Modelos de Deep Learning

Rumbo a capas convolucionales

Universidad ORT Uruguay

29 de Setiembre, 2025

Por qué capas densas no alcanzan (I)

- Imágenes como tensores: $\mathbf{X} \sim (h, w, c)$, donde
  - h=alto, w=ancho
  - c=canales (1 en B/N, 3 en color, mayor en hiperespectral)
- (h, w, c) no son equivalentes: h, w codifican la *estructura espacial* de píxeles; los canales c no tienen un orden intrínseco.
- Aplanado para usar una capa densa (sin bias por simplicidad):

$$\mathbf{a} = A\big(\mathrm{vect}(\mathbf{X})^\top \cdot \mathbf{W}\big), \qquad \mathrm{vect} : \ (s_1, \ldots, s_n) \mapsto \left(\prod_{i=1}^{n} s_i\right)$$

En PyTorch: `x.reshape(-1)`.

Por qué capas densas no alcanzan (II)

- Desventaja 1: composicionalidad. La entrada es imagen y la salida es vector

  Remedio: reshape de la salida:

$$\mathbf{H} = \text{unvect}\big(A(\text{vect}(\mathbf{X})^\top \cdot \mathbf{W})\big), \qquad \mathbf{H} \sim (h', w', c'),$$

- Desventaja 2: demasiados parámetros. Si $\mathbf{X} \sim (h, w, c)$ y $\mathbf{H} \sim (h, w, c')$, una capa densa tiene (sin contar los biases)

$$\underbrace{(hwc)}_{\text{entradas}} \times \underbrace{(hwc')}_{\text{salidas}} = (hw)^2\, c\, c' \quad \text{parámetros.}$$

  Ejemplo: imagen $(1024, 1024)$ RGB ($c = 3$) con $c' = 3$: $\approx 10^{13}$ parámetros.

  Razón: cada canal de cada *píxel de salida* es combinación ponderada de *todos* los canales de *todos* los píxeles de entrada.

Ejemplo

- **1D (4 "píxeles", 1 canal):**

$$\begin{bmatrix} h_1 & h_2 & h_3 & h_4 \end{bmatrix} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix} \begin{bmatrix} W_{11} & W_{12} & W_{13} & W_{14} \\ W_{21} & W_{22} & W_{23} & W_{24} \\ W_{31} & W_{32} & W_{33} & W_{34} \\ W_{41} & W_{42} & W_{43} & W_{44} \end{bmatrix}$$

- Motivación para convoluciones: restringir las conexiones para explotar **localidad** y **compartir pesos**.

Capas locales: intuición

- El arreglo espacial de píxeles induce una métrica sobre la grilla:

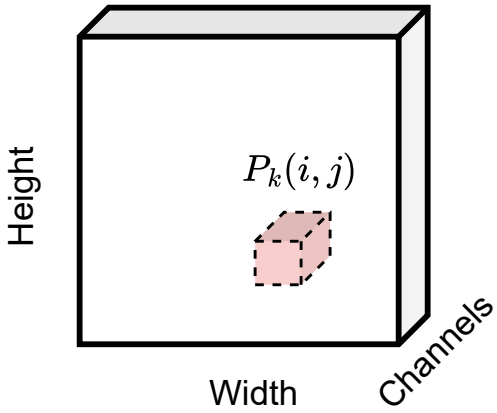
$$d\big((i,j),(i',j')\big) = \max\big(|i-i'|,\ |j-j'|\big).$$

- Idea clave: la **influencia** entre píxeles decrece con la distancia.
- Dada una imagen $\mathbf{X}$ y un entero $k \geq 0$, definimos el **patch** centrado en (i,j) como

$$\mathbf{P}_k(i,j) = \mathbf{X}_{i-k:i+k,j-k:j+k,:} = \text{píxeles a distancia} \leq k \text{ de } (i,j)$$

- El tamaño del patch es (s,s,c) con $s = 2k+1$; a s = **kernel size**.
- Válido si (i,j) está a $\geq k$ pasos del borde (veremos luego cómo manejar bordes).

Patch



Capas locales: definición

- Sea f una capa con entrada $\mathbf{X} \sim (h, w, c)$ y salida $\mathbf{H} = f(\mathbf{X}) \sim (h, w, c')$.

- f es k-**local** si para todo píxel (i, j),

$$H_{ij} = f\left(\mathbf{P}_k(i, j)\right),$$

- O sea: la salida en (i, j) depende solo del patch alrededor de (i, j).

