Redes Neuronales para Reconocimiento de Imágenes

Data Science (CC3084) - 2026

Redes Neuronales para Reconocimiento de Imágenes

Semestre 02, 2026

Introducción

El reconocimiento de imágenes es una de las tareas centrales de la visión por computadora: enseñar a una máquina a "ver" y entender el contenido de una imagen. Es también el problema donde el deep learning demostró por primera vez su ventaja decisiva sobre los métodos clásicos.

Esta lección continúa el tema de Deep Learning y se concentra en las redes neuronales convolucionales (CNN), la arquitectura que domina la visión por computadora desde 2012.

Las tareas de la visión por computadora

No todo "reconocer una imagen" es lo mismo. Conviene distinguir las tareas, de la más simple a la más compleja:

- Clasificación: asignar una etiqueta a toda la imagen. "Esto es un gato".
- Localización: además de la etiqueta, dibujar un recuadro (bounding box) alrededor del objeto.
- Detección de objetos: encontrar y clasificar varios objetos en la misma imagen, cada uno con su recuadro.
- Segmentación semántica: clasificar cada píxel de la imagen (todo el "cielo", toda la "carretera").
- Segmentación de instancias: separar además cada objeto individual (este gato vs. aquel otro gato).

La clasificación es la puerta de entrada: casi todas las demás tareas se construyen sobre las mismas ideas.

Por qué es un problema difícil

Para una persona, reconocer un gato es instantáneo. Para una computadora no lo es, porque el mismo objeto puede verse de infinitas maneras:

- Variación de punto de vista: el mismo gato desde arriba, de lado o de frente produce píxeles completamente distintos.
- Variación de iluminación: sombras, contraluz, luz cálida o fría cambian todos los valores.
- Deformación y pose: el gato puede estar acostado, saltando o hecho un ovillo.
- Oclusión: parte del objeto puede estar tapada.
- Fondo desordenado y variación intra-clase: hay miles de razas y colores de gato.

Una buena solución debe ser robusta a todas estas variaciones y aun así reconocer la clase correcta.

Cómo ve una computadora una imagen

Una imagen digital no es más que una cuadrícula de números. Cada píxel guarda una intensidad de luz.

- Imagen en escala de grises: una matriz de alto × ancho, con valores típicamente de 0 (negro) a 255 (blanco).
- Imagen a color: tres matrices apiladas, una por canal — rojo, verde y azul (RGB). Su forma es alto × ancho × 3.

Es decir, una foto a color de 224 × 224 píxeles es un tensor de 224 × 224 × 3 = 150 528 números. Para la red, la imagen es exactamente eso: un tensor de números.

El preprocesamiento habitual

Antes de entrar a la red, las imágenes suelen prepararse:

- Redimensionar todas al mismo tamaño (por ejemplo 224 × 224), porque la red espera una entrada fija.
- Normalizar los valores de píxel: pasar del rango [0, 255] a [0, 1] o restar la media y dividir por la desviación estándar. Las redes aprenden mejor con valores pequeños y centrados.
- Aumentar los datos (data augmentation): crear variantes de las imágenes de entrenamiento (rotarlas, voltearlas, recortarlas) para que la red aprenda a ser robusta.

Por qué el MLP no basta para imágenes

La primera idea sería "aplanar" la imagen en un vector y meterla a un perceptrón multicapa (MLP) totalmente conectado. En la práctica funciona muy mal, por tres razones:

- Explosión de parámetros: una imagen de 224 × 224 × 3 aplanada tiene 150 528 entradas. Si la primera capa oculta tiene 1000 neuronas, eso son más de 150 millones de pesos solo en la primera capa. Es inmanejable y se sobreajusta.
- Pierde la estructura espacial: al aplanar, dos píxeles que estaban uno junto al otro quedan tan lejos como dos píxeles en esquinas opuestas. La red ya no sabe qué está cerca de qué.
- No es invariante a la posición: si el MLP aprende a reconocer un ojo en la esquina superior, no reconoce ese mismo ojo si aparece en el centro. Tendría que reaprenderlo en cada posición.

