

Inteligencia Artificial

CC-421

César Lara Avila

Universidad Nacional de Ingeniería

(actualización: 2021-02-02)

Bienvenidos

Redes convolucionales

- Introducción
- Bloques de construcción básicos de CNN
- Arquitecturas de CNN
- Propagación hacia adelante y hacia atrás en CNN

Introducción

¿Cómo leen los autos sin conductor las señales de tráfico? ¿Cómo te etiqueta Facebook automáticamente en imágenes? ¿Cómo logra una computadora la clasificación "dermatológica" de las enfermedades de la piel?

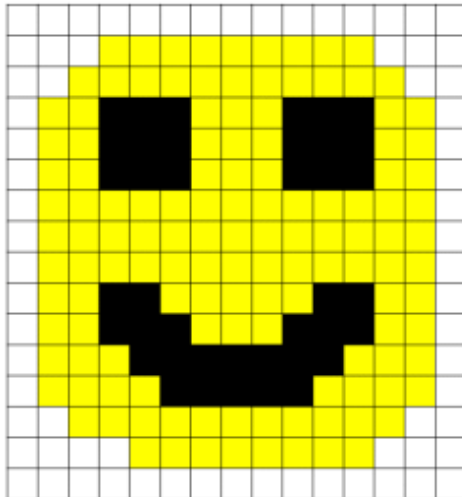
En todas estas aplicaciones, una computadora debe **ver** el mundo: toma una representación numérica de la radiación electromagnética (por ejemplo, una fotografía) y descubre qué significa esa radiación.

La visión por computadora es un campo amplio que combina inteligencia artificial, ingeniería, procesamiento de señales y otras técnicas, para permitir que las computadoras **vean**.

Una **red neuronal convolucional** (CNN) es un tipo de modelo de visión por computadora.

La entrada a una CNN para una aplicación de visión por computadora es una imagen o un video.




picture display



picture representation

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	2	2	2	2	2	2	2	2	2	2	2	0	0	0	0	0	0
0	0	2	2	2	2	2	2	2	2	2	2	2	2	0	0	0	0	0	0
0	2	2	9	9	9	2	2	2	9	9	9	2	2	0	0	0	0	0	0
0	2	2	9	9	9	2	2	2	9	9	9	2	2	0	0	0	0	0	0
0	2	2	9	9	9	2	2	2	9	9	9	2	2	0	0	0	0	0	0
0	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	0
0	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	0
0	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	0
0	2	2	9	9	2	2	2	2	9	9	2	2	2	0	0	0	0	0	0
0	2	2	9	9	9	2	2	2	9	9	9	2	2	0	0	0	0	0	0
0	2	2	2	9	9	9	9	9	9	9	9	2	2	2	2	2	2	2	0
0	2	2	2	2	9	9	9	9	9	9	2	2	2	2	2	2	2	2	0
0	0	2	2	2	2	2	2	2	2	2	2	2	2	0	0	0	0	0	0
0	0	0	0	2	2	2	2	2	2	2	2	2	2	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

La salida de un CNN depende de la tarea.

Task	Input	Output
object classification		<div>fireplace</div> <div>table</div> <div>couch</div> <div>chair</div> <div>mirror</div> <div>bed</div> <div>$\begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$</div>
emotion classification		<div>smiling</div> <div>crying</div> <div>laughing</div> <div>angry</div> <div>$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$</div>
animal classification		<div>dog</div> <div>fish</div> <div>cat</div> <div>bird</div> <div>$\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$</div>

Idea general

En una CNN, diferentes **filtros** (pequeñas cuadrículas de números) se deslizan a través de una imagen completa, calculando la operación de convolución. Diferentes filtros con diferentes números en ellos detectarán diferentes aspectos de la imagen, como bordes horizontales vs. verticales.

Se utilizan muchos filtros diferentes en una CNN para identificar muchos aspectos diferentes de una imagen.

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	2	2	2	2	2	2	2	2	2	0	0	0
0	0	2	2	2	2	2	2	2	2	2	2	0	0	0
0	2	2	9	9	9	2	2	2	9	9	9	2	2	0
0	2	2	9	9	9	2	2	2	9	9	9	2	2	0
0	2	2	9	9	9	2	2	2	9	9	9	2	2	0
0	2	2	2	2	2	2	2	2	2	2	2	2	2	0
0	2	2	2	2	2	2	2	2	2	2	2	2	2	0
0	2	2	2	2	2	2	2	2	2	2	2	2	2	0
0	2	2	9	9	2	2	2	2	2	9	9	2	2	0
0	2	2	9	9	9	2	2	2	9	9	9	2	2	0
0	2	2	2	9	9	9	9	9	9	9	2	2	2	0
0	2	2	2	2	9	9	9	9	9	2	2	2	2	0
0	0	2	2	2	2	2	2	2	2	2	2	0	0	0
0	0	0	0	2	2	2	2	2	2	2	2	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bloques de construcción básicos de CNN

Sistemas lineales invariantes en el tiempo

En el procesamiento de señales o el análisis de series de tiempo, una transformación o un sistema que es lineal e invariante en el tiempo se denomina sistema lineal invariante en el tiempo (LTI), es decir, si $y(t) = T(x(t))$, entonces $y(t - s) = T(x(t - s))$, donde $x(t)$ y $y(t)$ son las entradas y salidas, mientras que $T()$ es la transformación.

