Redes Neuronales Convolucionales (CNN)

Orlando Ramos Flores

Contenido

Introducción

- Las CNN o ConvNets son diseñadas para procesar datos que vienen en forma de matrices múltiples, por ejemplo:
  - Una imagen en color compuesta por tres matrices 2D que contienen intensidades de píxeles en los tres canales (RGB) de color.
- Muchas modalidades de datos se encuentran en forma de matrices múltiples:
  - 1D para señales y secuencias, incluido el idioma
  - 2D para imágenes o espectrogramas de audio
  - 3D para video o imágenes volumétricas

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. nature, 521(7553), 436-444.

Introducción

- Hay cuatro ideas clave detrás de las CNN que aprovechan las propiedades de las señales naturales: conexiones locales, pesos compartidos, agrupamiento (pooling) y el uso de muchas capas o niveles.
- La arquitectura CNN está estructurada como una serie de etapas o estados.
- Las primeras etapas se componen de dos tipos de capas:
  - Capas convolucionales y
  - Capas de agrupación (pooling).

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. nature, 521(7553), 436-444.

Capa de convolución

- Su trabajo es detectar características locales.
- Las unidades del nivel de convolución están organizadas en mapas de características, en los cuales cada unidad se conecta a “parches” locales en los mapas de características del nivel anterior, a través de pesos, llamados “bancos de filtros”.
- El resultado de esta suma ponderada local, se pasa a través de una no linealidad (función no lineal) como ReLU.
- Todas las unidades en un mapa de características comparten el mismo banco de filtros.
- Diferentes mapas de características en una capa usan diferentes bancos de filtros.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. nature, 521(7553), 436-444.

Aspectos de la arquitectura CNN

- La razón de esta arquitectura contempla dos aspectos:
  - En primer lugar, en los datos de una matriz, como las imágenes, los grupos locales de valores suelen estar altamente correlacionados, formando patrones locales distintivos que se detectan fácilmente.
  - En segundo lugar, las estadísticas locales de imágenes y otras señales no varían con la ubicación. En otras palabras, si un patrón puede aparecer en una parte de la imagen, podría aparecer en cualquier lugar, de ahí la idea de unidades en diferentes ubicaciones que comparten los mismos pesos y detectan el mismo patrón en diferentes partes de la matriz. Matemáticamente, la operación de filtrado realizada por un mapa de características es una convolución discreta, de ahí el nombre.
-

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. nature, 521(7553), 436-444.

Capa de agrupamiento

- La función de la capa de agrupación es fusionar características semánticamente similares en una sola característica.
- Debido a que las posiciones relativas de las características que forman un patrón pueden variar un poco, la detección confiable del patrón se puede realizar mediante una granularidad gruesa de la posición de cada característica.
- Una unidad de pooling típica calcula el máximo de un parche local de unidades en un mapa de características (o en unos pocos mapas de características).
- Las unidades de pooling vecinas toman entradas de parches que se desplazan en más de una fila o columna, lo que reduce la dimensión de la representación y crea una invariancia a pequeños cambios y distorsiones.
- Se apilan dos o tres etapas de convolución, no linealidad y pooling, seguidas de más capas convolucionales o capas totalmente conectadas.
- Todos los pesos en todos los bancos de filtros se entrenan con la retropropagación de gradientes a través de la CNN, es tan simple como a través de una red profunda regular.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. nature, 521(7553), 436-444.

Capas convolucionales

Convolución en 1D

Para la **convolución** entre dos funciones, se tiene $\mathbf{f, g: \mathbb{R}^D \rightarrow \mathbb{R}}$, es definida como:

$$s(t) = \int x(a)w(t-a)da$$

En la terminología de red convolucional, el primer argumento (en este ejemplo, la función $\mathbf{x}$) de la convolución se suele denominar **entrada** y el segundo argumento (en este ejemplo, la función $\mathbf{w}$) como **kernel**. La salida a veces se denomina **mapa de características**. Esta operación es llamada convolución. Típicamente es denotada con un (*): $s(t) = (x * w)(t)$

Referencias:
Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press.
Murphy, K. P. (2022). Probabilistic machine learning: an introduction. MIT press.
Ketkar, N., & Santana, E. (2017). Deep learning with Python (Vol. 1). Berkeley: Apress.

Convolución 1D

Por lo general, cuando trabajamos con datos en una computadora, el tiempo se discretiza y nuestro sensor proporcionará datos a intervalos regulares. Entonces:

$$s(t) = (x * w)(t) = \sum_a x(a) \cdot w(t - a)$$

Una forma equivalente de esta operación dada la conmutatividad de la operación de convolución es la siguiente:

$$(x * w)(t) = \sum_a x(t - a) \cdot w(a)$$

Referencias:
Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press.
Murphy, K. P. (2022). Probabilistic machine learning: an introduction. MIT press.
Ketkar, N., & Santana, E. (2017). Deep learning with Python (Vol. 1). Berkeley: Apress.

Convolución 1D

En la literatura de aprendizaje profundo, el término "convolución" generalmente se usa para significar correlación cruzada; seguiremos esta convención:

$$(x * w)(t) = \sum_a x(t + a) \cdot w(a)$$

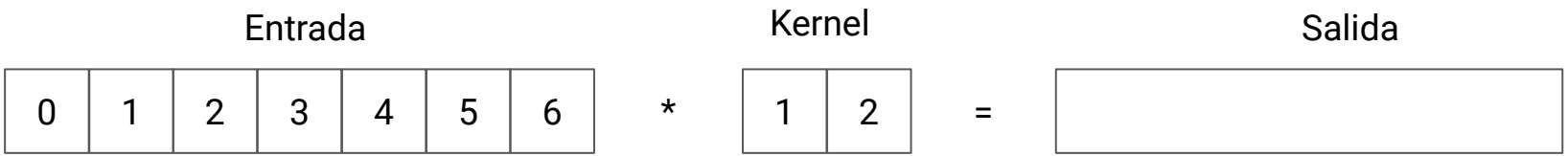
En general la operación de convolución se puede ver de la siguiente forma:

$$(w * x)(i) = \sum a{=}0 w_a \cdot x_{(i+a)}$$

Referencias:
Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press.
Murphy, K. P. (2022). Probabilistic machine learning: an introduction. MIT press.
Ketkar, N., & Santana, E. (2017). Deep learning with Python (Vol. 1). Berkeley: Apress.

Convolución 1D: Ejemplo

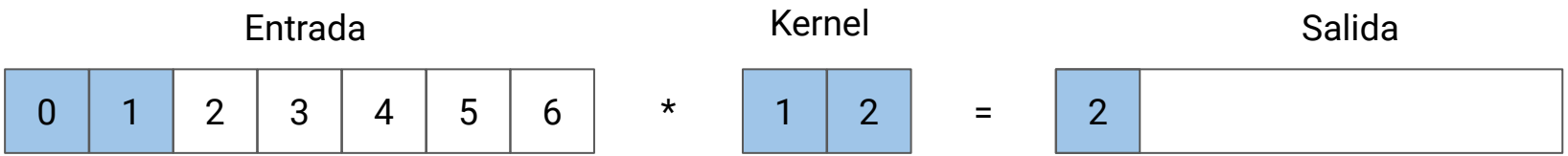
x = [0,1,2,3,4,5,6] w = [1,2]



Referencias: Murphy, K. P. (2022). Probabilistic machine learning: an introduction. MIT press.

Convolución 1D: Ejemplo

x = [0,1,2,3,4,5,6] w = [1,2]

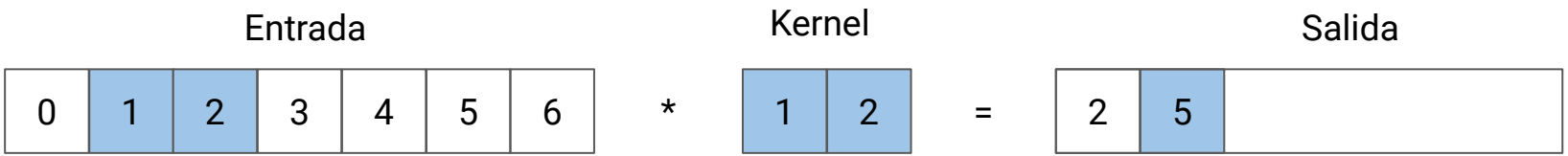


(0 * 1) + (1 * 2) = 2

Referencias: Murphy, K. P. (2022). Probabilistic machine learning: an introduction. MIT press.

