Redes Convolucionales (CNN): Fundamentos y Práctica

Antonio Falcó y Juan Pardo

Introducción a la Inteligencia Artificial

Lectura 3

Objetivos de la sesión

- Comprender qué es una **red convolucional (CNN)** y por qué se usa en visión artificial.
- Ver cómo una CNN aprende **automáticamente filtros** que detectan bordes, texturas y formas.
- Explorar las partes básicas de una CNN:
  - Capa **convolucional** (extracción de rasgos).
  - Capa de **activación** (introduce no linealidad).
  - Capa de **pooling** (reduce tamaño y resalta lo importante).
  - Capa **densa** (toma decisiones).
- Entender cómo las CNN **aprenden a partir de imágenes etiquetadas** mediante entrenamiento.
- Probar en el cuaderno de Jupyter un modelo simple con `Keras/TensorFlow` o `PyTorch`.

Objetivo final

Ser capaces de reconocer cómo una CNN transforma una imagen en una serie de representaciones cada vez más abstractas hasta clasificarla.

El modelo general de una red neuronal

Idea principal: Toda red neuronal, ya sea una CNN o cualquier otro tipo, puede verse como una **función** que transforma una entrada (imagen, texto, sonido) en una salida o **etiqueta**.

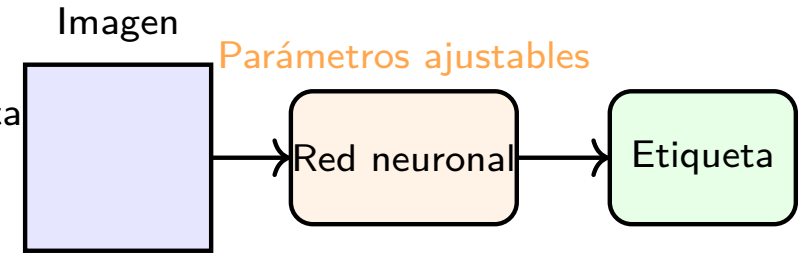
$$f(\text{Imagen, Parámetros}) = \text{Etiqueta}$$

Qué significa esto

- **Imagen:** los datos que damos al modelo (por ejemplo, una foto de un gato o un perro).
- **Parámetros:** los valores internos de la red (pesos y filtros) que determinan cómo procesa la imagen.
- **Etiqueta:** el resultado esperado (por ejemplo, "gato" o "perro").

Objetivo del aprendizaje:

- Encontrar los valores de los **parámetros** que hacen que $f(\text{Imagen, Parámetros})$ dé la etiqueta correcta el mayor número de veces posible.
- Es decir, **ajustar** la red para que sus predicciones sean cada vez más precisas.
- Este ajuste se realiza automáticamente mediante el entrenamiento.



Resumen

El entrenamiento busca que la función f **aprenda** los parámetros ideales para transformar correctamente cada imagen en su etiqueta correspondiente.

Los parámetros de una red neuronal

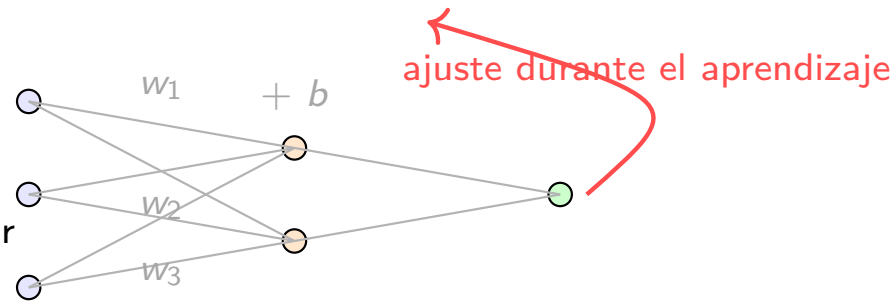
Idea principal: Los **parámetros** (también llamados **pesos** y **sesgos**) son los valores internos que determinan cómo la red combina la información para tomar decisiones.