La solución a estos tres problemas es la convolución.

La convolución

La convolución es la operación central de una CNN. En lugar de conectar cada neurona con todos los píxeles, usa un filtro (o kernel) pequeño que se desliza por toda la imagen.

- Un filtro es una matriz pequeña de pesos, típicamente de 3 × 3 o 5 × 5.
- El filtro se coloca sobre una región de la imagen, se multiplican sus valores por los píxeles que cubre, se suman, y ese número va a la salida.
- Luego el filtro se desliza (por ejemplo un píxel a la derecha) y repite la operación por toda la imagen.

El resultado es un mapa de características (feature map): una nueva "imagen" que marca dónde apareció el patrón que ese filtro detecta.

Tres ideas clave de la convolución

- Conectividad local: cada neurona de salida solo mira una pequeña región de la entrada (su campo receptivo), no toda la imagen. Esto respeta la estructura espacial.
- Pesos compartidos: el mismo filtro (los mismos pesos) se usa en toda la imagen. Por eso hay poquísimos parámetros: un filtro de $3 \times 3 \times 3$ tiene solo 27 pesos, sin importar el tamaño de la imagen.
- Invariancia a la traslación: como el filtro recorre toda la imagen, detecta su patrón (por ejemplo un borde vertical) sin importar dónde aparezca.

Qué aprenden los filtros

Nadie programa los filtros: la red los aprende durante el entrenamiento. Al analizar redes ya entrenadas se ve un patrón claro:

- Los filtros de las primeras capas detectan bordes y colores simples.
- Las capas intermedias combinan bordes en texturas y formas (esquinas, curvas, rejillas).
- Las capas profundas combinan formas en partes de objetos (un ojo, una rueda, una oreja).
- Las últimas capas reconocen objetos completos.

Esta jerarquía de características —de lo simple a lo complejo— es exactamente lo que hace poderoso al deep learning en visión.

Múltiples filtros y canales

Una capa convolucional no aprende un solo filtro, sino muchos (por ejemplo 32 o 64). Cada uno detecta un patrón distinto, y la salida de la capa es un apilado de mapas de características (uno por filtro). Esa profundidad se convierte en los "canales" de entrada de la siguiente capa.

Parámetros de la convolución

El comportamiento de una capa convolucional se controla con unos pocos hiperparámetros:

- Tamaño del kernel: el ancho y alto del filtro (3 × 3 es el más común). Kernels grandes ven más contexto pero cuestan más.
- Stride (paso): cuántos píxeles se desplaza el filtro en cada paso. Un stride de 1 recorre píxel por píxel; un stride de 2 salta de dos en dos y reduce el tamaño de la salida a la mitad.
- Padding (relleno): agregar un borde de ceros alrededor de la imagen. El padding "same" mantiene el tamaño de salida igual al de entrada; sin padding ("valid") la salida se encoge en cada capa.

El tamaño de salida de una convolución se calcula así:

$$ O = \left\lfloor \frac{N - K + 2P}{S} \right\rfloor + 1 $$

donde N es el tamaño de entrada, K el del kernel, P el padding y S el stride.

Pooling (submuestreo)

Después de una o varias convoluciones se suele aplicar una capa de pooling, que reduce el tamaño de los mapas de características.

- Max pooling: divide el mapa en regiones (por ejemplo 2 × 2) y de cada una conserva solo el valor máximo.
- Average pooling: en lugar del máximo, conserva el promedio de cada región.

El pooling cumple tres funciones:

- Reduce la resolución y, con ella, el número de cálculos y parámetros de las capas siguientes.
- Aporta invariancia a pequeñas traslaciones: si el patrón se mueve un píxel, el máximo de la región probablemente no cambia.
- Amplía el campo receptivo: tras el pooling, cada neurona "resume" una región más grande de la imagen original.

Función de activación ReLU

Igual que en cualquier red profunda, después de cada convolución se aplica una función de activación no lineal. En las CNN la estándar es ReLU:

$$ \text{ReLU}(z) = \max(0, z) $$

- Deja pasar los valores positivos y pone en cero los negativos.
- Es rápida de calcular y evita en gran medida el problema del gradiente que se desvanece.
- Sin una activación no lineal, apilar convoluciones sería inútil: toda la red se reduciría a una sola operación lineal.

