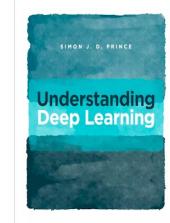


Bibliografía

- Understanding Deep Learning. Capítulo 10.



Tema 6 – Redes de Neuronas Convolucionales (CNNs)

Aprendizaje Automático II - Grado en Inteligencia Artificial
Universidad Rey Juan Carlos

Iván Ramírez Díaz
ivan.ramirez@urjc.es

José Miguel Buenaposada Biencinto
josemiguel.buenaposada@urjc.es

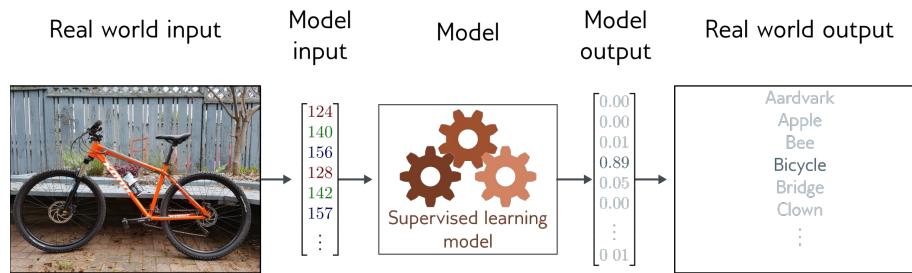
6.1 Redes de Neuronas Convolucionales

- Redes para imágenes
- Invarianza y Equivarianza
- Convoluciones 1D
- Capas convolucionales
- Canales
- Campo receptivo (Receptive field)
- Red convolucional para MNIST 1D
- Backpropagation con CNNs

6.1 Redes de Neuronas Convolucionales

- Redes para imágenes
- Invarianza y Equivarianza
- Convoluciones 1D
- Capas convolucionales
- Canales
- Campo receptivo (Receptive field)
- Red convolucional para MNIST 1D
- Backpropagation con CNNs

Clasificación de imágenes

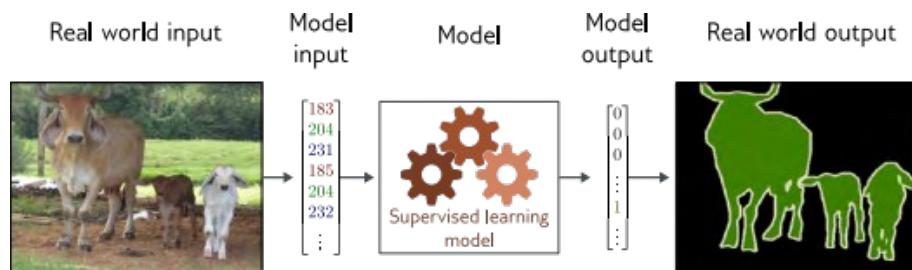


- Clasificación multiclase (clases discretas, >2 clases)
- Convolutional Neural Network (CNN)

Detección de objetos



Segmentación de imágenes



- Clasificación binaria multivariante (muchas salidas, 2 clases)
- Convolutional encoder-decoder Network (CNN)

Problemas de las MLPs para imágenes

1. Tamaño (número de parámetros)
 - $224 \times 224 \text{ píxeles/imagen} = 150528 \text{ dimensiones}$
 - Capas ocultas con más parámetros que la entrada
 - 1 capa oculta = $150520 \times 150528 \rightarrow 22000 \text{ Millones}$
2. Píxeles cercanos relacionados estadísticamente
 - Pero con la FCN podría permutar píxeles, reentrenar y obtener los mismos resultados
3. No son estables a transformaciones geométricas
 - Necesitamos entrenar con un objeto en todas las posibles posiciones de la imagen

Redes convolucionales

- Los **parámetros afectan localmente** a una región rectangular de la imagen (no a toda la imagen).
- Los mismos **parámetros se comparten** en todas las regiones rectangulares de la imagen.

6.1 Redes de Neuronas Convolucionales

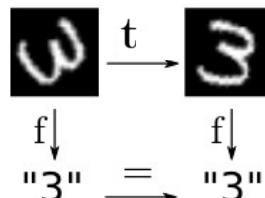
- Redes para imágenes
- Invarianza y Equivarianza
- Convoluciones 1D
- Capas convolucionales
- Canales
- Campo receptivo (Receptive field)
- Red convolucional para MNIST 1D
- Backpropagation con CNNs

Invarianza a transformaciones

- Una función $f[x]$ es **invariante a una transformación** $t[]$ si:

$$f[t[x]] = f[x]$$

la salida de la función es la misma incluso después de aplicarle la transformación a x



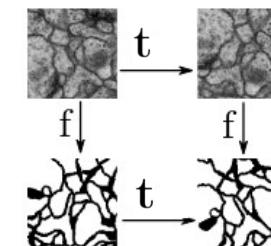
Diego Marcos, Michele Volpi, Nikos Komodakis, Devis Tuia:
Rotation Equivariant Vector Field Networks. ICCV 2017:
5058-5067

Equivarianza a transformaciones

- Una función $f[x]$ es **equivariante a una transformación** $t[]$ si:

$$f[t[x]] = t[f[x]]$$

la salida de la función se transforma igual que la entrada x



Diego Marcos, Michele Volpi, Nikos Komodakis, Devis Tuia:
Rotation Equivariant Vector Field Networks. ICCV 2017:
5058-5067

Equivarianza a transformaciones

- **Ejemplo:** segmentación de imágenes
 - La imagen se traslada y queremos que la segmentación (separación en regiones coherentes) se traslade con ella



6.1 Redes de Neuronas Convolucionales

- Redes para imágenes
- Invarianza y Equivarianza
- **Convoluciones 1D**
- Capas convolucionales
- Canales
- Campo receptivo (Receptive field)
- Red convolucional para MNIST 1D
- Backpropagation con CNNs

Convolución en 1D

Sean $f(x)$ y $g(y)$ dos funciones discretas definidas en los dominios $x \in \{0, \dots, a-1\}$, $y \in \{0, \dots, b-1\}$

$f(x)$	3	4	5
--------	---	---	---

$g(y)$	1	2	3	4	5
--------	---	---	---	---	---

Convolución en 1D

Sean $f(x)$ y $g(y)$ dos funciones discretas definidas en los dominios $x \in \{0, \dots, a-1\}$, $y \in \{0, \dots, b-1\}$.

La convolución de f con g es otra función $s(r) = f * g$ donde $r \in \{0, \dots, s-1\}$; $s = a + b - 1$.

$$s(r) = f * g$$

--	--	--	--	--	--	--

$f(x)$	3	4	5
--------	---	---	---

$g(y)$	1	2	3	4	5
--------	---	---	---	---	---

Convolución en 1D

Sean $f(x)$ y $g(y)$ dos funciones discretas definidas en los dominios $x \in \{0, \dots, a-1\}$, $y \in \{0, \dots, b-1\}$.

La convolución de f con g es otra función $s(r) = f * g$ donde $r \in \{0, \dots, s-1\}$; $s = a + b - 1$.

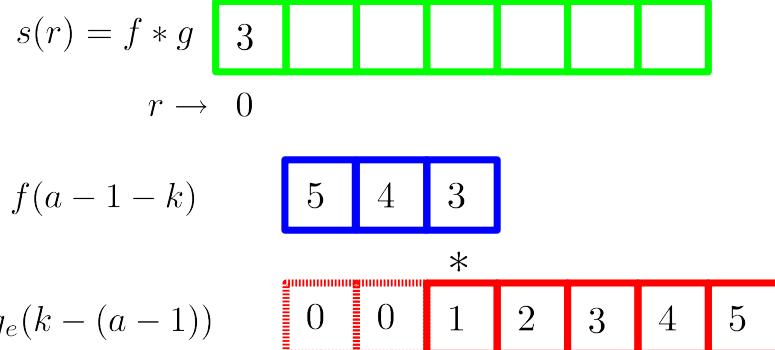
$$s(r) = \sum_{k=0}^{a-1} f(a-1-k)g_e(k+r-(a-1))$$

siendo

$$g_e(y) = \begin{cases} g(y) & \text{si } y \in \{0, \dots, b-1\} \\ 0 & \text{en otro caso} \end{cases}$$

Convolución en 1D

$$s(r) = \sum_{k=0}^{a-1} f(a-1-k)g_e(k+r-(a-1))$$



Convolución en 1D

$$s(r) = \sum_{k=0}^{a-1} f(a-1-k)g_e(k+r-(a-1))$$

$$s(r) = f * g$$

3	10					
---	----	--	--	--	--	--

$r \rightarrow 1$

$f(a-1-k)$

5	4	3
---	---	---

*

$g_e(k+1-(a-1))$

0	0	1	2	3	4	5
---	---	---	---	---	---	---

Convolución en 1D

$$s(r) = \sum_{k=0}^{a-1} f(a-1-k)g_e(k+r-(a-1))$$

$$s(r) = f * g$$

3	10	22				
---	----	----	--	--	--	--

$r \rightarrow 2$

$f(a-1-k)$

5	4	3
---	---	---

*

$g_e(k+2-(a-1))$

0	0	1	2	3	4	5
---	---	---	---	---	---	---

Convolución en 1D

$$s(r) = \sum_{k=0}^{a-1} f(a-1-k)g_e(k+r-(a-1))$$

$$s(r) = f * g \quad \boxed{3 \quad 10 \quad 22 \quad 34 \quad \boxed{} \quad \boxed{}}$$

$r \rightarrow \quad \quad \quad 3$

$$f(a-1-k) \quad \quad \quad \boxed{5 \quad 4 \quad 3}$$

* *

$$g_e(k+3-(a-1)) \quad \boxed{0 \quad 0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5}$$

Convolución en 1D

$$s(r) = \sum_{k=0}^{a-1} f(a-1-k)g_e(k+r-(a-1))$$

$$s(r) = f * g \quad \boxed{3 \quad 10 \quad 22 \quad 34 \quad 46 \quad \boxed{} \quad \boxed{}}$$

$r \rightarrow \quad \quad \quad 4$

$$f(a-1-k) \quad \quad \quad \boxed{5 \quad 4 \quad 3}$$

* *

$$g_e(k+4-(a-1)) \quad \boxed{0 \quad 0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5}$$

Convolución en 1D

$$s(r) = \sum_{k=0}^{a-1} f(a-1-k)g_e(k+r-(a-1))$$

$$s(r) = f * g \quad \boxed{3 \quad 10 \quad 22 \quad 34 \quad 46 \quad 40 \quad \boxed{}}$$

$r \rightarrow \quad \quad \quad 5$

$$f(a-1-k) \quad \quad \quad \boxed{5 \quad 4 \quad 3}$$

* *

$$g_e(k+5-(a-1)) \quad \boxed{0 \quad 0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 0}$$

Convolución en 1D

$$s(r) = \sum_{k=0}^{a-1} f(a-1-k)g_e(k+r-(a-1))$$

$$s(r) = f * g \quad \boxed{3 \quad 10 \quad 22 \quad 34 \quad 46 \quad 40 \quad 25}$$

$r \rightarrow \quad \quad \quad 6$

En la convolución se invierte la máscara $f()$ y "se pasa" por la función $g()$ extendida con 0s, $g_e()$

$$f(a-1-k) \quad \quad \quad \boxed{5 \quad 4 \quad 3}$$

* *

$$g_e(k+6-(a-1)) \quad \boxed{0 \quad 0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 0 \quad 0}$$

Correlación cruzada en 1D

- Diferencia con la convolución: La máscara no se invierte pero por lo demás el algoritmo para calcular el resultado es el mismo:
 - Se pasa la máscara (kernel) por la función $g()$ se multiplica elemento a elemento donde se encuentre situada y se suman los resultados.

Redes de convolución

- ¿Qué se usa en redes de convolución? El algoritmo que se utiliza es el común a correlación cruzada / convolución:
 - Como los valores en la máscara (kernel) son aprendidos en realidad se puede interpretar como una convolución (con los pesos dados la vuelta).

IMPORTANTE: A partir de este momento hablaremos de convolución en cualquier caso (aunque se trate de una correlación cruzada y no se de la vuelta al kernel de convo

Respuesta de la correlación cruzada

- En cada paso de la correlación cruzada se calcula el producto escalar de w con una parte de la entrada x , $p=(x_i, \dots x_{i+k-1})^T$.
- Este producto escalar es:
$$w \cdot p = \|w\| \|p\| \cos \alpha$$
 donde α es el ángulo entre w y p
- Suponiendo $\|w\| \|p\|$ de valor fijo,
 - La máxima respuesta del filtro w (el valor de $w \cdot p$) se dará cuando se encuentre en la misma dirección que p ¡cuando w y p sean iguales!

