DEEP LEARNING

# Regularización y transfer learning

## Cómo hacer que una CNN funcione en el mundo real

Dr. León Felipe Palafox Novack

lfpalafox@up.edu.mx

01 Repaso de CNN

02 El problema práctico

03 Visión moderna

04 Overfitting y Dropout

05 Batch Normalization

06 Data augmentation

07 Transfer learning

08 Evaluación y práctica

02

# Repaso: lo que vimos sobre CNN

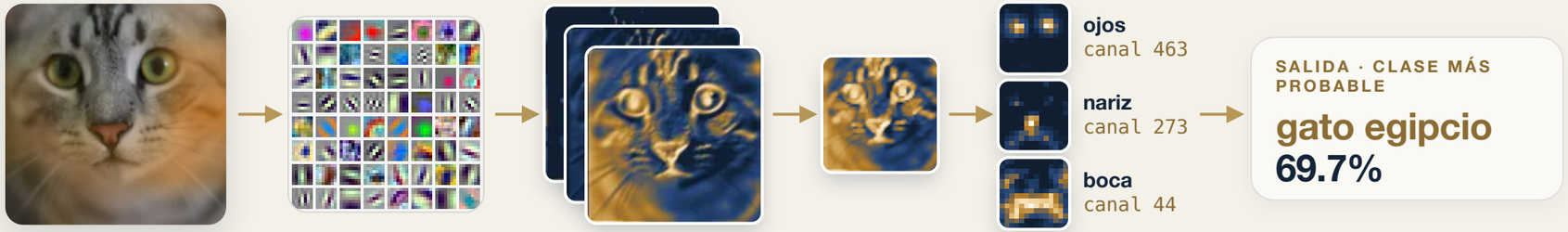
## De píxeles a patrones visuales

**Convolución**: un filtro pequeño recorre la imagen.

**Filtros y mapas** de activación: dónde aparece cada patrón.

**Pooling**: reduce el mapa y conserva lo importante.

**Jerarquía**: de bordes a partes, de partes a objetos.




LA RED DICE · LAS 5 CLASES MÁS PROBABLES

| | |
|---|---|
| **gato egipcio** | **69.7%** |
| gato atigrado | 16.1% |
| gato tigre | 10.4% |
| lince | 2.6% |
| leopardo de las nieves | 0.7% |

**96.3%**
gato: suma de las 5 clases de gato doméstico de ImageNet

**< 0.01%**
perro, conejo, coche o ave (cada uno)

**Todo es real:** esta foto pasó por una ResNet-50 entrenada en ImageNet. Los filtros son los pesos de su primera capa y los mapas, sus respuestas; los canales 463, 273 y 44 se encienden en ojos, nariz y boca. Al final, en dorado sobre la foto: dónde «miró» la red para decidir (CAM).

03

# ¿Por qué usamos CNN para imágenes?

## Aprovechan la estructura espacial de la imagen

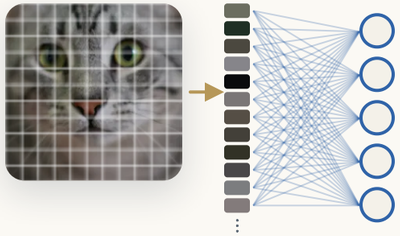
Detectan patrones **locales**.

**Reutilizan** filtros en toda la imagen.

Son más eficientes que una red totalmente conectada.

Aprenden de **bordes a objetos**.

**RED DENSA**



**150,528**

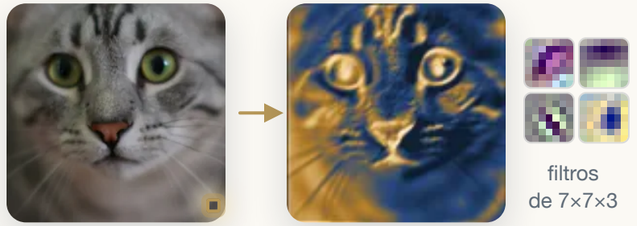
entradas (224×224×3)

**× 1,000**

neuronas → **150.5** M pesos

La imagen se «aplana»: cada píxel se conecta con todas las neuronas y se pierden las relaciones espaciales.

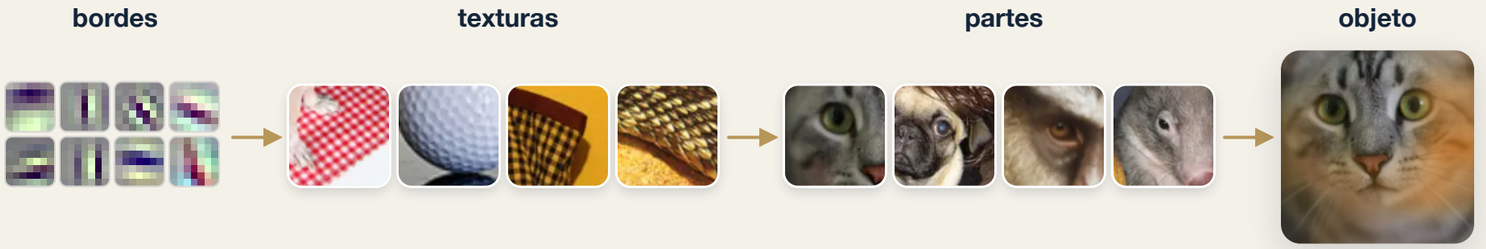
**CNN**



filtros de 7×7×3

El mismo filtro recorre toda la imagen: **12,544 posiciones** con los mismos 147 pesos. Los 64 filtros suman **9,408 pesos**.

Una sola capa densa sobre la imagen tendría **150.5 M** de pesos: 5.9 veces **toda** ResNet-50 (25.6 M).