Un sistema lineal posee las siguientes dos propiedades:

- Escala: $T(ax(t)) = aT(x(t))$
- Superposición $T(x_1(t) + x_2(t)) = T(x_1(t)) + T(x_2(t))$

El operador de convolución y sus propiedades

La convolución es una operación matemática realizada en sistemas LTI en la que una función de entrada $x(t)$ se combina con una función $h(t)$ (kernel o filtro) para dar una nueva salida que significa una superposición entre $x(t)$ y la versión inversa $h(t)$.

$$y(t) = (h \times x)(t) = \int_{-\infty}^{\infty} h(\tau)x(t - \tau)d\tau$$

En el dominio discreto, en una dimensión, se puede definir como:

$$y(i) = (h \times x)(i) = \sum_n h(n)x(i - n)$$

Del mismo modo en dos dimensiones, más utilizado en **visión artificial** con imágenes fijas:

$$y(i, j) = (h \times x)(i, j) = \sum_n \sum_m h(m, n)x(i - m, j - n)$$

El resultado anterior se puede escribir como correlación cruzada o kernel invertido o rotado:

$$y(i, j) = (h \times x)(i, j) = \sum_n \sum_m x(i + m, i + n) h(-m, -n)$$

$$y(i, j) = (h \times x)(i, j) = x(i + m, i + n) \times \text{rotación}_{180}\{h(m, n)\}$$

La convolución exhibe las propiedades conmutativas, distributivas, asociativas y diferenciables generales.

Correlación cruzada y sus propiedades

La correlación cruzada es una operación matemática muy similar a la convolución y es una medida de similitud o de la fuerza de la correlación entre dos señales $x(t)$ y $h(t)$. Está dado por:

$$y(t) = (h \otimes x)(t) = \int_{-\infty}^{\infty} h(\tau)x(t + \tau)$$

En el dominio discreto, en una dimensión, se puede definir como:

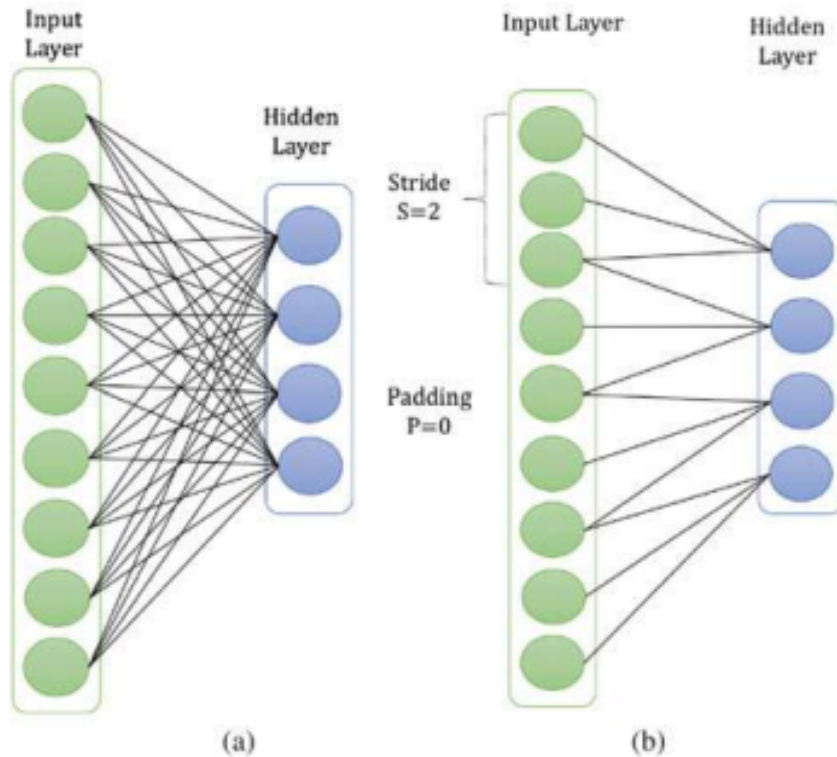
$$y(i) = (h \otimes x)(i) = \sum_n h(n)x(i + n)$$

Del mismo modo en dos dimensiones:

$$y(i, j) = (h \otimes x)(i, j) = \sum_n \sum_m h(m, n)x(i + m, i + n)$$

Muchas CNN emplean el operador de correlación cruzada, pero la operación se llama **convolución**.

