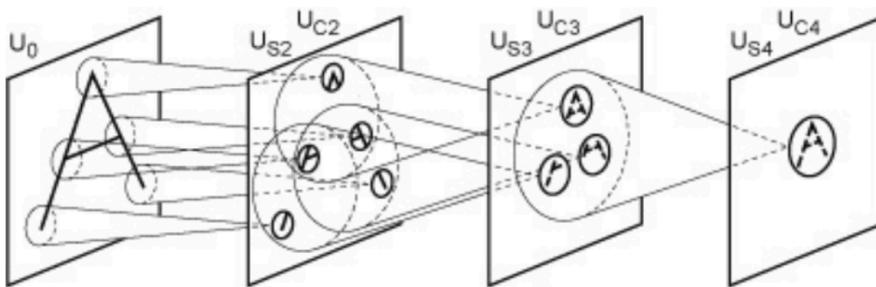


# Clase 3

## Introducción

- Las redes neuronales convolucionales (CNN) están inspiradas en el sistema visual de los humanos.
  - Las neuronas de la corteza visual se organizan en forma jerárquica, donde las capas principales se concentran en capturar formas simples y las capas posteriores se concentrar en patrones más complejos y abstractos.
  - El campo receptivo es el área de la imagen que, si cambia, cambia la salida de la neurona. Es decir, el campo receptivo de las neuronas son locales.
  - Las neuronas en la capa principal tienen un campo receptivo más chico que las neuronas posteriores. Esto permite que las neuronas de la capa principal se concentren en formas simples, de bajo nivel de abstracción, como vértices y bordes, y las neuronas de la capas posteriores se concentren en objetos más abstractos como, en la imagen, una letra.



- Las CNN son introducidas por Yann Lecun. Están definidas como redes neuronales que tienen, al menos, una capa convolucional.

## Capas convolucionales

### Convolución

- Matemáticamente, una convolución es la integral del producto entre una matriz  $X$  y una matriz  $W$ . La matriz  $W$  se conoce como filtro, y es una matriz que vamos a ir desplazando para obtener una nueva matriz a partir del producto interno de las matrices.

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

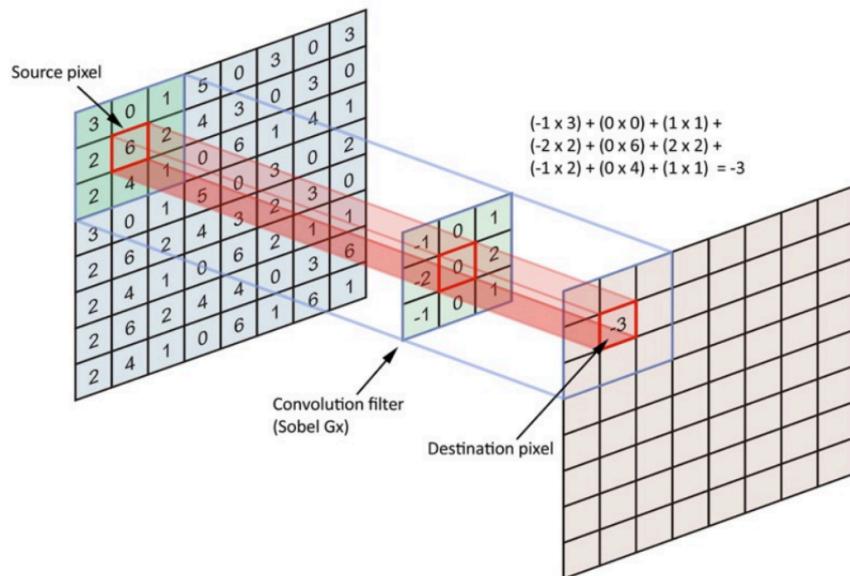
Señal convolucionada      Señal de entrada      Filtro

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a)$$

- En otras palabras, una convolución es una multiplicación de señales que nos devuelve una nueva señal convolucionada.