**Las CNN aprenden jerarquías:** de características simples a representaciones complejas. Parches reales: las fotos de ImageNet que más activan cada canal.

04

# La pregunta de hoy

## ¿Cómo hacemos que una CNN funcione bien en el mundo real?

Cuatro situaciones que aparecen en cualquier proyecto real, y la herramienta de hoy para cada una (el número es la diapositiva).

| | | |
|---|---|---|
| Cuando hay **pocos datos** | → | Data augmentation 14 · Transfer learning 15–17 |
| Cuando aparece **overfitting** | → | Dropout 10–11 · Data augmentation 14 · Early stopping 9 |
| Cuando entrenar se vuelve **inestable** | → | Batch Normalization 12–13 |
| Cuando necesitamos resultados **rápido** | → | Modelo preentrenado 15 · Feature extractor 16–17 |

05

# El problema práctico de entrenar CNNs

### Muchos parámetros

ResNet-50: **25.6 M**. Nuestra CNN pequeña: **1.12 M**.

### Pocos datos

ImageNet: **1.28 M** imágenes de entrenamiento. Nuestro experimento: **2,000**.

### Overfitting

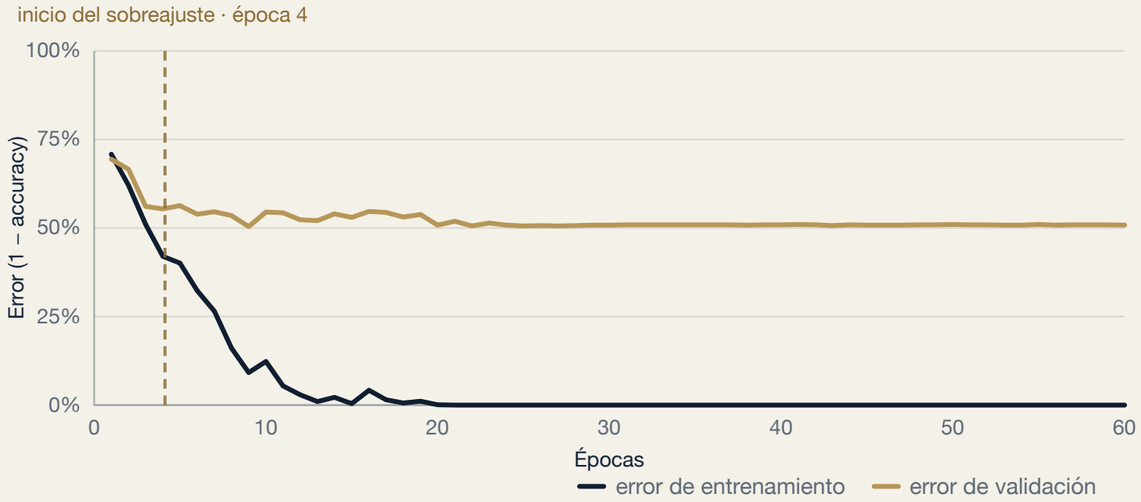
Tras 60 épocas: **100%** en entrenamiento, **49%** en validación.

### Entrenamiento inestable

Con SGD y lr = 0.1, nuestra CNN sin BatchNorm no aprendió en **1 de 3** intentos; con BatchNorm, en **3 de 3** sí.

### Generalización difícil

PlantVillage: **99.35%** en su prueba, **31.4%** con fotos de internet (Mohanty et al., 2016).




⚠ Entender las CNNs no es suficiente: hay que **entrenarlas bien** y lograr que **generalicen**.

Curvas reales: la CNN pequeña de nuestro experimento (1.12 M de parámetros) entrenada desde cero con 2,000 imágenes de CIFAR-10. El entrenamiento llega a 100% de aciertos; la validación se estanca: eso es overfitting.

06

# Ejemplos modernos: visión en medicina

Detección de hallazgos en **radiografías**.

Apoyo en **dermatología** y **patología**.

Herramientas de análisis **asistido** por IA.

Las CNN no solo clasifican gatos: también **apoyan decisiones reales**.




RADIOGRAFÍAS

**CheXNet · Rajpurkar et al., 2017**

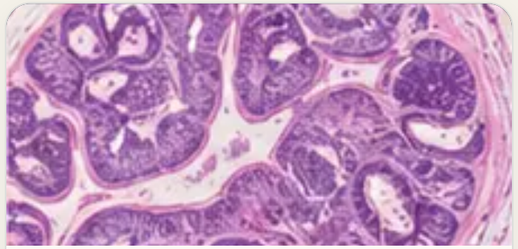
- DenseNet-121 **preentrenada en ImageNet** y ajustada
- **112,120** radiografías de tórax de **30,805** pacientes
- Neumonía: F1 **0.435** vs **0.387**, promedio de 4 radiólogos




DERMATOLOGÍA

**Esteva et al., Nature 2017**

- Inception v3 **preentrenada en ImageNet** + transfer learning
- **129,450** imágenes clínicas de **2,032** enfermedades
- A la par de **21** dermatólogos certificados



PATOLOGÍA

**Campanella et al., Nature Medicine 2019**

- CNN entrenada con el diagnóstico de cada laminilla, sin anotar regiones
- **44,732** laminillas completas de **15,187** pacientes
- AUC **> 0.98** en próstata, carcinoma basocelular y metástasis de mama

**1,614** dispositivos médicos con IA autorizados por la **FDA** (hasta junio de 2026)

76% son de radiología · la decisión final es del profesional de la salud

Imágenes ilustrativas de la presentación original; cifras de los artículos citados (ver notas).

