Clase 143 — Transformers: arquitectura, BERT, GPT (+ Flash Attention, RoPE, GQA)

Parte: 2 — Deep Learning · Fuente: Géron, cap. 16 § The Transformer Architecture + Vaswani et al. (2017) + papers BERT, GPT, FlashAttention. Duración estimada: 100 min.

Clase 143 — Transformers: arquitectura, BERT, GPT (+ Flash Attention, RoPE, GQA)

> *Parte: 2 — Deep Learning · Fuente: Géron, cap. 16 § The Transformer Architecture + Vaswani et al. (2017) + papers BERT, GPT, FlashAttention. Duración estimada: 100 min.*

Objetivo

Dominar la arquitectura Transformer —encoder, decoder, ambas variantes (BERT encoder-only, GPT decoder-only, T5 encoder-decoder)— a nivel de poder implementarla a mano. Conocer las mejoras clave 2022-2024 que hacen a los LLMs modernos rápidos y eficientes: Flash Attention v2/v3, RoPE (Rotary Position Embeddings), Grouped-Query Attention (GQA).

Resultados de aprendizaje

Al finalizar, el estudiante podrá:

- Dibujar el block de Transformer: LayerNorm → MultiHeadAttention → +residual → LayerNorm → FFN → +residual.
- Implementar positional encoding sinusoidal o aprendible.
- Reconocer 3 variantes: encoder-only (BERT), decoder-only (GPT), encoder-decoder (T5, Whisper).
- Saber qué hace cada mejora moderna: Flash Attention (O(N) memoria, 2-3× speedup), RoPE (mejor extrapolación a secuencias largas), GQA (compartir KV heads → menos KV cache en inference).
- Cargar y usar un Transformer chico con keras.layers.MultiHeadAttention o desde Hugging Face.

Temas

- Block Transformer: LN → MHA → res → LN → FFN → res.
- Positional encoding: por qué se necesita (attention es permutation-invariant) — sin → aprendible → RoPE.
- BERT: encoder-only, MLM + NSP, bidireccional.
- GPT: decoder-only, next-token, causal mask.
- T5 / BART: encoder-decoder, span-corruption / denoising.
- Complemento moderno: Flash Attention, RoPE, GQA, MQA.

Versión profundizada — 2026

El tema moderno que vivía como complemento dentro de esta clase ahora tiene clase propia dedicada con patrón completo, ejercicios y homework:

- Clase 126b — Flash Attention v2/v3, RoPE, GQA: el motor de los LLMs modernos

Definiciones y características

- Transformer block: bloque básico repetido N veces.
- MHA (Multi-Head Attention): H heads paralelos, cada uno con Q, K, V proyectados.

- Positional encoding: información de orden inyectada en embeddings.
- BERT: encoder-only, bidirectional, masked LM pre-training.
- GPT: decoder-only, causal mask, autoregressive pre-training.
- T5: encoder-decoder, span-corruption pre-training.
- Flash Attention: implementación memory-efficient de scaled-dot-product attention.
- RoPE: rotación de Q/K en función de posición.
- GQA: heads de KV compartidos entre grupos de heads de Q.
- KV cache: almacenamiento de K y V de tokens previos para inference autoregresiva eficiente.

Dataset / recursos

- WikiText-2 o similar.
- Modelos preentrenados desde HF (clase 127).
- Librerías: tensorflow, keras, transformers.

Ejercicios

1. Transformer block desde cero: implementar def transformer_block(x): x = x + mha(LN(x)); x = x + ffn(LN(x)); return x con MultiHeadAttention y FFN.
2. Positional encoding sinusoidal: implementar la fórmula original de Vaswani; visualizar como heatmap.
3. Mini-GPT: 4 capas Transformer con causal mask. Entrenar en next-token sobre Tiny Shakespeare. Comparar con char-RNN (122).
4. RoPE manual: implementar la rotación. Aplicar y verificar que attention(Q_rot, K_rot) da diferencia relativa.
5. HuggingFace check: cargar bert-base-uncased y gpt2; imprimir model.config. Identificar num_attention_heads, hidden_size, intermediate_size.

Homework verificable

Entrenar un mini-GPT desde cero sobre Tiny Shakespeare:

1. 4 capas Transformer decoder con causal mask.
2. d_model=128, n_heads=4, vocab=char-level.
3. Train 20 épocas; generar 1000 caracteres con temperatura 0.7.
4. Comparar val_loss y muestras vs char-RNN (clase 122).

Criterio de aceptación: val_loss < 1.3 (mejor que char-RNN); samples más coherentes con estructura de obra teatral.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
Modelo entrena pero outputs son random	Olvido del causal mask en decoder. Fix: us
LayerNorm antes vs después: dudas	"Pre-LN" (antes) entrena mucho mejor que "
Positional encoding aprendible explota	Init mal. Fix: small std (0.02) o usar RoP
Sin gradient checkpointing → OOM en secuen	Fix: tf.recompute_grad o gradient checkpoi
FFN sin expansión	Default es d_ff = 4 * d_model. Fix: respet

Preguntas frecuentes

¿Encoder-only o decoder-only o ambos?

Encoder-only (BERT): clasificación, NER, similaridad. Decoder-only (GPT): generación, LLMs modernos. Encoder-decoder (T5, Whisper): traducción, transcripción.

¿GPT-style "es todo lo que necesitás" hoy?

Para LLMs generales sí. BERT-style sigue siendo eficiente para clasificación.

¿LayerNorm o RMSNorm?

RMSNorm (Zhang & Sennrich 2019) es estándar moderno — más rápida sin pérdida de calidad. Lo usan Llama, Mistral, etc.

¿Por qué RoPE en lugar de aprendible?

(a) Mejor extrapolación a longitudes no vistas, (b) propiedad relativa útil, (c) no agrega parámetros.

¿GQA solo para inference?

Se entrena con GQA desde el principio. Beneficio principal es inference; training también es marginalmente más rápido.

Referencias

- Géron, cap. 16 — The Transformer Architecture.
- Vaswani et al. (2017), Attention Is All You Need, NeurIPS.
- Devlin et al. (2018), BERT, NAACL.
- Radford et al. (2018-2019), GPT, GPT-2, OpenAI.
- Dao et al. (2022, 2023, 2024), FlashAttention v1/v2/v3.
- Su et al. (2021), RoFormer: Enhanced Transformer with Rotary Position Embedding.
- Ainslie et al. (2023), GQA: Training Generalized Multi-Query Transformer Models.

Siguiente clase

Clase 144 — Flash Attention v2/v3, RoPE, GQA: el motor de los LLMs modernos

Apéndice: notebook (primer bloque)

Primera celda ejecutable del notebook de la clase.

```
# Imports y configuración inicial
```

Archivos complementarios

- notebook.ipynb