Convolución 1D: Ejemplo

x = [0,1,2,3,4,5,6] w = [1,2]

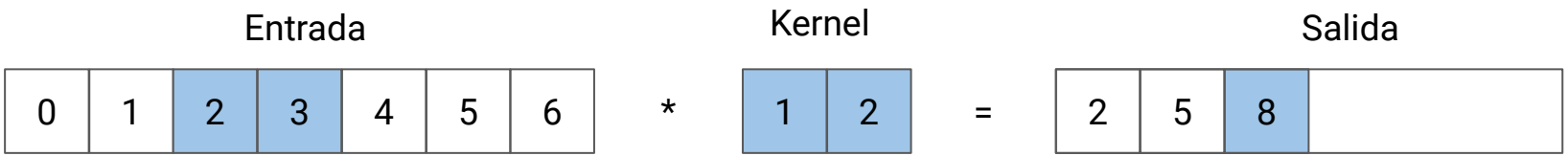


(1 * 1) + (2 * 2) = 5

Referencias: Murphy, K. P. (2022). Probabilistic machine learning: an introduction. MIT press.

Convolución 1D: Ejemplo

x = [0,1,2,3,4,5,6] w = [1,2]

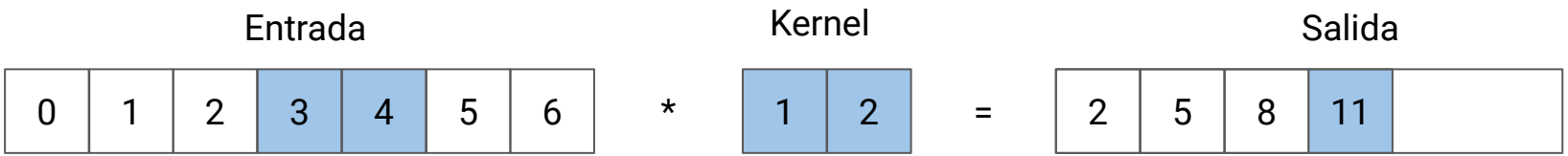


(2 * 1) + (3 * 2) = 8

Referencias: Murphy, K. P. (2022). Probabilistic machine learning: an introduction. MIT press.

Convolución 1D: Ejemplo

x = [0,1,2,3,4,5,6] w = [1,2]

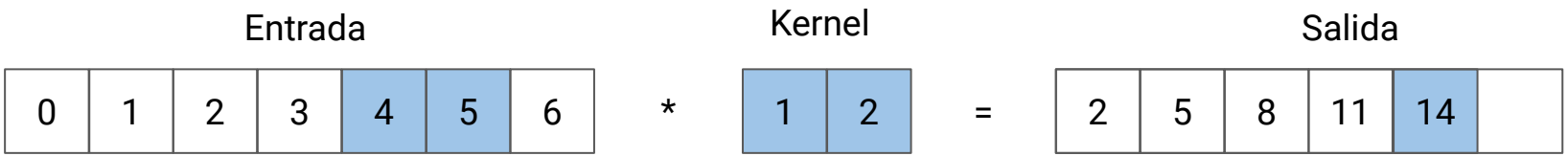


(3 * 1) + (4 * 2) = 11

Referencias: Murphy, K. P. (2022). Probabilistic machine learning: an introduction. MIT press.

Convolución 1D: Ejemplo

x = [0,1,2,3,4,5,6] w = [1,2]

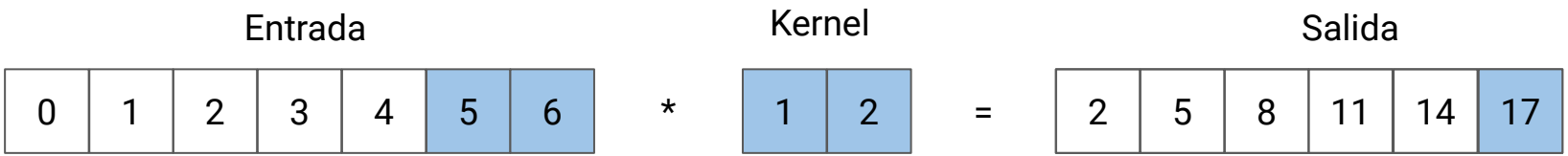


(4 * 1) + (5 * 2) = 14

Referencias: Murphy, K. P. (2022). Probabilistic machine learning: an introduction. MIT press.

Convolución 1D: Ejemplo

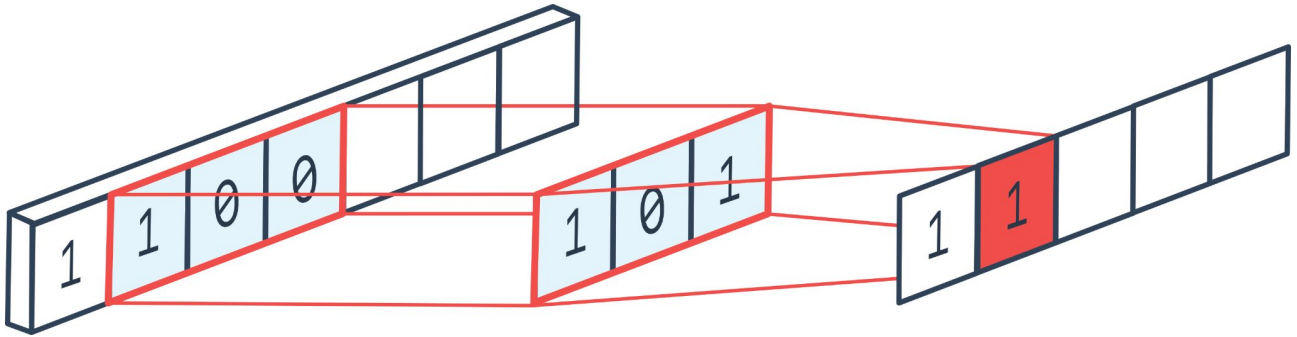
x = [0,1,2,3,4,5,6] w = [1,2]



(5 * 1) + (6 * 2) = 17

Referencias: Murphy, K. P. (2022). Probabilistic machine learning: an introduction. MIT press.

Convolución 1D



Muestra una convolución de 1D con 1 kernel de tamaño 3 y 1 paso (stride).

Para una operación convolucional o de agrupación, el **stride** denota el número de píxeles por los que se mueve la ventana después de cada operación.

Source:
https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/blocks/1d-convolution

Convolución 1D:
Calcular matriz de salida de la operación (*)

$$o = \left(\frac{i - k + 2 \cdot p}{s} \right) + 1$$

$\boldsymbol{o}$ = es el tamaño de la matriz resultante de la operación convolución
$\boldsymbol{i}$ = el tamaño de la entrada
$\boldsymbol{p}$ = se refiere al relleno (padding) default=0
$\boldsymbol{k}$ = el tamaño del kernel
$\boldsymbol{s}$ = (stride) número de pixeles por los que se mueve la ventana default=1

Referencias: Murphy, K. P. (2022). Probabilistic machine learning: an introduction. MIT press.

Convolución 1D:
Del ejemplo, ¿cuál es el tamaño del vector de salida?

$$o = \left(\frac{i - k + 2 \cdot p}{s} \right) + 1 \quad = 6$$

Entrada

0	1	2	3	4	5	6

*

Kernel

1	2

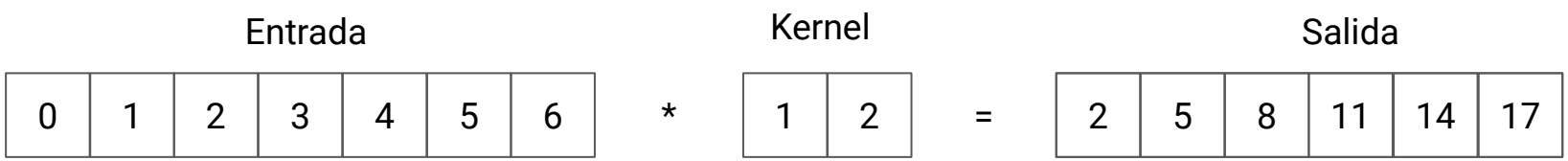
$o = ?$
$i = 7$
$p = 0$
$k = 2$
$s = 1$

Referencias: Murphy, K. P. (2022). Probabilistic machine learning: an introduction. MIT press.