07

# Ejemplos modernos: búsqueda visual

Buscar **productos** desde una foto.

Reconocer **objetos** y **texto**.

Traducir o identificar elementos **en tiempo real**.

GOOGLE LENS · MAYO DE 2025

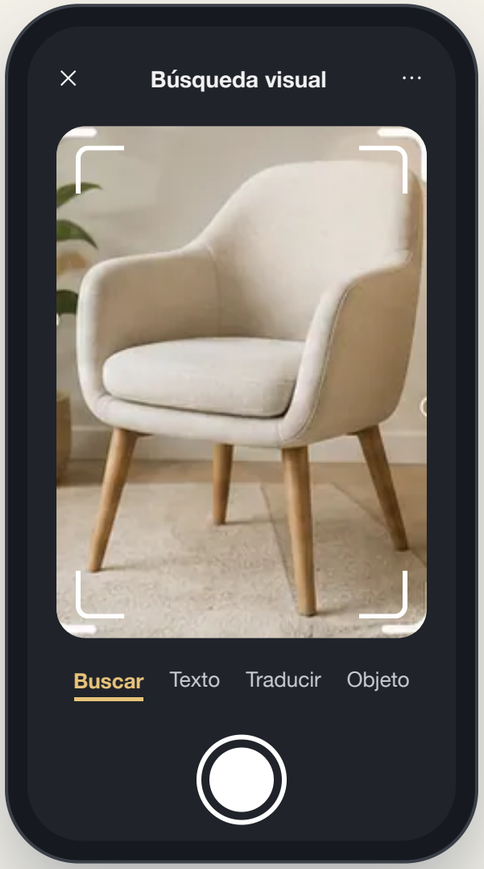
**> 100 mil millones**

de búsquedas visuales en lo que iba del año; +65% anual

RESULTADOS · SIMILITUD COSENO

| | | |
|---|---|---|
|  | sillón (catálogo) | 0.91 |
|  | sillón (catálogo) | 0.91 |
|  | sillón (catálogo) | 0.90 |
|  | mesa de comedor | 0.72 |
|  | persiana | 0.69 |

Los 3 sillones vienen de la maqueta original; las otras fotos, de ImageNet. La consulta no está en el catálogo.




LA RED VE · RESNET-50

| | |
|---|---|
| mecedora | 70.9% |
| sofá cama | 18.4% |
| silla plegable | 2.4% |

ImageNet no tiene la clase «sillón»: la red usa lo más parecido que conoce. Aun así, su **embedding** encuentra los sillones.

TEXTO




Horario de atención
9:00 – 18:00

TRADUCIR



Salida de emergencia
→ **Emergency exit**

Texto y traducción: ilustrativos (otros modelos). La búsqueda y la clasificación son reales.

08

# Ejemplos modernos: segmentación y video

Separar objetos **del fondo**.

Seguir objetos **en video**.

Modelos **fundacionales** de visión.

**SEGMENT ANYTHING (META)**

| | |
|---|---|
| 2023 | SAM: más de 1,000 millones de máscaras en 11 M de imágenes |
| 2024 | SAM 2: imágenes y video, con memoria |
| 2025 | SAM 3: todos los «perros» de un video, pedidos con texto |

**1 Imagen original**

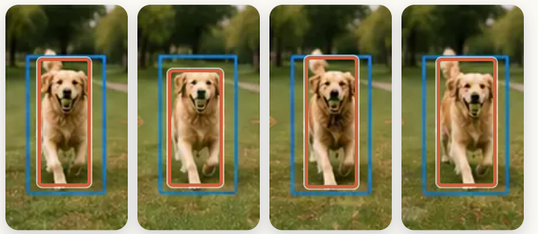


**2 Máscara de segmentación**



perro · 21% de los píxeles · confianza 96%

**3 Seguimiento en video**



0.90 0.79 0.83

tiempo →

Una máscara por cuadro; el traslape entre cajas consecutivas (IoU, en dorado) liga al mismo perro a lo largo del tiempo.

La visión moderna ya no solo clasifica: también **localiza y separa**.

Máscaras calculadas con FCN-ResNet50 (ONNX Model Zoo) sobre las imágenes de la presentación original.

09

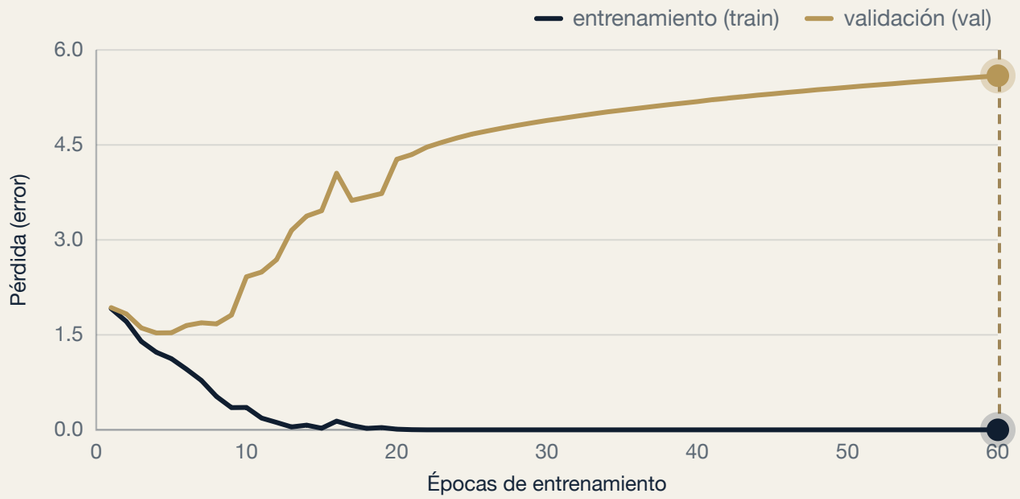
# Overfitting: cuando el modelo memoriza

Aprende **demasiado bien** el entrenamiento.