Qué son los parámetros

- Cada conexión entre neuronas tiene un **peso** que indica su importancia.
- Cada neurona puede tener un **sesgo** que ajusta el nivel mínimo de activación.
- Durante el entrenamiento, estos valores se van modificando para reducir el error del modelo.

Intuición práctica:

- Los pesos son como los **controles de volumen** de una mesa de mezcla: deciden qué señales se amplifican y cuáles se atenúan.
- El sesgo permite desplazar la señal para mejorar el ajuste final.
- El entrenamiento ajusta estos "controles" automáticamente hasta obtener el mejor sonido (la mejor predicción).



Resumen

Los pesos y sesgos son los **parámetros ajustables** que permiten a la red aprender. El entrenamiento busca sus valores ideales para minimizar el error en las predicciones.

Cómo cambian los parámetros durante el entrenamiento

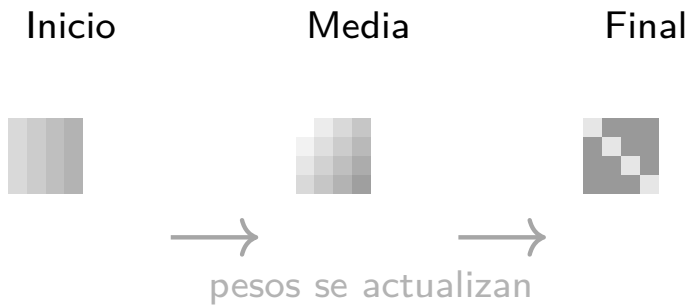
Idea principal: Al inicio del entrenamiento, los **pesos** de la red son valores aleatorios. A medida que la red aprende, esos valores se ajustan poco a poco para que las predicciones sean cada vez más correctas.

Evolución de los filtros (conceptualmente)

- Al principio, los filtros no "ven" nada concreto (valores aleatorios → ruido).
- Después de varias **épocas**, comienzan a detectar bordes o texturas.
- En capas más profundas, los filtros combinan rasgos simples para reconocer partes de objetos.

Analogía intuitiva:

- Es como ajustar el enfoque de una cámara: al principio todo está borroso, pero poco a poco la imagen se vuelve nítida.
- Cada ajuste de pesos es un pequeño paso hacia una mejor visión del problema.



Resumen

El entrenamiento es un proceso de **ajuste progresivo de los pesos**: de valores aleatorios a filtros que reconocen patrones visuales.

¿Qué es una red convolucional (CNN)?

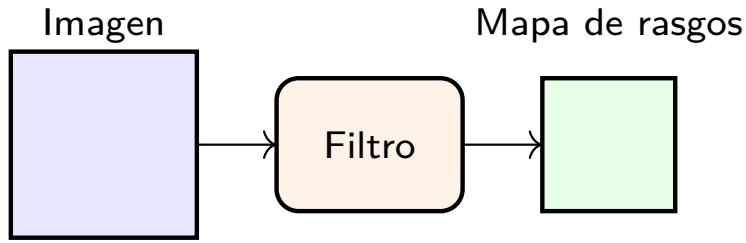
Idea principal: Una **red convolucional** es un tipo de red neuronal que puede "ver" imágenes. Imita la forma en que el cerebro analiza la información visual: primero detecta bordes, luego formas simples y, al final, objetos completos.

Cómo funciona, en palabras sencillas

- Una CNN **recorre la imagen con filtros** (pequeñas ventanas) que buscan patrones.
- Cada filtro se **entrena solo** para detectar algo útil: un borde, una textura, un color, etc.
- Las capas más profundas combinan esos rasgos para reconocer formas completas.

Comparación intuitiva

- Como el ojo humano, que detecta primero contrastes y luego reconoce figuras.
- Cada capa de la red "mira" la imagen de manera diferente.



Resumen

Una CNN aprende automáticamente a identificar patrones visuales sin que nadie le diga qué filtros usar.

¿Por qué usamos redes convolucionales en visión artificial?