Convolución en 1D para redes CNN

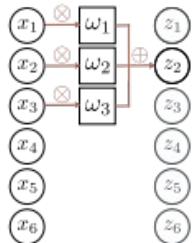
- Vector de entrada x :
$$x = [x_1, x_2, \dots, x_I]$$
- La salida es la suma ponderada de los vecinos:
$$z_i = \omega_1 x_{i-1} + \omega_2 x_i + \omega_3 x_{i+1}$$
- Un filtro, máscara o kernel de convolución:

$$\omega = [\omega_1, \omega_2, \omega_3]^T \quad \text{Kernel de tamaño 3}$$

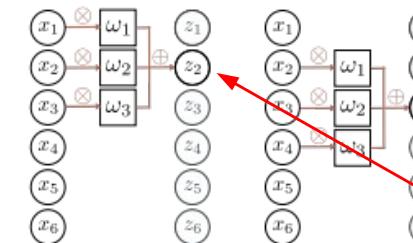
Convolución con kernel de tamaño 3

Convolución con kernel de tamaño 3

a)



a)

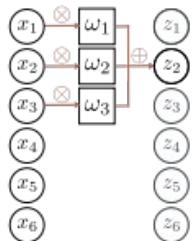


b)

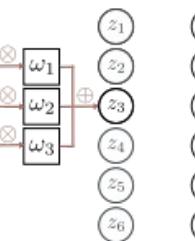
Equivariante a la traslación de la entrada:
 $f[t[x]] = t[f[x]]$

Relleno con ceros (zero padding)

a)



b)



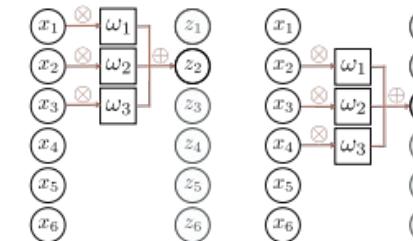
c)



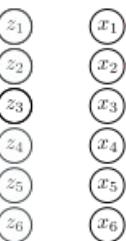
Las localizaciones "fuera" de la entrada se tratan como si tuvieran valor 0

Convoluciones "válidas"

a)



b)



c)



d)

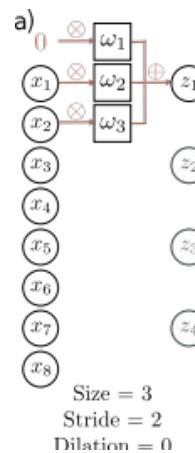


En este caso únicamente se procesan las localizaciones donde el kernel se encuentra completamente dentro de la entrada (salida más pequeña que con relleno – padding –)

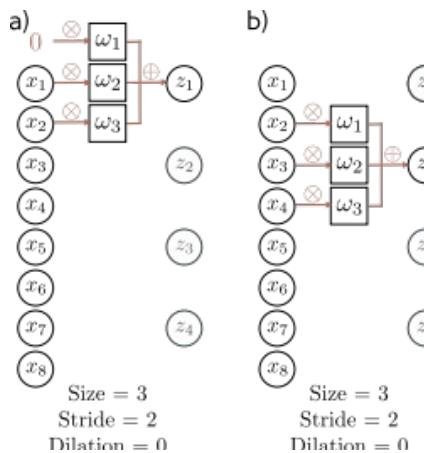
Stride (paso), tamaño del kernel, dilatación

- **Stride** = desplazar k posiciones para cada salida
 - Decrementa el tamaño de la salida con respecto a la entrada.
- **Kernel size** = aplicar pesos a un número diferente de componentes de la entrada para obtener cada componente de salida.
 - Combinar información de un área más grande
 - Pero en 1D, kernel size = 5 implica utilizar 5 parámetros.
- **Dilated o atrous convolutions** = aplicar pesos a un número diferente de componentes de la entrada para obtener cada componente de salida.

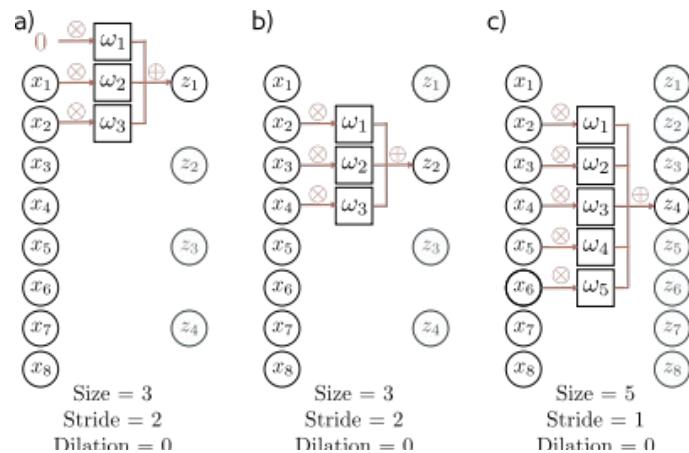
Stride (paso), tamaño del kernel, dilatación



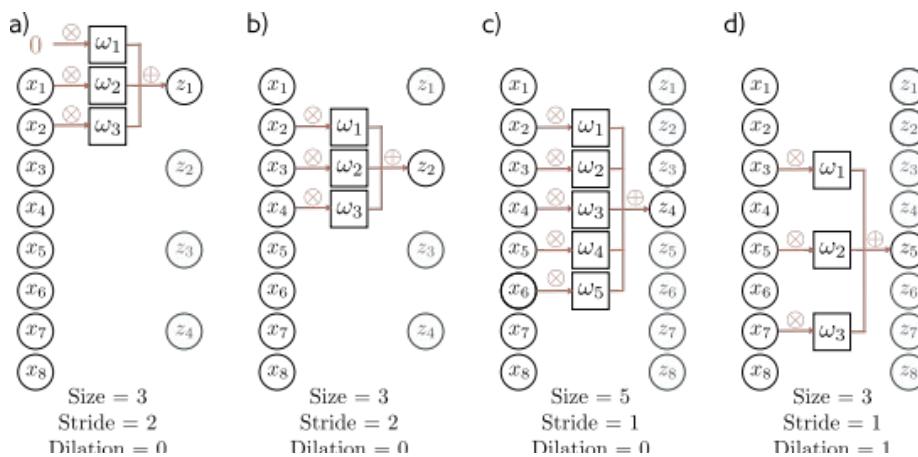
Stride (paso), tamaño del kernel, dilatación



Stride (paso), tamaño del kernel, dilatación



Stride (paso), tamaño del kernel, dilatación



Capa convolucional 1D

- Activación i -ésima (h_i) en una **capa convolucional**:

$$h_i = a [\beta + \omega_1 x_{i-1} + \omega_2 x_i + \omega_3 x_{i+1}]$$

$$= a \left[\beta + \sum_{j=1}^3 \omega_j x_{i+j-2} \right]$$

← Capa completa:
3 pesos, 1 sesgo

- Activación i -ésima (h_i) en una **capa de una red MLP**:

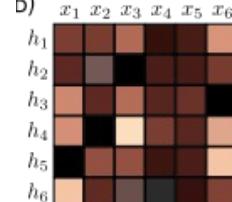
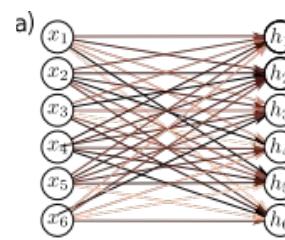
$$h_i = a \left[\beta_i + \sum_{j=1}^D \omega_{ij} x_j \right]$$

← Capa completa
(nº entradas = nº salidas)
 $D \times D$ pesos, D sesgos

6.1 Redes de Neuronas Convolucionales

- Redes para imágenes
- Invarianza y Equivarianza
- Convoluciones 1D
- Capas convolucionales**
- Canales
- Campo receptivo (Receptive field)
- Red convolucional para MNIST 1D
- Backpropagation con CNNs

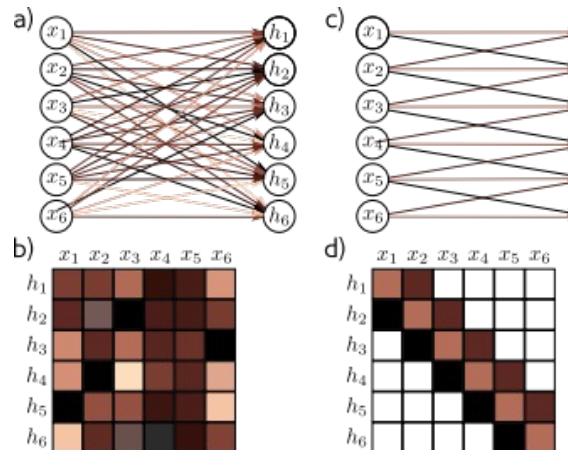
Caso especial de una red Fully Connected



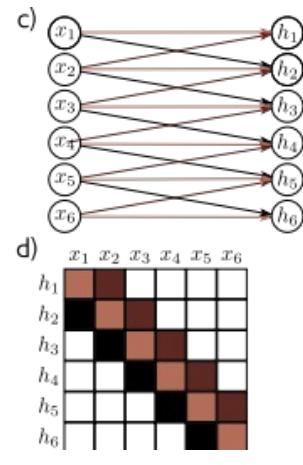
Fully connected network

Convolution, size 3, stride 1,
dilation 0, zero padding

Caso especial de una red Fully Connected



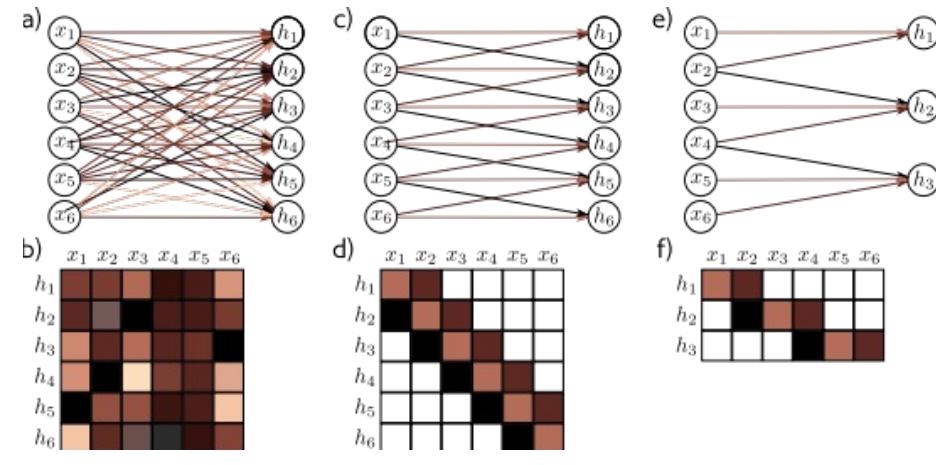
Fully connected network



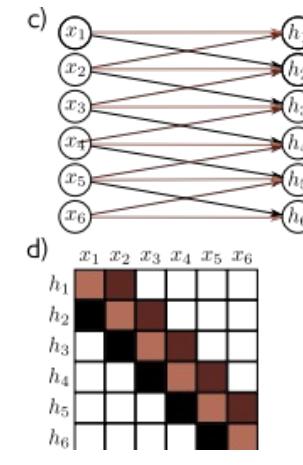
Convolution, size 3, stride 1,
dilation 0, zero padding

Convolution, size 3, stride 2,
dilation 0, zero padding

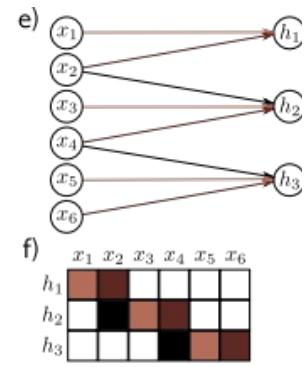
Caso especial de una red Fully Connected



Fully connected network



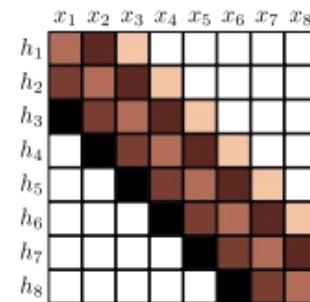
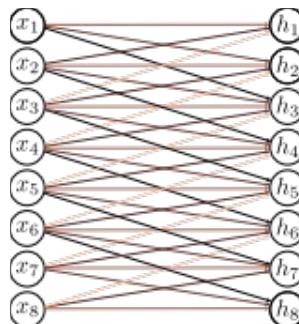
Convolution, size 3, stride 1,
dilation 0, zero padding



Convolution, size 3, stride 2,
dilation 0, zero padding

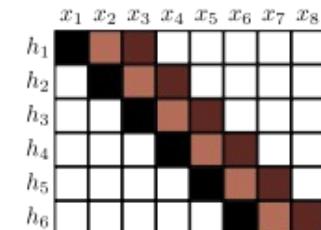
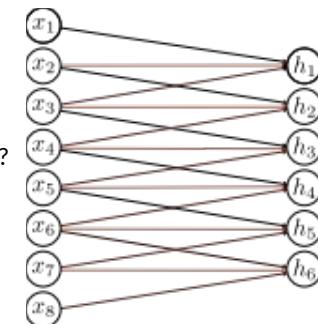
Pregunta 1

- ¿Kernel size?
- ¿Stride?
- ¿Dilation?
- Zero padding / valid?



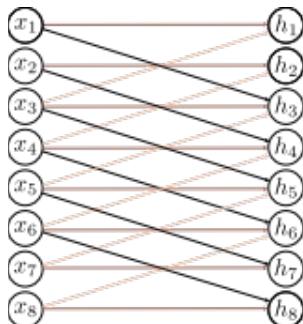
Pregunta 2

- ¿Kernel size?
- ¿Stride?
- ¿Dilation?
- Zero padding / valid?



Pregunta 3

- ¿Kernel size?
- ¿Stride?
- ¿Dilation?
- Zero padding / valid?



	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
h_1	■							
h_2		■						
h_3			■					
h_4				■				
h_5					■			
h_6						■		
h_7							■	
h_8								■

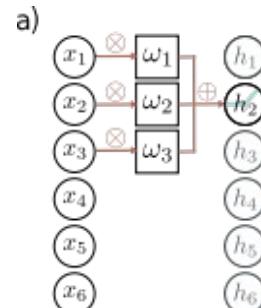
Canales

- La operación de convolución promedia todas las entradas
- Además se utilizar la activación ReLU
- Tiene que perder información
- Solución:
 - Aplicar diferentes convoluciones y apilarlas en canales
 - También se llaman **feature maps** (mapas de características)

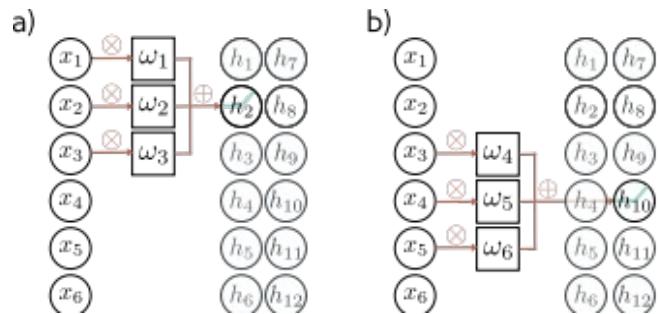
6.1 Redes de Neuronas Convolucionales

- Redes para imágenes
- Invarianza y Equivarianza
- Convoluciones 1D
- Capas convolucionales
- **Canales**
- Campo receptivo (Receptive field)
- Red convolucional para MNIST 1D
- Backpropagation con CNNs

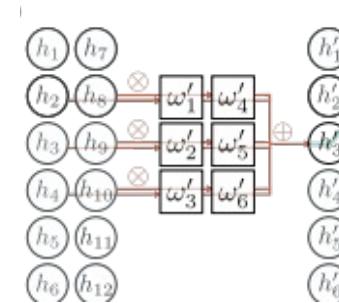
Dos canales de salida, uno de entrada



Dos canales de salida, uno de entrada



Un canal de salida, dos de entrada



¿Cuántos parámetros?

