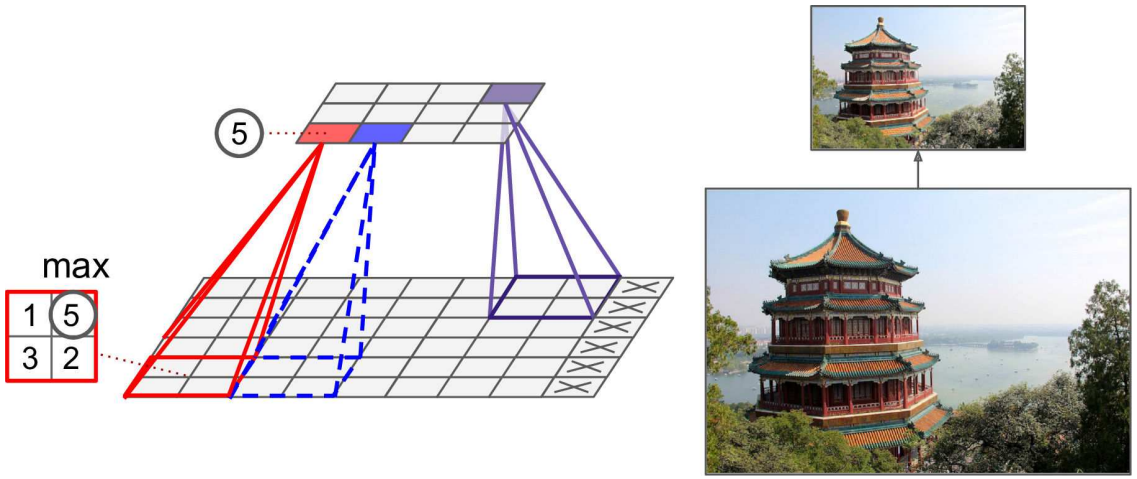
Pooling layer I

Las capas de *pooling* tienen como objetivo es submuestrear la imagen de entrada para **reducir** la carga computacional, el uso de memoria y el número de parámetros (lo que limita el riesgo de sobreajuste).



Cada neurona en una capa de pooling está conectada a las salidas de un número de neuronas en la capa anterior. Una neurona de pooling **no tiene pesos**; agrega las entradas usando una **función de agregación** como `max` o `mean`