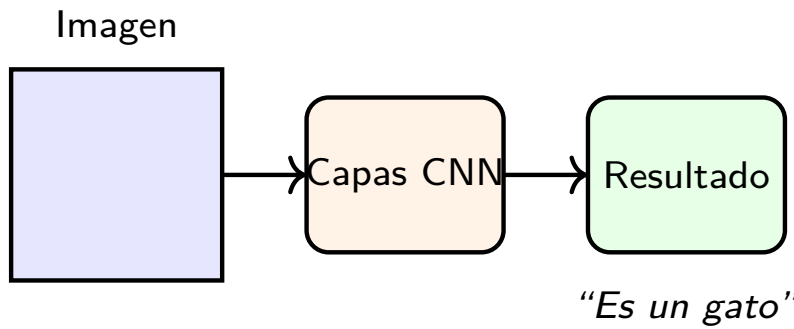
Motivación: Las imágenes son enormes (millones de píxeles) y cada píxel tiene poco sentido por sí solo. Las CNNs aprenden a **organizar la información visual** de forma jerárquica y eficiente.

Ventajas principales

- **Detectan patrones locales**: reconocen bordes, esquinas, texturas.
- **Reutilizan filtros**: el mismo filtro se aplica en toda la imagen.
- **Reducción de parámetros**: mucho menos costosas que redes densas normales.
- **Escalabilidad**: funcionan con fotos, vídeos, rayos X o imágenes de satélite.

Ejemplos de uso

- Clasificar imágenes (gato vs. perro).
- Detectar objetos o rostros.
- Analizar radiografías o resonancias.
- Reconocimiento de escritura y señales de tráfico.



Mensaje final

Las CNNs son el **pilar de la visión por computador**: permiten que los sistemas de IA **entiendan imágenes** como los humanos.

Cómo una CNN aprende sus propios filtros

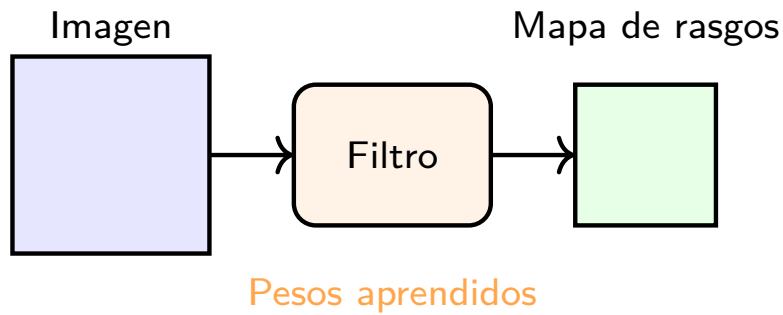
Idea clave: En una CNN, los filtros (*kernels*) no se eligen a mano: la red **aprende automáticamente** qué patrones son importantes durante el entrenamiento.

Durante el aprendizaje

- La CNN recibe muchas imágenes (por ejemplo, gatos y perros).
- Compara su salida con la etiqueta correcta.
- Si se equivoca, ajusta ligeramente los valores de los filtros (*pesos*).
- Tras miles de ejemplos, los filtros "aprenden" a detectar patrones útiles.

Comparación con filtros clásicos:

- Filtros de Sobel o Gauss son fijos y diseñados por humanos.
- Filtros de una CNN se **actualizan solos** para mejorar la precisión.



Resumen

La CNN no necesita saber qué es un "borde" o una "textura": aprende esos filtros al intentar reconocer correctamente las imágenes.

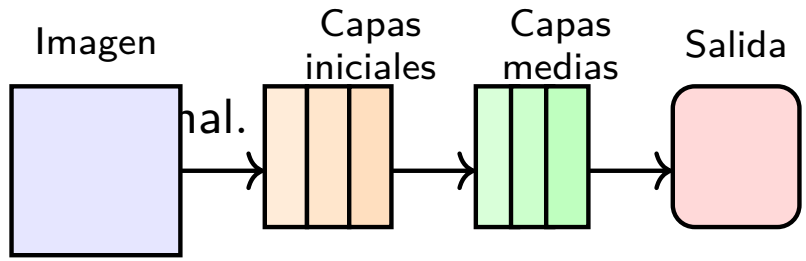
Qué aprende cada capa de una CNN

Cada capa ve la imagen de forma distinta:

- Las primeras capas aprenden filtros muy simples: **bordes y contrastes**.
- Las capas intermedias aprenden combinaciones: **texturas y patrones repetidos**.
- Las últimas capas combinan todo eso para detectar **formas completas u objetos**.

