

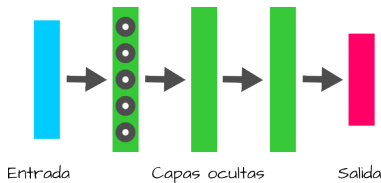


# Arquitectura

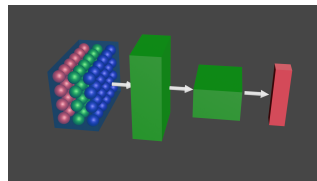
- 1 Arquitectura
- 2 Capa convolucional
- 3 Capa de Unión (*Pooling*)
- 4 Conversión Completamente conectada ↔ Convolucional
- 5 En la práctica

# Características

- Las *Redes Neuronales Convolucionales* (*Convolutional Neural Networks* o CNNs) se especializan en el procesamiento de imágenes.
- Sus capas se organizan en bloques 3D ancho  $\times$  alto  $\times$  profundidad



(a) FFNN



(b) CNN

**Figura:** Las capas son 1D en una FFNN y 3D en una CNN. En esta la capa de entrada contiene la imagen (ancho  $\times$  alto  $\times$  profundidad) donde sus tres canales RGB darían la profundidad.

# Tipos de capas

Hay varios tipos de capas que se pueden apilar para formar la arquitectura de una red convolucional:

**Convolución** Ejecuta la operación de convolución sobre la capa anterior con tantos filtros como sean especificados y puede sumar un sesgo, su salida es otra capa 3D.

**Activación** Opcionalmente se puede aplicar una función de activación sobre cada valor en la capa resultante.

**Unión (pooling)** Realiza una operación de *submuestreo* sobre las dimensiones espaciales (ancho × alto)

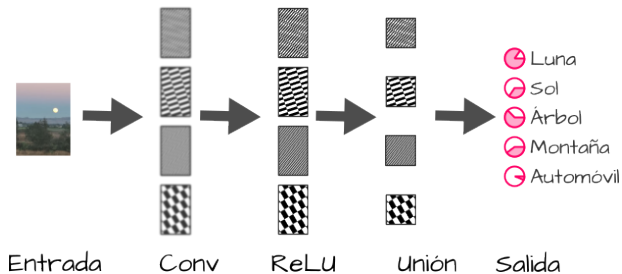
**No tiene parámetros entrenables.**

**Completamente conectada (fully connected)** Cada neurona de esta capa se encuentra conectada con todas las neuronas de la capa anterior y puede agregarse un sesgo.

# Ejemplo de especificación

Una arquitectura se especifica listando los tipos de capas que contiene.

Entrada → Conv → ReLU → Unión → Completa



**Figura:** Para visualizar los valores de activación en cada capa se puede usar una imagen 2D por canal. Una salida típica es un capa donde cada neurona representa la probabilidad de que algún objeto se encuentre en la imagen.

## Capa convolucional

- 1 Arquitectura
- 2 **Capa convolucional**
- 3 Capa de Unión (*Pooling*)
- 4 Conversión Completa
- 5 En la práctica

# Temas

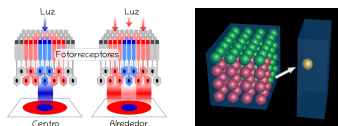
## 2 Capa convolucional

- Entrada
- Salida
- Alternativas





# Filtro

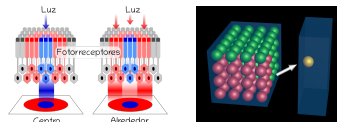


(a) ¿Recuerdan esta imagen?

(b) Filtro convolucional

- El valor de activación de una neurona en la capa siguiente depende del producto escalar entre las neuronas de la capa anterior en su *campo receptivo* y los pesos que la conectan con ellas más el valor del sesgo.
- Los valores de los pesos de las conexiones definen un *filtro* que será aplicado sobre la capa anterior.

# Filtro



(a) ¿Recuerdan esta imagen?

(b) Filtro convolucional

- El **ancho** y **alto** de un *filtro* puede variar, pero la **profundidad** siempre es igual a la profundidad de la capa anterior.
- Se suelen elegir valores impares entre 3 y 9.
- Por cada filtro se obtiene un nivel de profundidad para la capa siguiente.

# Temas

## 2 Capa convolucional

- Entrada
- Salida
- Alternativas

# Profundidad (*Depth*)

- La *profundidad* indica el número de filtros que deseamos aplicar sobre una capa.
- Esto determina la profundidad de la capa siguiente.
- El conjunto de neuronas que son activadas por la misma región en la capa anterior forman una *columna en profundidad* o *fibra* y son el resultado de aplicar filtros diferentes sobre la misma región.