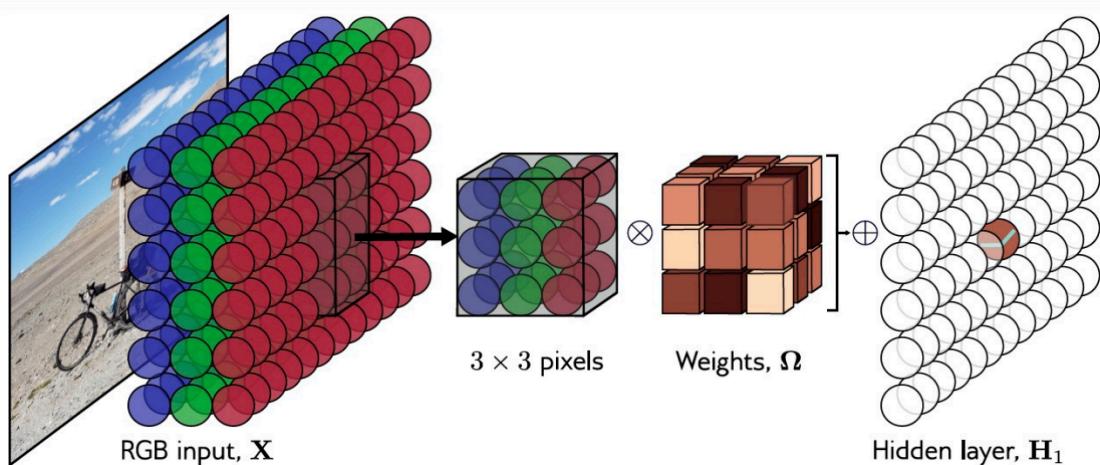
## Kernels (filtros)

- Cuando aplicamos un kernel (filtro) sobre nuestra imagen, lo que obtenemos es el producto interno entre ambas matrices. Si convolucionamos (trasladamos) el kernel a lo largo de la matriz, obtenemos el producto interno entre el kernel y la ventana de la matriz que estamos multiplicando, y con eso construimos una **representación (feature map)** de la imagen original.



- Una capa convolucional se ve de la siguiente manera.

1. Tenemos un imagen de entrada con tres canales (porque es a color), la cual multiplicamos por una matriz de pesos (o kernel) de  $3 \times 3$  (se puede elegir otro tamaño si se quiere).
2. El resultado es el producto interno entre el kernel y la matriz de entrada. Ese producto interno es nuestra neurona convolucional (esfera de color rojo).
3. Aplicamos la función de activación a cada una de las neuronas, es decir, a cada uno de los productos internos.

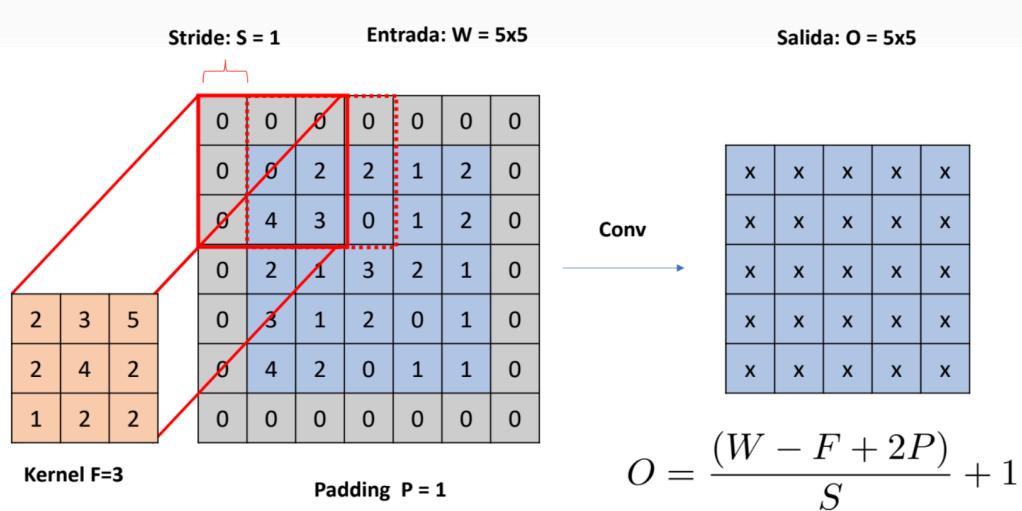


- Una red neuronal convolucional construye representaciones (feature maps) de la imagen original que nos permiten discriminar más facilmente entre clases.
  - En una CNN, los parámetros son los kernels (filtros). Es decir, al entrenar la red neuronal lo que vamos a aprender van a ser los filtros que utilizamos para construir las representaciones.
    - Las neuronas de un feature map comparte los mismos pesos, es decir, el mismo kernel.
    - La cantidad de feature maps que tenemos como salida de nuestra capa convolucional es equivalente a la cantidad de kernels que tenemos.
    - Por ejemplo, podemos tener dos kernels, un kernel que extraiga objetos acomodados verticalmente y otra que extraiga objetos horizontales. Ahí tendríamos dos feature maps.

