Clase 137 — LSTM, GRU

Parte: 2 — Deep Learning · Fuente: Géron, cap. 15 § Tackling Short-Term Memory Problems. Duración estimada: 70 min.

Clase 137 — LSTM, GRU

> *Parte: 2 — Deep Learning · Fuente: Géron, cap. 15 § Tackling Short-Term Memory Problems. Duración estimada: 70 min.*

Objetivo

Entender LSTM (Hochreiter & Schmidhuber 1997) y GRU (Cho et al. 2014) — celdas recurrentes con gates que solucionan el vanishing gradient de SimpleRNN: la información puede fluir sin atenuación por la cell state, y los gates aprenden qué olvidar, recordar y emitir.

Resultados de aprendizaje

Al finalizar, el estudiante podrá:

- Explicar los 3 gates de LSTM: forget, input, output, y la cell state que viaja "horizontal".
- Diferenciar LSTM (3 gates + 2 estados) de GRU (2 gates + 1 estado, más simple, casi igual de bueno).
- Usar keras.layers.LSTM y keras.layers.GRU con return_sequences, return_state, recurrent_dropout.
- Aplicar Bidirectional cuando la tarea lo permite.
- Reconocer que con T > 100, incluso LSTM lucha — preferir Transformers.

Temas

- Cell state c_t (memoria larga) vs hidden state h_t (memoria corta).
- Forget gate: cuánto borrar de c_{t-1}.
- Input gate: cuánto agregar nuevo a c_t.
- Output gate: cuánto exponer en h_t.
- GRU: combina forget e input en una sola "update gate".
- Bidirectional + stacked LSTMs como receta clásica pre-Transformers.

Definiciones y características

- LSTM cell: 3 gates con sigmoid (0-1) que controlan flujo, + tanh para candidatos.
- GRU cell: update + reset gate. Menos parámetros que LSTM.
- recurrent_dropout: dropout aplicado a las conexiones recurrentes (entre timesteps).
- CuDNN kernel: LSTM/GRU tienen implementación cuDNN ultra rápida si cumplís restricciones (default activation tanh, recurrent activation sigmoid, no recurrent_dropout).
- Stacked LSTM: apilar varios → mejor capacidad pero más caro.

Dataset / recursos

- IMDB para sentimiento.
- Serie temporal del ejercicio anterior.
- Librerías: tensorflow, keras.

Ejercicios

1. LSTM vs SimpleRNN: en una serie de 100 pasos con dependencia long-range, comparar accuracy.
2. GRU vs LSTM: misma tarea, comparar. GRU ~ 25 % menos params; accuracy casi igual.
3. Stacked: 2-3 capas LSTM apiladas con return_sequences=True en las primeras.
4. Bidirectional: para sentimiento IMDB, comparar LSTM vs Bidirectional(LSTM).
5. cuDNN check: medir velocidad LSTM vanilla vs con recurrent_dropout (desactiva cuDNN).

Homework verificable

Clasificación de sentimiento sobre IMDB con LSTM:

1. Tokenizar con TextVectorization(max_tokens=20_000, output_sequence_length=200).
2. Embedding(20_000, 128) → Bidirectional(LSTM(64)) → Dense(64) → Dense(1, sigmoid).
3. Entrenar 5 épocas.
4. Reportar accuracy y comparar contra una baseline Dense sin RNN.

Criterio de aceptación: accuracy ≥ 0.85; el modelo recurrente claramente supera al baseline Dense.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
LSTM lento en GPU	Tal vez recurrent_dropout > 0 o unroll=Tru
Padding/masking ignorado	mask_zero=True en Embedding + LSTM que res
Bidirectional rompe en task causal (foreca	Bidirectional ve el futuro. Fix: solo Forw
Stacked LSTM overfittea	Demasiado expresivo. Fix: dropout entre ca
GRU vs LSTM comparados sin suficientes see	Diferencia chica puede ser ruido. Fix: ≥ 3

Preguntas frecuentes

¿LSTM o GRU?

Empezá con LSTM (más conocido). GRU si necesitás eficiencia. La diferencia de accuracy es típicamente < 0.5 pp.

¿LSTM aún se usa en 2026?

Sí, para forecasting de series temporales chicas, sistemas embedded, NER simple. Para NLP serio → Transformers.

¿Cuántas capas?

1-3. Más de 3 raramente justifica el costo. Para tareas serias, mejor un Transformer.

¿recurrent_dropout vs dropout?

dropout se aplica a inputs (entre capas); recurrent_dropout entre timesteps (intra-secuencia). El segundo desactiva cuDNN.

¿Cómo manejo secuencias de longitud variable?

Embedding(..., mask_zero=True) + padding con 0s. LSTM/GRU respetan la máscara automáticamente.

Referencias

- Géron, cap. 15 — Tackling Short-Term Memory Problems.
- Hochreiter & Schmidhuber (1997), Long Short-Term Memory, Neural Computation.
- Cho et al. (2014), Learning Phrase Representations using RNN Encoder–Decoder (GRU), EMNLP.
- Chung et al. (2014), Empirical Evaluation of Gated Recurrent Neural Networks.

Siguiente clase

Clase 138 — 1D CNNs y WaveNet

Apéndice: notebook (primer bloque)

Primera celda ejecutable del notebook de la clase.

```
# Imports y configuración inicial
```

Archivos complementarios

- notebook.ipynb