Clase 131 — Transfer learning con CNNs preentrenadas

Parte: 2 — Deep Learning · Fuente: Géron, cap. 14 § Using Pretrained Models from Keras.

Duración estimada: 70 min.

Clase 131 — Transfer learning con CNNs preentrenadas

> *Parte: 2 — Deep Learning · Fuente: Géron, cap. 14 § Using Pretrained Models from Keras. Duración estimada: 70 min.*

Objetivo

Aplicar transfer learning específicamente en visión — el caso de uso más común del campo. Profundizar la clase 101 con receta industrial: data augmentation, fine-tuning gradual, learning rate diferencial, y manejo de BatchNorm en fine-tuning (un gotcha clásico).

Resultados de aprendizaje

Al finalizar, el estudiante podrá:

- Construir pipeline con image_dataset_from_directory + augmentation (RandomFlip, RandomRotation, RandomZoom).
- Aplicar el preprocess_input específico del modelo base.
- Hacer fine-tuning en 2 fases con LR diferencial.
- Manejar correctamente BatchNorm en fine-tuning (mantenerlo en inference mode si descongelaste pocas capas).
- Comparar feature extraction (frozen base + nueva head) vs fine-tune (descongelar todo) según tamaño del dataset.

Temas

- Augmentation como capa: RandomFlip, RandomRotation, RandomZoom, RandomCrop, RandomContrast.
- preprocess_input por modelo (cada red espera su escalado).
- 2-stage training: freeze + head warmup → unfreeze + LR bajo.
- BN gotcha: cuando descongelás, BN sigue actualizando moving averages → puede dañar pretraining.
- MixUp, CutMix, RandAugment como augmentations modernas (anticipo).

Definiciones y características

- Augmentation layer: capa Keras que transforma random las imágenes durante training, no en inference.
- preprocess_input: función específica del modelo (keras.applications.<modelo>.preprocess_input).
- Feature extraction: base frozen, solo entreno head. Bueno para n < 1000.
- Fine-tune: descongelar parte/total y reentrenar con LR muy bajo.
- LR diferencial: capas tempranas LR bajo (1e-5), tardías más alto (1e-4).

Dataset / recursos

- Dataset chico propio o tfds.load('cats_vs_dogs') o tfds.load('tf_flowers').
- Modelo base: EfficientNetB0 o MobileNetV3Small.
- Librerías: tensorflow, keras, keras.applications.

Ejercicios

1. Pipeline con augmentation: RandomFlip + RandomRotation(0.1) + RandomZoom(0.1) como capas. Visualizar imágenes augmentadas.
2. Modelo con base + head: base = EfficientNetB0(include_top=False, weights='imagenet'); base.trainable = False; model = Sequential([data_aug, preprocess_input, base, GlobalAvgPool, Dropout, Dense(num_classes)]).
3. Etapa 1 (head warmup): Adam(1e-3), entrenar 5 épocas.
4. Etapa 2 (fine-tune): base.trainable = True, recompilar con Adam(1e-5). Entrenar 10 épocas. Verificar mejora.
5. BN gotcha: con base.trainable = True pero pasando training=False a la base durante fine-tuning → BN no se actualiza. Comparar.

Homework verificable

Sobre un dataset de 5 categorías (e.g., flores):

1. Pipeline completo con augmentation.
2. Transfer learning con EfficientNetB0 en 2 fases.
3. Reportar accuracy de cada fase.
4. Comparar contra entrenar EfficientNet desde cero (sin transfer).

Criterio de aceptación: la etapa 2 mejora la 1 en ≥ 2 pp; transfer learning supera "desde cero" por mucho margen con dataset chico.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
Imágenes pasadas en [0, 255] cuando el mod	Fix: agregar preprocess_input correcto.
Augmentation aplicada en val/test también	Augmentation debe ser solo en train. Fix:
Fine-tuning con LR=1e-3 → catastrophic for	Fix: LR 10-100× más bajo.
Forgot to recompile después de cambiar tra	El optimizer no ve los nuevos pesos como t
BatchNorm en fine-tuning actualiza moving	Para datasets chicos, mantener BN en infer

Preguntas frecuentes

¿Cuántas imágenes mínimas?

50-100 por clase para feature extraction. 200-500 por clase para fine-tuning completo. < 50 → considerar data augmentation agresiva o few-shot learning.

¿Qué proporción de capas descongelo?

Empezá con todo. Si overfittea, congelar las primeras N. Si underfittea, descongelar todo.

¿include_top=False siempre?

Sí para transfer learning (querés head custom).

¿Augmentation en GPU o CPU?

GPU es más rápido si está disponible (with tf.device('/GPU:0')). Augmentation pesada es lo más comúnmente bottleneck en CPU.

¿Test-time augmentation (TTA)?

Predecir varias versiones augmentadas de la misma imagen y promediar. Mejora accuracy 0.5-2 pp. Buen truco para Kaggle.

Referencias

- Géron, cap. 14 — Using Pretrained Models from Keras.
- Yosinski et al. (2014), How transferable are features in deep neural networks?, NeurIPS.
- Cubuk et al. (2020), RandAugment.
- Keras — Image data preprocessing.

Siguiente clase

Clase 132 — Localización, detección, segmentación (+ DETR, Segment Anything, YOLOv11)

Apéndice: notebook (primer bloque)

Primera celda ejecutable del notebook de la clase.

```
# Imports y configuración inicial
```

Archivos complementarios

- notebook.ipynb