## Hiper-parámetros

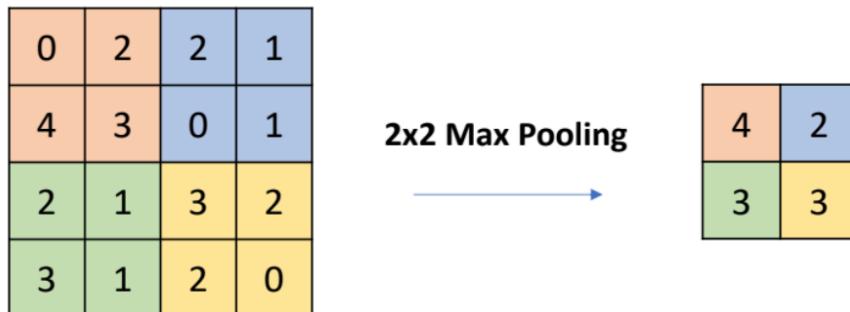
- **Tamaño del kernel.** Define de que tamaño es el filtro que le aplicamos a nuestra imagen.
- **Stride.** Define el salto que damos cuando movemos el filtro.
  - Cuanto más grande sea el stride, más chica es la salida.

- **Padding.** Agrega bordes alrededor de la imagen. Con eso podemos evitar que la salida tenga menor tamaño que la entrada.



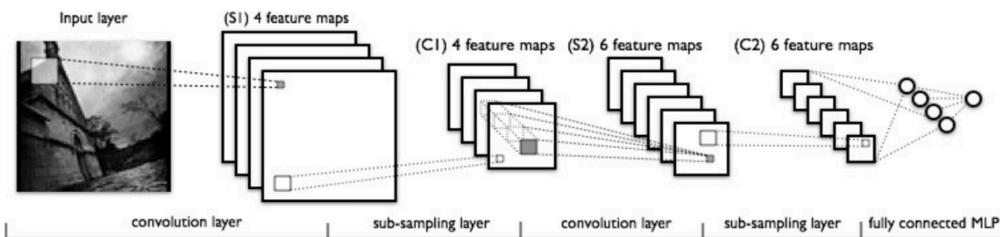
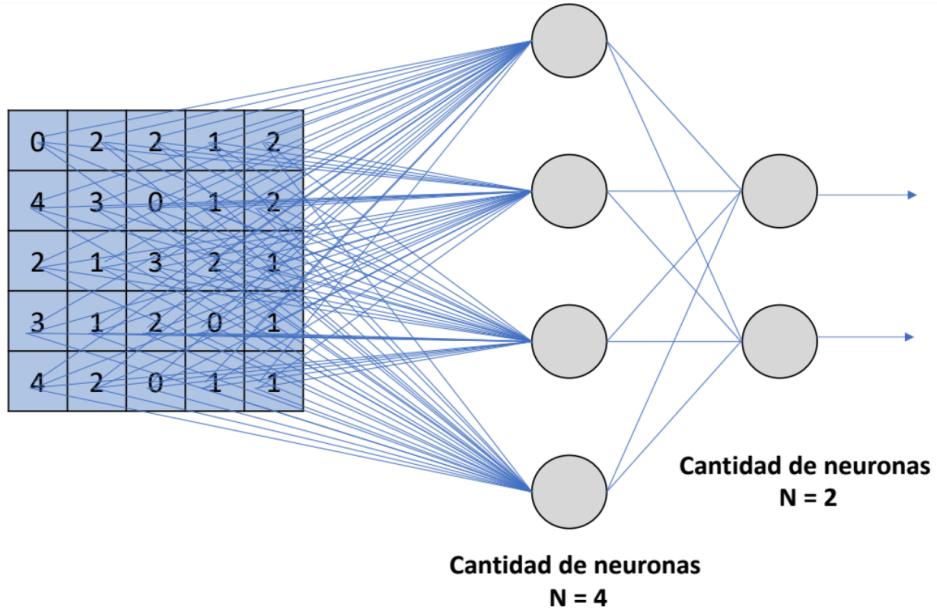
## Capas de pooling

- El objetivo de las capas de pooling es reducir la dimensión de los feature maps.
- Hay varios tipos de pooling. Por ejemplo, un max pooling de 2x2 va a tomar matrices de 2x2 dentro de nuestro feature map y se va a quedar con el máximo valor, como podemos ver en la imagen.