- Si tenemos C_i canales de entrada y un kernel de tamaño K :

Capa completa:
 $C_i \cdot K + 1$ parámetros

$$\Omega \in \mathbb{R}^{C_i \times K} \quad \beta \in \mathbb{R}$$

- Si hay C_i canales de entrada y C_o canales de salida:

Capa completa:
 $C_i \cdot C_o \cdot K + C_o$ parámetros

$$\Omega \in \mathbb{R}^{C_i \times C_o \times K} \quad \beta \in \mathbb{R}^{C_o}$$

¿Tamaño de la salida?

- Tenemos una capa convolucional con:
 - C_i canales de entrada de longitud D_i
 - C_o canales de salida,
 - kernel de tamaño K ,
 - Stride S
 - número de celdas añadidas con ceros P (*zero padding*)
- El resultado será un feature map de tamaño $D_o \times C_o$, donde:

$$D_o = \text{floor}[(D_i - K + 2 \cdot P)/S + 1]$$

Convolución 1D en Pytorch

Docs > torch.nn > Conv1d

Conv1d

CLASS `torch.nn.Conv1d(in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True, padding_mode='zeros', device=None, dtype=None)` [SOURCE]

Applies a 1D convolution over an input signal composed of several input planes.

In the simplest case, the output value of the layer with input size (N, C_{in}, L) and output $(N, C_{\text{out}}, L_{\text{out}})$ can be precisely described as:

$$\text{out}(N_i, C_{\text{out},j}) = \text{bias}(C_{\text{out},j}) + \sum_{k=0}^{C_{\text{in}}-1} \text{weight}(C_{\text{out},j}, k) * \text{input}(N_i, k)$$

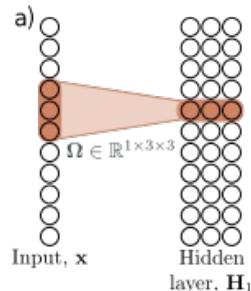
where $*$ is the valid cross-correlation operator, N is a batch size, C denotes a number of channels, L is a length of signal sequence.

This module supports `TensorFloat32`.

On certain ROCm devices, when using float16 inputs this module will use different precision for backward.

- `stride` controls the stride for the cross-correlation, a single number or a one-element tuple.
- `padding` controls the amount of padding applied to the input. It can be either a string `{'valid', 'same'}` or a tuple of ints giving the amount of implicit padding applied on both sides.
- `dilation` controls the spacing between the kernel points; also known as the à trous algorithm. It is harder to describe, but this [link](#) has a nice visualization of what `dilation` does.
- `groups` controls the connections between inputs and outputs. `in_channels` and `out_channels` must both be divisible by `groups`. For example,
 - At `groups=1`, all inputs are convolved to all outputs.
 - At `groups=2`, the operation becomes equivalent to having two conv layers side by side, each seeing half the input channels and producing half the output channels, and both subsequently concatenated.
 - At `groups= in_channels`, each input channel is convolved with its own set of filters (of size $\frac{\text{out_channels}}{\text{in_channels}}$).

Campo Receptivo (receptive field)



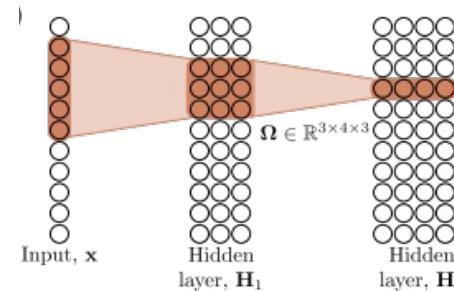
$$\Omega \in \mathbb{R}^{C_i \times C_o \times K}$$

$$\beta \in \mathbb{R}^{C_o}$$

6.1 Redes de Neuronas Convolucionales

- Redes para imágenes
- Invarianza y Equivarianza
- Convoluciones 1D
- Capas convolucionales
- Canales
- Campo receptivo (Receptive field)
- Red convolucional para MNIST 1D
- Backpropagation con CNNs

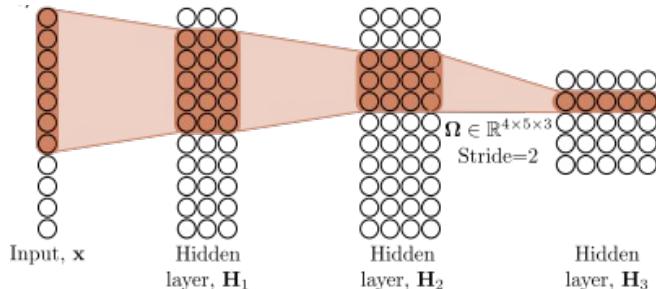
Campo Receptivo (receptive field)



$$\Omega \in \mathbb{R}^{C_i \times C_o \times K}$$

$$\beta \in \mathbb{R}^{C_o}$$

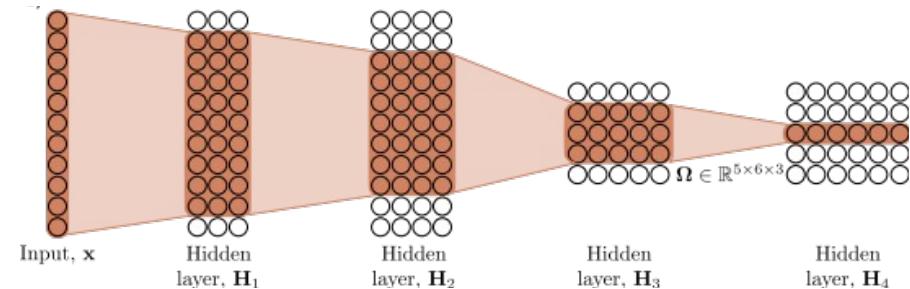
Campo Receptivo (receptive field)



$$\Omega \in \mathbb{R}^{C_i \times C_o \times K}$$

$$\beta \in \mathbb{R}^{C_o}$$

Campo Receptivo (receptive field)



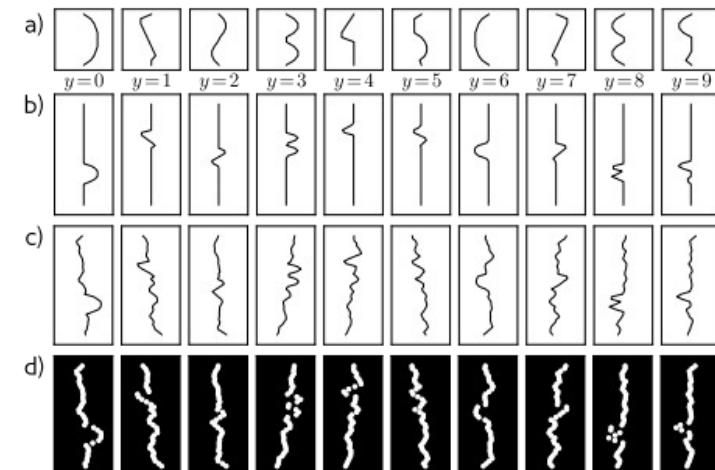
$$\Omega \in \mathbb{R}^{C_i \times C_o \times K}$$

$$\beta \in \mathbb{R}^{C_o}$$

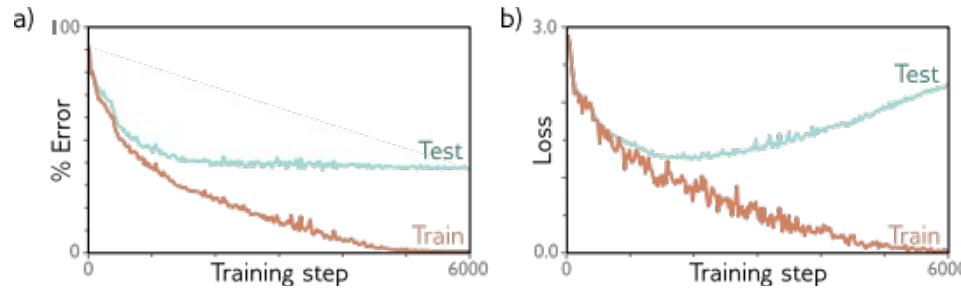
6.1 Redes de Neuronas Convolucionales

- Redes para imágenes
- Invarianza y Equivarianza
- Convoluciones 1D
- Capas convolucionales
- Canales
- Campo receptivo (Receptive field)
- **Red convolucional para MNIST 1D**
- Backpropagation con CNNs

MNIST 1D Dataset



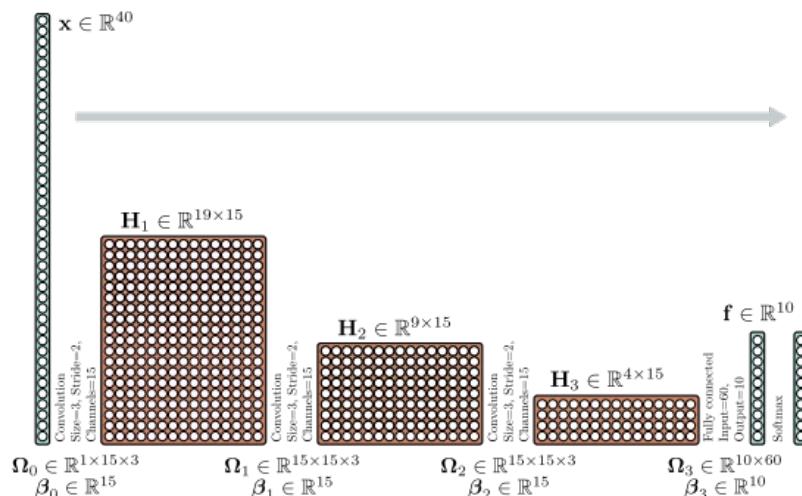
MNIST 1D: resultados red Fully Connected



Red convolucional

- 4 capas ocultas:
 - 3 capas convolucionales
 - Una capa fully connected (FC)
- Softmax al final
- Parámetros = 2050
- Entrenada 100000 iteraciones con SGD, LR=0.01, batch size 100

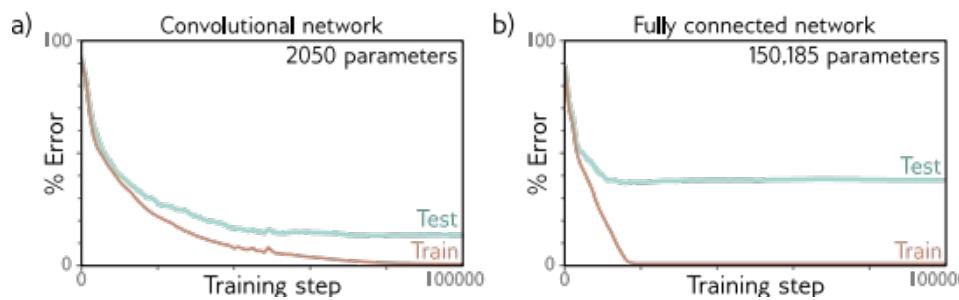
MNIST 1D red convolucional



MNIST 1D red Fully Connected

- 4 capas ocultas con mismo número de unidades ocultas
- Todas capas fully connected
- Parámetros = 150185

Resultados



¿Por qué la CNN es mejor?

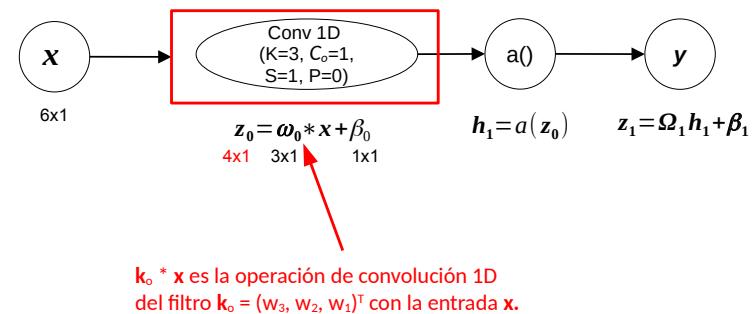
- Mejor sesgo inductivo (inductive bias) en la CNN para ayudar a entrenar:
 - Se fuerza a la red a procesar cada localización de la misma manera.
 - Comparte información entre localizaciones
- Se busca en una familia de funciones más pequeña, siendo todas ellas plausibles.

Sesgo inductivo: el conjunto de suposiciones que el modelo utiliza para predecir salidas dadas unas entradas que no ha visto nunca

6.1 Redes de Neuronas Convolucionales

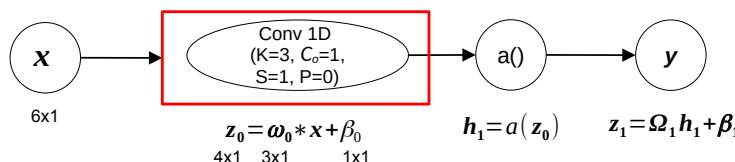
- Redes para imágenes
- Invarianza y Equivarianza
- Convoluciones 1D
- Capas convolucionales
- Canales
- Campo receptivo (Receptive field)
- Red convolucional para MNIST 1D
- Backpropagation con CNNs

Backprop con capa convolucional 1D



Pregunta: ¿Cuál es la dimensión de salida C_o , de esta capa

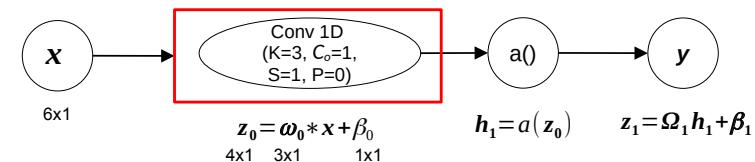
Backprop con capa convolucional 1D



- Supongamos conocida la *derivada de la función de pérdida J*: $\frac{\partial J}{\partial z_0} = (a b c d)^T$
- Y ahora queremos calcular:

$$\frac{\partial L}{\partial x} = \frac{\partial z_0}{\partial x} \cdot \frac{\partial L}{\partial z_0}$$

Backprop con capa convolucional 1D

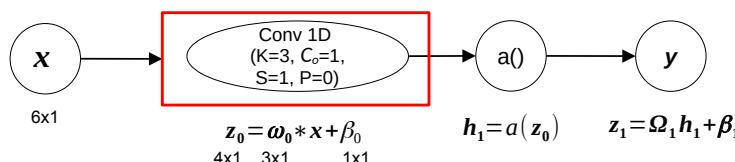


- Supongamos conocida la *derivada de la función de pérdida J*: $\frac{\partial J}{\partial z_0} = (a b c d)^T$
- Y ahora queremos calcular:

$$\frac{\partial L}{\partial x} = \frac{\partial z_0}{\partial x} \cdot \frac{\partial L}{\partial z_0}$$