Conectividad local o interacciones dispersas

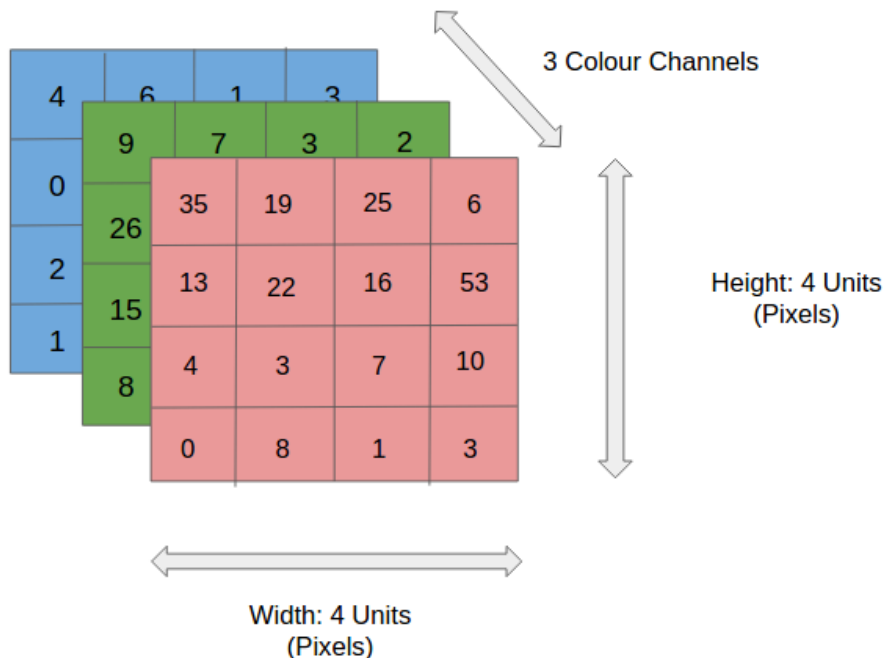


Palabras claves: **Filtro, Kernel, Campo receptivo**

Volumen de entrada

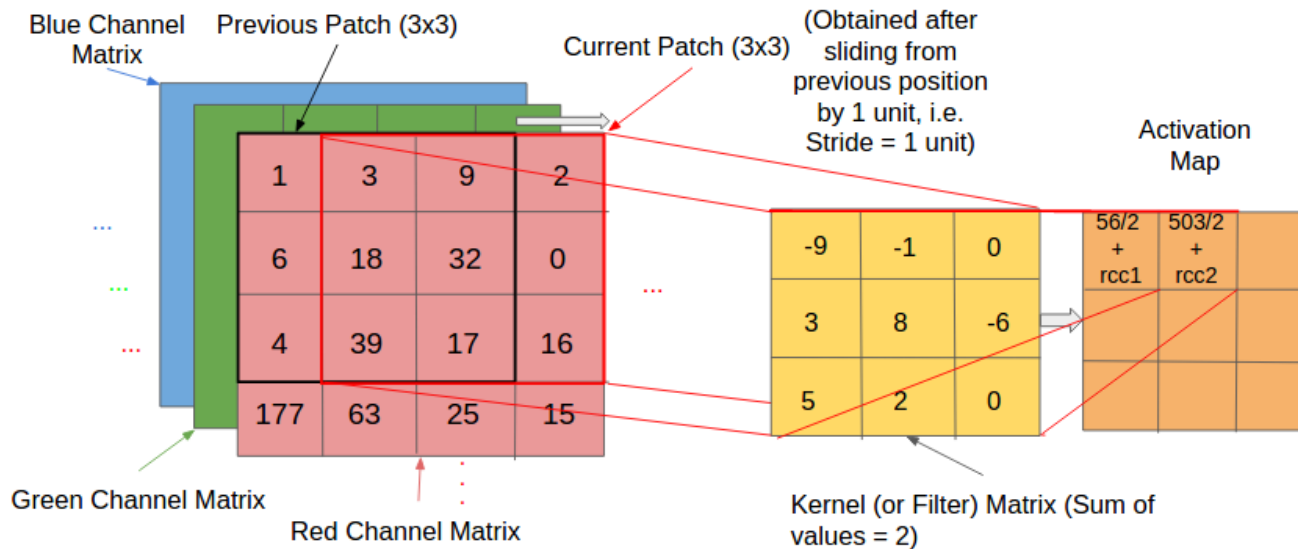
Las CNN se suelen aplicar a datos de imágenes. Cada imagen es una matriz de valores de píxeles.

Si tenemos imágenes basadas en RGB la presencia de canales de color separados introduce un campo de **profundidad** adicional a los datos, lo que hace que la entrada 3- dimensional.



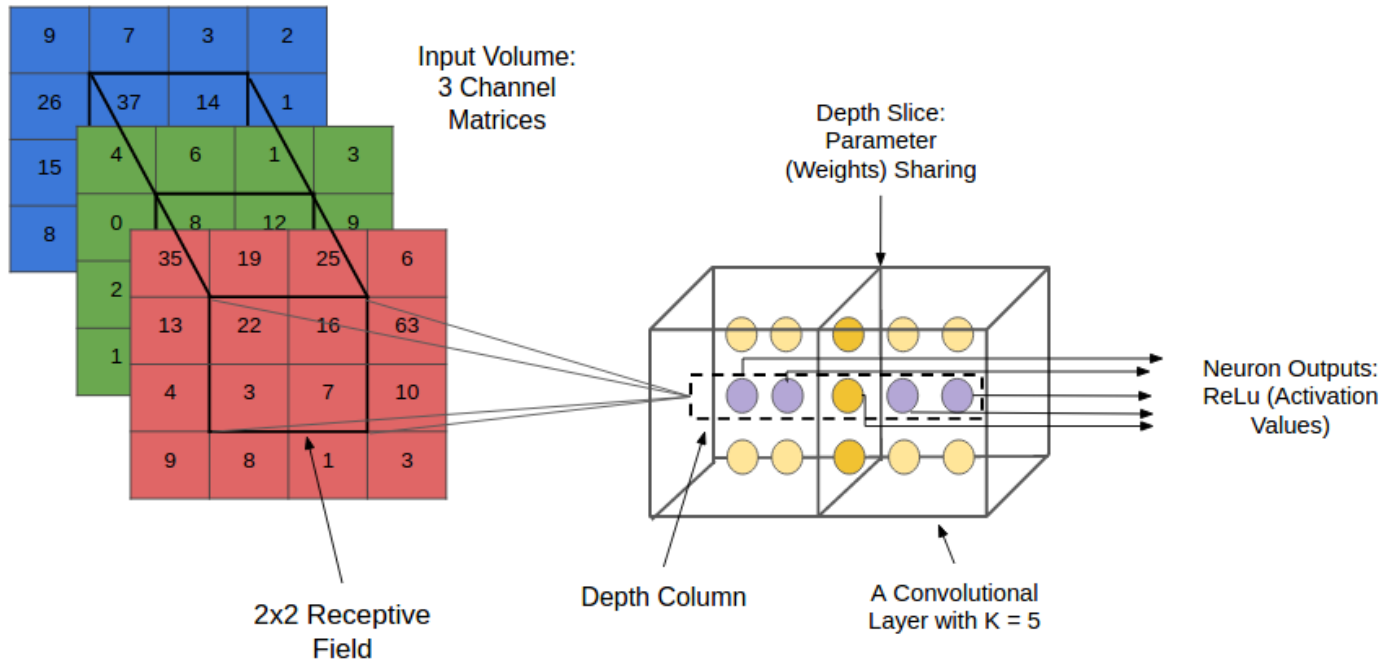
Operaciones del kernel

El procedimiento exacto para convolucionar un kernel (digamos, de tamaño 16×16) con el volumen de entrada (una imagen RGB de tamaño $256 \times 256 \times 3$ en nuestro caso) implica tomar parches de la imagen de entrada de tamaño igual al del kernel (16×16) y convolucionar (o calcular el producto escalar) entre los valores del parche y los de la matriz del kernel.