Convolución 1D: Ejercicio

- Implementar la convolución en 1D

$$x = [0,1,2,3,4,5,6] \qquad k = [1,2]$$



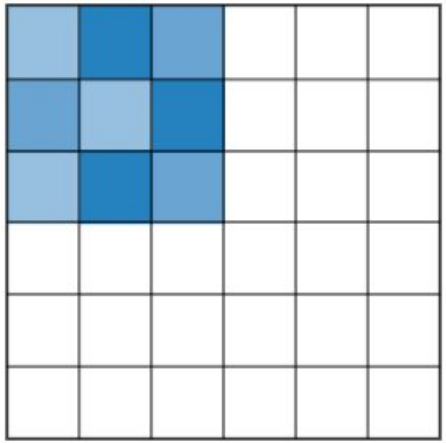
Padding

Padding

- El padding es el proceso de agregar uno o más píxeles de ceros alrededor de los límites de una imagen para aumentar su tamaño efectivo.
- Las capas convolucionales devuelven por defecto una imagen más pequeña que la entrada.
- Si se unen muchas capas convolucionales, la imagen de salida se reduce progresivamente en tamaño hasta que, finalmente, puede volverse inutilizable.
- Al rellenar una imagen (es decir, "aumentar" su tamaño) antes de una capa convolucional, se puede mitigar este efecto.

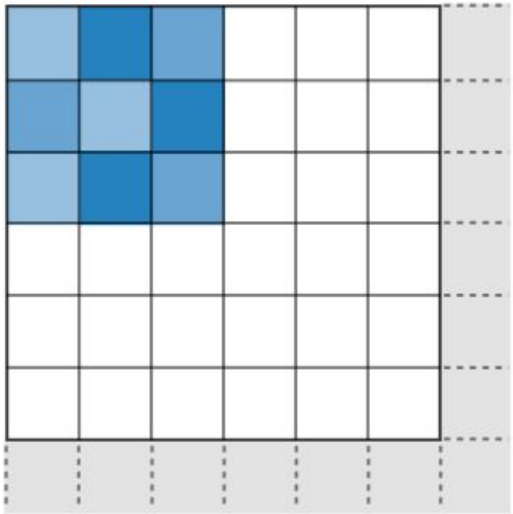
Source: https://peltarion.com/knowledge-center/documentation/glossary

Padding: valid



- Sin relleno
- Omite la última convolución si las dimensiones no coinciden

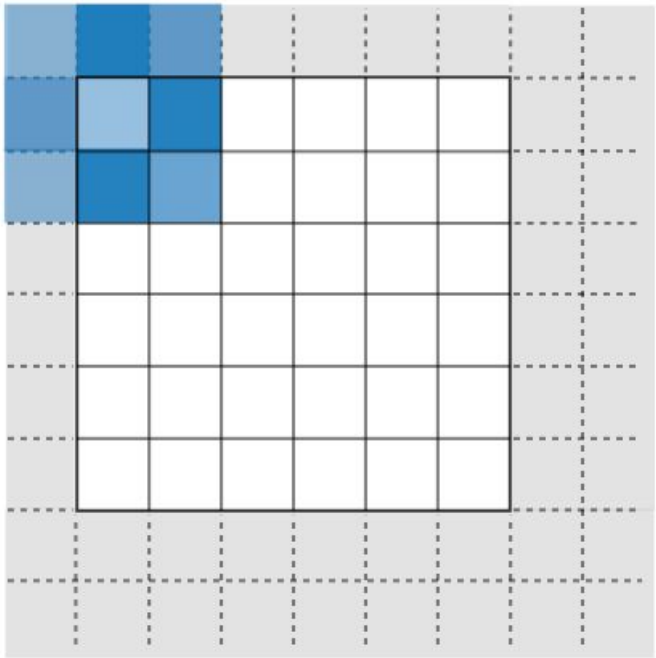
Padding: same



- Padding tal que el tamaño del mapa de características tenga el tamaño $\left\lceil \frac{I}{S} \right\rceil$ donde I es la longitud del tamaño de la entrada y S stride (salto)
- El tamaño de salida es matemáticamente conveniente
- También llamado padding 'medio'

CS 230 - Convolutional Neural Networks Cheatsheet, https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks#

Padding: full



- Relleno máximo, tal que las convoluciones finales se aplican en los límites de la entrada
- El filtro “ve” la entrada de extremo a extremo

CS 230 - Convolutional Neural Networks Cheatsheet, https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks#

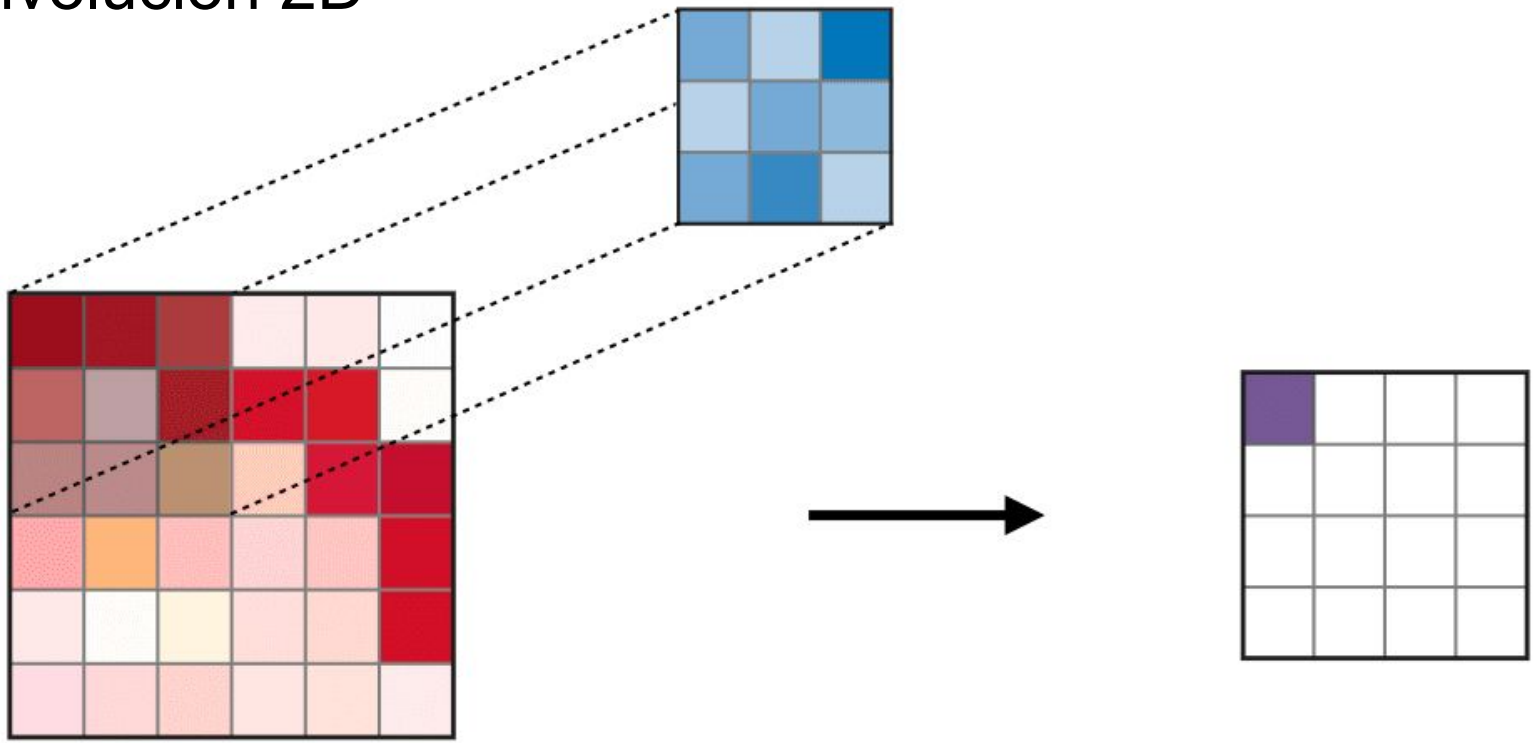
Convolución 2D

La ecuación para la ecuación en 2D es:

$$(w * x)(i,j) = \sum a=0 \sum b=0 w_{a,b} \cdot x_{(i+a),(j+b)}$$

Referencias: Murphy, K. P. (2022). Probabilistic machine learning: an introduction. MIT press.