Dado que: $z_0 = \begin{bmatrix} w_1 & w_2 & w_3 & 0 & 0 & 0 \\ 0 & w_1 & w_2 & w_3 & 0 & 0 \\ 0 & 0 & w_1 & w_2 & w_3 & 0 \\ 0 & 0 & 0 & w_1 & w_2 & w_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} + \begin{bmatrix} \beta_0 \\ \beta_0 \\ \beta_0 \\ \beta_0 \\ \beta_0 \\ \beta_0 \end{bmatrix}$

Backprop con capa convolucional 1D

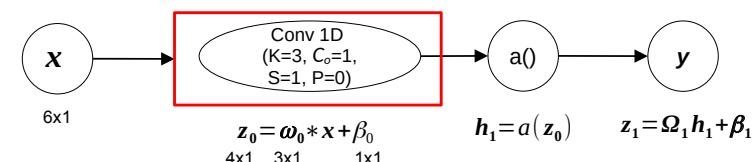


- Supongamos conocida la *derivada de la función de pérdida J*: $\frac{\partial J}{\partial z_0} = (a b c d)^T$
- Y ahora queremos calcular:

$$\frac{\partial L}{\partial x} = \frac{\partial z_0}{\partial x} \cdot \frac{\partial L}{\partial z_0}$$

Dado que: $z_0 = \begin{bmatrix} w_1 & w_2 & w_3 & 0 & 0 & 0 \\ 0 & w_1 & w_2 & w_3 & 0 & 0 \\ 0 & 0 & w_1 & w_2 & w_3 & 0 \\ 0 & 0 & 0 & w_1 & w_2 & w_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} + \begin{bmatrix} \beta_0 \\ \beta_0 \\ \beta_0 \\ \beta_0 \\ \beta_0 \\ \beta_0 \end{bmatrix} = \begin{bmatrix} w_1 \cdot x_1 & w_2 \cdot x_2 & w_3 \cdot x_3 + \beta_0 \\ w_1 \cdot x_2 & w_2 \cdot x_3 & w_3 \cdot x_4 + \beta_0 \\ w_1 \cdot x_3 & w_2 \cdot x_4 & w_3 \cdot x_5 + \beta_0 \\ w_1 \cdot x_4 & w_2 \cdot x_5 & w_3 \cdot x_6 + \beta_0 \end{bmatrix}$

Backprop con capa convolucional 1D

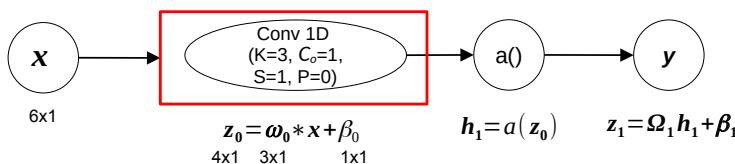


- Supongamos conocida la *derivada de la función de pérdida J*: $\frac{\partial J}{\partial z_0} = (a b c d)^T$
- Y ahora queremos calcular:

$$\frac{\partial L}{\partial x} = \frac{\partial z_0}{\partial x} \cdot \frac{\partial L}{\partial z_0}$$

Dado que: $z_0 = \begin{bmatrix} w_1 \cdot x_1 & w_2 \cdot x_2 & w_3 \cdot x_3 + \beta_0 \\ w_1 \cdot x_2 & w_2 \cdot x_3 & w_3 \cdot x_4 + \beta_0 \\ w_1 \cdot x_3 & w_2 \cdot x_4 & w_3 \cdot x_5 + \beta_0 \\ w_1 \cdot x_4 & w_2 \cdot x_5 & w_3 \cdot x_6 + \beta_0 \end{bmatrix}$ $\rightarrow \frac{\partial z_0}{\partial x} = \begin{bmatrix} \frac{\partial z_0(1)}{\partial x_1} & \dots & \frac{\partial z_0(4)}{\partial x_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial z_0(1)}{\partial x_6} & \dots & \frac{\partial z_0(4)}{\partial x_6} \end{bmatrix}$

Backprop con capa convolucional 1D

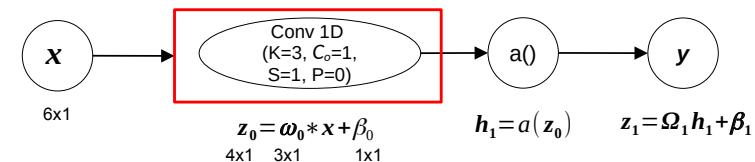


- Supongamos conocida la *derivada de la función de pérdida J*: $\frac{\partial J}{\partial \mathbf{z}_0} = (a \ b \ c \ d)^T$
- Y ahora queremos calcular:

$$\frac{\partial L}{\partial \mathbf{x}} = \frac{\partial \mathbf{z}_0}{\partial \mathbf{x}} \cdot \frac{\partial L}{\partial \mathbf{z}_0}$$

Dado que: $\mathbf{z}_0 = \begin{bmatrix} w_1 \cdot x_1 & w_2 \cdot x_2 & w_3 \cdot x_3 + \beta_0 \\ w_1 \cdot x_2 & w_2 \cdot x_3 & w_3 \cdot x_4 + \beta_0 \\ w_1 \cdot x_3 & w_2 \cdot x_4 & w_3 \cdot x_5 + \beta_0 \\ w_1 \cdot x_4 & w_2 \cdot x_5 & w_3 \cdot x_6 + \beta_0 \end{bmatrix}$ $\frac{\partial \mathbf{z}_0}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial \mathbf{z}_0(1)}{\partial \mathbf{x}_1} & \dots & \frac{\partial \mathbf{z}_0(4)}{\partial \mathbf{x}_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial \mathbf{z}_0(1)}{\partial \mathbf{x}_6} & \dots & \frac{\partial \mathbf{z}_0(4)}{\partial \mathbf{x}_6} \end{bmatrix} = \begin{bmatrix} w_1 & 0 & 0 & 0 \\ w_2 & w_1 & 0 & 0 \\ w_3 & w_2 & w_1 & 0 \\ 0 & w_3 & w_2 & w_1 \\ 0 & 0 & w_3 & w_2 \\ 0 & 0 & 0 & w_3 \end{bmatrix}$

Backprop con capa convolucional 1D



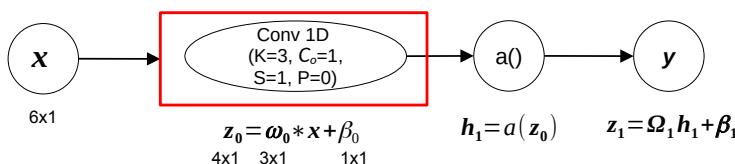
- Supongamos conocida la *derivada de la función de pérdida J*: $\frac{\partial J}{\partial \mathbf{z}_0} = (a \ b \ c \ d)^T$
- Y ahora queremos calcular:

$$\frac{\partial L}{\partial \mathbf{x}} = \frac{\partial \mathbf{z}_0}{\partial \mathbf{x}} \cdot \frac{\partial L}{\partial \mathbf{z}_0}$$

Convolución con el filtro
 $r(\mathbf{k}_0) = (w_1, w_2, w_3)^T$

Dado que: $\mathbf{z}_0 = \begin{bmatrix} w_1 \cdot x_1 & w_2 \cdot x_2 & w_3 \cdot x_3 + \beta_0 \\ w_1 \cdot x_2 & w_2 \cdot x_3 & w_3 \cdot x_4 + \beta_0 \\ w_1 \cdot x_3 & w_2 \cdot x_4 & w_3 \cdot x_5 + \beta_0 \\ w_1 \cdot x_4 & w_2 \cdot x_5 & w_3 \cdot x_6 + \beta_0 \end{bmatrix}$ $\frac{\partial \mathbf{z}_0}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial \mathbf{z}_0}{\partial \mathbf{x}_1} & \dots & \frac{\partial \mathbf{z}_0}{\partial \mathbf{x}_6} \end{bmatrix} = \begin{bmatrix} w_1 & 0 & 0 & 0 \\ w_2 & w_1 & 0 & 0 \\ w_3 & w_2 & w_1 & 0 \\ 0 & w_3 & w_2 & w_1 \\ 0 & 0 & w_3 & w_2 \\ 0 & 0 & 0 & w_3 \end{bmatrix}$ $\frac{\partial L}{\partial \mathbf{x}} = \frac{\partial \mathbf{z}_0}{\partial \mathbf{x}} \cdot \frac{\partial L}{\partial \mathbf{z}_0} = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$

Backprop con capa convolucional 1D



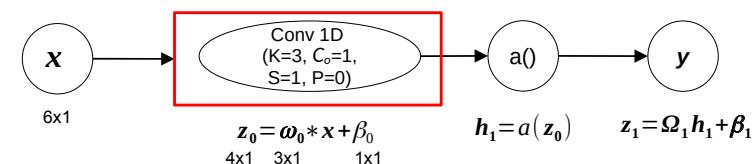
- Supongamos conocida la *derivada de la función de pérdida J*: $\frac{\partial J}{\partial \mathbf{z}_0} = (a \ b \ c \ d)^T$
- Y ahora queremos calcular:

$$\frac{\partial L}{\partial \mathbf{x}} = \frac{\partial \mathbf{z}_0}{\partial \mathbf{x}} \cdot \frac{\partial L}{\partial \mathbf{z}_0} = r(\omega_0) * \frac{\partial L}{\partial \mathbf{z}_0}$$

Convolución con el filtro
 $r(\mathbf{k}_0) = (w_1, w_2, w_3)^T$

Dado que: $\mathbf{z}_0 = \begin{bmatrix} w_1 \cdot x_1 & w_2 \cdot x_2 & w_3 \cdot x_3 + \beta_0 \\ w_1 \cdot x_2 & w_2 \cdot x_3 & w_3 \cdot x_4 + \beta_0 \\ w_1 \cdot x_3 & w_2 \cdot x_4 & w_3 \cdot x_5 + \beta_0 \\ w_1 \cdot x_4 & w_2 \cdot x_5 & w_3 \cdot x_6 + \beta_0 \end{bmatrix}$ $\frac{\partial L}{\partial \mathbf{x}} = \frac{\partial \mathbf{z}_0}{\partial \mathbf{x}} \cdot \frac{\partial L}{\partial \mathbf{z}_0} = \begin{bmatrix} w_1 & 0 & 0 & 0 \\ w_2 & w_1 & 0 & 0 \\ w_3 & w_2 & w_1 & 0 \\ 0 & w_3 & w_2 & w_1 \\ 0 & 0 & w_3 & w_2 \\ 0 & 0 & 0 & w_3 \end{bmatrix} = r(\omega_0) * \frac{\partial L}{\partial \mathbf{z}_0}$

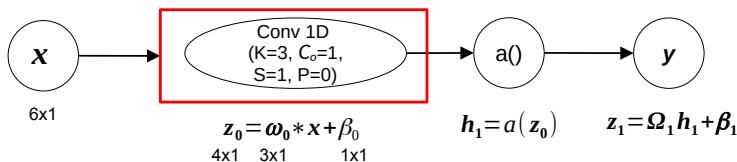
Backprop con capa convolucional 1D



- Supongamos conocida la *derivada de la función de pérdida J*: $\frac{\partial J}{\partial \mathbf{z}_0} = (a \ b \ c \ d)^T$
- Si ahora queremos calcular:

$$\frac{\partial L}{\partial \omega_0} = \frac{\partial \mathbf{z}_0}{\partial \omega_0} \cdot \frac{\partial L}{\partial \mathbf{z}_0}$$

Backprop con capa convolucional 1D

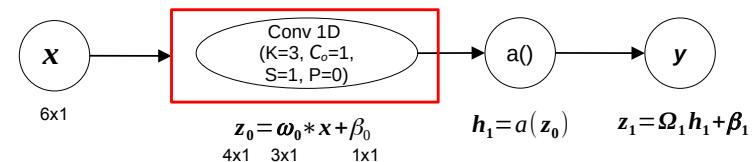


- Supongamos conocida la *derivada de la función de pérdida J*: $\frac{\partial J}{\partial \mathbf{z}_0} = (a \ b \ c \ d)^T$
- Si ahora queremos calcular:

$$\frac{\partial L}{\partial \mathbf{w}_0} = \frac{\partial \mathbf{z}_0}{\partial \mathbf{w}_0} \cdot \frac{\partial L}{\partial \mathbf{z}_0}$$

Dado que: $\mathbf{z}_0 = \begin{bmatrix} w_1 \cdot x_1 & w_2 \cdot x_2 & w_3 \cdot x_3 + \beta_0 \\ w_1 \cdot x_2 & w_2 \cdot x_3 & w_3 \cdot x_4 + \beta_0 \\ w_1 \cdot x_3 & w_2 \cdot x_4 & w_3 \cdot x_5 + \beta_0 \\ w_1 \cdot x_4 & w_2 \cdot x_5 & w_3 \cdot x_6 + \beta_0 \end{bmatrix}$ $\frac{\partial \mathbf{z}_0}{\partial \mathbf{w}_0} = \begin{bmatrix} \frac{\partial \mathbf{z}_0(1)}{\partial \mathbf{w}_1} & \dots & \frac{\partial \mathbf{z}_0(4)}{\partial \mathbf{w}_1} \\ \frac{\partial \mathbf{z}_0(1)}{\partial \mathbf{w}_2} & \dots & \frac{\partial \mathbf{z}_0(4)}{\partial \mathbf{w}_2} \\ \frac{\partial \mathbf{z}_0(1)}{\partial \mathbf{w}_3} & \dots & \frac{\partial \mathbf{z}_0(4)}{\partial \mathbf{w}_3} \end{bmatrix}$

Backprop con capa convolucional 1D

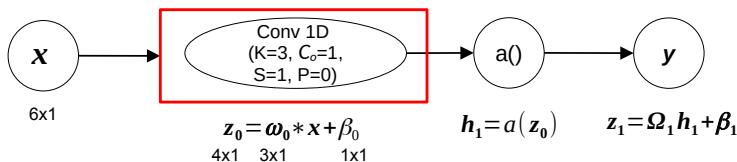


- Supongamos conocida la *derivada de la función de pérdida J*: $\frac{\partial J}{\partial \mathbf{z}_0} = (a \ b \ c \ d)^T$
- Si ahora queremos calcular:

$$\frac{\partial L}{\partial \mathbf{w}_0} = \frac{\partial \mathbf{z}_0}{\partial \mathbf{w}_0} \cdot \frac{\partial L}{\partial \mathbf{z}_0}$$