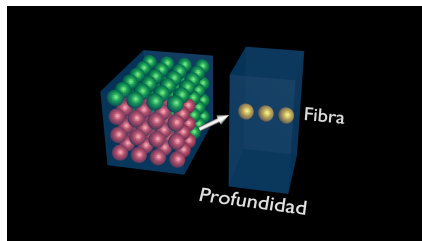
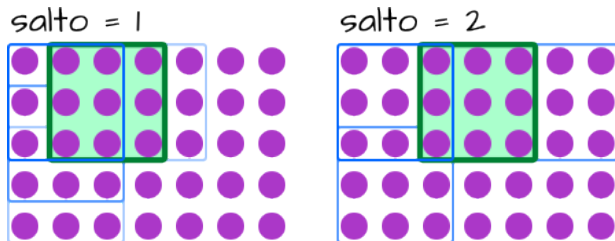


Figura: Columna o fibra.

# Salto (*Stride*)

- El *salto* indica la distancia entre campos receptivos consecutivos tanto vertical como horizontalmente.
- En la práctica se suelen elegir saltos de 1 ó 2.



**Figura:** Salto = 1 corresponde a la operación de convolución. Salto = 2 omite el cálculo de ciertos valores, reduciendo el número de neuronas necesarias para guardar los valores de activación en la capa siguiente.

# Relleno (*Padding*)

- Se utiliza para rellenar el volumen de entrada.
- Se pueden utilizar las mismas opciones que vimos en la operación de convolución:
  - Relleno con ceros.
  - Reflexión.
  - Cíclico.
  - No se suele usar extrapolación.

# Salida

Las dimensiones cada lado del volumen de salida están dadas por:

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

donde:

- W es la longitud del lado de entrada,
- F el del campo receptivo, núcleo o *kernel*,
- S es el salto,
- P número de valores agregados alrededor de la entrada para este lado.

# Temas

## 2 Capa convolucional

- Entrada
- Salida
- Alternativas



# Capas conectadas localmente

- Si se desea reconocer características en la imagen que sí dependen de la posición, es posible buscar filtros diferentes para campos receptivos en posiciones distintas.
- En este caso cada neurona en la capa siguiente tiene un conjunto de pesos único que la conecta con su campo receptivo.
- Por el contrario, en una capa convolucional, todas las neuronas en la misma profundidad en la capa siguiente comparten los valores de los pesos que las conectan con su campo receptivo en la capa anterior.

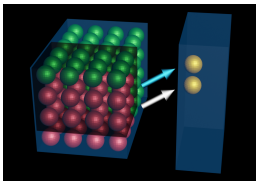


Figura: Los conjuntos de pesos para ambas neuronas difieren.

# Convoluciones dilatadas

- La *dilatación* permite aplicar filtros sobre celdas con espacios entre ellas.
- Los filtros dilatados permiten analizar información en regiones más grandes de la entrada con menos capas en la red.

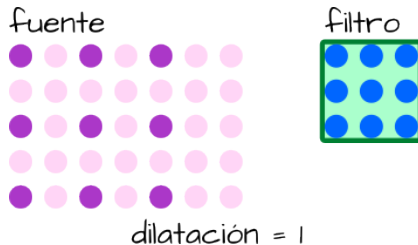


Figura: Columna o fibra.

## Capa de Unión (*Pooling*)

- 1 Arquitectura
- 2 Capa convolucional
- 3 Capa de Unión (*Pooling*)
- 4 Conversión Completamente conectada ↔ Convolucional
- 5 En la práctica

# Capa de unión

- También se dice que aplican filtros, pero su objetivo es **reducir** el tamaño de la representación.
- Las funciones típicas a utilizar son:
  - Máximo Toma el valor más grande en el área.
  - Promedio Toma el promedio de los valores en el área.
  - Promedio de potencias <sup>[1]</sup>

$$f(X) = \sqrt[p]{\sum_{x \in X} x^p}$$

[1] <https://pytorch.org/docs/stable/generated/torch.nn.LPPool1d.html#torch.nn.LPPool1d>

# Max, Prom y Potencias



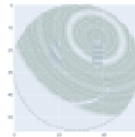
(a) Original



(b) Max



(c) Promedio



(d) Promedio de potencias

# Ejemplo

- Valores típicos de sus parámetros son:
  - $F = 3, S = 2$  (unión superpuesta)
  - $F = 2, S = 2$
  - Valores más grandes pierden mucha información.
  - Alternativamente se evita el uso de esta capa usando algunas capas convolucionales con saltos  $S$  grandes.