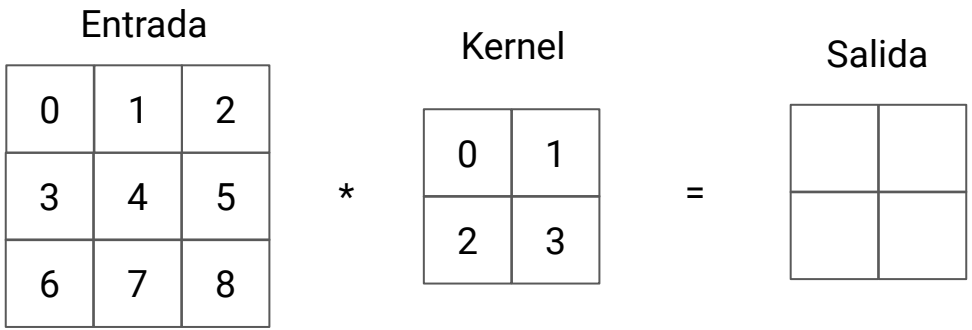
Convolución 2D



CS 230 - Convolutional Neural Networks Cheatsheet, https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks#

Convolución 2D: Ejemplo

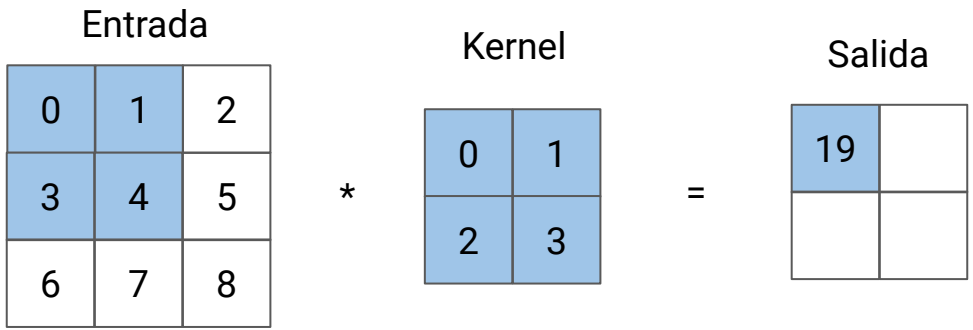
x = [[0,1,2],[3,4,5],[6,7,8]] **w = [[0,1][2,3]]**



Referencias: Murphy, K. P. (2022). Probabilistic machine learning: an introduction. MIT press.

Convolución 2D: Ejemplo

x = [[0,1,2],[3,4,5],[6,7,8]] **w = [[0,1][2,3]]**

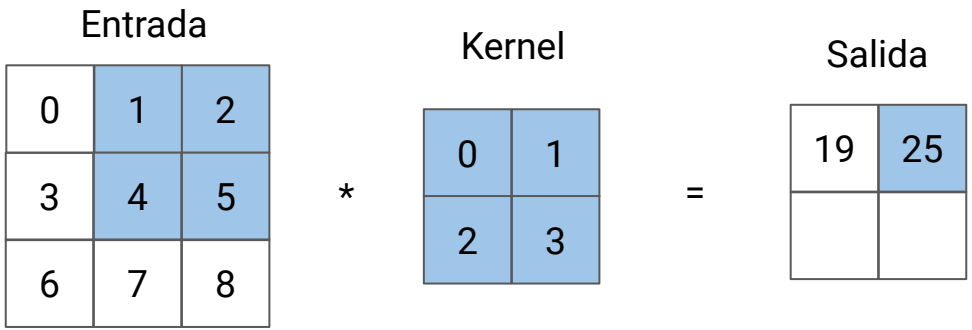


(0 * 0) + (1 * 1) + (3 * 2) + (4 * 3) = 0 + 1 + 6 + 12 = 19

Referencias: Murphy, K. P. (2022). Probabilistic machine learning: an introduction. MIT press.

Convolución 2D: Ejemplo

x = [[0,1,2],[3,4,5],[6,7,8]] **w = [[0,1][2,3]]**

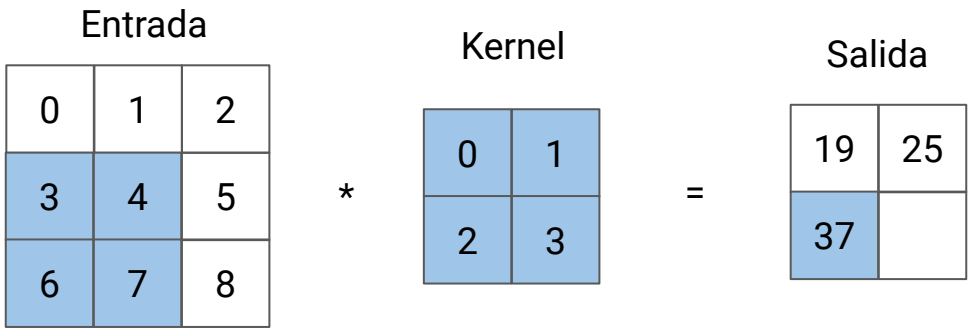


(1 * 0) + (2 * 1) + (4 * 2) + (5 * 3) = 0 + 2 + 8 + 15 = 25

Referencias: Murphy, K. P. (2022). Probabilistic machine learning: an introduction. MIT press.

Convolución 2D: Ejemplo

x = [[0,1,2],[3,4,5],[6,7,8]] **w = [[0,1][2,3]]**

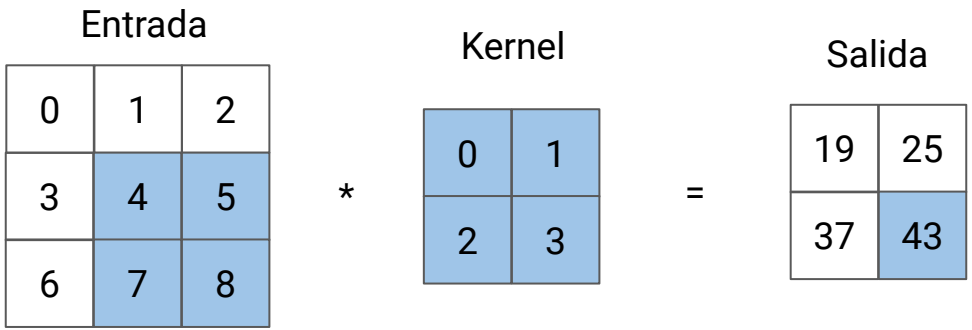


(3 * 0) + (4 * 1) + (6 * 2) + (7 * 3) = 0 + 4 + 12 + 21 = 37

Referencias: Murphy, K. P. (2022). Probabilistic machine learning: an introduction. MIT press.

Convolución 2D: Ejemplo

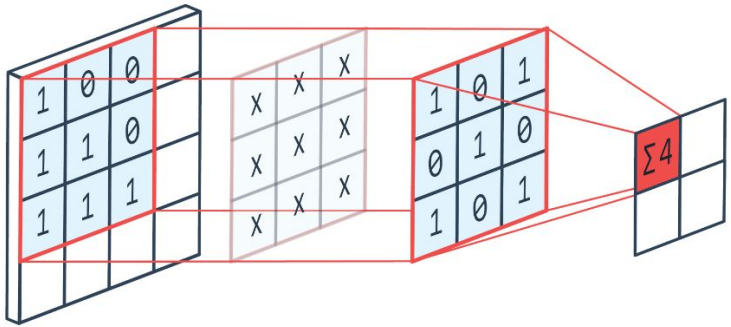
x = [[0,1,2],[3,4,5],[6,7,8]] **w = [[0,1][2,3]]**

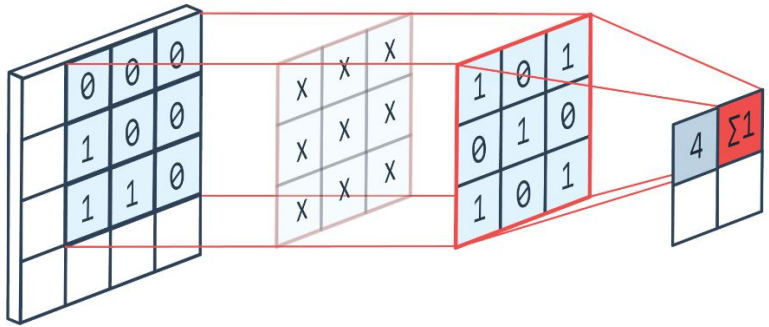


(4 * 0) + (5 * 1) + (7 * 2) + (8 * 3) = 0 + 5 + 14 + 24 = 43

Referencias: Murphy, K. P. (2022). Probabilistic machine learning: an introduction. MIT press.

