Capas convolucionales 2D

Cross-Correlation

- Invarianza de traslación (*translation invariance*).
- Principio de localidad (*locality principle*).

Podemos pensar a esta capa como un operador $T_{(D,k,C)} : \mathbb{R}^{C\times n\times n} \rightarrow \mathbb{R}^{D\times(n-k+1)\times(n-k+1)}$ tal que

$$Z = T(X) \qquad X = (x_{i,j,c}), Z = (z_{i,j,d})$$

$$z_{i,j,d} = u + \sum_{a\in A}\sum_{b\in B}\sum_{c\leq C} w_{a,b,c,d} \cdot x_{i+a,j+b,c} \qquad \forall d \leq D$$

$C \times n \times n \rightarrow D \times (n-k+1) \times (n-k+1)$

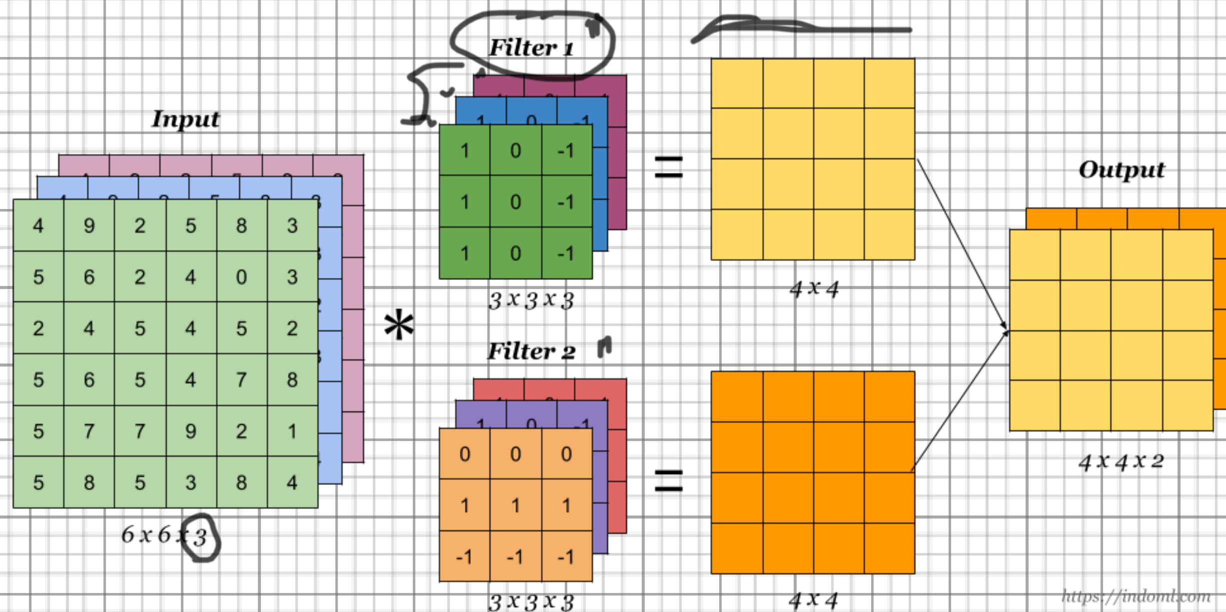
↳ tam Kernel

Capas CC. unidireccionales

$Z = XW^T + b^T$

Definen una ventana

A, B



Padding and Stride

Ajustar la dimensión de los datos a procesar

- *Padding* incrementa el ancho y largo de las imágenes de forma artificial.
- *Stride* reduce la resolución de la salida.

$$(n-k+p+1)\times(n-k+p+1)$$

$$(n_h-k_h+p_h+1)\times(n_w-k_w+p_w+1)$$

Resultado

3x3

2x2

0	0	0	0	0
0	0	1	2	0
0	3	4	5	0
0	6	7	8	0
0	0	0	0	0

2x2

Resultado 2x2

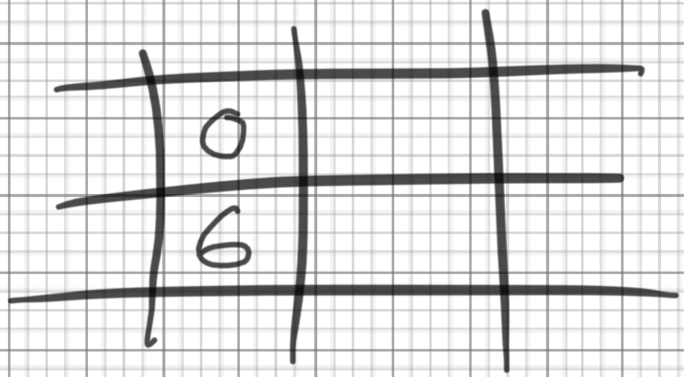
$$\left(\frac{n-k+p}{s}+1\right)\times\left(\frac{n-k+p}{s}+1\right)$$

$$\left(\frac{n_h-k_h+p_h}{s_h}+1\right)\times\left(\frac{n_w-k_w+p_w}{s_w}+1\right)$$

s:3

0	0	0	0	0
0	0	1	2	0
0	3	4	5	0
0	6	7	8	0
0	0	0	0	0

0	1
2	8



Pooling layer

Es un operador completamente determinista que al igual que las capas convolucionales actúa sobre una ventana de los valores de entrada, puede calcular el máximo y el promedio sobre dichos valores, no tiene kernel.