Evolución de la representación:

Imagen $\rightarrow$ bordes $\rightarrow$ texturas $\rightarrow$ formas $\rightarrow$ nal.



- **Capa 1:** detecta líneas y contornos.
- **Capa 2:** combina bordes para ver texturas (pelo, madera, agua).
- **Capa 3:** combina texturas en partes de objetos (ojos, patas, hojas).

Analogía

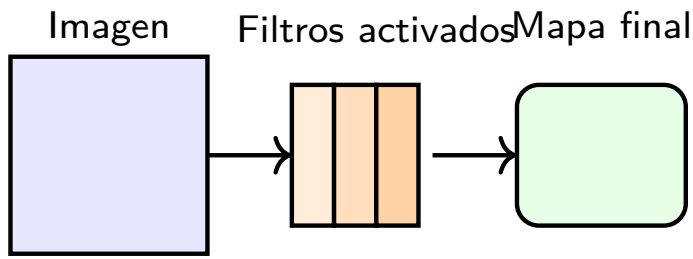
Las primeras capas son como "ojos" que detectan rasgos simples; las últimas, como "neuronas del cerebro" que reconocen conceptos.

Ejemplo visual: activaciones de filtros

Durante el entrenamiento: cada filtro aprende a activarse (encenderse) cuando "ve" un patrón que reconoce.

Interpretación visual:

- **Filtro 1:** responde a bordes horizontales.
- **Filtro 2:** responde a bordes verticales.
- **Filtro 3:** responde a texturas (rejillas, sombras).
- **Filtros posteriores:** se activan ante partes del objeto (ojos, ruedas, hojas...).



Conclusión

Las CNN **descubren solas los filtros** que antes diseñábamos a mano, aprendiendo a detectar patrones cada vez más complejos.

La capa convolucional: el ojo de la red

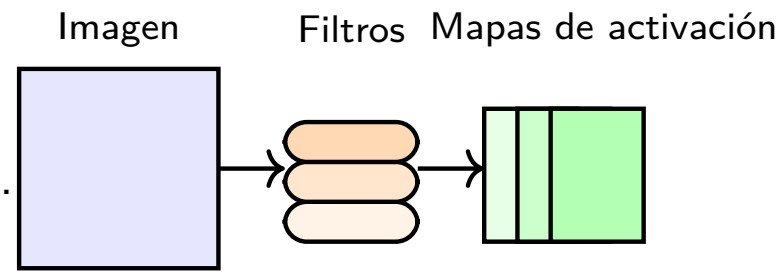
Idea principal: Una **capa convolucional** aplica muchos filtros pequeños sobre la imagen para descubrir patrones como líneas, bordes o texturas.

Qué hace esta capa

- Cada **filtro** (por ejemplo 3×3 o 5×5) recorre la imagen y combina los valores cercanos.
- Cada filtro "mira" la imagen de una manera distinta: detecta una forma o dirección específica.
- El resultado de aplicar un filtro se llama **mapa de activación** o **feature map**.
- La red aprende los valores de estos filtros durante el entrenamiento.

Ejemplo intuitivo:

- Filtro 1 $\rightarrow$ resalta bordes verticales.
- Filtro 2 $\rightarrow$ resalta bordes horizontales.
- Filtro 3 $\rightarrow$ detecta texturas.



Resumen

La capa convolucional actúa como un **conjunto de detectores de patrones**, que analizan cada zona de la imagen buscando coincidencias.

La capa de activación (ReLU): encendiendo lo importante

Idea principal: Después de la convolución, aplicamos una función llamada **ReLU** (*Rectified Linear Unit*) que decide qué valores "activan" realmente a una neurona.