Convolución 2D

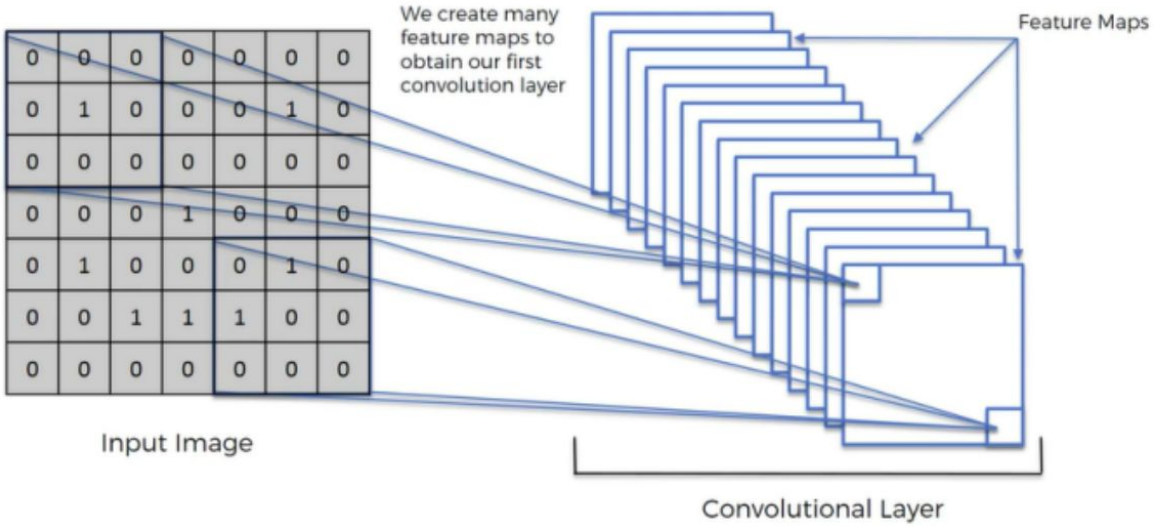




Muestra una convolución 2D. con 1 filtro (canal), altura (height) 3, ancho (width) 3, paso o salto (stride) 1 y 0 padding.

Source: https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/blocks/1d-convolution

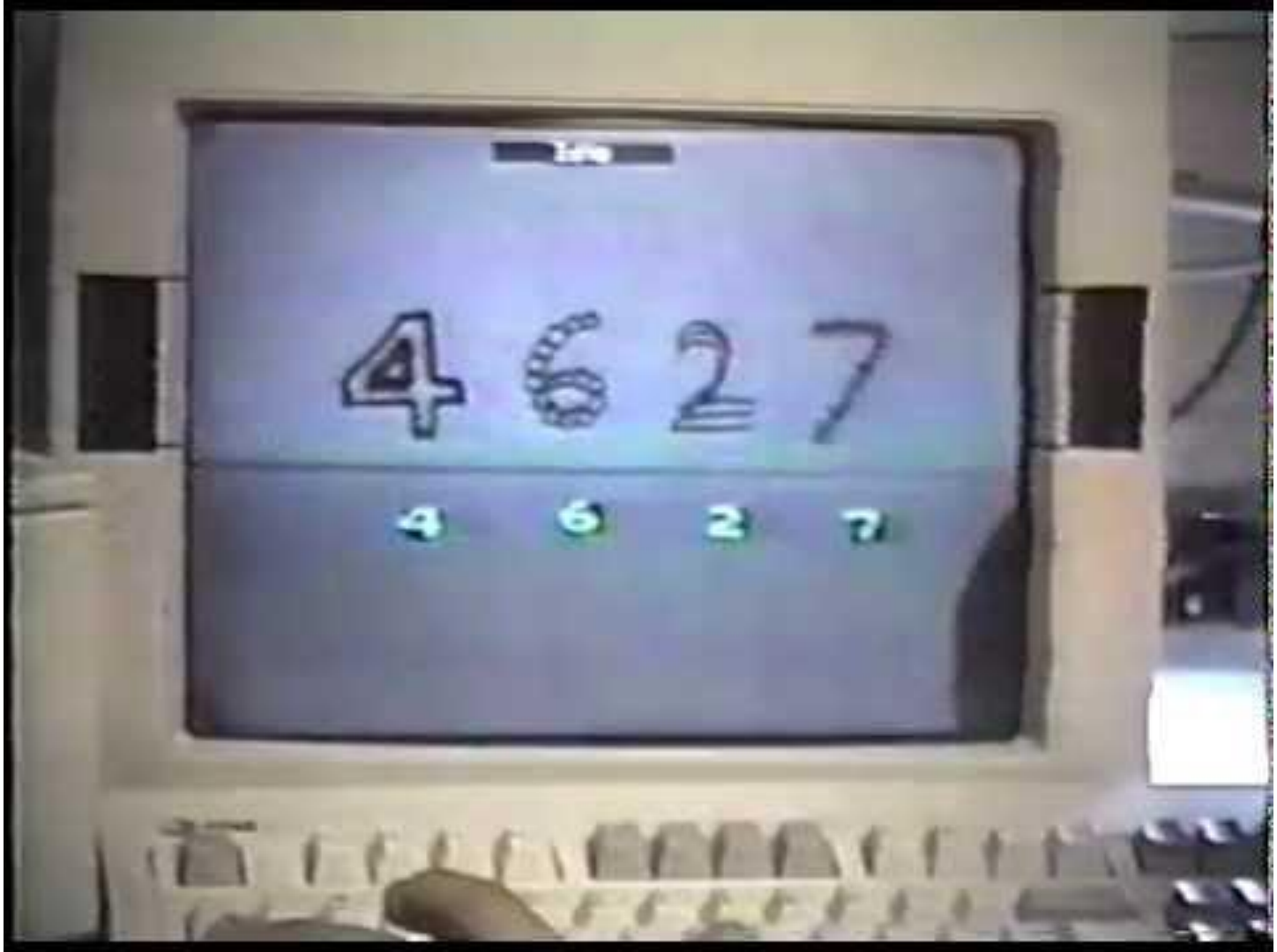
Convolución 2D



En la aplicaciones de la vida real, se trabaja con múltiples mapas de características o también llamada capa convolucional. Donde la característica individual contribuye a dar información particular.

Yann LeCun

Demostración: red convolucional de 1993


4627
4 6 2 7

Convolución 2D: Ejercicio

- Implementar la convolución en 2D

$$x = [[0,1,2],[3,4,5],[6,7,8]] \qquad w = [[0,1][2,3]]$$

Entrada

0	1	2
3	4	5
6	7	8

*

Kernel

0	1
2	3

=

Salida

19	25
37	43

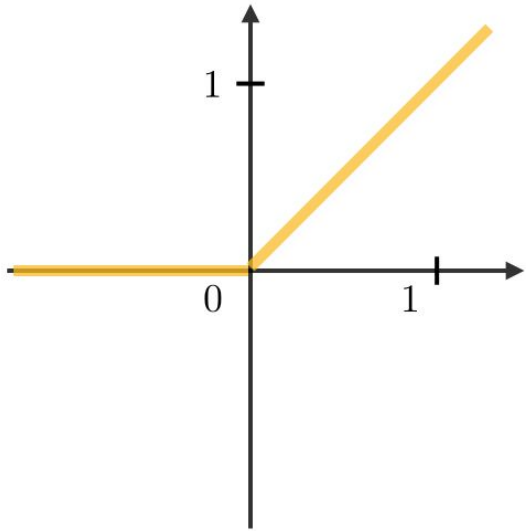
Referencias: Murphy, K. P. (2022). Probabilistic machine learning: an introduction. MIT press.

Funciones de activación comunes

Función ReLU

La función de activación lineal rectificada (ReLU) es una función de activación cuyo objetivo es introducir no linealidades en la red. También se emplean variantes de ella.