## Redes Neuronales Convolucionales

- Las redes neuronales convolucionales combinan capas convolucionales con capas de pooling para ir obteniendo distintas representaciones de la imagen original.
- Al final, las representaciones se convierten en vectores (de menor dimensión gracias al pooling) y se pasan por un MLP para hacer la clasificación.



## Tipos de redes neuronales convolucionales

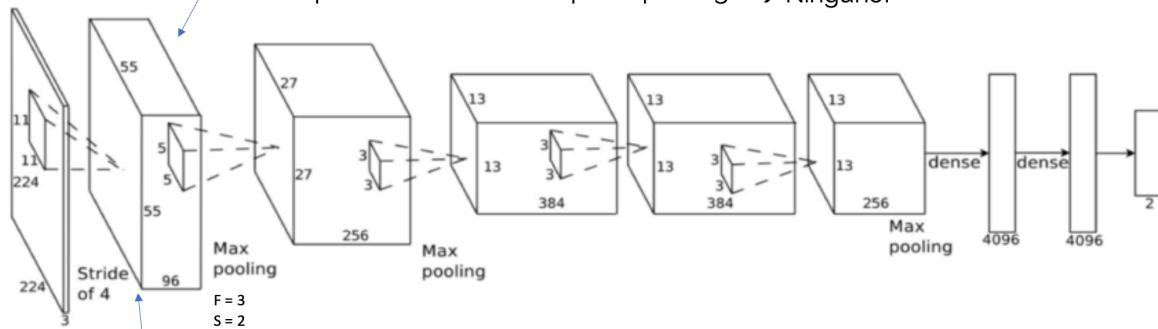
### LeNet

- **Recordar.** No se pueden procesar imágenes de cualquier tamaño con esta red neuronal. Si aumentamos el tamaño de la imagen de entrada va a cambiar el tamaño de los features maps finales, es decir, los que utilizamos como inputs del MLP. Ahora, el MLP tiene un número fijo de pesos que depende de la cantidad de inputs que recibe, por lo que nos limita que imágenes podemos clasificar con una red neuronal LeNet.
  - Por ejemplo, si entrenamos con imágenes de 32x32, el MLP que utilizamos para clasificar tiene una cantidad de parámetros fija, asumiendo que recibe 400 inputs de entrada. Si quisieramos clasificar una imagen de 40x40, necesitaríamos más pesos, porque tendríamos más de 400 inputs de entrada, lo que no es posible porque el modelo ya fue entrenado con esos pesos.

### AlexNet

- Cada capa convolucional tiene más kernels (es decir, más feature maps) que en la red LeNet.
  - Tiene mucho más parámetros: 60 millones.

- En total, las capas convolucionales solo aportan aproximadamente 2.3M de parámetros. Las capas fully-connected aportan 58M de parámetros.

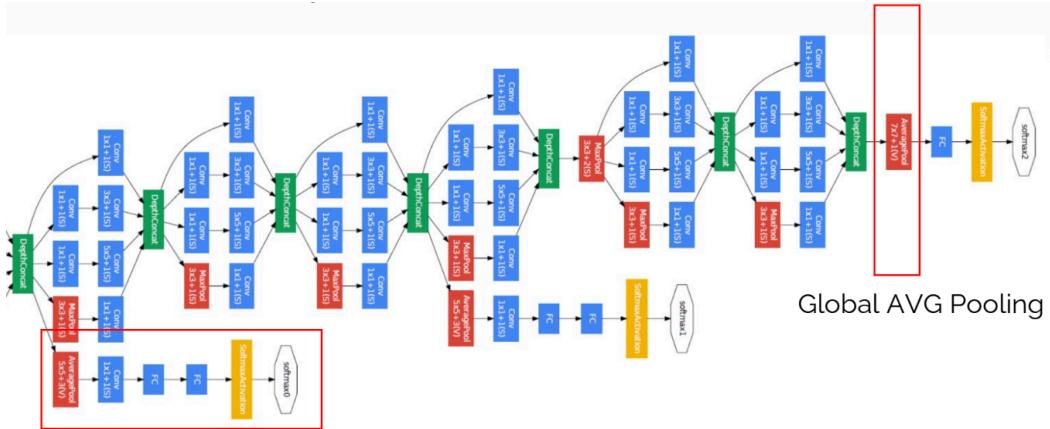


## VGGNet

- Es una red más profunda que, como característica principal, utiliza solo kernels de 3x3 y capas de pooling de 2x2.
- Al definir kernels de 3x3, la cantidad de parámetros que tenemos que estimar se reduce significativamente.