Dado que: $\mathbf{z}_0 = \begin{bmatrix} w_1 \cdot x_1 & w_2 \cdot x_2 & w_3 \cdot x_3 + \beta_0 \\ w_1 \cdot x_2 & w_2 \cdot x_3 & w_3 \cdot x_4 + \beta_0 \\ w_1 \cdot x_3 & w_2 \cdot x_4 & w_3 \cdot x_5 + \beta_0 \\ w_1 \cdot x_4 & w_2 \cdot x_5 & w_3 \cdot x_6 + \beta_0 \end{bmatrix}$ $\frac{\partial \mathbf{z}_0}{\partial \mathbf{w}_0} = \begin{bmatrix} \frac{\partial \mathbf{z}_0(1)}{\partial \mathbf{w}_1} & \dots & \frac{\partial \mathbf{z}_0(4)}{\partial \mathbf{w}_1} \\ \frac{\partial \mathbf{z}_0(1)}{\partial \mathbf{w}_2} & \dots & \frac{\partial \mathbf{z}_0(4)}{\partial \mathbf{w}_2} \\ \frac{\partial \mathbf{z}_0(1)}{\partial \mathbf{w}_3} & \dots & \frac{\partial \mathbf{z}_0(4)}{\partial \mathbf{w}_3} \end{bmatrix} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ x_2 & x_3 & x_4 & x_5 \\ x_3 & x_4 & x_5 & x_6 \end{bmatrix}$

Backprop con capa convolucional 1D



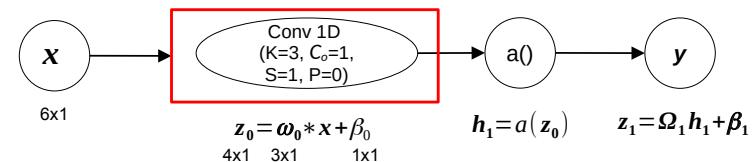
- Supongamos conocida la *derivada de la función de pérdida J*: $\frac{\partial J}{\partial \mathbf{z}_0} = (a \ b \ c \ d)^T$
- Si ahora queremos calcular:

$$\frac{\partial L}{\partial \mathbf{w}_0} = \frac{\partial \mathbf{z}_0}{\partial \mathbf{w}_0} \cdot \frac{\partial L}{\partial \mathbf{z}_0}$$

Convolución del filtro
(d c b a)^T con la entrada x

Dado que: $\mathbf{z}_0 = \begin{bmatrix} w_1 \cdot x_1 & w_2 \cdot x_2 & w_3 \cdot x_3 + \beta_0 \\ w_1 \cdot x_2 & w_2 \cdot x_3 & w_3 \cdot x_4 + \beta_0 \\ w_1 \cdot x_3 & w_2 \cdot x_4 & w_3 \cdot x_5 + \beta_0 \\ w_1 \cdot x_4 & w_2 \cdot x_5 & w_3 \cdot x_6 + \beta_0 \end{bmatrix}$ $\frac{\partial L}{\partial \mathbf{w}_0} = \frac{\partial \mathbf{z}_0}{\partial \mathbf{w}_0} \cdot \frac{\partial L}{\partial \mathbf{z}_0} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ x_2 & x_3 & x_4 & x_5 \\ x_3 & x_4 & x_5 & x_6 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$

Backprop con capa convolucional 1D



- Supongamos conocida la *derivada de la función de pérdida J*: $\frac{\partial J}{\partial \mathbf{z}_0} = (a \ b \ c \ d)^T$
- Si ahora queremos calcular:

$$\frac{\partial L}{\partial \mathbf{w}_0} = \frac{\partial \mathbf{z}_0}{\partial \mathbf{w}_0} \cdot \frac{\partial L}{\partial \mathbf{z}_0} = r \left(\frac{\partial L}{\partial \mathbf{z}_0} \right) * \mathbf{x}$$

Convolución del filtro
(d c b a)^T con la entrada x

Dado que: $\mathbf{z}_0 = \begin{bmatrix} w_1 \cdot x_1 & w_2 \cdot x_2 & w_3 \cdot x_3 + \beta_0 \\ w_1 \cdot x_2 & w_2 \cdot x_3 & w_3 \cdot x_4 + \beta_0 \\ w_1 \cdot x_3 & w_2 \cdot x_4 & w_3 \cdot x_5 + \beta_0 \\ w_1 \cdot x_4 & w_2 \cdot x_5 & w_3 \cdot x_6 + \beta_0 \end{bmatrix}$ $\frac{\partial L}{\partial \mathbf{w}_0} = \frac{\partial \mathbf{z}_0}{\partial \mathbf{w}_0} \cdot \frac{\partial L}{\partial \mathbf{z}_0} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ x_2 & x_3 & x_4 & x_5 \\ x_3 & x_4 & x_5 & x_6 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = r \left(\frac{\partial L}{\partial \mathbf{z}_0} \right) * \mathbf{x}$

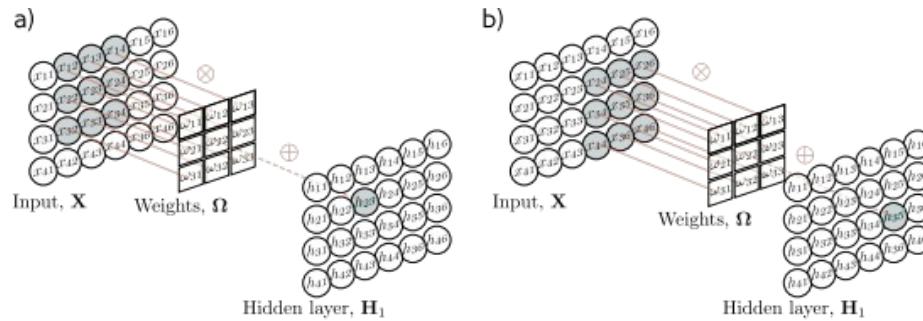
6.2 CNNs en 2D

- Convolución 2D
- Bajar la resolución y subirla, convolución 1x1
- Clasificación de imágenes
- Redes que devuelven una imagen
- Redes Residuales
- Redes U-Net y HourGlass

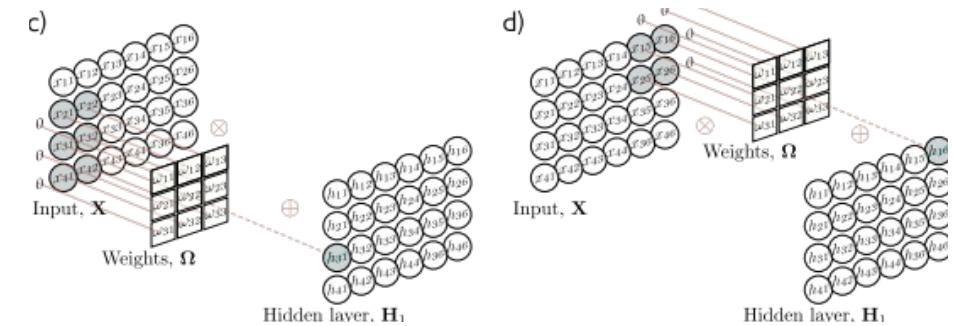
6.2 CNNs en 2D

- Convolución 2D
- Bajar la resolución y subirla, convolución 1x1
- Clasificación de imágenes
- Redes que devuelven una imagen
- Redes Residuales
- Redes U-Net y HourGlass

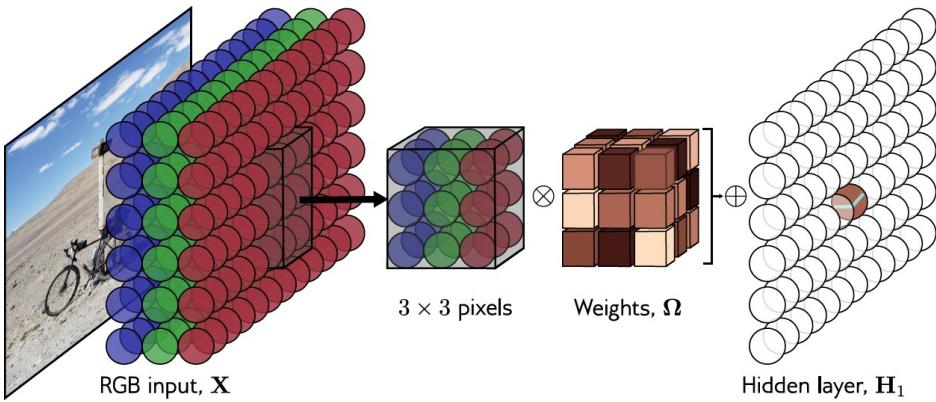
Convolución 2D



Convolución 2D



Canales en la Convolución 2D



Tamaño del kernel, paso (stride) y dilatación funcionan como se espera

Convolución 2D en Pytorch

Docs > torch.nn > Conv2d
Conv2d

CLASS `torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True, padding_mode='zeros', device=None, dtype=None)` [SOURCE]

Applies a 2D convolution over an input signal composed of several input planes.

In the simplest case, the output value of the layer with input size (N, C_{in}, H, W) and output $(N, C_{\text{out}}, H_{\text{out}}, W_{\text{out}})$ can be precisely described as:

$$\text{out}(N_i, C_{\text{out},j}) = \text{bias}(C_{\text{out},j}) + \sum_{k=0}^{C_{\text{in}}-1} \text{weight}(C_{\text{out},j}, k) * \text{input}(N_i, k)$$

where $*$ is the valid 2D cross-correlation operator, N is a batch size, C denotes a number of channels, H is a height of input planes in pixels, and W is width in pixels.

This module supports `TensorFloat32`.

On certain ROCm devices, when using float16 inputs this module will use different precision for backward.

- `stride` controls the stride for the cross-correlation, a single number or a tuple.
- `padding` controls the amount of padding applied to the input. It can be either a string `{'valid', 'same'}` or an int / a tuple of ints giving the amount of implicit padding applied on both sides.
- `dilation` controls the spacing between the kernel points; also known as the `à trous` algorithm. It is harder to describe, but this [link](#) has a nice visualization of what `dilation` does.
- `groups` controls the connections between inputs and outputs. `in_channels` and `out_channels` must both be divisible by `groups`. For example,
 - At `groups=1`, all inputs are convolved to all outputs.
 - At `groups=2`, the operation becomes equivalent to having two conv layers side by side, each seeing half the input channels and producing half the output channels, and both subsequently concatenated.
 - At `groups= in_channels`, each input channel is convolved with its own set of filters (of size $\frac{\text{out_channels}}{\text{in_channels}}$).

¿Cuántos parámetros?

- Con C_i canales de entrada y un kernel de tamaño $K \times K$:

Capa completa:
 $C_i \cdot K^2 + 1$ parámetros

$$\omega \in \mathbb{R}^{C_i \times K \times K} \quad \beta \in \mathbb{R}$$

- Si hay C_i canales de entrada y C_o canales de salida:

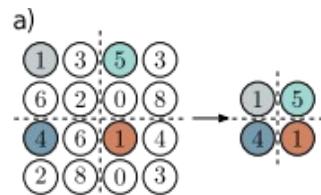
Capa completa:
 $C_i \cdot C_o \cdot K^2 + C_o$ parámetros

$$\omega \in \mathbb{R}^{C_i \times C_o \times K \times K} \quad \beta \in \mathbb{R}^{C_o}$$

6.2 CNNs en 2D

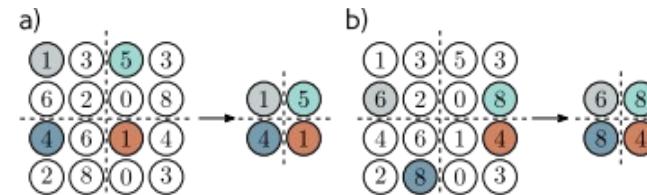
- Convolución 2D
- Bajar la resolución y subirla, convolución 1x1
- Clasificación de imágenes
- Redes que devuelven una imagen
- Redes Residuales
- Redes U-Net y HourGlass

Bajar la resolución (downsampling)



Quedarse uno de cada dos píxeles (= stride 2)

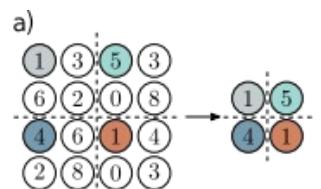
Bajar la resolución (downsampling)



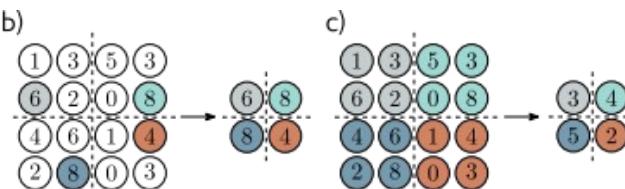
Quedarse uno de cada dos píxeles (= stride 2)

Max pooling
(Invarianza parcial a traslación)

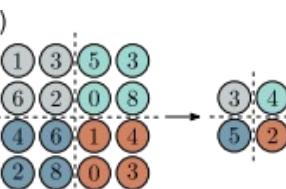
Bajar la resolución (downsampling)



Quedarse uno de cada dos píxeles (= stride 2)

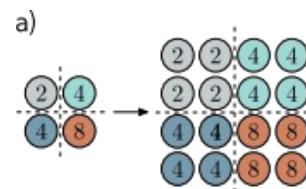


Max pooling
(Invarianza parcial a traslación)



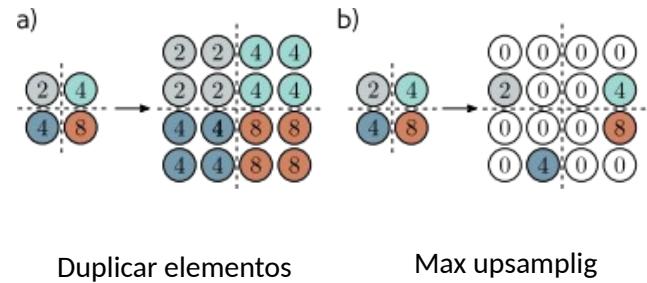
Mean (o average)
pooling

Subir la resolución (upsampling)