Qué hace ReLU

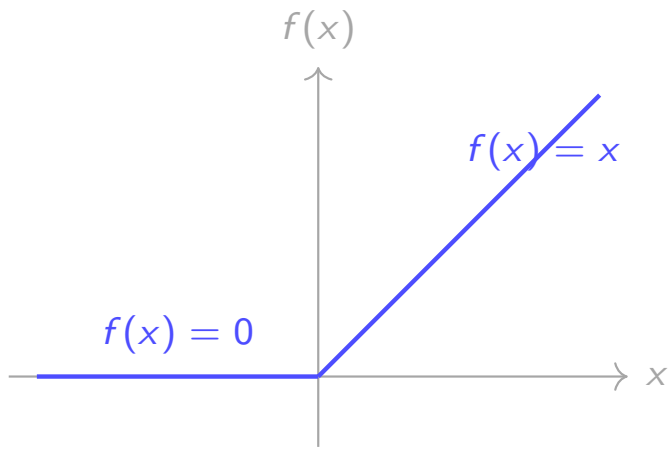
$$f(x) = \begin{cases} x, & \text{si } x > 0, \\ 0, & \text{si } x \leq 0. \end{cases}$$

- Mantiene los valores positivos (zonas donde el filtro "detecta" algo).
- Pone a cero los valores negativos (zonas donde no hay coincidencia).
- Así, sólo las respuestas más relevantes continúan hacia la siguiente capa.

Intuición visual:

- ReLU actúa como un interruptor: deja pasar la señal solo si es positiva.
- Hace que la red sea capaz de aprender **formas complejas**, no solo combinaciones lineales.

La función ReLU **introduce no linealidad**: permite que la red combine patrones simples (bordes, texturas) en estructuras más complejas.



La capa de pooling: resumiendo la información

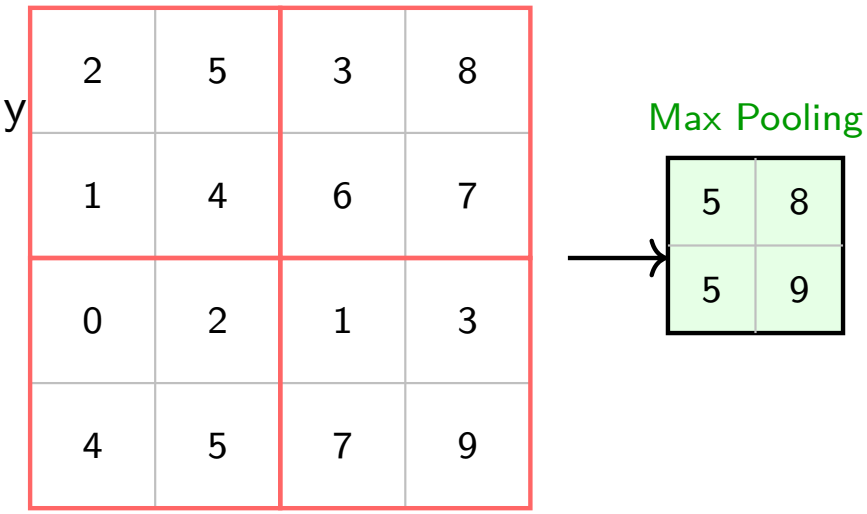
Idea principal: Después de detectar patrones con la convolución y la ReLU, la red reduce el tamaño de los mapas de activación usando una capa llamada **pooling**. El objetivo es quedarse con la información más importante y eliminar los detalles innecesarios.

Qué hace el pooling

- Divide el mapa en pequeños bloques (por ejemplo, 2×2).
- En cada bloque aplica una operación:
  - **Máximo (Max Pooling):** se queda con el valor más alto.
  - **Media (Average Pooling):** calcula el promedio.
- El resultado es un mapa más pequeño, pero con la información esencial.

Intuición:

- Reduce el tamaño $\rightarrow$ menos cálculos y menos riesgo de sobreajuste.
- Hace que la red sea más **robusta a pequeñas variaciones** (un borde desplazado sigue siendo reconocido).