Arquitectura de una CNN

Una CNN típica de clasificación encadena bloques que se repiten y termina en un clasificador:

1. Entrada: el tensor de la imagen (alto × ancho × canales).
2. Bloques convolucionales: se repite el patrón [Convolución → ReLU → Pooling] varias veces. A medida que se avanza, la resolución baja pero el número de filtros (canales) sube.
3. Aplanado (flatten): se convierte el último mapa de características en un vector.
4. Capas densas: una o dos capas totalmente conectadas combinan las características de alto nivel.
5. Capa de salida con softmax: produce una probabilidad para cada clase; la clase con mayor probabilidad es la predicción.

La intuición: las capas convolucionales extraen qué hay en la imagen (las características), y las capas densas finales deciden qué clase es a partir de esas características.

Entrenar una CNN

Una CNN se entrena igual que cualquier red neuronal, con el mismo ciclo:

1. Forward pass: la imagen atraviesa la red y produce probabilidades por clase.
2. Pérdida: se mide qué tan lejos está la predicción de la etiqueta real.
3. Backpropagation: se calcula el gradiente de la pérdida respecto a cada peso (incluidos los filtros).
4. Actualización: el optimizador ajusta los pesos, y se repite con el siguiente lote de imágenes.

La función de pérdida: entropía cruzada

Para clasificación se usa la entropía cruzada (cross-entropy), que penaliza asignar poca probabilidad a la clase correcta:

$$ L = -\sum_{c=1}^{C} y_c \log(\hat{y}_c) $$

donde y_c vale 1 para la clase correcta y 0 para las demás, y $\hat{y}_c$ es la probabilidad que la red asignó a la clase c. La pérdida es baja cuando la red da alta probabilidad a la clase correcta.

Data augmentation

Como las CNN necesitan muchos datos, una técnica clave es aumentar artificialmente el conjunto de entrenamiento generando variantes de cada imagen:

- Voltear horizontalmente, rotar, recortar o hacer zoom.
- Cambiar brillo, contraste o color.
- Desplazar la imagen.

Así la red ve "más" ejemplos y aprende a ser robusta a esas variaciones, lo que reduce el sobreajuste.

Regularización

Para que la red no memorice el entrenamiento se usan varias técnicas:

- Dropout: durante el entrenamiento se apagan al azar algunas neuronas, forzando a la red a no depender de ninguna en particular.
- Regularización L2 (weight decay): penaliza pesos grandes.
- Batch normalization: normaliza las activaciones dentro de la red, lo que estabiliza y acelera el entrenamiento.

- Early stopping: detener el entrenamiento cuando la pérdida de validación deja de bajar.

Transfer learning (aprendizaje por transferencia)

Entrenar una CNN grande desde cero requiere millones de imágenes y mucho cómputo. Casi nunca se hace. En su lugar se usa transfer learning:

- Se parte de un modelo ya entrenado en un conjunto enorme (típicamente ImageNet, con más de un millón de imágenes y 1000 clases).
- Ese modelo ya aprendió filtros útiles y generales (bordes, texturas, formas) que sirven para casi cualquier problema de visión.

Hay dos estrategias:

- Extracción de características: se congelan las capas convolucionales del modelo preentrenado y solo se entrena un nuevo clasificador al final. Ideal cuando se tienen pocos datos.
- Fine-tuning (ajuste fino): además del clasificador, se reentrena (con una tasa de aprendizaje pequeña) algunas de las últimas capas convolucionales para adaptarlas al nuevo problema.

El transfer learning permite lograr excelente precisión con unos pocos cientos o miles de imágenes, algo imposible entrenando desde cero.