## GoogLeNet (Inception)

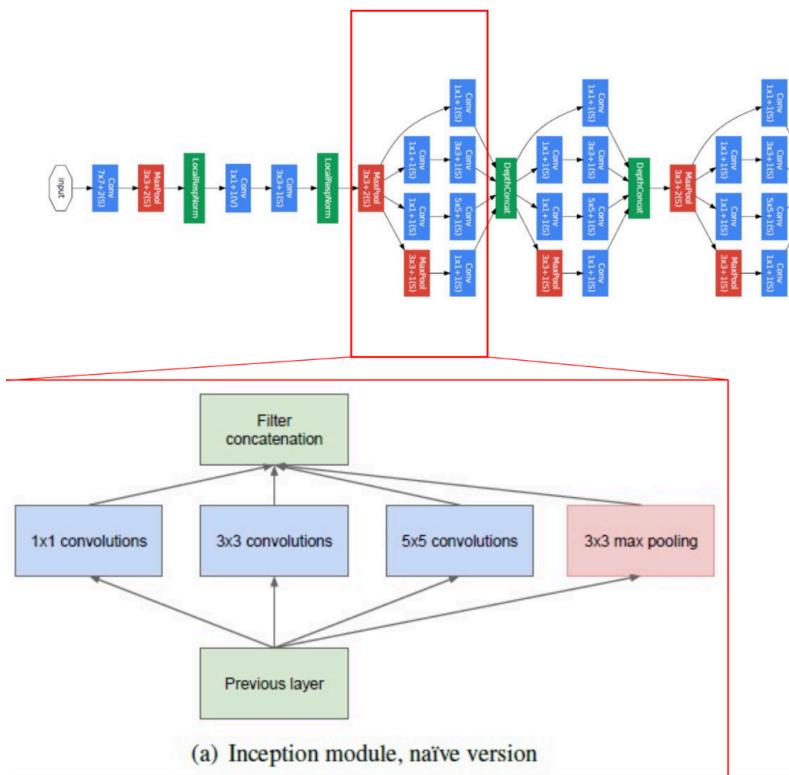
- Innovaciones que presenta este modelo:
  - Utiliza una capa especial llamada **Global Average Pooling**. La diferencia con el resto de las capas de pooling es que no se divide el feature map en ventanas más pequeñas para calcular el promedio de cada ventana, sino que calcula el promedio sobre todos los valores del feature map.  
Como resultado obtenemos un vector con dimensión igual al número de feature maps que tenemos (e.g., si tenemos cuatro feature maps, como resultado obtenemos un vector con cuatro elementos).
    - Esto nos permite reducir significativamente la cantidad de parámetros.
  - Utiliza **clasificadores auxiliares**. Esto quiere decir que la red neuronal toma algunos de los feature maps intermedios, les aplica una capa GAP, y después pasa ese resultado por un MLP.
    - Esto nos permite obtener clasificadores más robustos.
    - Nos ayuda a combatir el problema del gradiente que se desvanece. Esto porque al tener varias salidas, algunas más cerca del principio de la red que otras, vamos a poder ir sumando los gradientes y así evitando que los gradientes sean muy cercanos a cero.



Clasificadores auxiliares

- Utilizan módulos **inception**. A diferencia de las capas convolucionales, en donde los kernels que aplicamos para convolucionar la imagen tienen el mismo tamaño, los módulos inception permiten utilizar kernels de distinto tamaño dentro de una misma capa convolucional.
  - El problema es que ahora los feature maps que obtenemos van a tener distintos tamaños. Para poder agruparlos, vamos a realizar un padding, agregando ceros en los bordes para que todos los feature maps tengan la misma dimensión.

Inception Modules



(a) Inception module, naïve version