- *max pooling*
- *average pooling*.

$$h_{i,j} = \max\left(z_{i+a,j+b-1} : \forall a \in \{1, \cdots, m_h\} \text{ y } b \in \{1, \cdots, m_w\}\right)$$

$$h_{i,j} = \sum_{a=1}^{m_h} \sum_{b=1}^{m_w} \frac{z_{i+a,j+b-1}}{\#\underbrace{\{1, \cdots, m_h\} \times \{1, \cdots, m_w\}}_{A \times B}}$$

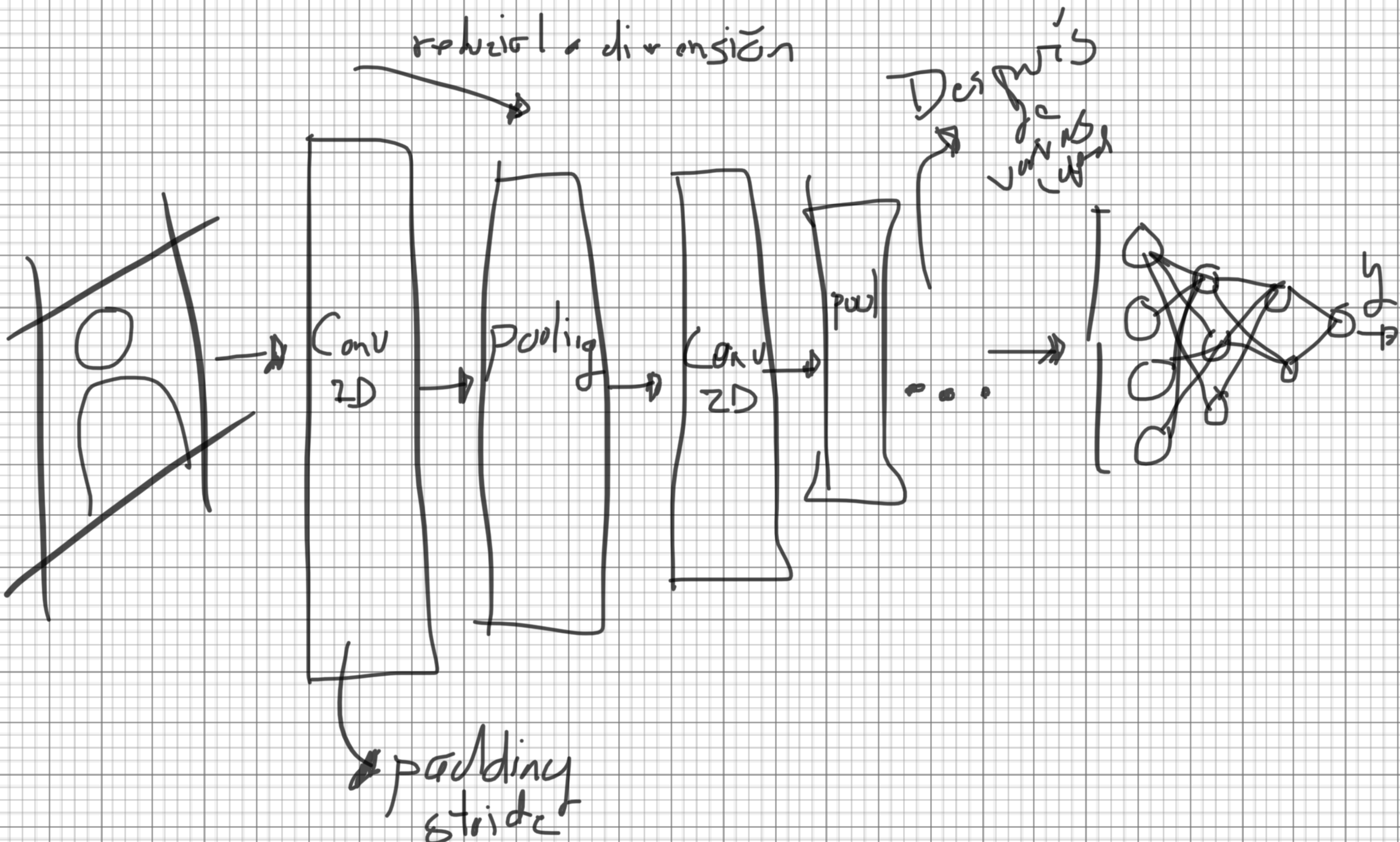
$A \times B \longrightarrow$ ventana

Ejemplo

0	1	2
3	4	5
6	7	8

2×2
max pooling

4	5
7	8


reducir la dimensión
Después de varias capas
Conv 2D
pooling
Conv 2D
pool
...
y
padding
stride