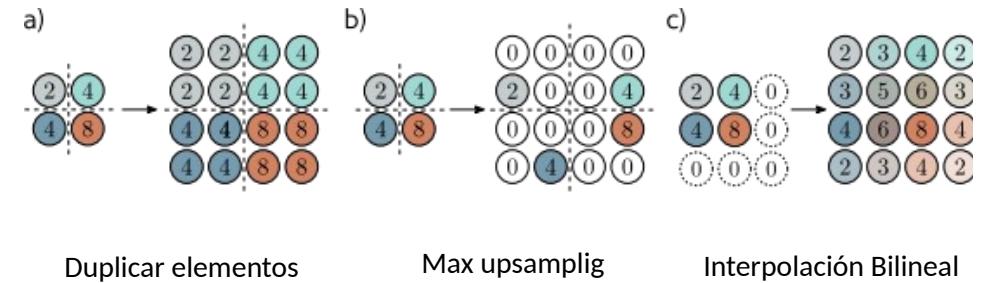


Duplicar elementos

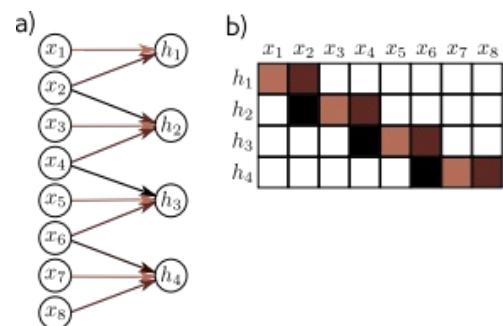
Subir la resolución (upsampling)



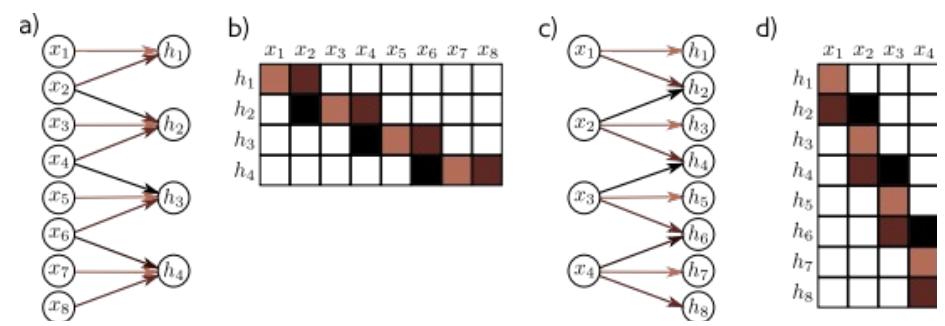
Subir la resolución (upsampling)



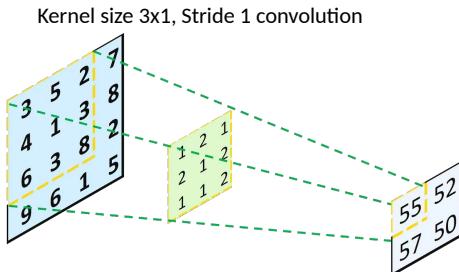
Convoluciones “traspuestas”



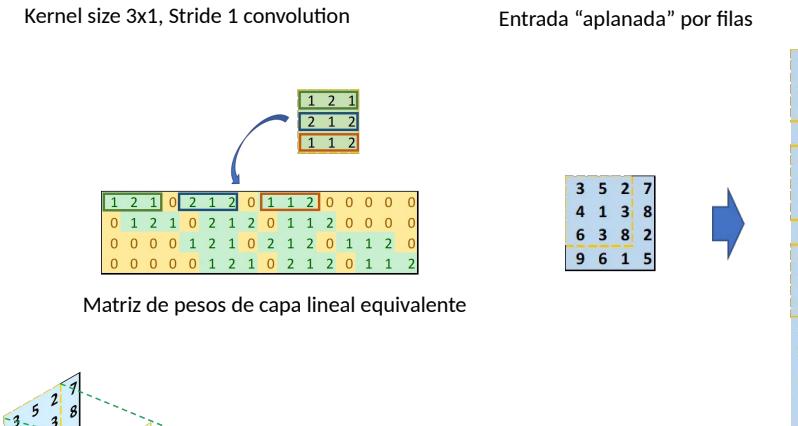
Convoluciones “traspuestas”



Convoluciones 2D como capa lineal

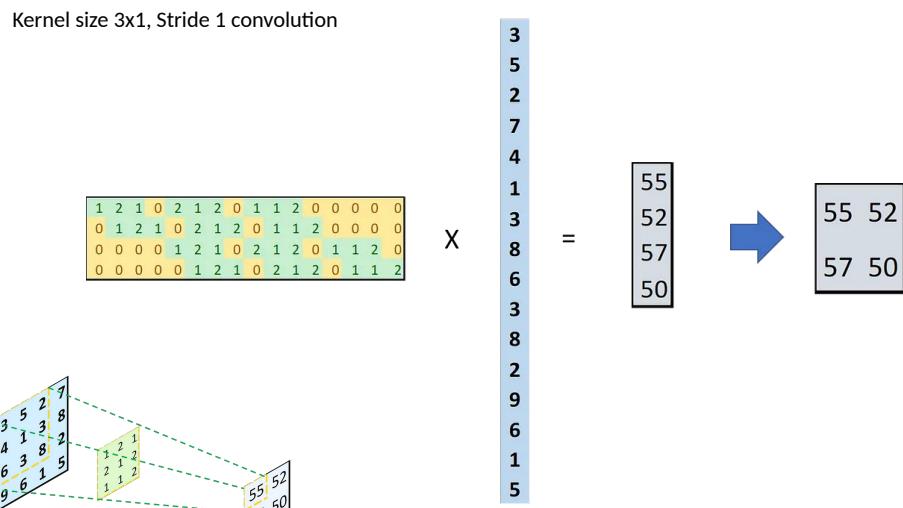


Convoluciones 2D como capa lineal

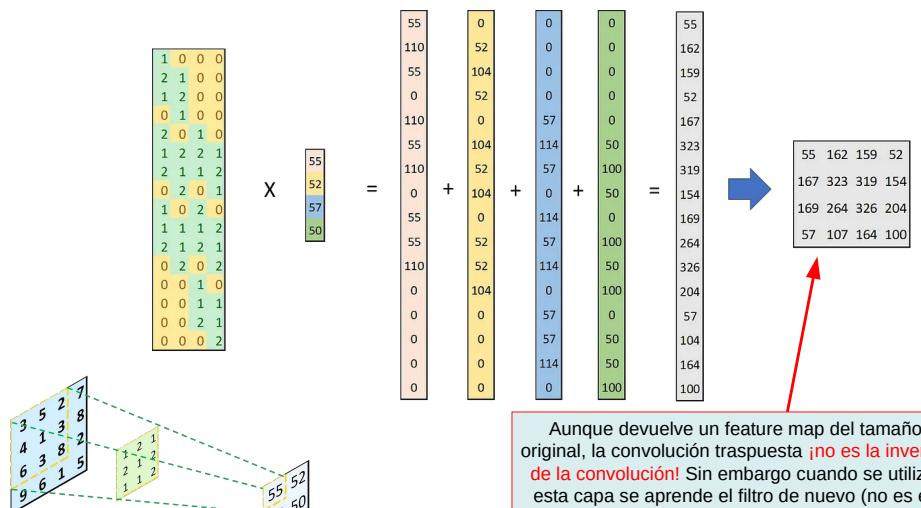


<https://towardsdatascience.com/understand-transposed-convolutions-and-build-your-own-transposed-convolution-layer-from-scratch-4f5d97b2967>

Convoluciones 2D como capa lineal

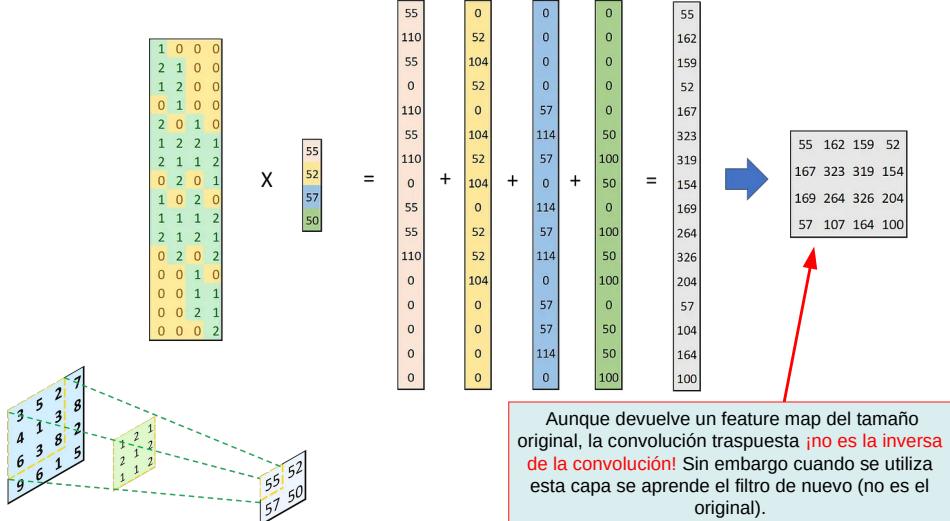


Convoluciones “traspuestas”



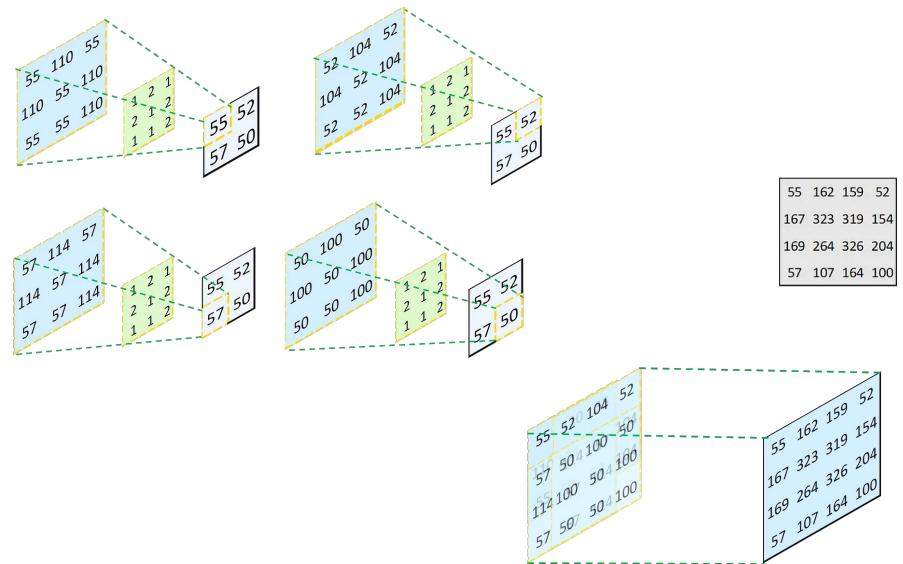
<https://towardsdatascience.com/understand-transposed-convolutions-and-build-your-own-transposed-convolution-layer-from-scratch-4f5d97b2967>

Convoluciones “traspuestas”



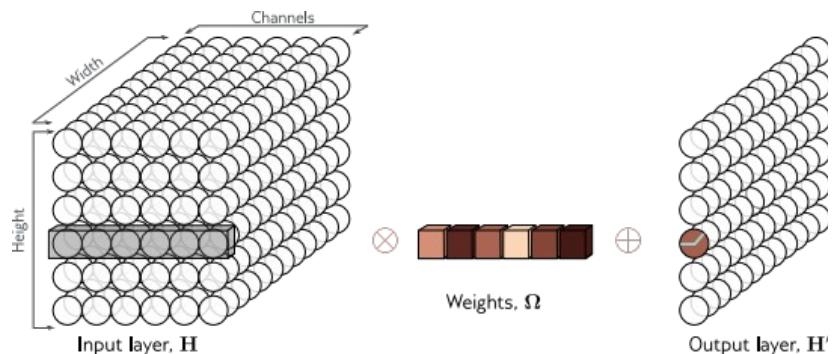
<https://towardsdatascience.com/understand-transposed-convolutions-and-build-your-own-transposed-convolution-layer-from-scratch-4f5d97b2967>

Convoluciones “traspuestas”



<https://towardsdatascience.com/understand-transposed-convolutions-and-build-your-own-transposed-convolution-layer-from-scratch-4f5d97b2967>

Convolución con kernel 1x1

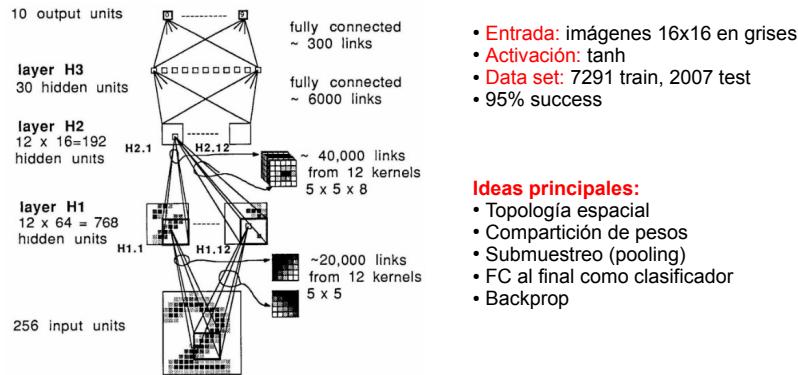


- Mezcla canales (combinación lineal de los mismos)
- Puede cambiar el número de canales

6.2 CNNs en 2D

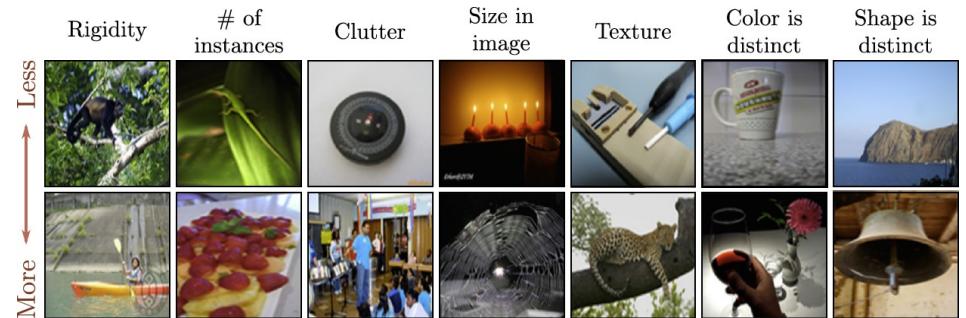
- Convolución 2D
- Bajar la resolución y subirla, convolución 1x1
- **Clasificación de imágenes**
- Redes que devuelven una imagen
- Redes Residuales
- Redes U-Net y HourGlass

LeNet: Primera CNN entrenada con backprop



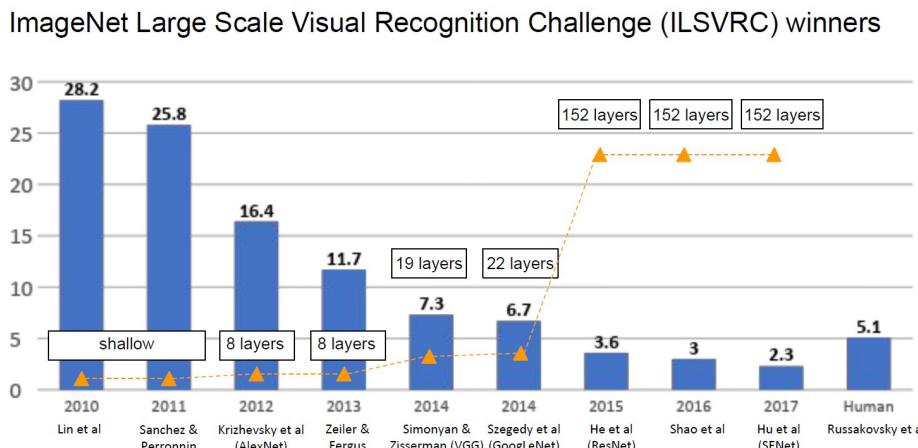
Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. *Backpropagation Applied to Handwritten Zip Code Recognition*. *Neural Computation*, Vol. 1, No. 4 , Pages: 541-551, 1989.