La capa densa: donde la red toma decisiones

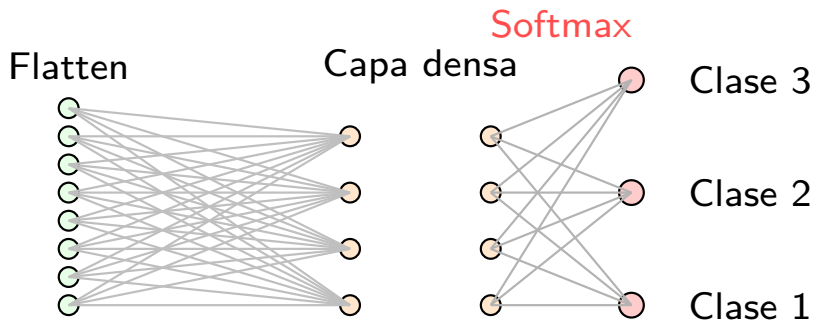
Idea principal: Después de extraer rasgos visuales (bordes, texturas, formas), la red necesita decidir qué representan. Esto lo hace la **capa densa** o **fully connected layer**.

Qué hace

- Cada neurona está conectada con todas las salidas de la capa anterior.
- Combina toda la información para **tomar una decisión global**.
- Es como una "votación ponderada" entre todas las características detectadas.
- Las últimas neuronas usan **Softmax** para convertir los resultados en probabilidades.

Ejemplo intuitivo

- Una neurona puede activarse más si ve "orejas puntiagudas" + "bigotes" → gato.
- Otra se activa si detecta "hocico grande" + "pelaje oscuro" → perro.
- Al final, la capa elige la clase con mayor probabilidad.



Resumen

La capa densa combina todos los rasgos aprendidos y genera la decisión final: "¿Qué objeto hay en la imagen?"

La función Softmax: de números a probabilidades

Idea principal: La última capa de una CNN usa la función **Softmax** para convertir los valores de salida en **probabilidades** que suman 1. Así, la red puede decir: "creo que es un gato con 85 % de seguridad".

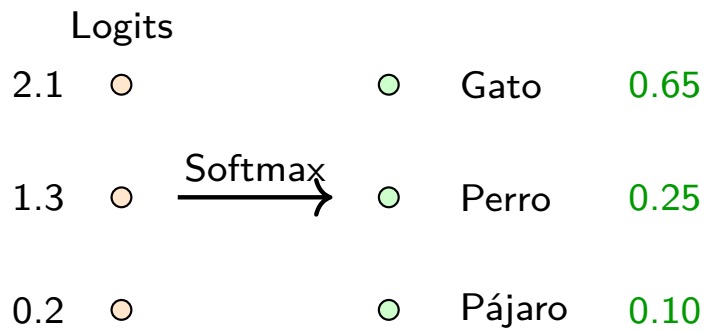
Qué hace Softmax

- Recibe los valores finales de la capa densa (llamados **logits**).
- Los transforma en probabilidades entre 0 y 1.
- El valor más alto indica la clase más probable.

$$\text{Ejemplo: } [2{,}1,\ 1{,}3,\ 0{,}2] \xrightarrow{\text{Softmax}} [0{,}65,\ 0{,}25,\ 0{,}10]$$

Interpretación:

- "El modelo está 65 % seguro de que es un gato, 25 % de que es un perro, y 10 % de que es un pájaro."
- Suma de probabilidades $= 1$.
- Se usa para clasificar entre múltiples categorías.



Resumen

Softmax convierte la salida de la red en una decisión comprensible: **la probabilidad de que la imagen pertenezca a cada clase**.