Campo receptivo

Es una región bidimensional (digamos de tamaño 5×5 unidades) que se extiende a toda la profundidad de la entrada ($5 \times 5 \times 3$ para un color de 3 canal de entrada), dentro del cual los píxeles incluidos están completamente conectados a la capa de entrada de la red neuronal.



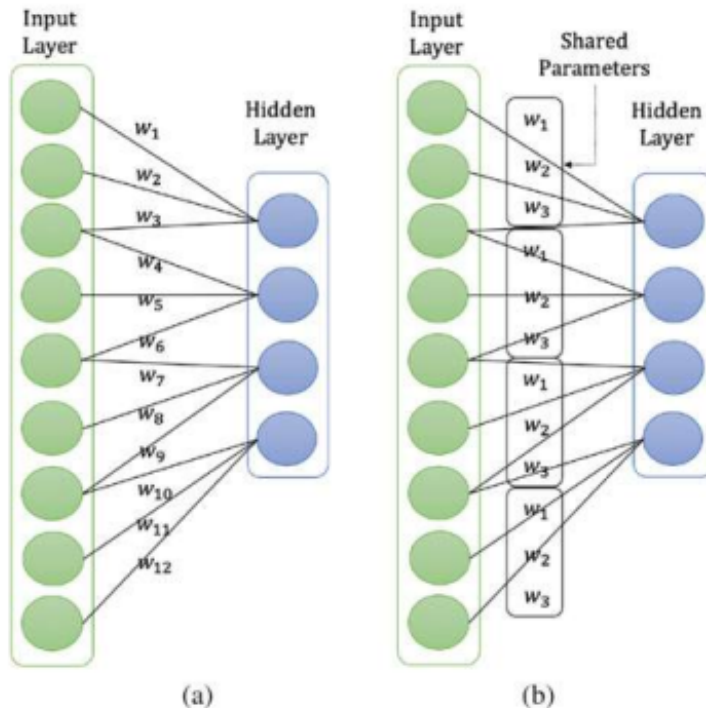
Zero-padding

Se refiere al proceso de agregar ceros simétricamente a la matriz de entrada.

Se utiliza principalmente en el diseño de las capas CNN cuando las dimensiones del volumen de entrada deben conservarse en el volumen de salida.

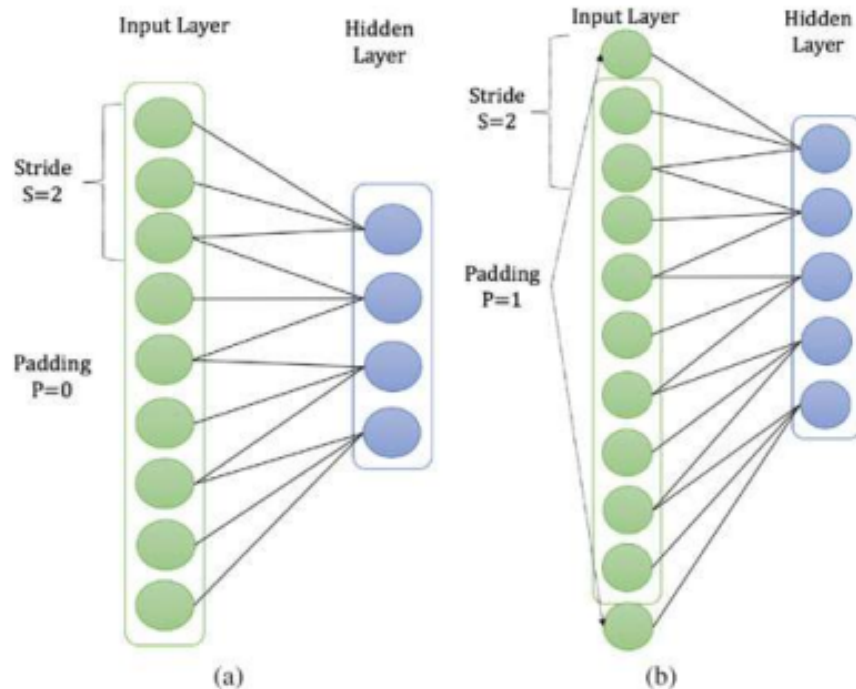
0	0	0	0	0	0
0	35	19	25	6	0
0	13	22	16	53	0
0	4	3	7	10	0
0	9	8	1	3	0
0	0	0	0	0	0

Compartir parámetros



La combinación de conectividad local y el uso compartido de parámetros da como resultado **filtros que capturan características comunes** que actúan como bloques de construcción en todas las entradas.