ImageNet data set

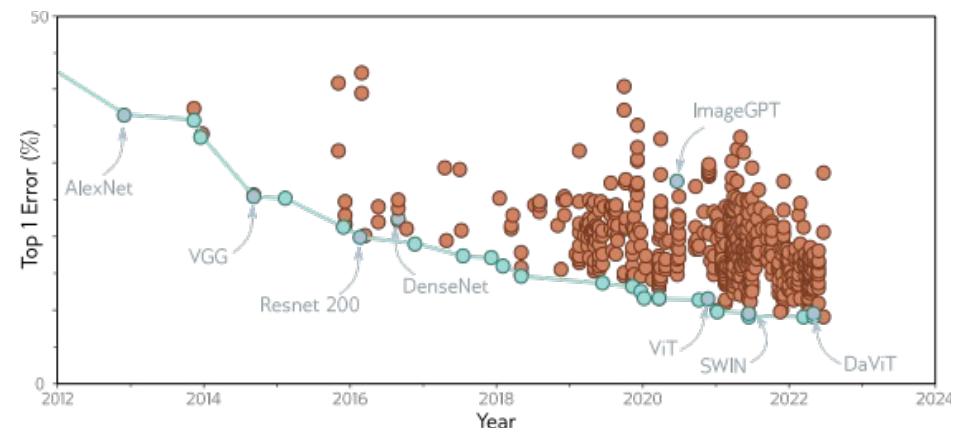


- Imágenes 224x224 píxeles
- 1281167 imágenes de train, 50K validación, 100K test.
- 1000 clases

ImageNet data set (top-5 error vs layers)

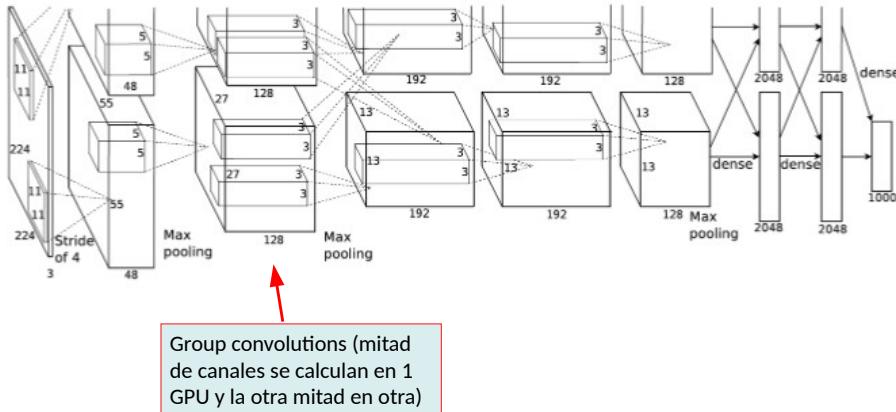


ImageNet data set



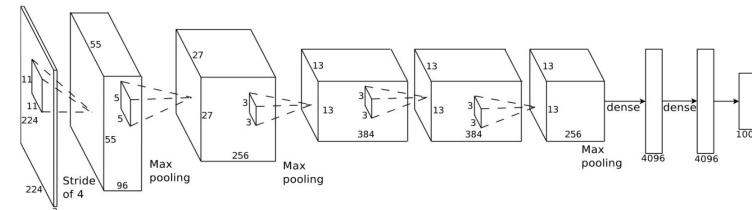
AlexNet (2012): grouped convolutions

Entrenado en 2 NVIDIA GTX 580 GPUs en 1 semana (velocidad ~x30)



A. Krizhevsky, I. Sutskever, G. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems* 25, pp. 1097–1105, 2012.

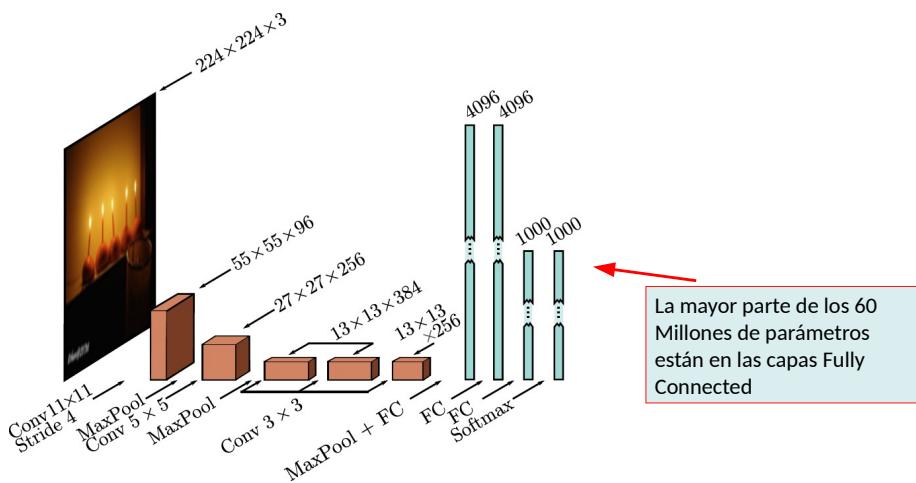
AlexNet (2012): red equivalente



Ideas principales:

- 650K neuronas, 60M parámetros, 630M conexiones.
- Entrenamiento más rápido por la ReLU (1er modelo profundo!)
- Data augmentation en train y test (5 imágenes generadas, resultados promediados)
- SGD, lr0=0.01, reducido manualmente, momentum=0.9, batch sz=128
- Regularización: L2 weight decay, Dropout (p=0.5) en las dos últimas capas FC
- Escrito en Python/C++/CUDA
- Ensemble de 7 CNNs: 16.4% top-5 error rate, 38.1% top-1 error rate (en validación)

AlexNet (2012)

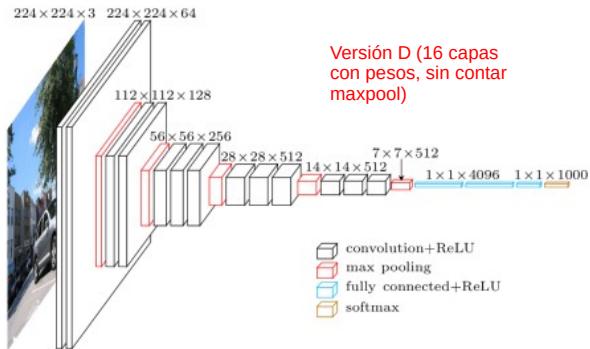


AlexNet (2012)

	FULL CONNECT
4M	FULL CONNECT
16M	FULL 4096/ReLU
37M	FULL 4096/ReLU
	MAX POOLING
442K	CONV 3x3/ReLU 256fm
1.3M	CONV 3x3ReLU 384fm
884K	CONV 3x3ReLU 384fm
	MAX POOLING 2x2sub
307K	LOCAL CONTRAST NORM
	CONV 11x11/ReLU 256fm
	MAX POOL 2x2sub
	LOCAL CONTRAST NORM
35K	CONV 11x11/ReLU 96fm



Visual Geometry Group (VGG) Net (2015)

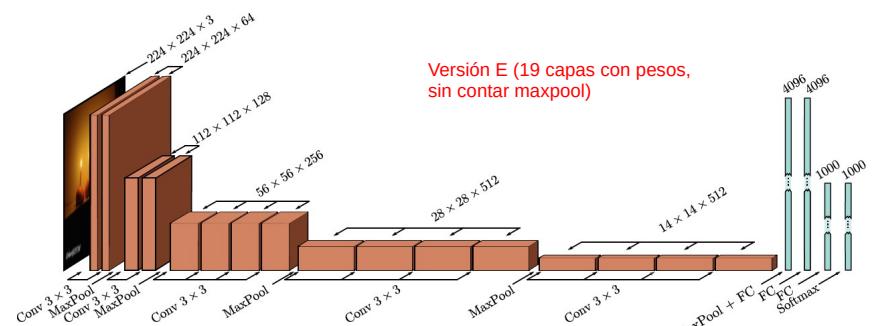


Ideas principales:

- Diseño sencillo y modular:
 - Convolución con ReLU + pooling.
 - Únicamente filtros 3x3 con stride 1 y 2x2 maxpool con stride 2
 - Se divide por 2 la resolución y se multiplica por 2 el número de canales.
- Versión de 19 capas, 144M de parámetros

K. Simonyan, A. Zisserman. Very Deep ConvNets for Large-Scale Image Recognition. ICLR 2015

Visual Geometry Group (VGG) Net (2015)



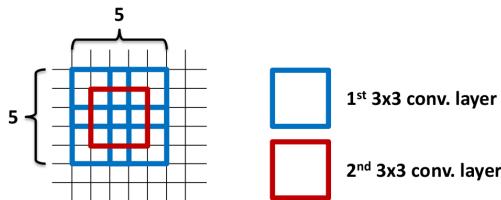
Ideas principales:

- Diseño sencillo y modular:
 - Convolución con ReLU + pooling,
 - Únicamente filtros 3x3 con stride 1 y 2x2 maxpool con stride 2
 - Se divide por 2 la resolución y se multiplica por 2 el número de canales.
- Versión de 19 capas, 144M de parámetros
- Ensemble 7 CNNs: 7.5% top-5 error rate, 24.7% top-1 error rate (en validación)

K. Simonyan, A. Zisserman. Very Deep ConvNets for Large-Scale Image Recognition. ICLR 2015

Visual Geometry Group (VGG) Net (2015)

- ¿Por qué convoluciones 3x3?
 - Si las apilas tienen un campo receptivo amplio
 - Dos capas conv. 3x3 → campo receptivo 5x5
 - Tres capas conv. 3x3 → campo receptivo 7x7
 - Más no linealidad
 - Menos parámetros que aprender

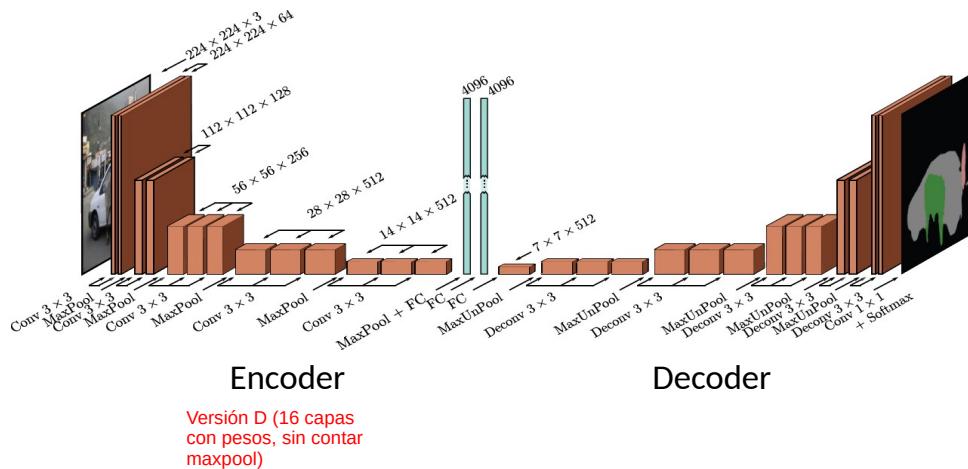


K. Simonyan, A. Zisserman. Very Deep ConvNets for Large-Scale Image Recognition. ICLR 2015

6.2 CNNs en 2D

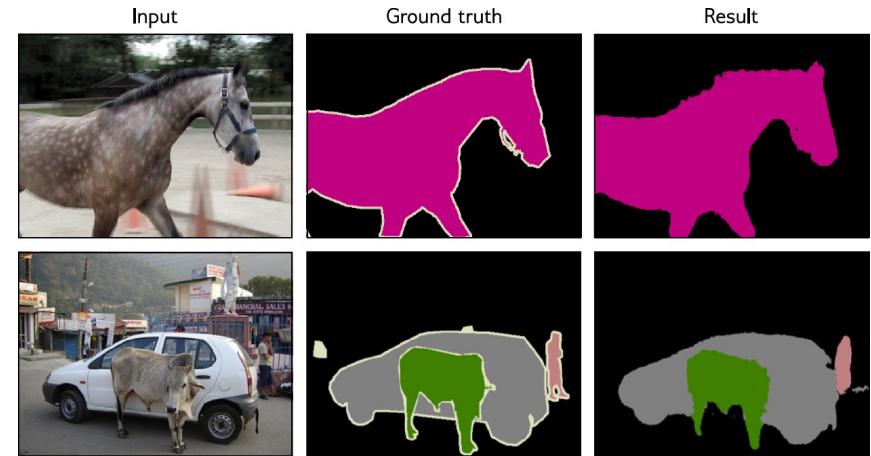
- Convolución 2D
- Bajar la resolución y subirla, convolución 1x1
- Clasificación de imágenes
- **Redes que devuelven una imagen**
- Redes Residuales
- Redes U-Net y HourGlass

Segmentación semántica (2015)



Noh, H., Hong, S., & Han, B. (2015). Learning deconvolution network for semantic segmentation. IEEE International Conference on Computer Vision, 1520–1528.

Segmentación semántica (2015)



Noh, H., Hong, S., & Han, B. (2015). Learning deconvolution network for semantic segmentation. IEEE International Conference on Computer Vision, 1520–1528.