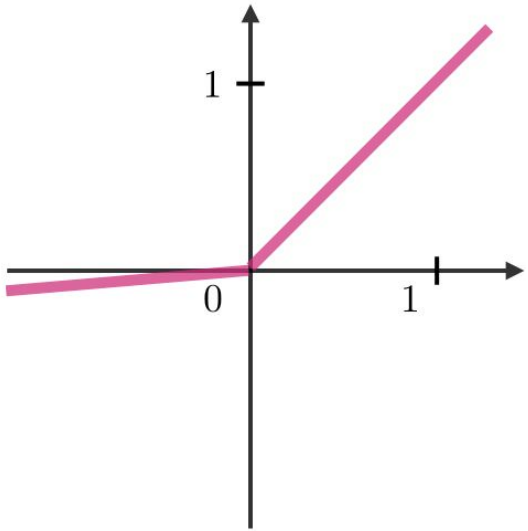
$$ReLU = g(z)=\max(0,z)$$



Leaky ReLU

Aborda el problema de ReLU moribundo para valores negativos

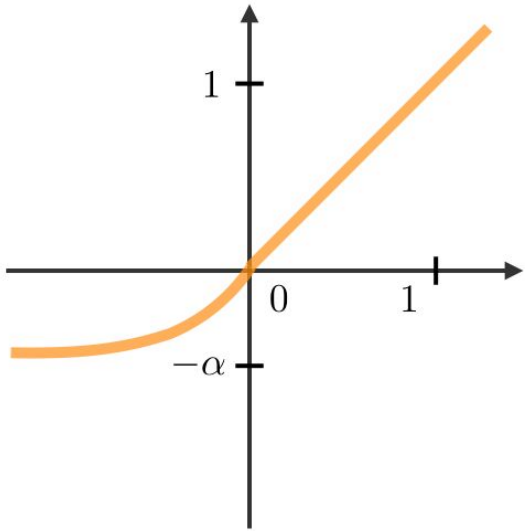
$$g(z) = max(\epsilon z, z) \text{ con } \epsilon < 1$$



ELU

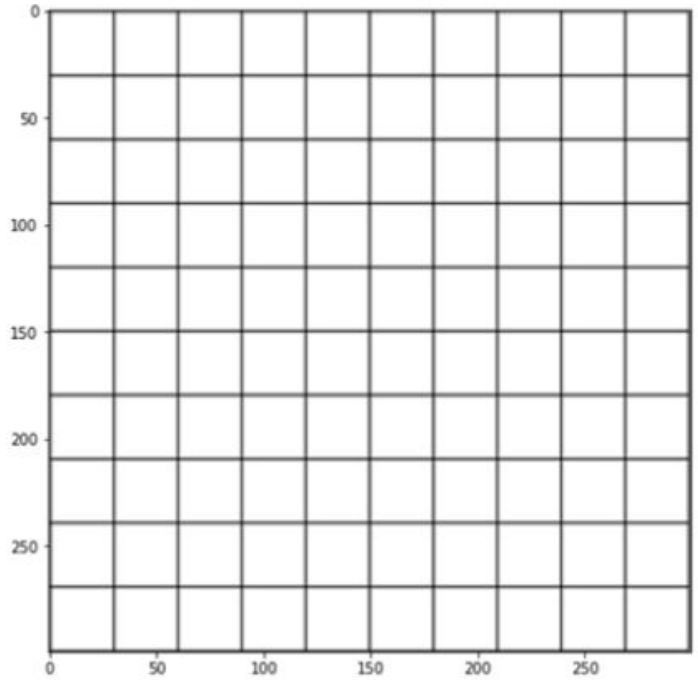
La función de activación ELU (Exponential Linear Units) es diferenciable en todas partes.