Disposición espacial



Palabras claves: **Profundidad, stride, zero-padding, padding, wide convolution, narrow convolution.**

Relación de hiperparámetros

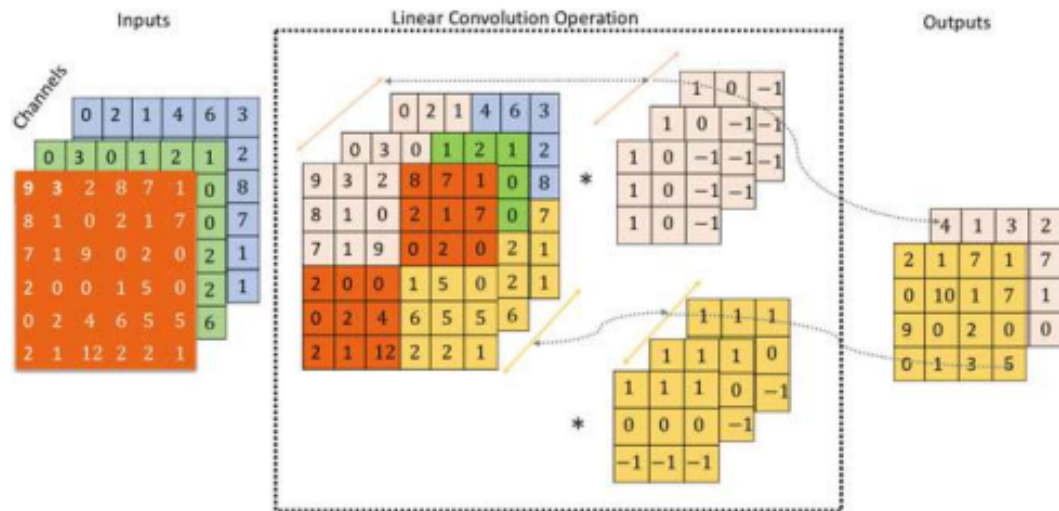
En las CNN, las propiedades que pertenecen a la estructura de capas y neuronas, como la disposición espacial y los valores de campo receptivo, se denominan hiperparámetros.

La relación entre el número de entradas (volumen de entrada) W , el campo receptivo (tamaño de filtro) F , el tamaño del stride S y el zero-padding P conduce al número de neuronas en la siguiente capa, como se indica en la ecuación:

$$N = \frac{W - F + 2P}{S} + 1$$

Convolución 3D

La figura siguiente extiende el proceso de convolución a una entrada 3D con tres canales (similar a los canales RGB en una imagen) y dos filtros, que muestran cómo la convolución lineal reduce el volumen entre capas.



Estos filtros actúan como un mecanismo complejo de detección de características. Dado un gran volumen de datos de entrenamiento, una CNN puede aprender filtros, como bordes horizontales, bordes verticales, contorno y más en el proceso de clasificación.

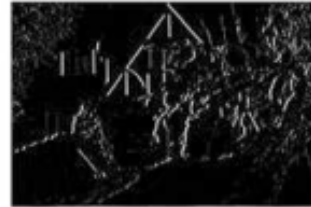
Horizontal Edge Kernel

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



Vertical Edge Kernel

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



Outline Kernel

-1	-1	-1
-1	8	-1
-1	-1	1



Emboss Kernel

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



El número de canales en los filtros debe coincidir con el número de canales en su entrada.

Ejemplo

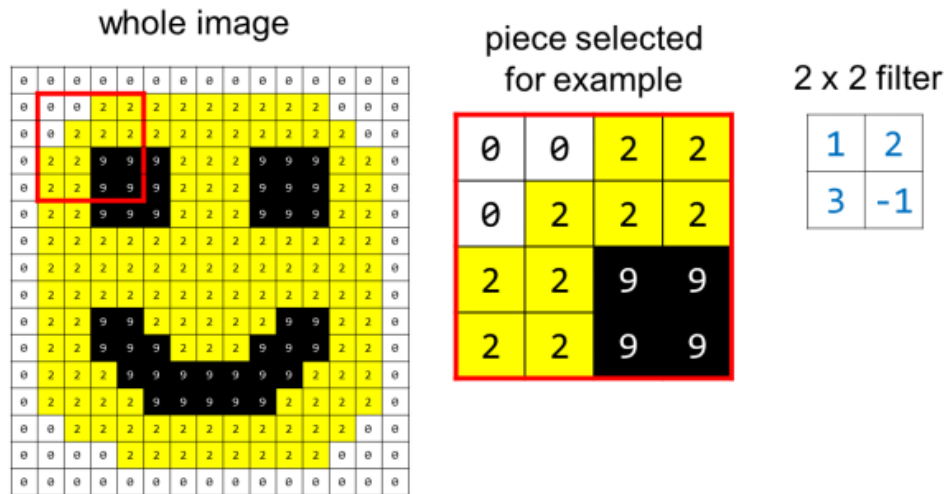
Recuerda: Un filtro CNN es una cuadrícula cuadrada de números. Los tamaños de filtro se especifican cuando se crea la CNN. Algunos tamaños de filtro utilizados comúnmente son 2×2 , 3×3 y 5×5 , pero pueden ser del tamaño que tu elijas.

Cuando se inicializa una CNN antes de que se produzca cualquier entrenamiento, todos los valores para los filtros se establecen en números aleatorios.

A través del proceso de entrenamiento, la CNN ajusta los valores en los filtros para que los filtros detecten características significativas de las imágenes.

		4 x 4 filter				6 x 6 filter					
2 x 2 filter		5	-2	1	0	0	4	-1	-5	1	2
-1	2	4	-3	6	2	5	-2	1	5	-3	0
3	0	0	1	-4	3	-2	1	0	4	2	1
		2	0	-1	4	4	-3	6	3	1	2
						0	1	-4	-1	5	3
						2	0	-1	0	-4	4

Así es como funciona la convolución. Tomemos una pequeña parte de la imagen de la cara sonriente y apliquemos una convolución, usando un filtro de 2×2 con los valores 1, 2, 3 y -1 :



Configuración para el ejemplo:

- El filtro que estamos utilizando se muestra a la izquierda como referencia, con sus valores en letra azul.
- La sección de 2×2 de la imagen que se está convolucionando con el filtro tiene sus valores resaltados en rojo.

		convolution calculation			
1	2	0	0	2	2
3	-1	0	2	2	2
		2	2	9	9
		2	2	9	9
		$(1)(0) + (2)(0) + (3)(0) + (-1)(2) = -2$			
		0	0	2	2
		0	2	2	2
		2	2	9	9
		2	2	9	9
		$(1)(0) + (2)(2) + (3)(2) + (-1)(2) = 8$			

convolution output			
-2			
-2	8		

		convolution calculation			
1	2	0	0	2	2
3	-1	0	2	2	2
		2	2	9	9
		2	2	9	9
		$(1)(2) + (2)(2) + (3)(2) + (-1)(2) = 10$			
		0	0	2	2
		0	2	2	2
		2	2	9	9
		2	2	9	9
		$(1)(2) + (2)(2) + (3)(2) + (-1)(2) = 10$			

convolution output			
-2	8	10	
-2	8	10	10

¿ Qué observas?