De capa densa a capa local

- Recordar una capa densa sobre imagen aplanada (sin bias):

$$\mathbf{a} = A\left(\text{vect}(\mathbf{X})^\top \cdot \mathbf{W}\right).$$

- Forzando localidad: anular pesos que conectan (i,j) con píxeles fuera de $\mathbf{P}_k(i,j)$:

$$H_{ij} = A\left(\text{vect}\left(\mathbf{P}_k(i,j)\right)^\top \cdot \mathbf{W}_{ij}\right).$$

- Parámetros: $\text{vect}(\mathbf{P}_k(i,j))$ tiene dimensión s^2c ($s = 2k+1$). Tomando c' salidas por píxel:

$$\mathbf{W}_{ij} \sim (s^2c, c') \quad \Rightarrow \quad \text{parámetros totales} = h\,w\,s^2c\,c'.$$

  Comparado con la capa densa se reduce por un factor $\dfrac{s^2}{hw}$.

Ejemplo con manejo de bordes (zero-padding)

- Ejemplo 1D: $\mathbf{x} = [x_1, x_2, x_3, x_4]^\top$, radio $k = 1$ ($s = 3$).
- La capa local puede escribirse como

$$\begin{bmatrix} h_1 & h_2 & h_3 & h_4 \end{bmatrix} = \begin{bmatrix} 0 & x_1 & x_2 & x_3 & x_4 & 0 \end{bmatrix} \begin{bmatrix} W_{11} & 0 & 0 & 0 \\ W_{21} & W_{22} & 0 & 0 \\ W_{31} & W_{32} & W_{33} & 0 \\ 0 & W_{42} & W_{43} & W_{44} \\ 0 & 0 & W_{53} & W_{54} \\ 0 & 0 & 0 & W_{64} \end{bmatrix}$$

- Zero-padding es agregar ceros en los bordes para aplicar el mismo patrón siempre.

Equivarianza por traslaciones

- Una capa local f es equivariante por traslaciones si

$$\mathbf{P}_k(i,j) = \mathbf{P}_k(i',j') \Longrightarrow f\left(\mathbf{P}_k(i,j)\right) = f\left(\mathbf{P}_k(i',j')\right),$$

  para todos los píxeles (i,j) y (i',j') donde los parches estén bien definidos.

- Intuición: Si se traslada un patch en la imagen desde (i,j) a (i',j'), entonces las activaciones "se mueven" junto con el patch.

Pesos compartidos $\Rightarrow$ equivarianza

- Idea: compartir pesos en todas las posiciones

$$H_{ij} = A\left(\text{vect}\left(\mathbf{P}_k(i,j)\right)^\top \cdot \mathbf{W}\right), \quad \mathbf{W} \text{ independiente de } (i,j).$$

Eficiencia en parámetros

- $\mathbf{W} \sim (s^2 c, c')$ con $s = 2k+1$.
- Independiente de (h,w): no crece con la resolución.
- Comparado con una capa local: reducción adicional por un factor $\frac{1}{hw}$.

Nombre: esta construcción es una **capa convolucional**.

Ejemplo con pesos compartidos

- Ejemplo 1D: $\mathbf{x} = [x_1, x_2, x_3, x_4]^\top$, radio $k = 1$ ($s = 3$).

- La capa local puede escribirse como

$$\begin{bmatrix} h_1 & h_2 & h_3 & h_4 \end{bmatrix} = \begin{bmatrix} 0 & x_1 & x_2 & x_3 & x_4 & 0 \end{bmatrix} \begin{bmatrix} W_1 & 0 & 0 & 0 \\ W_2 & W_1 & 0 & 0 \\ W_3 & W_2 & W_1 & 0 \\ 0 & W_3 & W_2 & W_1 \\ 0 & 0 & W_3 & W_2 \\ 0 & 0 & 0 & W_3 \end{bmatrix}$$

Capa convolucional 2D: definición completa

- Entrada $\mathbf{X} \sim (h, w, c)$, kernel size $s = 2k + 1$, número de canales de salida c'.
- Definición (elemento a elemento):

$$H_{ij} = \text{vect}\left(\mathbf{P}_k(i, j)\right)^\top \mathbf{W} + \mathbf{b}^\top,$$

- Parámetros entrenables son $\mathbf{W} \sim (s^2 c, c')$, $\quad \mathbf{b} \sim (c')$.
- Hiperparámetros: k (o s), c', y si se aplica zero-padding.
- Shape de salida: $\mathbf{H} \sim \begin{cases} (h, w, c') & \text{con padding de } k \text{ píxeles} \\ (h - 2k,\ w - 2k,\ c') & \text{sin padding.} \end{cases}$