Clase 128 — Capas convolucionales, filtros, feature maps

Parte: 2 — Deep Learning · Fuente: Géron, cap. 14 § Convolutional Layers. Duración estimada: 75 min.

Clase 128 — Capas convolucionales, filtros, feature maps

Parte: 2 — Deep Learning · Fuente: Géron, cap. 14 § Convolutional Layers. Duración estimada: 75 min.

Objetivo

Entender la operación de convolución 2D — un filtro K×K se desliza sobre la imagen produciendo un feature map —, los hiperparámetros (filters, kernel_size, strides, padding), por qué las CNN son parameter-efficient vs MLPs (sharing + locality + translation invariance) y cómo aprenden jerarquías visuales (bordes → texturas → partes → objetos).

Resultados de aprendizaje

Al finalizar, el estudiante podrá:

- Aplicar Conv2D(filters=32, kernel_size=3, strides=1, padding='same', activation='relu').
- Calcular el shape de la salida: H' = (H - K + 2P)/S + 1.
- Visualizar feature maps (activaciones intermedias) y filtros aprendidos.
- Diferenciar padding='same' (preserva tamaño) de 'valid' (reduce).
- Calcular el # de parámetros de una Conv2D: K K C_in * C_out + C_out (mucho menos que Dense equivalente).

Temas

- Operación convolución 2D paso a paso.
- Filters / kernels como features detectables.
- Feature maps como representación espacial.
- Stride: paso del filtro.
- Padding: bordes; 'same' agrega zeros para preservar shape.
- Parameter sharing: el mismo filtro se aplica en toda la imagen.

Definiciones y características

- Filter / kernel: matriz pequeña (3×3, 5×5) de pesos aprendibles.
- Feature map: salida de aplicar un filtro a toda la imagen. Una capa con 32 filtros produce 32 feature maps.
- Stride: cuántos píxeles se mueve el filtro entre aplicaciones. 1 = denso; 2 = subsampling.
- padding='same': agrega zeros para que la salida tenga la misma altura/ancho que la entrada.
- padding='valid': sin padding. Salida más chica.
- Receptive field: porción de la imagen original que afecta una neurona específica del feature map.

Dataset / recursos

- MNIST / Fashion-MNIST (1 canal) o CIFAR-10 (3 canales).
- Librerías: tensorflow, keras, matplotlib.

Ejercicios

1. Conv básica: Conv2D(8, 3, padding='same', activation='relu')(input_28x28x1). Verificar shape de salida: (batch, 28, 28, 8).
2. Conteo parámetros: para Conv2D(32, kernel_size=5) aplicado a entrada (28, 28, 1), calcular params (551*32 + 32 = 832).
3. Stride 2: Conv2D(32, 3, strides=2, padding='same')(x). Shape de salida: (batch, H/2, W/2, 32).
4. Visualizar feature maps: entrenar una mini-CNN en MNIST; tomar las activaciones intermedias para una imagen y visualizar.
5. Filtros aprendidos: visualizar model.layers[0].kernel.numpy() para los primeros filtros. Para CIFAR, suelen verse bordes/colores básicos.

Homework verificable

Mini-CNN para Fashion-MNIST:

1. Conv(32, 3) → MaxPool(2) → Conv(64, 3) → MaxPool(2) → Flatten → Dense(128) → Dense(10).
2. Entrenar y reportar accuracy.
3. Visualizar feature maps de la primera Conv para 3 imágenes test.
4. Calcular # parámetros total y compararlo con un MLP equivalente en tamaño (Dense(128) → Dense(64) → Dense(10) sin Conv).

Criterio de aceptación: accuracy de la CNN ≥ 0.91 (mejor que MLP); # parámetros de la CNN MLP equivalente.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
Input shape error expected ndim=4, got ndi	Olvidaste el canal: MNIST viene (60000, 28
Conv2D + Dense sin Flatten o GlobalAverage	Dense espera 2D. Fix: agregá Flatten() o G
padding='valid' con muchas capas → shape s	Cada conv reduce. Fix: usar 'same' por def
Stride alto pierde información	strides=4 con kernel=3 salta píxeles. Fix:
Filtros aprendidos no se ven nada	Inicialización mala o modelo no entrenado.

Preguntas frecuentes

¿Cuántos filtros?

Empieza chico y duplicá al profundizar. Patrón estándar: 32 → 64 → 128 → 256.

¿Kernel 3×3 o 5×5?

3×3 es default moderno (VGG, ResNet). Dos 3×3 apilados tienen el mismo receptive field que 5×5 con menos parámetros.

¿stride o MaxPool para downsampling?

Tradicionalmente MaxPool. ResNet moderno usa stride 2 en Convs y elimina pooling intermedio. Ambos funcionan.

¿Conv1D para series, Conv3D para video?

Sí. Conv1D para secuencias temporales (clase 121 WaveNet). Conv3D para video (T × H × W × C).

¿Convs vs Transformers en visión?

CNN domina en visión clásica (eficiente, buen prior espacial). ViT (Vision Transformer) supera con datasets muy grandes (>10M imágenes). ConvNeXt (2022) recuperó terreno para CNN. Ver clase 126.

Referencias

- Géron, cap. 14 — Deep Computer Vision Using Convolutional Neural Networks.
- LeCun et al. (1998), Gradient-based learning applied to document recognition — LeNet, paper canónico.
- Keras Conv2D.
- CS231n CNN notes — referencia clásica de Stanford.

Siguiente clase

Clase 129 — Pooling

Apéndice: notebook (primer bloque)

Primera celda ejecutable del notebook de la clase.

```
# Imports y configuración inicial
```

Archivos complementarios

- notebook.ipynb