Estructura completa de una CNN

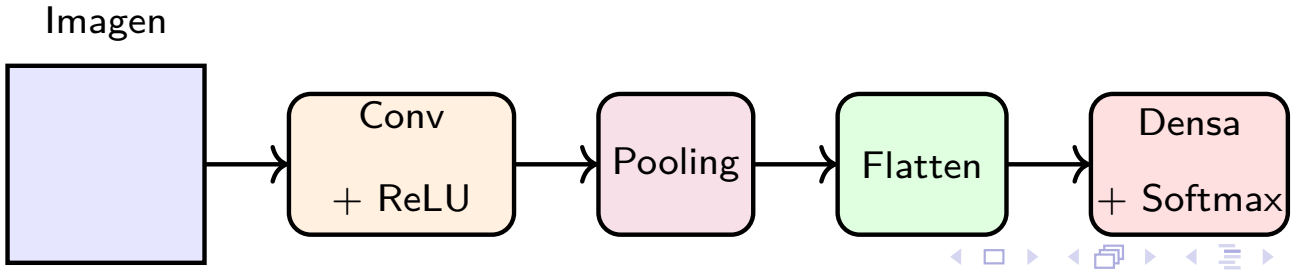
Flujo de información (de píxeles a clase):

$$\text{Imagen} \xrightarrow{\text{Conv}} \text{ReLU} \xrightarrow{\text{Pool}} \cdots \xrightarrow{\text{Flatten}} \text{Densa} \xrightarrow{\text{Softmax}} \text{Clase}$$

- **Conv 2D**: aplica *filtros aprendidos* y crea *mapas de activación*.
- **ReLU**: deja pasar activaciones positivas (no linealidad).
- **Pooling**: reduce tamaño, mantiene lo esencial (p.ej. max 2×2).
- **Flatten**: pasa de $H \times W \times C$ a un vector 1D.
- **Capa Densa**: combina rasgos para decidir.
- **Softmax**: convierte la salida en % de probabilidad por clase.

Mensaje clave

De **rasgos simples** (bordes) a **conceptos** (clases), capa a capa. Todo se aprende automáticamente a partir de datos etiquetados.



Cómo aprende una red convolucional

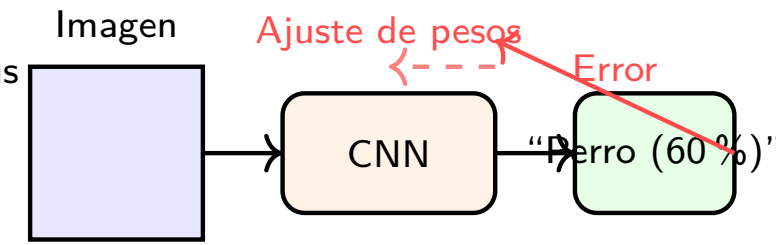
Idea principal: Una CNN aprende ajustando los valores de sus filtros y conexiones para que su respuesta sea cada vez más parecida a la correcta.

Proceso de aprendizaje (entrenamiento)

1. Se muestra a la red una imagen de entrenamiento (por ejemplo, un "gato").
2. La red genera una predicción (por ejemplo, "perro, 60
3. Se calcula el **error** comparando la salida con la etiqueta real.
4. El error se propaga hacia atrás y la red **ajusta sus pesos**.
5. Este proceso se repite miles de veces con muchas imágenes.

Intuición:

- La red mejora poco a poco, aprendiendo de sus errores.
- Filtros que ayudan a reducir el error se refuerzan; los que no, se corrigen.
- Tras muchas iteraciones, la red "reconoce" patrones por sí sola.



Resumen

Una CNN aprende por **retropropagación del error**: ajusta sus filtros hasta que las predicciones son correctas. Es como un estudiante que mejora al recibir retroalimentación.

Nota terminológica

El término técnico en inglés **epoch** representa una pasada completa del conjunto de entrenamiento por toda la red neuronal.