Arquitecturas famosas

La historia de las CNN es una carrera de arquitecturas, muchas nacidas del concurso ImageNet (ILSVRC):

- LeNet-5 (1998): la CNN pionera, de Yann LeCun, para reconocer dígitos escritos a mano.
- AlexNet (2012): ganó ImageNet por amplio margen y desató la revolución del deep learning. Popularizó ReLU, dropout y el entrenamiento en GPU.
- VGG (2014): demostró que apilar muchas convoluciones pequeñas (3 × 3) funciona muy bien. Simple y profunda (16 o 19 capas).
- GoogLeNet / Inception (2014): introdujo los módulos "inception", que aplican filtros de varios tamaños en paralelo.
- ResNet (2015): introdujo las conexiones residuales (skip connections), que permiten entrenar redes de cientos de capas sin que el gradiente se degrade. Es una de las arquitecturas más influyentes.
- EfficientNet (2019): escala de forma balanceada la profundidad, el ancho y la resolución para máxima eficiencia.

Vision Transformers (ViT)

Desde 2020, los transformers —la arquitectura que revolucionó el lenguaje— también llegaron a la visión:

- Un Vision Transformer parte la imagen en parches (por ejemplo de 16 × 16 píxeles), los trata como una secuencia y aplica el mecanismo de atención.
- Con suficientes datos, igualan o superan a las CNN.
- Requieren más datos para entrenarse bien; por eso las CNN siguen siendo muy usadas, y a menudo se combinan ambos enfoques (modelos híbridos).

Más allá de la clasificación

Sobre las mismas ideas convolucionales se construyen tareas más ricas:

- Detección de objetos: familias como R-CNN (precisas) y YOLO / SSD (rápidas, en tiempo real) encuentran y clasifican varios objetos con sus recuadros.
- Segmentación semántica: arquitecturas encoder–decoder como U-Net clasifican cada píxel; muy usadas en imágenes médicas y satelitales.
- Segmentación de instancias: modelos como Mask R-CNN separan además cada objeto individual.

Evaluar el modelo

Para clasificación de imágenes las métricas habituales son:

- Exactitud (accuracy): porcentaje de imágenes clasificadas correctamente. Útil cuando las clases están balanceadas.
- Precisión y recall: importantes cuando las clases están desbalanceadas o cuando un tipo de error cuesta más (por ejemplo, un diagnóstico médico).
- Matriz de confusión: tabla que muestra qué clases se confunden entre sí; revela dónde falla el modelo.
- Top-5 accuracy: en problemas con muchas clases (como ImageNet), se considera acierto si la clase correcta está entre las 5 más probables.

Aplicaciones

El reconocimiento de imágenes está hoy en todas partes:

- Medicina: detección de tumores en radiografías, resonancias y retinografías.
- Vehículos autónomos: identificar peatones, señales, carriles y otros autos.
- Seguridad y biometría: reconocimiento facial y de huellas.
- Agricultura: detección de plagas y enfermedades en cultivos por foto.

- Industria: control de calidad y detección de defectos en líneas de producción.
- Vida diaria: buscar y organizar fotos, filtros de cámara, lectura de documentos (OCR).

Consideraciones éticas

El poder de estos modelos trae responsabilidades:

- Sesgos: si los datos de entrenamiento no representan bien a todos los grupos, el modelo falla más con los subrepresentados (por ejemplo, reconocimiento facial menos preciso en ciertos tonos de piel).
- Privacidad y vigilancia: el reconocimiento facial masivo plantea serios problemas de derechos.
- Ataques adversariales: cambios diminutos e imperceptibles en una imagen pueden engañar a la red y hacerla clasificar mal.
- Explicabilidad: en decisiones críticas (medicina, justicia) es clave poder explicar por qué el modelo decidió lo que decidió.

Cierre

- El reconocimiento de imágenes es el problema donde el deep learning demostró primero su ventaja.
- Las CNN resuelven las limitaciones del MLP con tres ideas: conectividad local, pesos compartidos e invariancia a la traslación.
- Una CNN aprende una jerarquía de características: de bordes a formas, de formas a objetos.
- En la práctica casi nunca se entrena desde cero: el transfer learning con modelos preentrenados es la norma.

- Las mismas ideas se extienden a detección, segmentación y, más recientemente, a los Vision Transformers.