**Falla** al generalizar.

Buen desempeño en **train**, malo en **validation/test**.

⚠ El modelo se vuelve **especialista** en el set de entrenamiento.




**ÉPOCA 60**

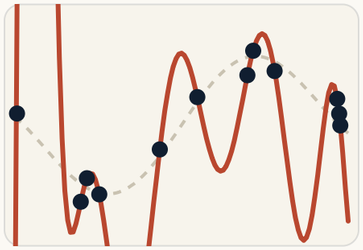
| | |
|---|---|
| entrenamiento | 100.0% |
| validación | 49.2% |
| brecha | 50.8 pts |

Memorizó el entrenamiento; validación estancada y su pérdida 3.7 veces la del mínimo.

inicio · mínimo val. · después · final

**MEMORIZAR EJEMPLOS**

Polinomio de grado 11: pasa por los 12 puntos



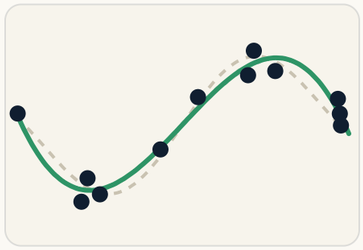
error en entrenamiento
**0.00**

error con datos nuevos
**68**

vs

**APRENDER PATRONES**

Polinomio de grado 3: sigue la tendencia



error en entrenamiento
**0.02**

error con datos nuevos
**0.07**

10

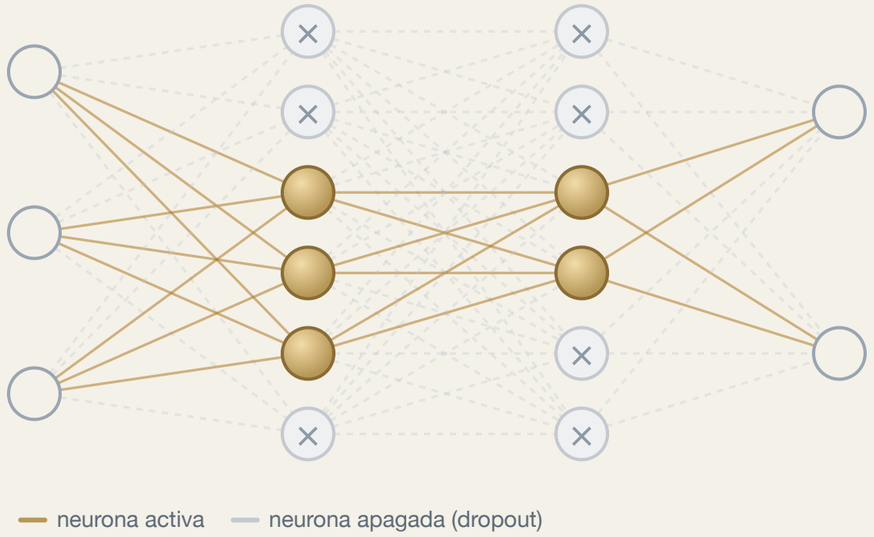
# Dropout: no depender de una sola ruta

Apaga neuronas **al azar** durante el entrenamiento.

Obliga a **repartir** el aprendizaje.

Ayuda a reducir **overfitting**.

> **Si una neurona falta, la red debe seguir funcionando.**




DROPOUT · P = 0.5

## Mini-lote 1

activas: 5 de 12 neuronas ocultas

lote 1 | lote 2 | lote 3 | lote 4 | lote 5 | lote 6

Cada mini-lote apaga **otra** mitad al azar: ninguna neurona puede ser indispensable.

PRUEBA REAL: APAGAR LA MITAD

Ya entrenadas, apagamos al azar 128 de las 256 neuronas ocultas (20 sorteos). Accuracy en validación:

| | |
|---|---|
| sin dropout | 49% → 43% |
| con dropout | 53% → 50% |

11

# ¿Qué hace realmente Dropout?

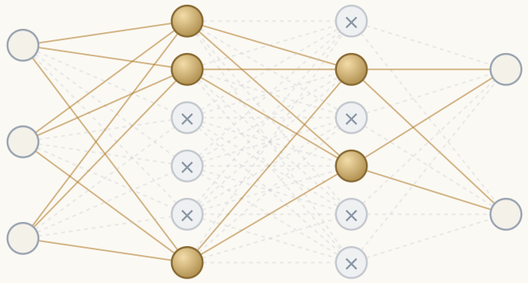
**Entrenamiento**: neuronas apagadas aleatoriamente.

**Inferencia**: red completa activa.

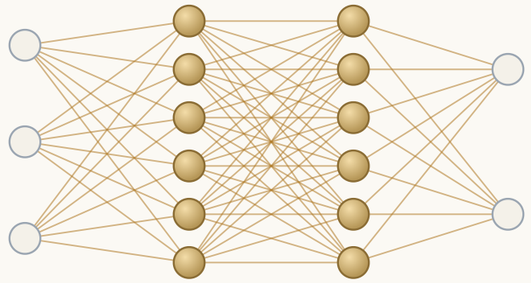
Efecto **regularizador**.

El modelo aprende **representaciones más robustas**.