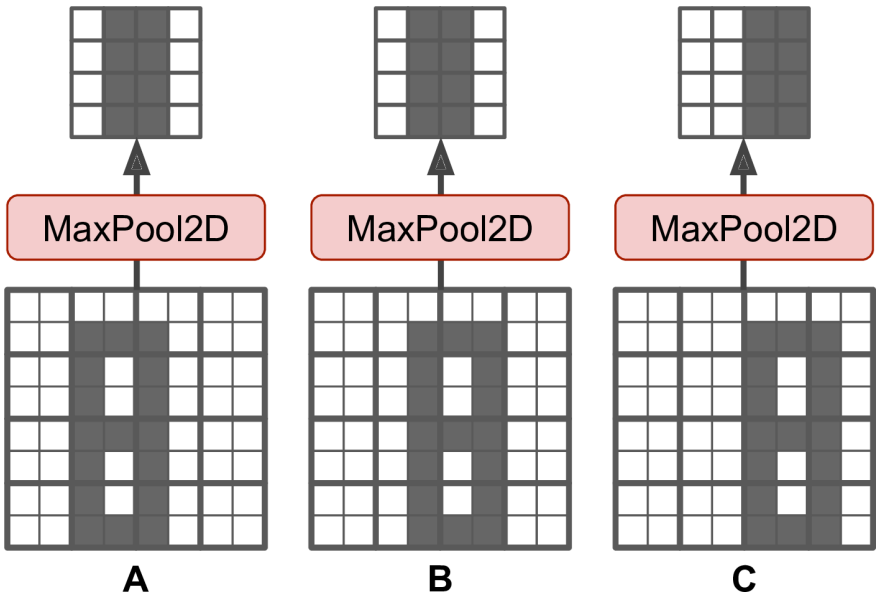
Pooling layer II



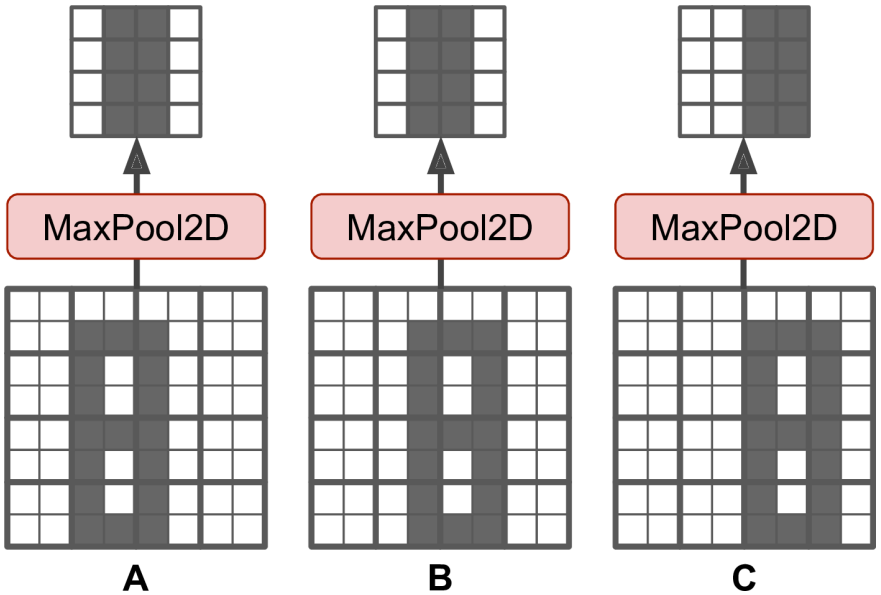
- En este ejemplo, usamos un pooling kernel de 2×2, con un stride de 2 y sin padding.
- Solo el valor de entrada máximo en cada campo receptivo llega a la siguiente capa, mientras que las otras entradas se descartan.
- Por ejemplo, en el campo receptivo inferior izquierdo, los valores de entrada son $1, 5, 3, 2$, por lo que solo el valor máximo, 5, se propaga a la siguiente capa
- Debido al stride de 2, la imagen de salida tiene la mitad de la altura y la mitad del ancho de la imagen de entrada.

Pooling layer III

- La capa de `max-pooling` también introduce cierto nivel de invariancia en las traducciones pequeñas.
- Asumimos que los píxeles brillantes tienen un valor más bajo que los píxeles oscuros.
- Consideramos que a las imágenes (A, B, C) se les aplica una capa de `max-pooling` con kernel y strides 2×2.
- Las imágenes B y C son los igual que la imagen A pero desplazada uno y dos píxeles a la derecha.



Pooling layer IV



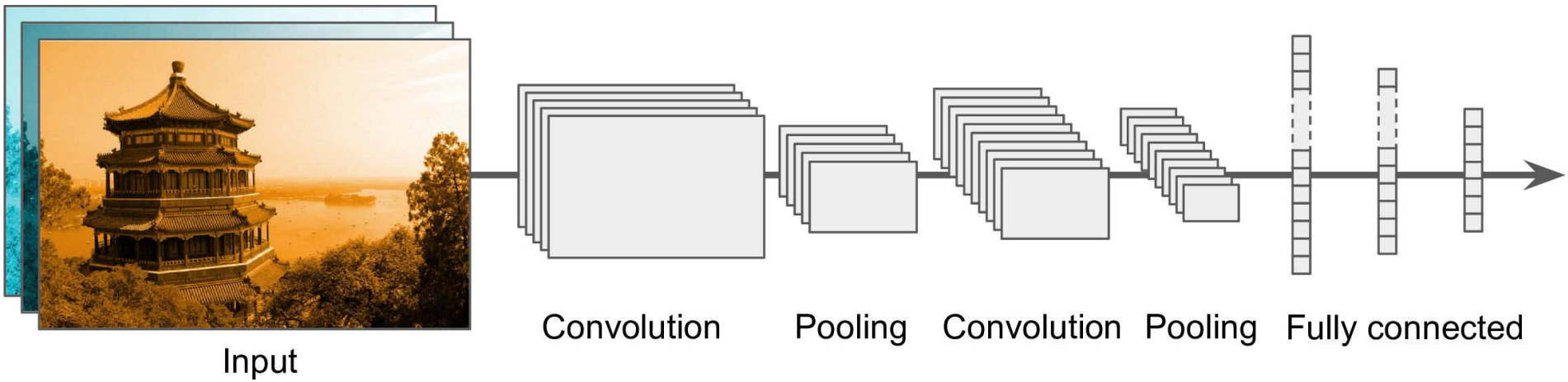
* Las salidas del `max-pooling` para las imágenes A y B son idénticas. Esto es lo que significa invariancia de traducción.
* Sin embargo, para la imagen C, la salida es diferente: se desplaza un píxel hacia la derecha (pero todavía hay un 75% de invariancia).
* Se tienen desventajas.

Implementación de max-pooling

Contenido

Arquitecturas CNN

Las arquitecturas típicas de CNN apilan algunas capas convolucionales (seguida de una capa ReLU), luego una capa de pooling, luego otras pocas capas convolucionales (+ ReLU), luego otra capa de pooling, y así sucesivamente.



La imagen se hace cada vez más pequeña a medida que avanza por la red, pero también se vuelve cada vez más profunda (es decir, con más mapas de características) gracias a las capas convolucionales.

Arquitectura CNN