Aquí está la salida de convolución para la región roja y el filtro que elegimos:

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	2	2	2	2	2	2	2	2	2	2	0	0	0
0	0	2	2	2	2	2	2	2	2	2	2	2	0	0	0
0	2	2	9	9	9	2	2	2	9	9	9	2	2	0	0
0	2	2	9	9	9	2	2	2	9	9	9	2	2	0	0
0	2	2	9	9	9	2	2	2	9	9	9	2	2	0	0
0	2	2	2	2	2	2	2	2	2	2	2	2	2	0	0
0	2	2	2	2	2	2	2	2	2	2	2	2	2	0	0
0	2	2	2	2	2	2	2	2	2	2	2	2	2	0	0
0	2	2	9	9	9	2	2	2	2	9	9	2	2	0	0
0	2	2	9	9	9	2	2	2	9	9	9	2	2	0	0
0	2	2	2	2	2	9	9	9	9	9	9	2	2	0	0
0	2	2	2	2	2	9	9	9	9	9	9	2	2	0	0
0	0	0	0	0	2	2	2	2	2	2	2	2	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

convolution output for red region
and specified filter

-2	8	10	10
8	3	24	24
10	17	45	45
10	17	45	45

Capa de detección

La salida de una convolución es una transformación afín que se alimenta en una capa o transformación no lineal conocida como **capa/etapa de detección**.

Pooling y subsampling

Las salidas de una capa pueden reducirse aún más para capturar estadísticas resumidas de neuronas o subregiones locales. Este proceso se denomina **pooling**, **subsampling** o **downsampling** según el contexto.

Por lo general, las salidas de la etapa del detector se convierten en las entradas para la capa pooling.

Existen diferentes métodos de pooling y la elección particular depende de la tarea en cuestión. Cuando el sesgo del método pooling coincide con los supuestos hechos en una aplicación CNN particular, como en el ejemplo de detección de rostros, se puede esperar una mejora significativa en los resultados

Max Pooling

Formalmente, para una salida 2D de la etapa de detección, una capa **max pooling** realiza la transformación:

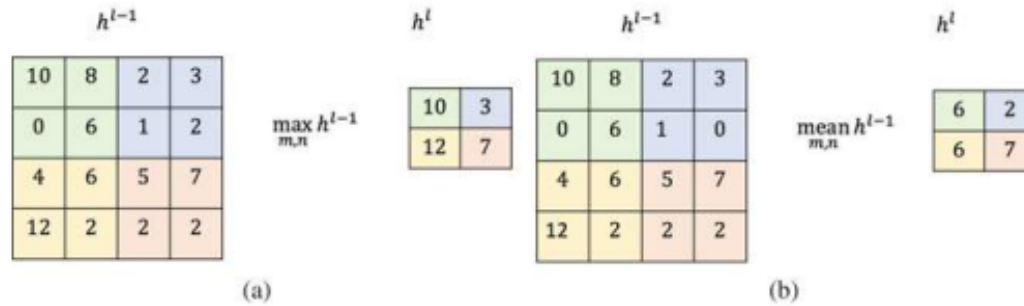
$$h_{i,j}^l = \max_{p,q} h_{i+p,j+q}^{l-1}$$

Average Pooling

En el average o mean pooling, los valores de las neuronas del vecindario local se promedian para dar el valor de salida, como se ilustra en la figura. Formalmente, el average pooling realiza la transformación:

$$h_{i,j}^l = \frac{1}{m^2} \sum_{p,q} h_{i+p,j+q}^{l-1}$$

donde $m \times m$ es la dimensión del kernel.



Pooling de la norma L2

Es una generalización del average pooling y es dado por:

$$h_{i,j}^l = \sqrt{\left(\sum_{p,q} h_{i+p,j+q}^{l-1}\right)^2}$$

Pooling estocástico

En el pooling estocástico, en lugar de elegir el máximo, la neurona elegida se extrae de una distribución multinomial. El pooling estocástico funciona de manera similar al dropout al abordar el problema del sobreajuste.

Arquitecturas de CNN

Hay 3 tipos principales de capas que se observan comúnmente en arquitecturas de redes neuronales complejas:

- **La Capa convolucional:** forma la base de la CNN y realiza las operaciones centrales del entrenamiento y en consecuencia, dispara las neuronas de la red.
- **La capa ReLu:** se refiere a la Unidad Rectificadora, la función de activación más comúnmente implementada para las salidas de las neuronas CNN.
- **La capa pooling:** generalmente se coloca después de la capa convolucional. Su utilidad principal radica en reducir las dimensiones espaciales ($\text{ancho} \times \text{alto}$) del volumen de entrada para la siguiente capa Convolucional. No afecta la dimensión de profundidad del volumen.