ENTRENAMIENTO



INFERENCIA

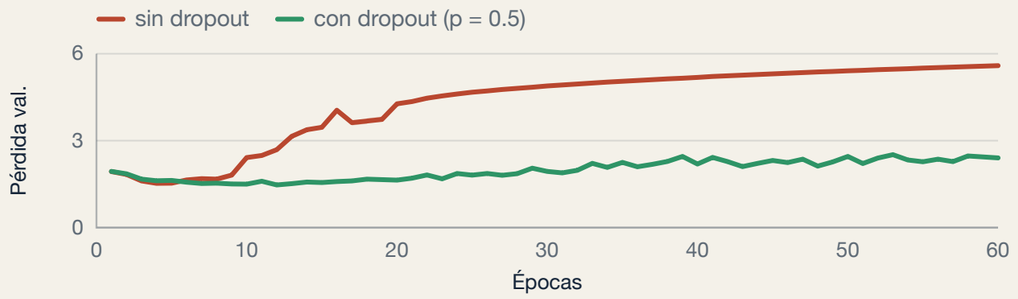


$$\tilde{h} = m \odot h \,/\, (1 - p)$$

m: 0 o 1 al azar (se queda con probabilidad 1 – p). Las que sobreviven se multiplican por 2 si p = 0.5.

$$\tilde{h} = h$$

Red completa, sin máscara y sin escalar: en PyTorch, `model.eval()`.




**CNN PEQUEÑA · 60 ÉPOCAS**

| | |
|---|---|
| val. sin dropout | 49% |
| val. con dropout | 53% |
| pérdida val. final | 5.6 → 2.4 |

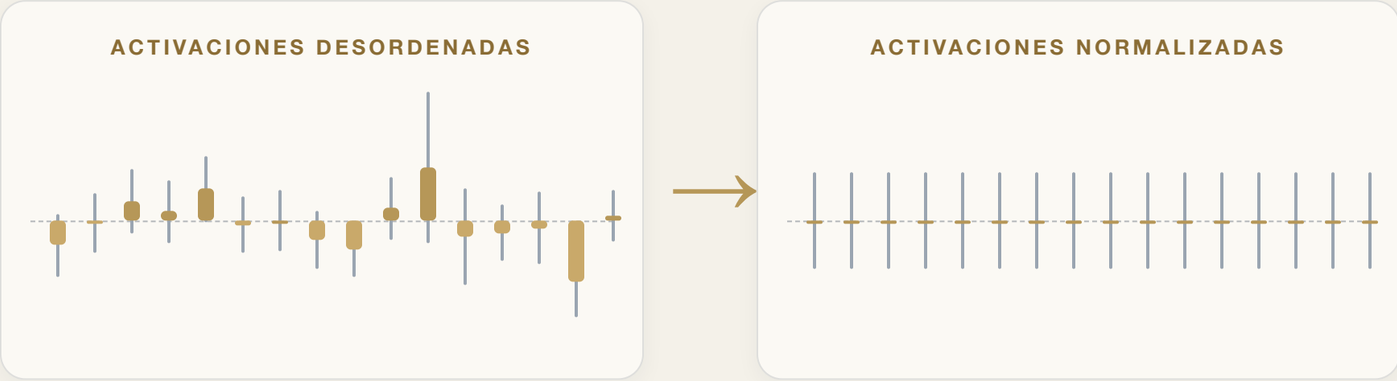
12

# BatchNorm: estabilizar para aprender mejor

Normaliza activaciones **por lote**.

Facilita el entrenamiento.

Hace más **estable** el flujo de información.




16 canales reales de ResNet-50 (etapa 2, conv 1×1): media ± desviación en un lote de 32 fotos.

Después de restar la media del lote y dividir entre su desviación: todos en 0 ± 1.

**PARA CADA CANAL, EN CADA LOTE**

$\mu_B, \sigma_B$

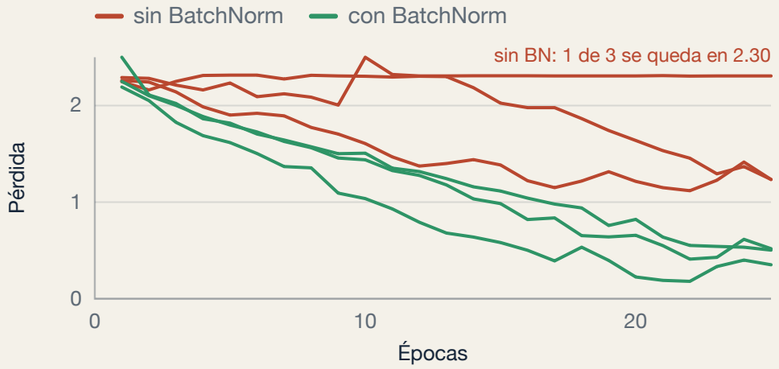
media y desviación del lote

$\hat{x} = (x - \mu_B) / \sqrt{(\sigma_B^2 + \varepsilon)}$

normalizar

$y = \gamma \hat{x} + \beta$

γ y β se aprenden




**Experimento real:** la misma CNN, la misma tasa de aprendizaje alta (SGD, lr = 0.1), 3 semillas cada una. Sin BatchNorm, 1 de 3 no aprende (la pérdida se queda en 2.30, el azar) y los demás llegan a 26%, 24% en validación; con BatchNorm, las 3 semillas llegan a 38%–39%.

⚠ Si las activaciones cambian demasiado, **entrenar se vuelve más difícil**.

