Clase 130 — Arquitecturas CNN: LeNet, AlexNet, VGG, GoogLeNet, ResNet, Xception, SENet, EfficientNet, ConvNeXt

Parte: 2 — Deep Learning · Fuente: Géron, cap. 14 § CNN Architectures + papers originales.

Duración estimada: 95 min.

Clase 130 — Arquitecturas CNN: LeNet, AlexNet, VGG, GoogLeNet, ResNet, Xception, SENet, EfficientNet, ConvNeXt

> *Parte: 2 — Deep Learning · Fuente: Géron, cap. 14 § CNN Architectures + papers originales. Duración estimada: 95 min.*

Objetivo

Conocer la historia y evolución de las arquitecturas CNN desde LeNet-5 (1998) hasta ConvNeXt (2022) — qué innovación introdujo cada una, por qué importaba, y cuál usar hoy. Identificar 3 patrones clave: profundidad creciente, módulos con paths múltiples, eficiencia paramétrica.

Resultados de aprendizaje

Al finalizar, el estudiante podrá:

- Trazar la línea temporal: LeNet → AlexNet → VGG → GoogLeNet (Inception) → ResNet → Xception → SENet → EfficientNet → ConvNeXt.
- Reconocer qué innovación trajo cada una: ReLU + dropout (AlexNet), kernels 3×3 (VGG), módulos Inception (GoogLeNet), skip connections (ResNet), depthwise-separable (Xception), squeeze-and-excite (SENet), compound scaling (EfficientNet), modernización con receta ConvNeXt.
- Implementar un mini-ResNet con Add() y skip connections.
- Cargar arquitecturas de keras.applications y leer sus tamaños (MobileNetV3, EfficientNetB0, ConvNeXtTiny).
- Elegir la arquitectura adecuada según constraint (latency, accuracy, memory).

Temas

- LeNet-5 (1998): 7 capas, MNIST. La cuna.
- AlexNet (2012): GPU + ReLU + Dropout. Ganó ImageNet → revolución DL.
- VGG (2014): muy uniforme — solo conv 3×3 + maxpool. 138M parámetros.
- GoogLeNet / Inception (2014): módulos paralelos con kernels 1×1, 3×3, 5×5.
- ResNet (2015): skip connections → entrenamiento de redes de 152 capas posible. Cambió el juego.
- Xception (2017): depthwise-separable convolutions → eficiencia.
- SENet (2017): attention sobre canales (squeeze-and-excite).
- EfficientNet (2019): scaling balanceado de depth/width/resolution.
- ConvNeXt (2022): "modernizar ResNet" con trucos de ViT.

Definiciones y características

- Skip / residual connection: y = x + F(x). Permite gradiente fluir y entrenar redes muy profundas.
- Bottleneck: 1×1 conv → 3×3 conv → 1×1 conv (reduce → opera → expande). Reduce parámetros.
- Depthwise-separable conv: una conv que actúa por canal, seguida de 1×1 conv que mezcla canales. ~10× menos parámetros.
- Squeeze-and-Excite: re-pesa los canales aprendido (GlobalAvgPool → Dense → sigmoid → multiply).
- Compound scaling (EfficientNet): escalar depth, width, resolution juntos con coeficientes balanceados.

- keras.applications: catálogo con pesos ImageNet preentrenados de todas estas arquitecturas.

Dataset / recursos

- ImageNet (vía transfer) o un dataset propio chico.
- Librerías: tensorflow, keras.applications.

Ejercicios

1. Cargar varios modelos: keras.applications.{ResNet50, EfficientNetB0, ConvNeXtTiny}(weights='imagenet'). Comparar model.count_params() y model.summary().
2. Mini-ResNet: implementar 4 bloques residuales: def res_block(x): return x + Conv(64,3,padding='same')(ReLU()(Conv(64,3,padding='same')(x))). Apilar y entrenar.
3. Skip connection a mano: comparar entrenamiento de un "ResNet" sin skip vs con skip a 50 capas. Sin skip no entrena.
4. Depthwise-separable: usar SeparableConv2D en lugar de Conv2D en el mismo modelo. Comparar params y accuracy.
5. Squeeze-Excite: implementar un bloque SE manualmente: s = GlobalAvgPool(x); s = Dense(C//r)(s); s = Dense(C, sigmoid)(s); return x * Reshape((1,1,C))(s).

Homework verificable

Comparar 4 arquitecturas en un dataset propio de imágenes (≥ 5 clases) con transfer learning:

1. ResNet50 (clásico).
2. EfficientNetB0 (eficiente).
3. MobileNetV3Small (móvil).
4. ConvNeXtTiny (moderno).

Para cada uno: feature extraction (frozen) + fine-tune. Reportar accuracy, # parámetros, tiempo de inference.

Criterio de aceptación: ConvNeXt o EfficientNet deben ganar en accuracy; MobileNet en velocidad de inference. ResNet sirve de baseline.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
Cargar ResNet50 sin include_top=False y pa	El head ImageNet espera 1000 clases. Fix:
Cargar EfficientNet y pasarle imágenes en	Espera preprocesamiento específico. Fix: k
Implementar ResNet desde cero y Add() fall	Skip connection requiere shapes iguales —
MobileNet para batch grande en CPU lento	Está optimizado para GPU/mobile. Fix: para
ConvNeXt en TF Lite no exporta	Algunos ops de ConvNeXt aún no portados a

Preguntas frecuentes

¿Cuál arquitectura uso en 2026?

- Default: EfficientNetB0 o ConvNeXtTiny.

- Movil/embedded: MobileNetV3Small.
- Máxima accuracy con budget: ConvNeXt-L o EfficientNet-L2.
- Visión + texto unificado: CLIP / SigLIP (clase 129).

¿ResNet aún se usa?

Sí, como baseline universal. Confiable, bien entendida, todo framework la soporta.

¿ViT supera a CNNs?

Con datasets muy grandes (>10M imágenes), ViT (clase 126) gana. Con datasets típicos del cliente (<100k imágenes), CNNs ganan o empatan. ConvNeXt mostró que CNNs bien modernizadas igualan a ViT.

¿"Compound scaling" qué significa?

EfficientNet showed que escalar depth × width × resolution con coeficientes balanceados (φ) es más eficiente que escalar uno solo. Por eso B0, B1, B2... son la misma red escalada.

¿Tantas opciones por qué?

Cada una optimizó una métrica distinta (accuracy puro, params, latencia, eficiencia mobile). No hay "una mejor" — depende del constraint.

Referencias

- Géron, cap. 14 — CNN Architectures.
- Krizhevsky et al. (2012), AlexNet, NeurIPS.
- Simonyan & Zisserman (2014), VGG.
- Szegedy et al. (2015), Inception (GoogLeNet), CVPR.
- He et al. (2015), Deep Residual Learning (ResNet), CVPR.
- Chollet (2017), Xception.
- Hu et al. (2018), Squeeze-and-Excitation Networks, CVPR.
- Tan & Le (2019), EfficientNet, ICML.
- Liu et al. (2022), A ConvNet for the 2020s (ConvNeXt), CVPR.

Siguiente clase

Clase 131 — Transfer learning con CNNs preentrenadas

Apéndice: notebook (primer bloque)

Primera celda ejecutable del notebook de la clase.

```
# Imports y configuración inicial
```

Archivos complementarios

- notebook.ipynb