Clase 102 — Keras Sequential API

Parte: 2 — Deep Learning · Fuente: Géron, cap. 10 § The Sequential API. Duración estimada: 60 min.

Clase 102 — Keras Sequential API

> *Parte: 2 — Deep Learning · Fuente: Géron, cap. 10 § The Sequential API. Duración estimada: 60 min.*

Objetivo

Dominar la Sequential API de Keras — la forma más simple y declarativa de construir un modelo cuando es una pila lineal de capas. Saber cuándo NO alcanza (cualquier topología con ramas, skip connections, multi-input/multi-output → Functional API, clase 093).

Resultados de aprendizaje

Al finalizar, el estudiante podrá:

- Construir un modelo con keras.Sequential([...]) o model.add(...) incrementalmente.
- Inspeccionar la arquitectura con model.summary() (parámetros por capa, output shape, total).
- Calcular a mano el número de parámetros de una Dense(n) (= input_dim * n + n por el bias).
- Compilar (compile), entrenar (fit), evaluar (evaluate) y predecir (predict).
- Guardar y cargar con el formato moderno .keras (HDF5 legacy).

Temas

- Dos formas equivalentes: lista en el constructor vs .add() incremental.
- Input layer explícito vs input_shape en la primera capa.
- model.summary(): leer parámetros por capa + total trainable / non-trainable.
- compile(optimizer, loss, metrics).
- fit(X, y, epochs, batch_size, validation_split, callbacks, verbose).
- model.save('m.keras') (formato nativo Keras 3+) y keras.models.load_model('m.keras').

Definiciones y características

- Sequential model: stack lineal — la salida de una capa es la entrada de la siguiente. No permite branching, skip connections, ni múltiples inputs/outputs.
- Parámetros trainable: pesos + bias que el optimizador actualiza. Dense(n_out) aplicada a entrada n_in: n_in * n_out + n_out params.
- Parámetros non-trainable: capas como BatchNormalization tienen estadísticas (running mean/var) que no se actualizan por backprop sino por moving average.
- .keras format: ZIP con config.json + metadata.json + pesos. Reemplaza al .h5 desde Keras 3.
- batch_size: cuántas muestras procesa antes de actualizar pesos. Default 32. Más grande = más estable pero menos updates por época.

Dataset / recursos

- Reutilizar Fashion-MNIST de la clase anterior.
- Librerías: tensorflow, keras (≥ 3.0).

Ejercicios

1. Dos sintaxis: construir el mismo modelo dos veces — una con Sequential([...]) y otra con model = Sequential(); model.add(...). Verificar que summary() produce el mismo resultado.
2. Conteo de parámetros: para Sequential([Dense(128, input_shape=(784,)), Dense(64), Dense(10)]), calcular a mano los parámetros y verificar contra model.summary().
3. Guardado/carga: entrenar 5 épocas, model.save('m.keras'), recargar con load_model, verificar que predict da idéntico.
4. Predict en batch vs individual: predecir 1 sola muestra (¿cómo cambia la shape?) vs predecir 100. Cuidado con la dimensión batch.
5. Verbose: probá fit(..., verbose=0), verbose=1 (barra), verbose=2 (1 línea por época). Útil para notebooks vs scripts.

Homework verificable

Entrenar tres arquitecturas distintas sobre Fashion-MNIST y comparar:

1. Tiny: Dense(64) → Dense(10).
2. Medium: Dense(256) → Dense(128) → Dense(10).
3. Wide: Dense(1024) → Dense(10).

Reportar: # parámetros, accuracy en test, tiempo de entrenamiento.

Criterio de aceptación: el medium debe ganar en accuracy/tiempo balanceado; el wide tiene más parámetros que el medium pero NO necesariamente mejor accuracy (lección: profundo > ancho).

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
ValueError: Input shape mismatch al cargar	El modelo guardado tiene un input shape di
model.summary() muestra ? en output shape	No definiste el input shape de la primera
Olvido compile() y fit() falla	Sin compile, no hay optimizer/loss. Fix: m
Modelo guardado en .h5 y se quiere cargar	.h5 está deprecado. Fix: model.save('m.ker
model.add() después de compile() y fit() y	No se puede modificar la arquitectura post

Preguntas frecuentes

¿Cuándo Sequential no alcanza?

Cuando hay branching (skip connections de ResNet), multiple inputs (modelos de fusión), multiple outputs (multi-task), o capas compartidas. Para todo eso → Functional API (clase 093).

¿Es más lento Sequential que Functional?

No, son la misma cosa por dentro. La diferencia es solo sintáctica.

¿Por qué Input separado y no input_shape=?

Equivalentes funcionalmente. Input(shape=(...)) es más explícito y es lo único que funciona en Functional API; mantenerlo es buena costumbre.

¿batch_size=32 siempre?

Depende. Imágenes: 32–128. Texto/NLP: a menudo 8–16 por memoria. Tabular grande: 256–1024. Probá; el batch_size afecta no solo velocidad sino también la generalización (batches grandes tienden a converger a minima "más planos" pero peor generalizables).

¿validation_split=0.2 o validation_data=(X_val, y_val)?

validation_split=0.2 toma el último 20 % del array; útil si los datos están bien mezclados. validation_data=(...) es explícito y obligatorio si los datos están ordenados.

Referencias

- Géron, cap. 10 — The Sequential API.
- Keras — The Sequential model.
- Keras 3 release notes.

Siguiente clase

Clase 103 — Keras Functional API y Subclassing

Apéndice: notebook (primer bloque)

Primera celda ejecutable del notebook de la clase.

```
# Imports y configuración inicial
```

Archivos complementarios

- notebook.ipynb