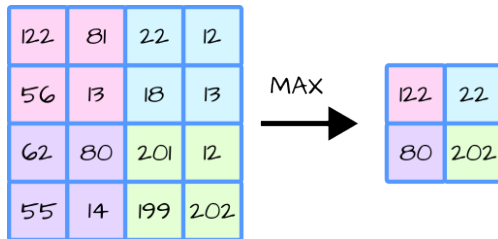


Figura: ¡Los más grandes a la alberca!

# Entrenamiento

- En el caso de la unión máxima (*max pooling*), al calcular el gradiente sólo se obtiene el gradiente de la componente que tuvo el valor más grande.
- El índice del peso que conecta con el valor más grande se puede guardar desde la ejecución de la alimentación hacia adelante.

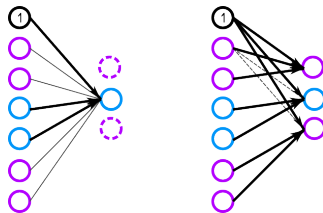
## Conversión Completamente conectada ↔ Convolucional

- 1 Arquitectura
- 2 Capa convolucional
- 3 Capa de Unión (*Pooling*)
- 4 Conversión Completamente conectada ↔ Convolucional
- 5 En la práctica



# Convolutional → Completamente conectada

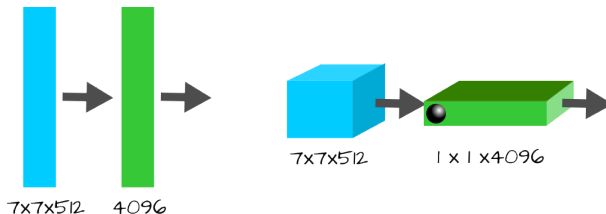
- Convertir la conectividad local en total: los pesos de las conexiones entre neuronas que no estaban conectadas antes a la neurona en la capa siguiente se fijan como cero.
- Los valores de los pesos se repiten para todas las neuronas, aunque las posiciones de estos cambian en las matrices, según la posición en que se aplicaba el filtro.



**Figura: Izq:** La conectividad local se convierte en total, fijando en ceros los pesos las conexiones antes inexistentes. **Der:** Los valores de los pesos se repiten cuando los filtros se comparten. **Nota:** Se usó en 1D y no todos los pesos se dibujaron por simplicidad.

# Completamente conectada → Convolucional

- Acomodar la capa de entrada  $F$  en un volumen 3D con ancho y alto  $W$  iguales.
- Usar  $V$  filtros del mismo ancho y alto que una rebanada de profundidad, sin relleno  $P = 0$  y con salto  $S = 1$ .



**Figura:**  $V$  filtros del tamaño de la entrada, sin relleno es equivalente a una capa completamente conectada.

# Aplicación

- Una aplicación es trabajar sobre imágenes aumentadas o de tamaño distinto a las imágenes sobre las que se entrenó a la red.
  - Se convierte a la red de modo que todas las capas sean convolucionales.
  - Se aplica la red completa sobre parches de la imagen más grande.
  - Se toma como clase ganadora a la que obtenga la calificación más alta al tomar el promedio sobre todas las aplicaciones.

## En la práctica

- 1 Arquitectura
- 2 Capa convolucional
- 3 Capa de Unión (*Pooling*)
- 4 Conversión Completamente conectada ↔ Convolucional
- 5 En la práctica

# CNNs famosas I

- Usualmente se usa como partida alguna arquitectura conocida, con pesos preentrenados y sólo realizan ajustes finos para adaptar a algún problema concreto.
- Algunas de estas arquitecturas son:
  - LeNet** 1er aplicación de Yann LeCun al problema de reconocer dígitos de códigos postales.
  - AlexNet** Introduce el uso de varias capas convolucionales seguidas sin capas de unión entre ellas.
  - ZF Net** Usa saltos y filtros pequeños en la 1a capa para mejorar su desempeño.


# CNNs famosas II

- GoogLeNet** Introduce el *módulo de origen* (*inception module*), que aplica: 1 convolución  $1 \times 1$  para reducir la profundidad de la capa de entrada y luego aplica convoluciones y uniones  $3 \times 3$  y  $5 \times 5$  en paralelo de modo que la capa siguiente contiene rebanadas de profundidad con la misma área correspondiendo a variantes de estas operaciones. [2]
- VGGNet** Demostró que la profundidad de la red afecta críticamente el desempeño de esta.
- ResNet** Introduce el uso de **capas residuales** (aquellas que introducen saltos de capas) en las CNNc.

---

[2] <https://valentinaalto.medium.com/understanding-the-inception-module-in-googlenet-2e1b7c406106>

# Referencias I

-  Stanford University (2022). *Convolutional Neural Networks for Visual Recognition*. English. Stanford University. URL: <https://cs231n.github.io/convolutional-networks/>.

# Licencia

Creative Commons  
Atribución-No Comercial-Compartir Igual