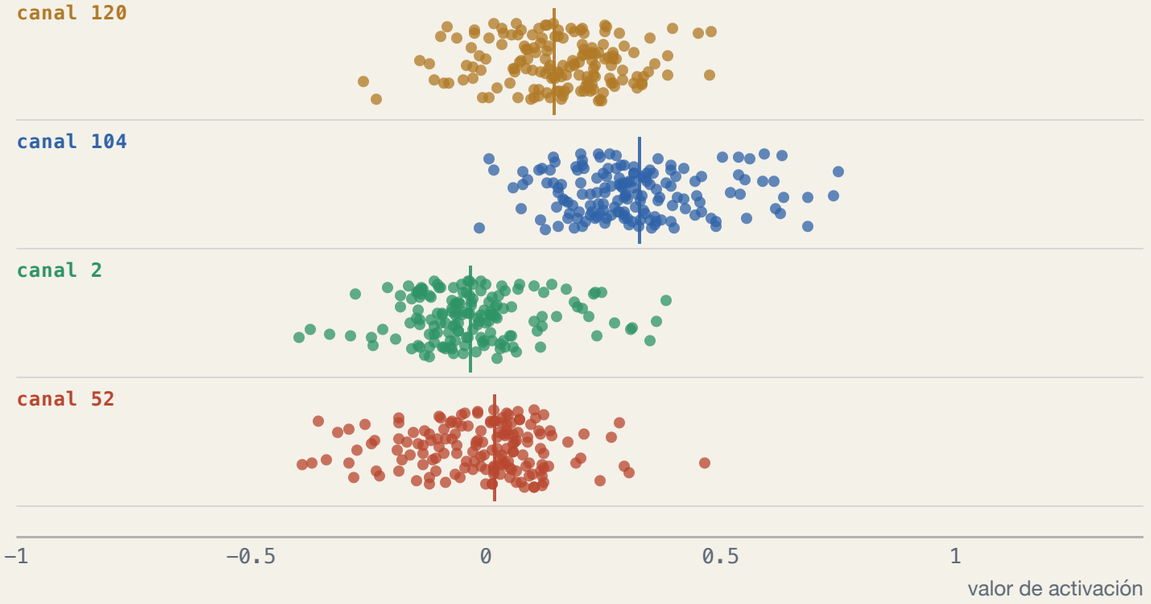
13

# Batch Norm: antes y después

**Antes**: activaciones inestables.

**Después**: activaciones más controladas.

**Resultado**: entrenamiento más confiable.




| | canal 120 | canal 104 | canal 2 | canal 52 |
|---|---|---|---|---|
| $\mu_B$ (media del lote) | −0.256 | 0.218 | 0.005 | −0.140 |
| $\sigma_B$ (desviación) | 0.143 | 0.313 | 0.037 | 0.080 |
| **γ, β (aprendidos)** | 0.129, 0.146 | 0.144, 0.325 | 0.131, −0.034 | 0.133, 0.017 |

Valores reales: 160 activaciones por canal, tomadas de un lote de 32 fotos en una capa de ResNet-50 (etapa 2, bloque 2, conv 1×1: 128 canales de 28×28). γ y β son los que la red aprendió.

PASO 4 DE 4

## Escalar y desplazar

$\gamma\ \hat{x} + \beta$

- Antes
- Restar la media
- Dividir entre la desviación
- **Escalar y desplazar**

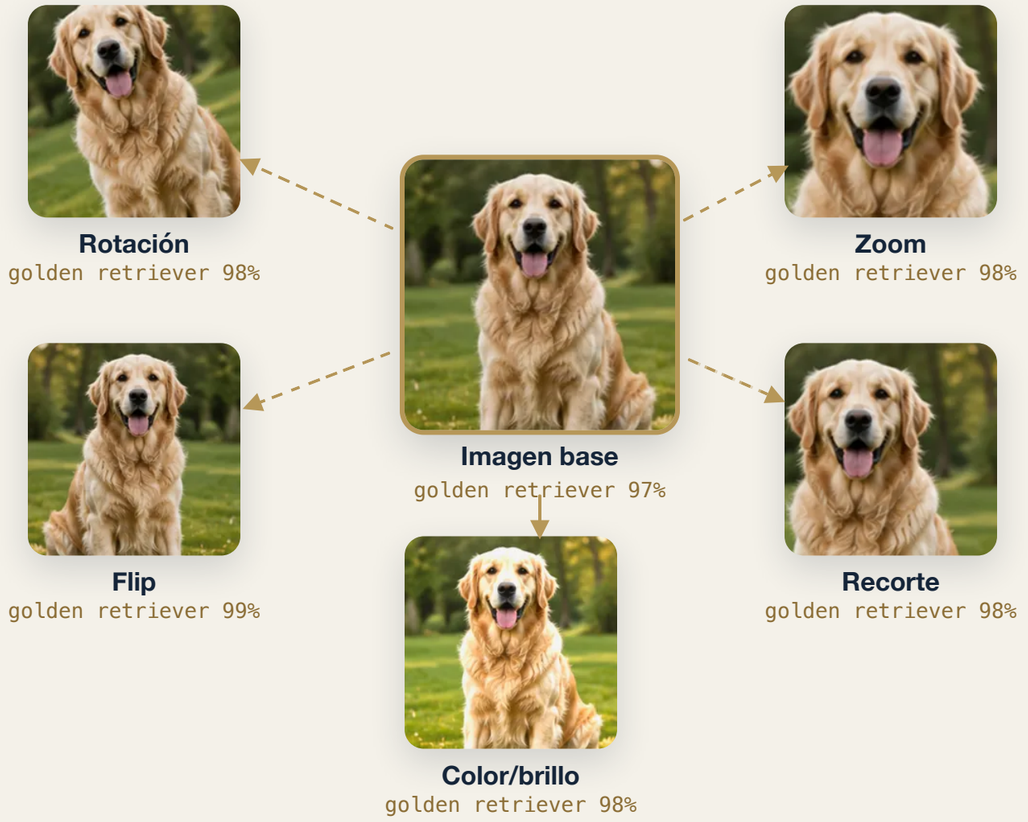
Entrada → Capa (conv) → **BatchNorm** → Activación

14

# Data augmentation: más variación

- Rotación
- Flip
- Zoom
- Recorte
- Pequeños cambios de **color o brillo**

> Aumenta **diversidad**, mejora **generalización**.