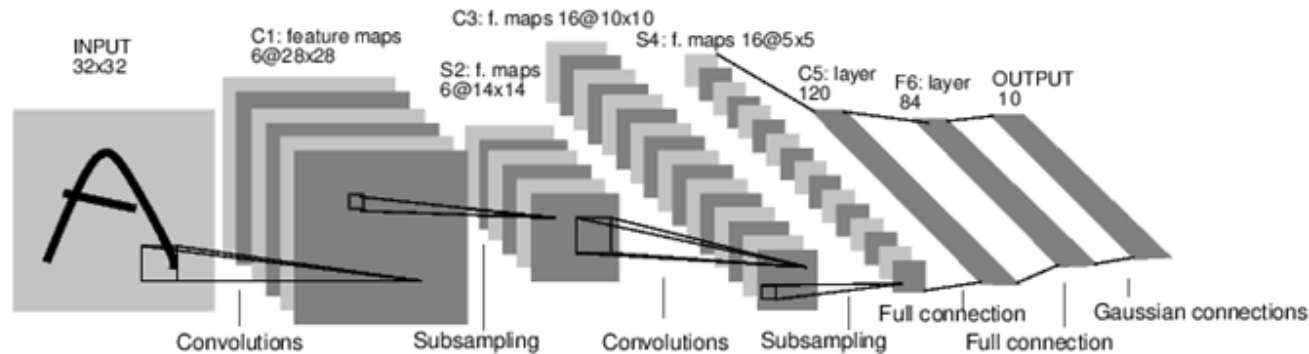
El pooling es opcional en las CNN y muchas arquitecturas no realizan operaciones pooling.

- **La capa completamente conectada (FC):** se configura exactamente como su nombre lo indica: está completamente conectada con la salida de la capa anterior.

Las capas completamente conectadas se utilizan normalmente en las últimas etapas de la CNN para conectarse a la capa de salida y construir la cantidad deseada de salidas.

Un clásica arquitectura se vería así:

Entrada- > Conv -> ReLU-> Pool->ReLU->Conv->ReLU->Pool->FC



A Full Convolutional Neural Network (LeNet)

Propagación hacia adelante y hacia atrás en CNN

Comencemos la derivación para la capa l , que es la salida de convolución en la capa $l - 1$. La capa l tiene altura h , ancho w y canales c . Supongamos que hay un canal, es decir, $c = 1$ y tiene iteradores i, j para las dimensiones de entrada.

Consideraremos las dimensiones del filtro $k_1 \times k_2$ con los iteradores m, n para la operación de convolución. La matriz de pesos con pesos $w_{m,n}^l$ y el sesgo b^l transforman la capa anterior $l - 1$ en la capa l mediante la operación de convolución.

La capa de convolución es seguida por una función de activación no lineal, como ReLU $f(\cdot)$.

La salida para la capa l es denotada por $O_{i,j}^l$.

Por lo tanto, la propagación hacia adelante para la capa l se puede escribir como:

$$X_{i,j}^l = \text{rotación}_{180}\{W_{m,n}^l\} \times O_{i,j}^{l-1} + b^l$$

Esto puede ser expandido como:

$$X_{i,j}^l = \sum_m \sum_n W_{m,n}^l O_{i+m,j+n}^{l-1} + b^l$$

$$O_{i,j}^l = f(X_{i,j}^l)$$

Suponemos que se utilizan mecanismos de error o pérdida, como el error cuadrático medio E . Los errores deben propagarse de nuevo y deben actualizar los pesos del filtro y las entradas recibidas en la capa l .

Por lo tanto, en el proceso de retropropagación, estamos interesados en el gradiente del error E con respecto a la entrada $\partial E / \partial \mathbf{X}$ y los pesos del filtro $\partial E / \partial \mathbf{W}$.

Gradientes con respecto a los pesos $\frac{\partial E}{\partial \mathbf{W}}$

Primero consideraremos el impacto de un solo píxel (m', n') del kernel, dado por $W_{m',n'}$ en el error E , usando la regla de la cadena:

$$\frac{\partial E}{\partial W_{m',n'}^l} = \sum_{i=0}^{h-k_1} \sum_{j=0}^{w-k_2} \frac{\partial E}{\partial X_{i,j}^l} \frac{\partial X_{i,j}^l}{\partial W_{m',n'}^l}$$

Si consideramos $\delta_{i,j}^l$ como el error de gradiente para la entrada de la capa con respecto a la ecuación anterior se puede reescribir como:

$$\frac{\partial E}{\partial W_{m',n'}^l} = \sum_{i=0}^{h-k_1} \sum_{j=0}^{w-k_2} \delta_{i,j}^l \frac{\partial X_{i,j}^l}{\partial W_{m',n'}^l}$$

Escribiendo el cambio en la salida en términos de las entradas (la capa anterior) obtenemos:

$$\frac{\partial X_{i,j}^l}{\partial W_{m',n'}^l} = \frac{\partial}{\partial W_{m',n'}^l} \left(\sum_m \sum_n W_{m,n}^l O_{i+m,j+n}^{l-1} + b^l \right)$$

Cuando tomamos derivadas parciales con respecto a $W_{m',n'}$, todos los valores se vuelven cero excepto los componentes que mapean $m = m'$ y $n = n'$.

$$\frac{\partial X_{i,j}^l}{\partial W_{m',n'}^l} = \frac{\partial}{\partial W_{m',n'}^l} \left(W_{0,0}^l O_{i+0,j+0}^{l-1} + \dots + W_{m',n'}^l O_{i+m',j+n'}^{l-1} + \dots + b^l \right)$$

$$\frac{\partial X_{i,j}^l}{\partial W_{m',n'}^l} = \frac{\partial}{\partial W_{m',n'}^l} \left(W_{m',n'}^l O_{i+m',j+n'}^{l-1} \right)$$

$$\frac{\partial X_{i,j}^l}{\partial W_{m',n'}^l} = O_{i+m',j+n'}^{l-1}$$

Reemplazando el resultado, se obtiene:

$$\frac{\partial E}{\partial W_{m',n'}^l} = \sum_{i=0}^{h-k_1} \sum_{j=0}^{w-k_2} \delta_{i,j}^l O_{i+m',j+n'}^{l-1}$$

Todo el proceso se puede resumir como la convolución de los gradientes con rotación de 180 grados $\sigma_{i,j}^l$ de la capa l con las salidas de la capa $l - 1$, es decir, $O_{i+m',j+n'}^{l-1}$.