$$g(z) = max(\alpha(e^z - 1), z) \text{ con } \alpha < 1$$



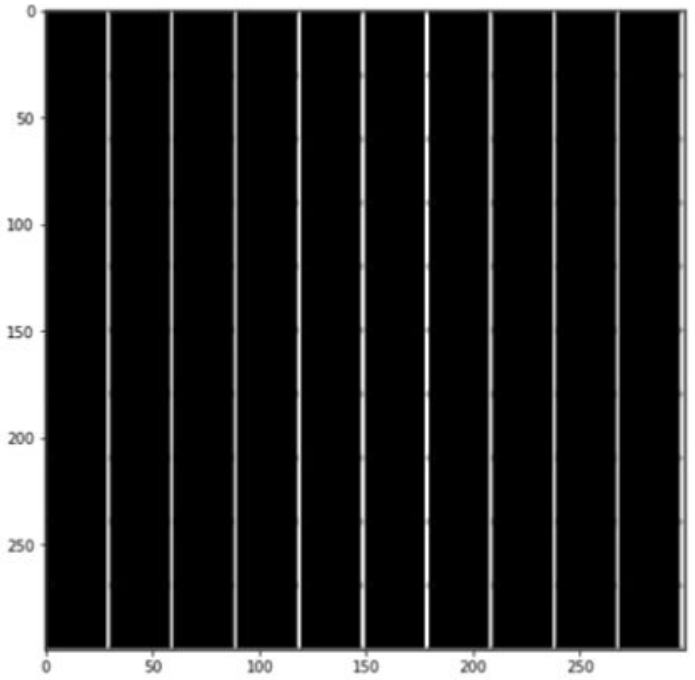
Filtros Comunes

Filtros comunes: Bordes verticales

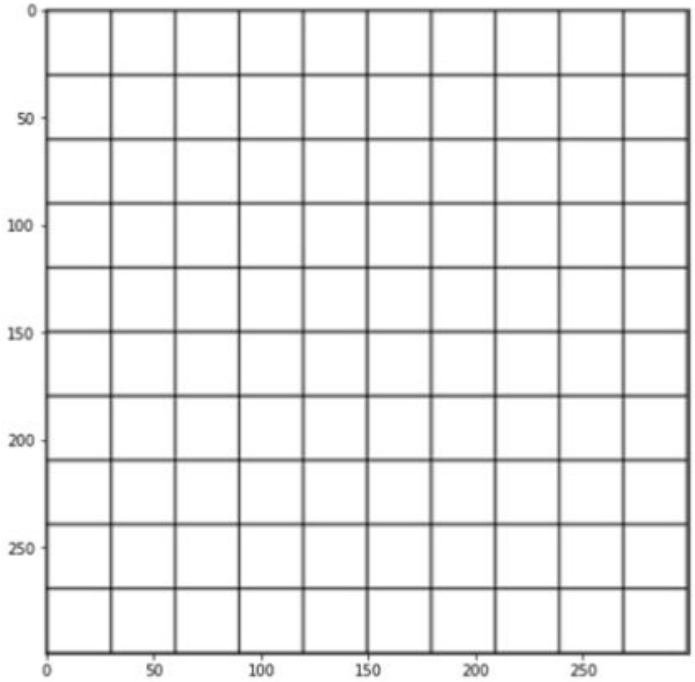


Vertical edge filter

1	0	-1
1	0	-1
1	0	-1

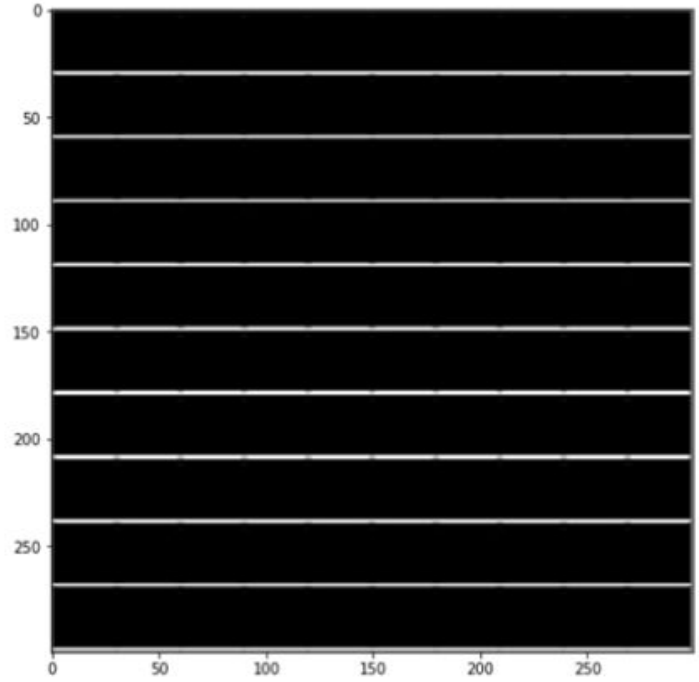


Filtros comunes: Bordes horizontales

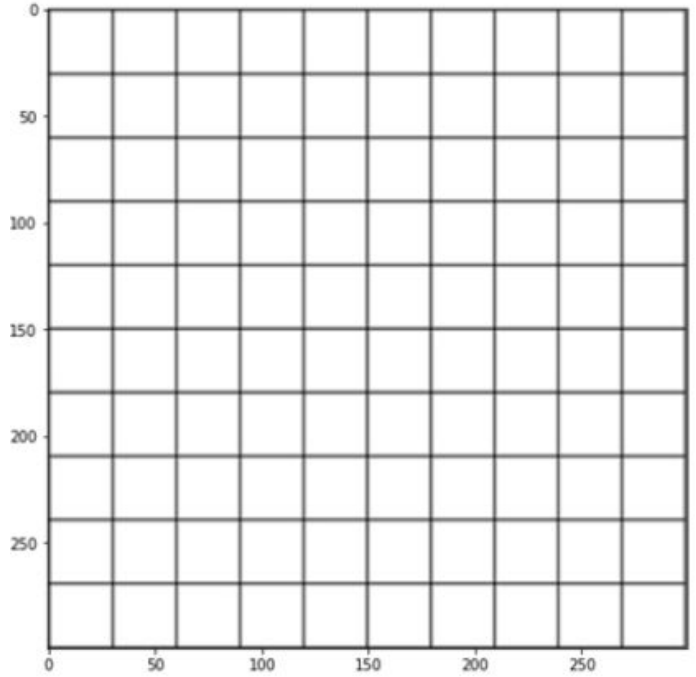


Horizontal edge filter

1	1	1
0	0	0
-1	-1	-1

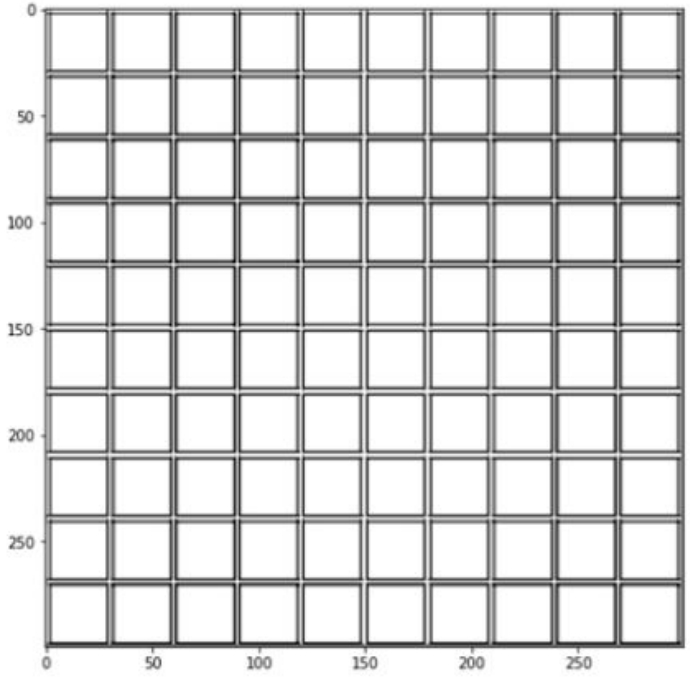


Filtros comunes: Bordes horizontales



Edge filter

1	1	1
1	-7	1
1	1	1



Filtros comunes

Bordes oscuros

$$\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$

desenfocado de caja

$$\frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

desenfocado gaussiano

$$\frac{1}{16}\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

Realzar

$$\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$

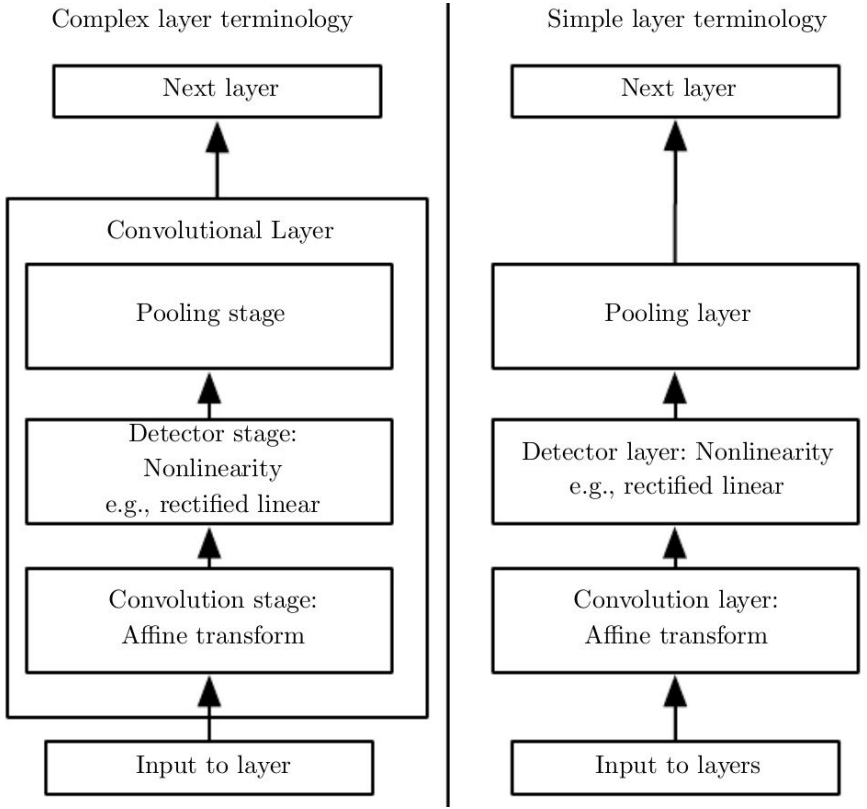
Filtros comunes: Ejercicio

- Implementar la operación convolución 2D para al menos 4 filtros:
  - bordes verticales
  - bordes horizontales
  - bordes
  - bordes oscuros
  - desenfocado de caja
  - desenfocado gaussiano
  - realzar
- Aplicar a un solo canal (escala de grises)
- Los filtros se aplicarán a una imagen
- Visualizar el resultado de aplicar el filtro
- *Aplicar a los tres canales (RGB)

Agrupamiento (Pooling)

Agrupamiento

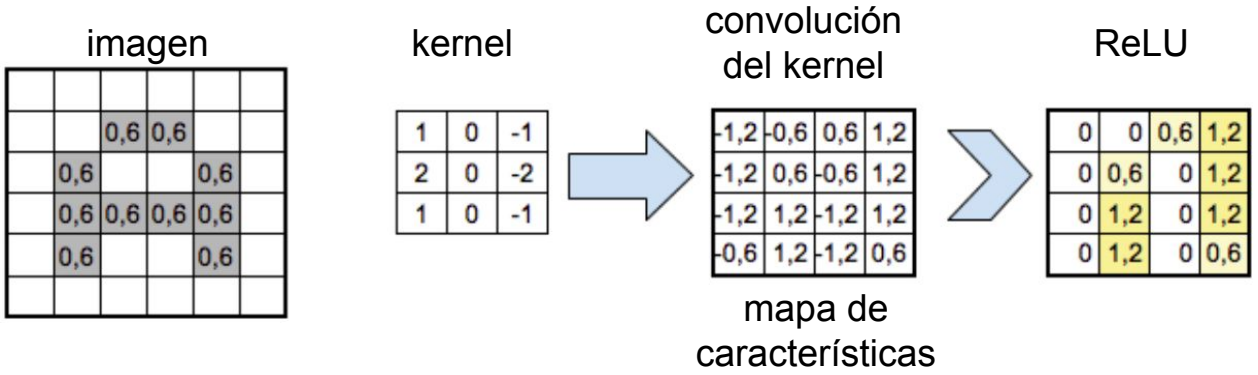
La capa de agrupación (Pooling) es una operación de reducción de muestreo, normalmente aplicada después de una capa de convolución, que produce cierta invariancia espacial. En particular, la agrupación máxima y promedio son tipos especiales de agrupación donde se toma el valor máximo y promedio, respectivamente.