**LO QUE VE LA RED EN CADA ÉPOCA**




**EXPERIMENTO REAL · CNN PEQUEÑA**

| | |
|---|---|
| sin augmentation | **48%** |
| con recorte + flip | **57%** |

15

# Transfer learning: reutilizar lo aprendido

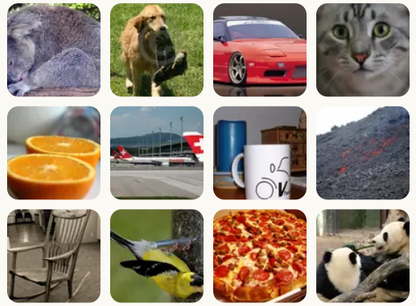
## Aprovechar conocimiento previo

Usar un modelo **preentrenado**.

Reutilizar características **visuales**.

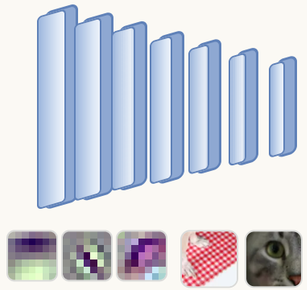
Adaptarlo a una **nueva tarea**.

**MUCHOS DATOS**



**ImageNet** · 1.28 M imágenes · 1,000 clases
Gran problema fuente

**MODELO PREENTRENADO**

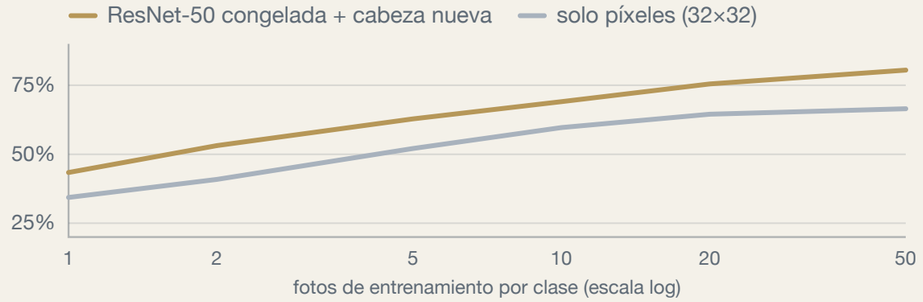


**ResNet-50** · 25.6 M parámetros
Conocimiento visual reutilizable

**NUEVA TAREA**




**Hojas de tomate** (PlantVillage) · 4 clases
Problema objetivo · pocos datos




**CON 20 FOTOS POR CLASE**

| 76% | 65% | 25% |
|---|---|---|
| ResNet-50 + cabeza | píxeles | azar |

Promedio de 40 sorteos; 200 fotos de prueba.

16

# Feature extractor vs fine-tuning

## Feature extractor

- Congelar el backbone
- Entrenar solo la cabeza final
- Más rápido, menos riesgo

## Fine-tuning

- Descongelar parte del modelo
- Ajustar capas profundas
- Más flexible, más delicado



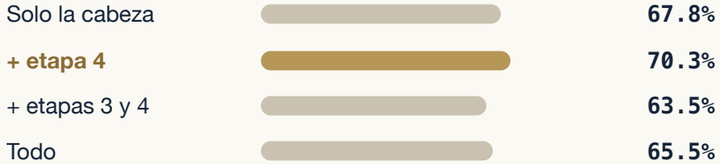

FINE-TUNING PARCIAL

**+ etapa 4**

| | |
|---|---|
| parámetros entrenables | 14,980,612 (64%) |
| aciertos en 200 hojas de prueba | 70.3% |
| tiempo por época (CPU) | 9 s |

Solo la cabeza | **+ etapa 4** | + etapas 3 y 4 | Todo

ACIERTOS EN LA PRUEBA · HOJAS DE TOMATE, 20 FOTOS POR CLASE




ResNet-50 real (pesos de ImageNet), 12 épocas, mismo aumento de datos; promedio de 2 sorteos de las 80 fotos de entrenamiento (1 en «Todo»). Descongelar todo entrena 2,872 veces más parámetros que la cabeza sola.

17

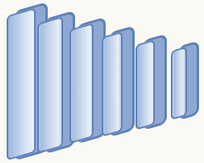
# Flujo práctico en un proyecto real

**1 Dataset**



**4 clases** · 20 fotos por clase para entrenar · **200** de prueba

**2 Modelo preentrenado**



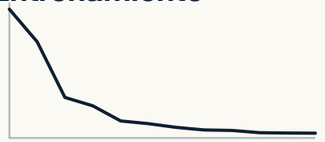
**ResNet-50** con pesos de ImageNet; se congela

**3 Nueva cabeza**

2048 → 4

**8,196** parámetros nuevos: una capa lineal + softmax

**4 Entrenamiento**



**12** épocas; la pérdida de entrenamiento baja · 7 s por época en CPU

**5 Evaluación**

| | | | |
|---|---|---|---|
| 19 | 7 | 11 | 13 |
| 8 | 35 | 5 | 2 |
| 4 | 6 | 32 | 8 |
| 0 | 0 | 0 | 50 |

**68.0%** de aciertos en 200 hojas de prueba; sanas: **50 de 50**

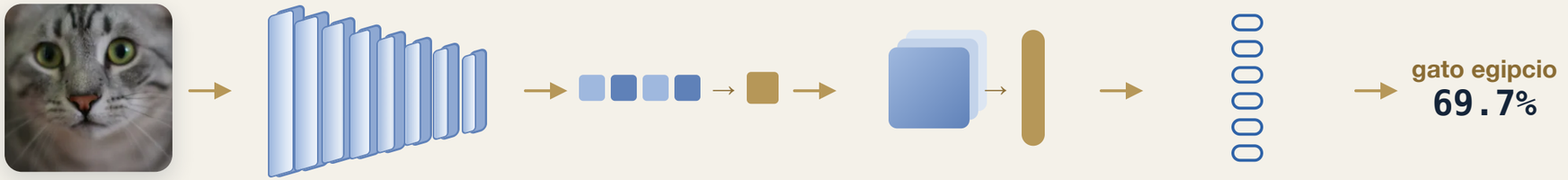
Este será el **flujo de la práctica**.