6.2 CNNs en 2D

- Convolución 2D
- Bajar la resolución y subirla, convolución 1x1
- Clasificación de imágenes
- Redes que devuelven una imagen
- **Redes Residuales**
- Redes U-Net y HourGlass

Redes más profundas ...

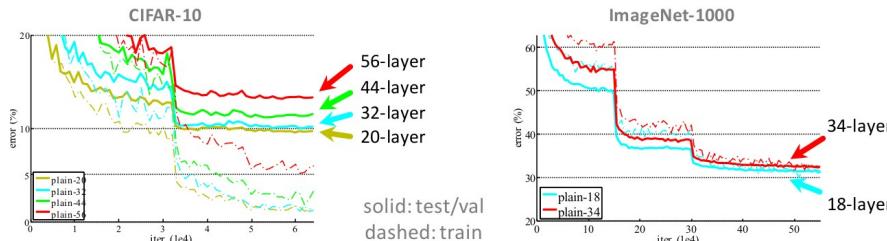


K. He, X. Zhang, S. Ren, Jian Sun. Deep Residual Learning for Image Recognition. CVPR, 2016.

Fuente: K. He

Redes más profundas ...

¿Si añadimos más capas mejora el rendimiento?

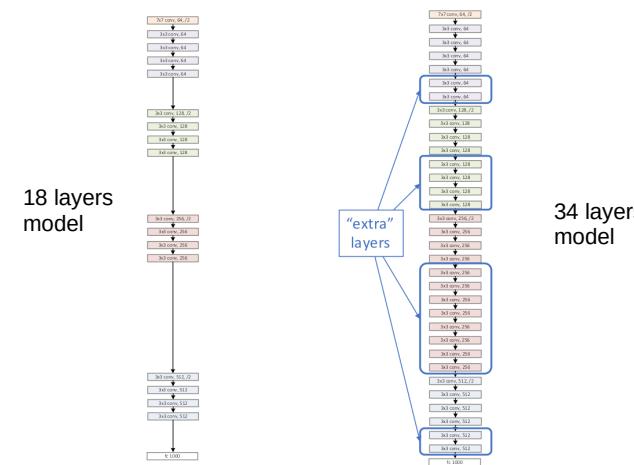


Problemas de optimización: ¡El optimizador no puede con tantas capas!

K. He, X. Zhang, S. Ren, Jian Sun. Deep Residual Learning for Image Recognition. CVPR, 2016.

Fuente: K. He

Redes más profundas ...

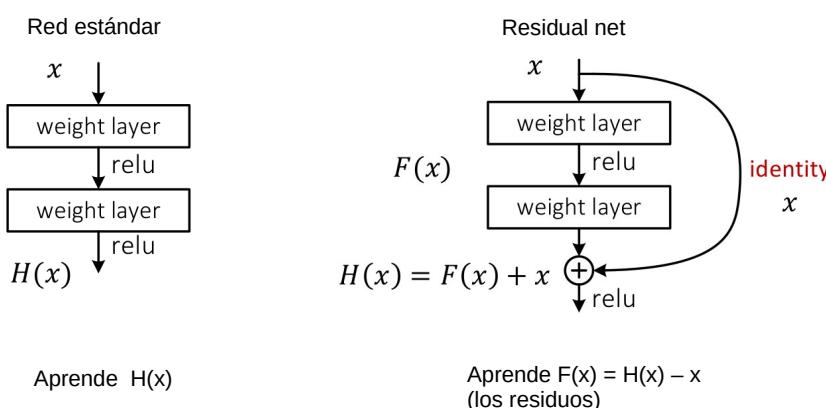


Mismo rendimiento si las capas extra son la identidad (se eliminan).

K. He, X. Zhang, S. Ren, Jian Sun. Deep Residual Learning for Image Recognition. CVPR, 2016.

Fuente: K. He

Residual networks (ResNets)

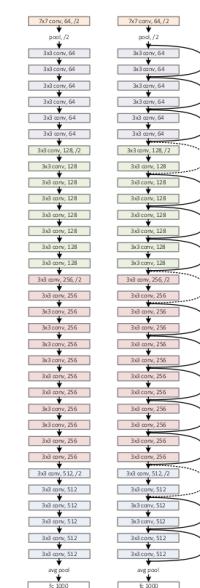


K. He, X. Zhang, S. Ren, Jian Sun. Deep Residual Learning for Image Recognition. CVPR, 2016.

Fuente: K. He

Residual Networks (ResNet)

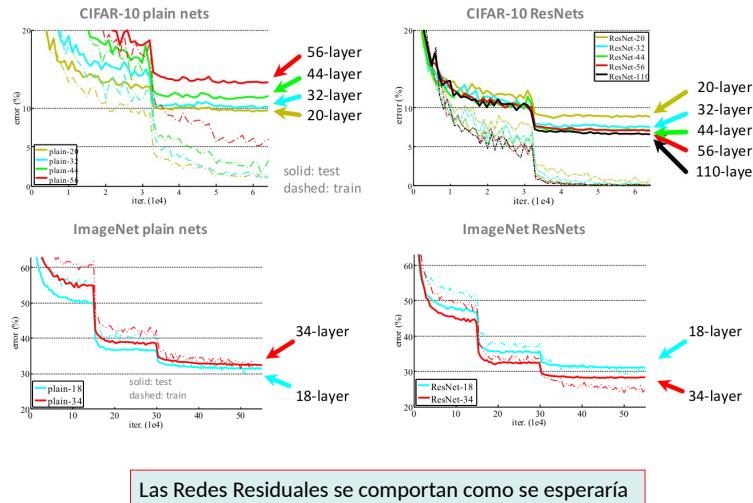
- Estilo VGG:
 - Únicamente conv. 3x3, stride 1
 - 2x2 max pooling, stride 2
 - Dobra n.º de filtros cada capa de pooling
 - 1 única capa FC
- Batch normalization
- No dropout
- Data Augmentation



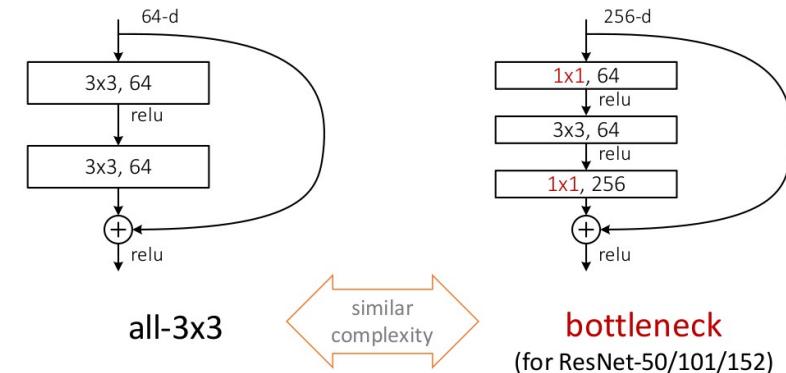
K. He, X. Zhang, S. Ren, Jian Sun. Deep Residual Learning for Image Recognition. CVPR, 2016.

Fuente: K. He

Resultados ResNets



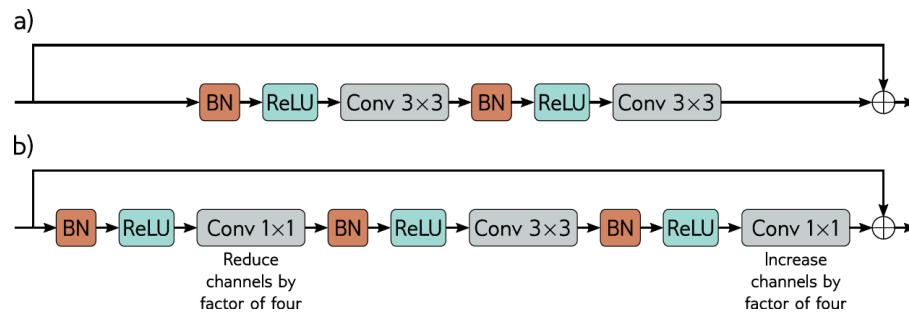
Refinamiento del bloque básico



K. He, X. Zhang, S. Ren, Jian Sun. Deep Residual Learning for Image Recognition. CVPR, 2016.

Fuente: K. He

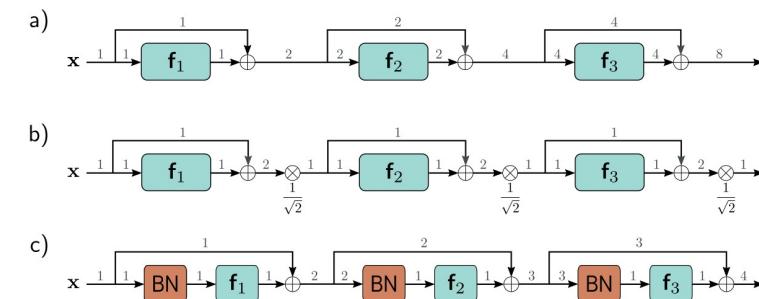
Refinamiento del bloque básico (detalle)



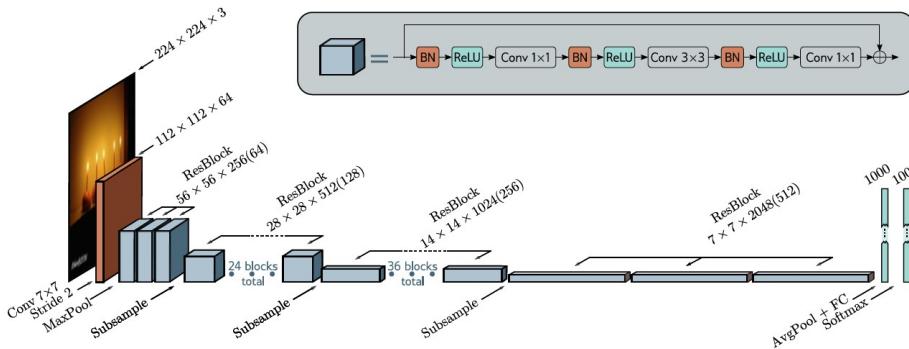
K. He, X. Zhang, S. Ren, Jian Sun. Deep Residual Learning for Image Recognition. CVPR, 2016.

¿Por qué batch normalization?

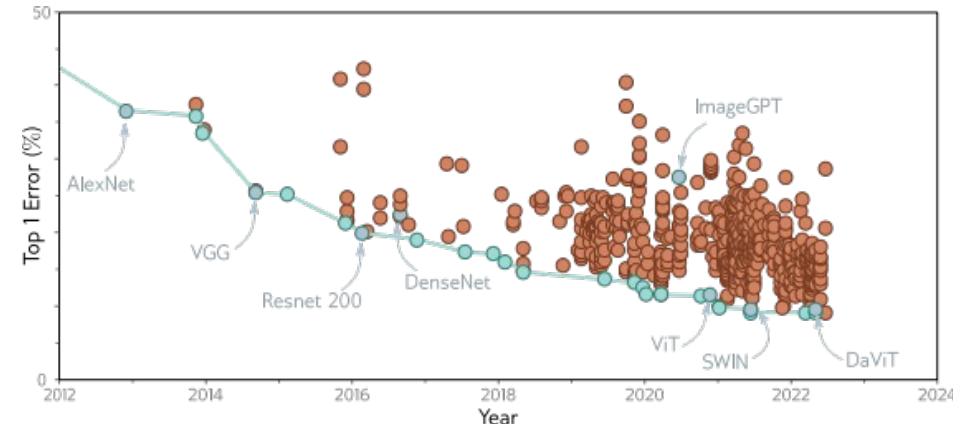
- Con la inicialización de He, la varianza es constante en las capas lineales.
- Pero con conexiones residuales ¡Se multiplica x2 la varianza en cada nodo suma!
- Es necesario normalizar para que la varianza de las activaciones no crezca exponencialmente y lo haga linealmente.



ResNet 200 (2016)



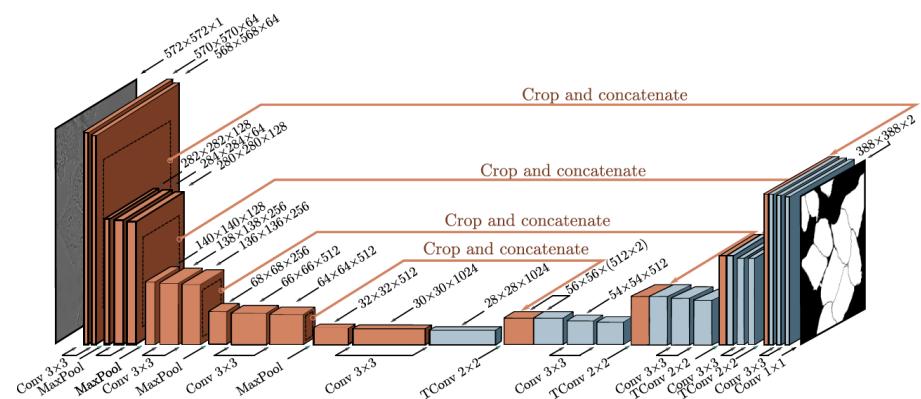
ImageNet data set



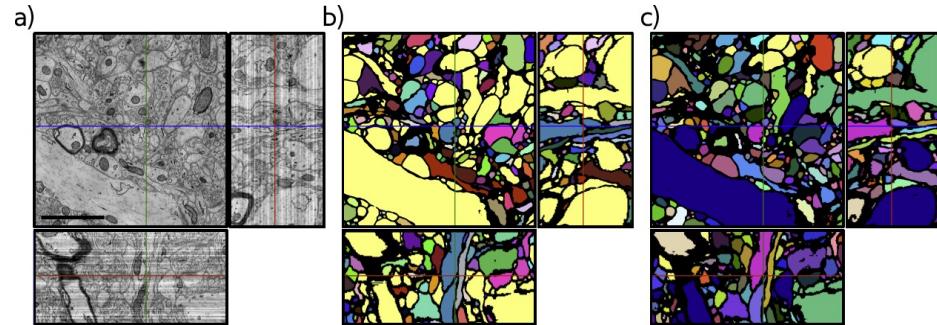
6.2 CNNs en 2D

- Convolución 2D
- Bajar la resolución y subirla, convolución 1x1
- Clasificación de imágenes
- Redes que devuelven una imagen
- Redes Residuales
- **Redes U-Net y HourGlass**

UNet (2016)



Resultados UNet



Stacked Hourglass Networks (2016)