Referencias: Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press.
CS 230 - Convolutional Neural Networks Cheatsheet, ttps://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks#

Agrupamiento

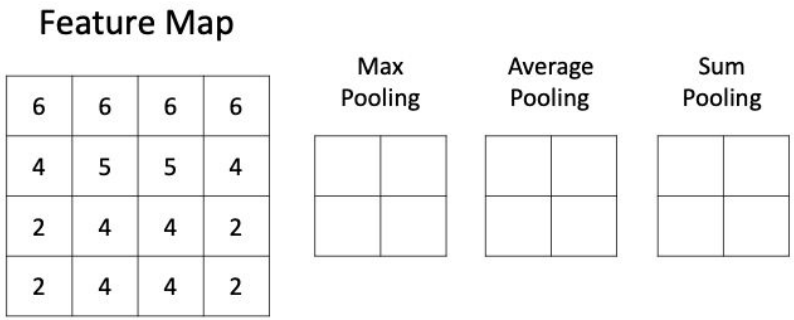
Max Pooling ayuda a reducir el mapa de características para hacer la clasificación con mayor precisión.



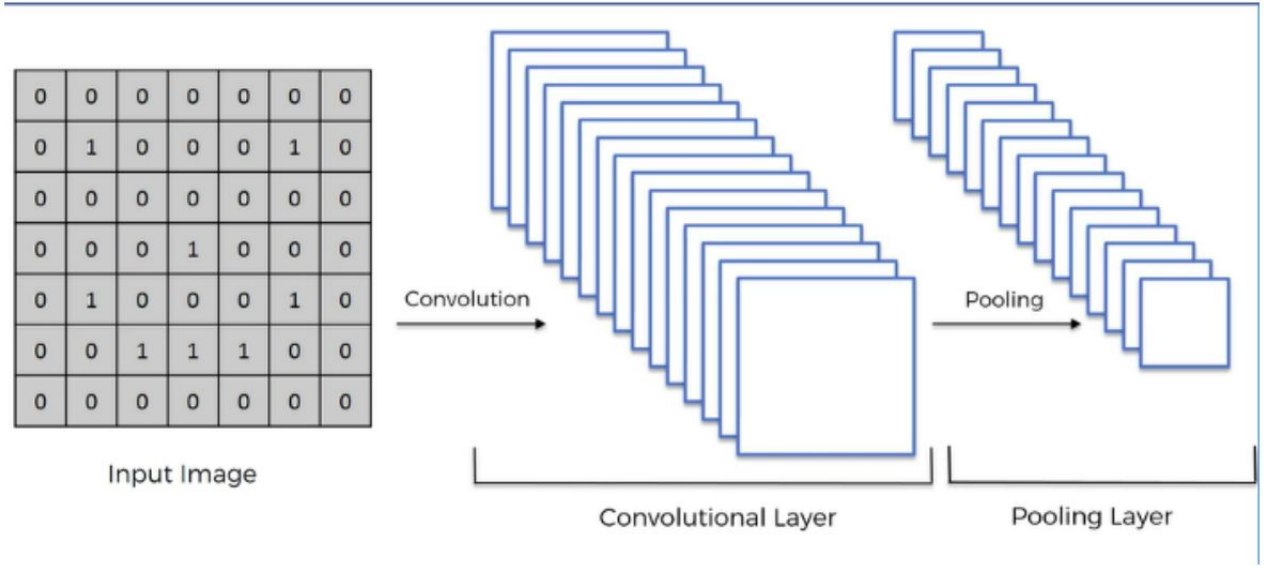
Referencias: Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press.
CS 230 - Convolutional Neural Networks Cheatsheet, ttps://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks#

Agrupamientos comunes

- **Max Pooling**. Toma el valor máximo de píxel dentro del filtro.
- **Average Pooling**. Toma el valor de píxel promedio dentro del filtro.
- **Sum Pooling**. Suma los valores de píxel dentro del filtro.

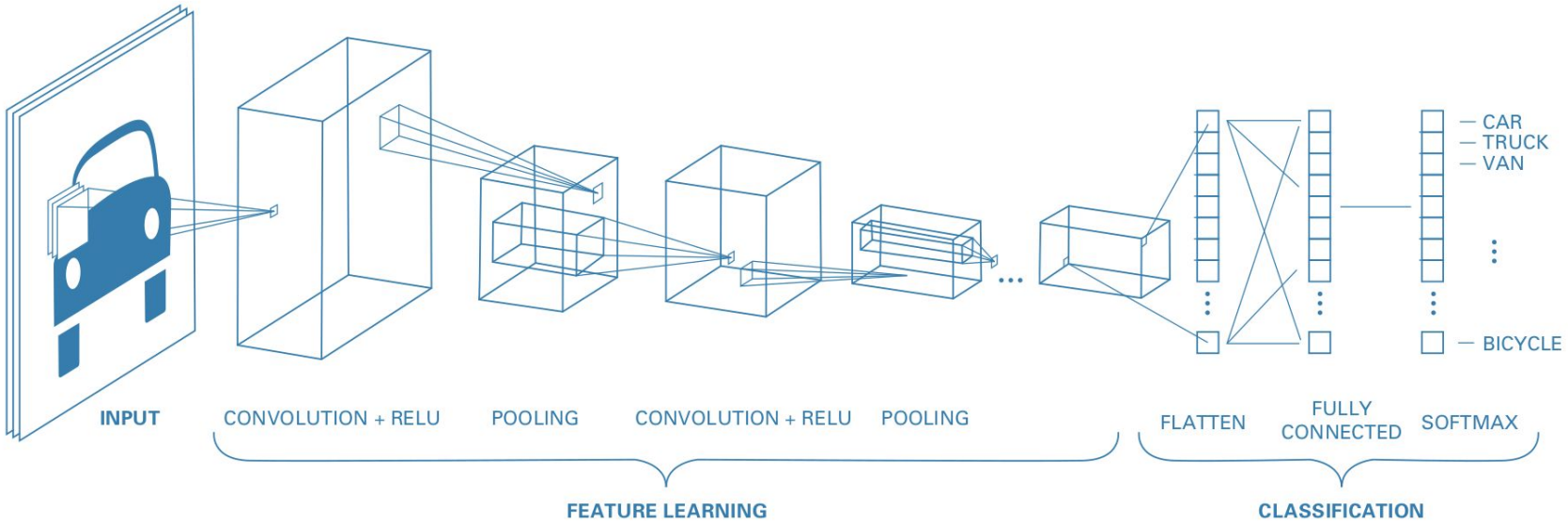


Agrupamiento



Ahora para cada uno de los mapas de características, estamos aplicando max-pooling para obtener un mapa de características agrupadas.

Agrupamiento



Referencias: Murphy, K. P. (2022). Probabilistic machine learning: an introduction. MIT press.

Comprendiendo la complejidad del Modelo

Complejidad del modelo:

Input Layer (28, 28, 1) = **(Alto * Ancho * Canal)** = 28 * 28 * 1 = **784**

Conv Layer (28, 28, 16) = **(Alto * Ancho * Canal)** = 28 * 28 * 16 = **12,544**

Pooling Layer (14, 14, 16) = **(Alto * Ancho * Canal)** = 14 * 14 * 16 = **3,136**

Fully Connected Layer (100, 1) = **(#neuronas * canal) =** 100 * 1 = **100**

Activation Shape

Activation Size

Complejidad del modelo: # Parámetros

Conv Layer:

$$\text{param} = ((\text{kernel}_H * \text{kernel}_W * \#\text{filtros\_capa\_previa}) + 1) * \#\text{filtros\_actuales}$$

Fully Connected Layer (100, 1):

$$\text{param} = (\#\text{neuronas} * \text{tamaño\_activación\_previo} * 1) + \text{tamaño\_activación\_actual}$$

Complejidad del modelo: Convolución

No		Activation Shape	Activation Size	# Parameters
1	Input Layer	(28, 28, 1)	784	0
2	CONV1	(28, 28, 16) (5, 5)	12,544	((5 * 5 * 1) + 1) * 16 = (26) * 16 = 416
3	POOL1	(14, 14, 16)	3,136	0
4	CONV2	(14, 14, 32) (5, 5)	6,272	((5 * 5 * 16) + 1) * 32 = (401) * 32 = 12,932
5	POOL2	(7, 7, 32)	1,568	0
6	FC3	(100, 1)	100	100 * 1568 * 1 + 100 = 156,800 + 100 = 156,900
7	FC4	(10, 1)	10	10 * 100 * 1 + 10 = 1000 + 10 = 1,010
8	Total parameters			171,158