Por tanto, las nuevas ponderaciones que se actualizarán se pueden calcular de forma muy similar al paso hacia adelante.

$$\frac{\partial E}{\partial W_{m',n'}^l} = \text{rotación}_{180}\{\sigma_{i,j}^l\} \times O_{m',n'}^{l-1}.$$

Gradientes con respecto a los pesos $\frac{\partial E}{\partial \mathbf{X}}$

A continuación, nos interesa cómo un cambio a una entrada única, dado por $X_{i',j'}$ afecta al error E . El píxel de entrada $X_{i',j'}$ después de la convolución afecta a una región delimitada por la parte superior izquierda $(i' + k_1 - 1, j' + k_2 - 1)$ y abajo a la derecha (i', j') .

Entonces, obtenemos:

$$\frac{\partial E}{\partial X_{i',j'}^l} = \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} \frac{\partial E}{\partial X_{i'-m,j'-n}^{l+1}} \frac{\partial X_{i'-m,j'-n}^{l+1}}{\partial X_{i',j'}^l}$$
$$\frac{\partial E}{\partial X_{i',j'}^l} = \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} \delta_{i'-m,j'-n}^{l+1} \frac{\partial X_{i'-m,j'-n}^{l+1}}{\partial X_{i',j'}^l}$$

Expandiendo solo la tasa de cambio de las entradas, se obtiene:

$$\frac{\partial X_{i'-m, j'-n}^{l+1}}{\partial X_{i', j'}^l} = \frac{\partial}{\partial X_{i', j'}^l} \left(\sum_{m'} \sum_{n'} W_{m', n'}^{l+1} O_{i'-m+m', j'-n+n'}^l + b^{l+1} \right)$$

Escribiendo esto en términos de la capa de entrada l , obtenemos:

$$\frac{\partial X_{i'-m, j'-n}^{l+1}}{\partial X_{i', j'}^l} = \frac{\partial}{\partial X_{i', j'}^l} \left(\sum_{m'} \sum_{n'} W_{m', n'}^{l+1} f(X_{i'-m+m', j'-n+n'}^l) + b^{l+1} \right)$$

Todas las derivadas parciales resultan en cero excepto donde $m = m'$ y $n = n'$, $f(X_{i'-m+m, j'-n'+n}^l) = f(X_{i', j'}^l)$ y $W_{m', n'}^{l+1} = W_{m, n}^{l+1}$ en la región de salida relevante:

$$\frac{\partial X_{i'-m, j'-n}^{l+1}}{\partial X_{i', j'}^l} = \frac{\partial}{\partial X_{i', j'}^l} \left(W_{m', n'}^{l+1} f(X_{0-m+m', 0-n+n'}^l) + \dots + W_{m, n}^{l+1} f(X_{i', j'}^l) \right. \\ \left. + \dots + b^{l+1} \right)$$

$$\frac{\partial X_{i'-m, j'-n}^{l+1}}{\partial X_{i', j'}^l} = \frac{\partial}{\partial X_{i', j'}^l} \left(W_{m,n}^{l+1} f \left(X_{i', j'}^l \right) \right)$$

$$\frac{\partial X_{i'-m, j'-n}^{l+1}}{\partial X_{i', j'}^l} = W_{m,n}^{l+1} f' \left(X_{i', j'}^l \right)$$

Sustituyendo, se obtiene:

$$\frac{\partial E}{\partial X_{i', j'}^l} = \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} \delta_{i'-m, j'-n}^{l+1} W_{m,n}^{l+1} f' \left(X_{i', j'}^l \right)$$

El término $\sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} \delta_{i'-m, j'-n}^{l+1} W_{m,n}^{l+1}$ puede verse como un filtro girado 180 grados realizando una convolución con la matriz δ .

Capa Max-pooling

La capa pooling no tiene ningún peso, sino que solo reduce el tamaño de la entrada en función de las operaciones espaciales realizadas. En el paso directo, la operación pooling conduce a la conversión de una matriz o un vector a un solo valor escalar.

En el max pooling la neurona ganadora se recuerda. Cuando se necesita hacer la propagación hacia atrás, el error completo se pasa a la neurona ganadora.

En el average pooling, un bloque pooling $n \times n$ durante la propagación hacia atrás divide el valor escalar total por $1/n \times n$ y lo distribuye equitativamente a través del bloque.

Continuación del ejemplo

Una vez que terminamos de realizar la convolución, aplicamos una función de activación que permitirá a la CNN aprender patrones más complicados en general. Usamos ReLU:

convolution output for red region
and specified filter

-2	8	10	10
8	3	24	24
10	17	45	45
10	17	45	45

apply ReLU nonlinearity:
make all negative
numbers zero



after applying ReLU nonlinearity

0	8	10	10
8	3	24	24
10	17	45	45
10	17	45	45

El último paso es el pooling. Este paso da como resultado una reducción del tamaño de la representación. Por lo general, elegimos una ventana pooling de la misma dimensión que el filtro.

Elegimos un filtro de 2×2 , por lo que elegimos una ventana pooling de 2×2 .

ReLU output with 2×2 pooling
regions colored

0	8	10	10
8	3	24	24
10	17	45	45
10	17	45	45

apply 2×2 max pooling:
choose the biggest
number in each 2×2 box



output of max pooling

8	24
17	45

Fin!

Estas notas están basados en el trabajo de Rachel Raelos y el texto de Deep Learning for NLP and Speech Recognition de Kamath, Liu y Whitaker.