Experimento real: 80 fotos de hojas de tomate (PlantVillage, CC0), ResNet-50 de ImageNet congelada y una cabeza lineal nueva entrenada con aumento de datos. Matriz de confusión (sorteo 1): filas = clase real, columnas = predicción, en este orden: tizón temprano, tizón tardío, Septoria y sana. Las tres enfermedades se confunden entre sí; las hojas sanas casi nunca.

18

# Anatomía mínima de una solución de visión

- Convolutional base
- Pooling
- GAP o Flatten
- Capa densa
- Softmax / salida final
- Dropout / BatchNorm como apoyo




BatchNorm × 53 · sin Dropout · Dropout (en una cabeza nueva)

En ResNet-50 casi toda la reducción la hacen convoluciones con paso 2; solo hay un max pooling (después de conv1) y un GAP al final. Tiene 53 capas BatchNorm y ninguna de Dropout.

### ¿GAP O FLATTEN ANTES DE LA CAPA DENSA?

**Global Average Pooling** · Flatten

Promedia cada uno de los 2048 mapas de 7×7: un número por canal. Es lo que usa ResNet-50.

| | |
|---|---|
| vector que entra a la capa densa | 2,048 |
| parámetros de la capa densa (1,000 clases) | 2,049,000 |
| comparado con todo ResNet-50 (25.6 M) | 8% |

19

# Evaluar bien: accuracy no siempre basta

**Accuracy** = aciertos / total

**Precision** = VP / (VP + FP)

**Recall** = VP / (VP + FN)

**Matriz de confusión**

Ver **errores reales**

Ver fallos del modelo **también enseña**.

PREDICCIÓN

| REAL | Gato | No gato |
|---|---|---|
| Gato | 66 verdadero positivo | 34 falso negativo |
| No gato | 19 falso positivo | 881 verdadero negativo |

UMBRAL: «GATO» SI P(GATO) ≥ 0.5 · 1,000 IMÁGENES DE PRUEBA, 100 GATOS

| | |
|---|---|
| Accuracy | 94.7% |
| Precision | 77.6% |
| Recall | 66.0% |

Línea gris: un modelo que **siempre** dice «no gato» tiene 90% de accuracy y 0% de recall.

umbral 0.2 · umbral 0.3 · **umbral 0.5** · umbral 0.7

CLASIFICACIÓN CORRECTA



→ gato

real: gato · P(gato) = 1.00

CLASIFICACIÓN INCORRECTA



→ gato

real: perro · P(gato) = 0.69

UN GATO QUE NO VIO



→ no gato

real: gato · P(gato) = 0.01

20

# Práctica: lo que vamos a programar

- Cargar imágenes
- Usar una CNN preentrenada
- Añadir capa final
- Probar Dropout y BatchNorm
- Entrenar y comparar resultados

Ya entendimos la idea, **ahora la implementamos**.

1 Cargar imágenes | 2 Usar CNN preentrenada | 3 Añadir capa final | 4 Probar Dropout y BatchNorm | 5 Entrenar y comparar

```
# Bloque 5 · Entrenar y evaluar
opt = torch.optim.Adam(modelo.fc.parameters(), lr=1e-3)
for epoca in range(10):
    modelo.eval(); modelo.fc.train()   # backbone fijo, cabeza en entrenamiento
    for x, y in train_dl:
        perdida = nn.functional.cross_entropy(modelo(x), y)
        opt.zero_grad(); perdida.backward(); opt.step()
    modelo.eval()   # dropout apagado; BatchNorm con estadísticas fijas
```

**EN NUESTRO EXPERIMENTO** Solo la cabeza: 67.8% en 200 hojas de prueba; ajustando también la etapa 4: 70.3%.

21

# Mapa mental final

### CNNs aprenden patrones visuales

Detectan patrones locales, combinan características y construyen representaciones cada vez más abstractas hasta clasificar.

ResNet-50 en la foto del repaso: **96.3%** gato



gato

Filtros → mapas → características → clasificación

### Dropout ayuda a generalizar

Apaga neuronas al azar durante el entrenamiento para evitar coadaptación y reducir el overfitting.

Validación: **49%** → **53%** con dropout

### BatchNorm estabiliza el entrenamiento

Normaliza las activaciones de cada lote; permite tasas de aprendizaje más altas y acelera la convergencia.

SGD, lr = 0.1: validación **10%–26%** sin BN, **38%–39%** con BN

Entender estas piezas es el paso de la **teoría a la práctica**.

### Transfer learning hace viable el trabajo real

Reutiliza representaciones aprendidas en grandes conjuntos de datos y las adapta a tu problema con pocos datos.

**48%** desde cero → **86%** con ResNet-50

NUESTRO EXPERIMENTO · CIFAR-10 CON 2,000 IMÁGENES · ACCURACY EN 1,000 DE PRUEBA