Epochs y conjuntos de datos en el entrenamiento de una CNN

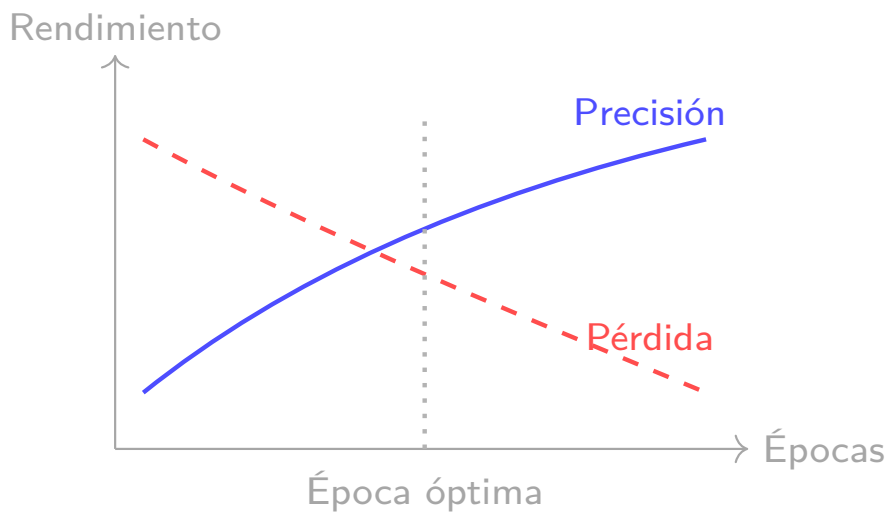
Idea principal: Entrenar una red no consiste en mostrarle una sola vez las imágenes, sino en hacerlo muchas veces y medir cómo mejora su rendimiento.

Estructura del entrenamiento

- Se divide el conjunto total de imágenes en tres partes:
  - **Entrenamiento (train)**: la red aprende ajustando sus pesos.
  - **Validación (val)**: se evalúa durante el entrenamiento (sin aprender).
  - **Prueba (test)**: se usa al final para medir el rendimiento real.
- Cada recorrido completo por todas las imágenes de entrenamiento se llama un **epoch**.
- Cuantas más epochs, más aprende la red (hasta cierto punto).

Durante el entrenamiento:

- En cada epoch, la red ve todas las imágenes y calcula el error.
- Observamos dos curvas:
  - **Pérdida (loss)** $\rightarrow$ debe bajar con el tiempo.
  - **Precisión (accuracy)** $\rightarrow$ debe subir hasta estabilizarse.
- Si la red mejora en "train" pero no en "val", puede estar **sobreajustada**.



Resumen

Una CNN aprende **por iteraciones (epochs)** sobre los datos de entrenamiento. El conjunto de validación sirve para controlar el aprendizaje y evitar el sobreajuste.

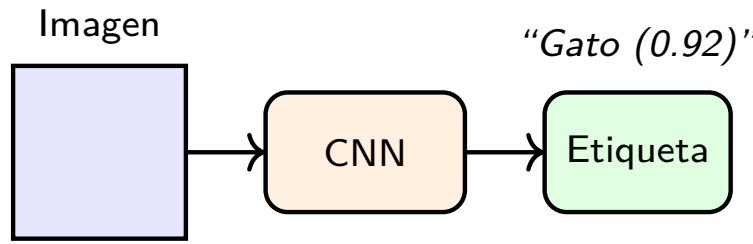
Conclusiones de la lección: Redes Convolucionales (CNNs)

Resumen del recorrido:

1. Partimos de la idea de una red como una **función** $f(\text{Imagen}, \text{Parámetros}) = \text{Etiqueta}$.
2. Vimos cómo las CNN aprenden **filtros** que detectan bordes, texturas y formas.
3. Entendimos las partes básicas:
   - **Capa convolucional:** detecta patrones locales.
   - **ReLU:** mantiene activaciones positivas (no linealidad).
   - **Pooling:** resume la información más relevante.
   - **Densa + Softmax:** toma la decisión final (clasificación).
4. Vimos que el entrenamiento ajusta los **pesos** y **sesgos** para minimizar el error, mejorando la precisión del modelo.

Qué debemos recordar:

- Una CNN aprende de los datos, no de reglas programadas.
- Los filtros evolucionan de ruido → bordes → formas → objetos.
- El proceso de aprendizaje es iterativo (por **epochs**) y controlado por el error.
- Los modelos modernos combinan muchas capas CNN para tareas complejas.



Próxima lección

Redes preentrenadas y transferencia del conocimiento: cómo aprovechar redes ya entrenadas (como VGG, ResNet o MobileNet) para resolver nuevos problemas con pocos datos y menos tiempo de entrenamiento.