Facsímil 03

Arquitecturas y modelos

Capítulo 05

Arquitecturas modernas: familias, MoE, razonamiento y multimodalidad

686f6c61 · 2026

Capítulo 05: Arquitecturas modernas: familias, MoE, razonamiento y multimodalidad

Tres formas de hacer que un modelo sea más capaz

Hasta ahora hemos mirado el Transformer clásico por dentro: tokens, tensores, atención, Q, K, V, residual, LayerNorm, MLP, logits y sampling. Ya tenemos la maquinaria básica. Ahora aparece una pregunta natural: si eso funciona, ¿cómo lo hacemos más capaz sin limitarnos a hacerlo todo más grande?

Las arquitecturas modernas suelen tocar tres palancas:

PALANCA	PREGUNTA QUE RESPONDE	EJEMPLO
Capacidad	¿Cómo meto más conocimiento y patrones sin pagar todo el coste en cada token?	Mixture of Experts, o MoE.
Proceso	¿Cómo doy más tiempo de cálculo a problemas que necesitan varios pasos?	Razonamiento, varias muestras, verificación.
Entrada	¿Cómo hago que el modelo trabaje con texto, imágenes, audio o acciones?	Modelos multimodales.

Este capítulo va de esas tres palancas. No son trucos aislados. Son respuestas distintas a la misma tensión: queremos modelos más útiles, pero tenemos límites de memoria, latencia, coste, datos y evaluación.

Qué no significan estas palabras

MoE no significa que haya expertos humanos dentro del modelo. La palabra experto engaña un poco. En la práctica, un experto suele ser una red interna, normalmente parecida al MLP del capítulo anterior, y un router decide qué expertos se activan para cada token.

Razonamiento no significa que el modelo piense como una persona. En IA moderna, muchas veces llamamos razonamiento a dedicar más pasos de generación, probar varias rutas, verificar resultados o entrenar el modelo para resolver tareas paso a paso. Eso puede mejorar resultados, pero no convierte al modelo en una mente humana.

Multimodal no significa que el modelo vea como tú. Un modelo puede procesar una imagen, pero lo hace transformándola en representaciones numéricas. A veces esas representaciones son muy útiles; otras veces fallan en detalles espaciales, texto pequeño, conteos o relaciones visuales delicadas.

Más moderno no significa automáticamente mejor para tu caso. Un modelo MoE puede ser excelente, pero más complejo de servir. Un modelo multimodal puede ser muy útil, pero más caro y más difícil de evaluar. Un modelo con razonamiento puede resolver mejor ciertas tareas, pero tardar más o necesitar controles de calidad distintos.

El mapa amplio de arquitecturas

Antes de entrar en MoE, razonamiento y multimodalidad, conviene abrir el plano completo. No existe una lista cerrada de «todas las arquitecturas», porque la investigación mezcla familias continuamente. Pero sí hay un mapa útil para no perderse.

Una primera separación:

NIVEL	QUÉ ES	EJEMPLO
Arquitectura neuronal	La forma interna de la red.	Transformer, CNN, RNN, MoE, Mamba, difusión.
Modelo entrenado	Una arquitectura con pesos aprendidos en datos concretos.	BERT, GPT-2, T5, Mixtral, CLIP.
Arquitectura de sistema	Cómo conectas modelos, datos, herramientas y validadores.	RAG, agente con herramientas, pipeline multimodal.

Esta distinción evita una confusión muy común: RAG, por ejemplo, no es una capa neuronal como LayerNorm. Es una arquitectura de sistema que combina recuperación de información con generación.[1]

Familias neuronales clásicas y actuales

FAMILIA	IDEA CENTRAL	SE USA MUCHO EN	QUÉ DEBES RECORDAR
MLP	Capas densas que transforman vectores.	Tabular, bloques internos de Transformers.	No mezcla posiciones; transforma representaciones.
CNN	Filtros locales que detectan patrones espaciales.	Imagen, audio, visión industrial.	Muy fuertes cuando importa la vecindad local.[2]
RNN / LSTM / GRU	Procesan secuencias manteniendo un estado.	Series temporales, lenguaje antes del dominio Transformer.	Buen punto histórico para entender secuencia, aunque hoy compiten con Transformers y SSM.[3]
Transformer	Atención para mezclar información entre posiciones.	LLMs, traducción, visión, multimodalidad.	Es la familia dominante en lenguaje moderno, pero no la única.
GNN	Mensajes entre nodos de un grafo.	Moléculas, redes, conocimiento estructurado.	Sirven cuando la estructura no es una secuencia sino un grafo.[4]
Autoencoders / VAE	Comprimir y reconstruir datos mediante una representación latente.	Generación, compresión, representación.	Enseñan la idea de espacio latente.[5]
Diffusion models	Aprenden a quitar ruido paso a paso.	Imagen, audio, vídeo, generación visual.	Generan invirtiendo un proceso de ruido.[6]
State Space Models	Mantienen un estado y procesan secuencias con coste favorable.	Secuencias largas, lenguaje, audio, señales.	Mamba reabrió el interés por alternativas al Transformer puro.[7]

Una de esas familias merece una nota aparte, porque es la alternativa al Transformer que más interés ha despertado: los **State Space Models** (SSM), popularizados por Mamba. Su atractivo es de coste. Recuerda que la atención del capítulo 3 compara cada token con cada token, un coste $O(n^2)$. Un SSM, en cambio, procesa la secuencia manteniendo un **estado** que resume el pasado y se actualiza token a token, como una RNN moderna: su coste crece **lineal** con la longitud, $O(n)$, no cuadrático. Eso lo hace muy atractivo para secuencias muy largas (audio, genómica, documentos enteros). El precio es que comprimir todo el pasado en un estado fijo puede perder el detalle fino que la atención sí captura, y por eso muchas arquitecturas recientes son **híbridas**: combinan capas de atención con capas de SSM para quedarse con lo mejor de cada una.

Familias Transformer en lenguaje

Dentro de Transformers hay varias arquitecturas que conviene distinguir:

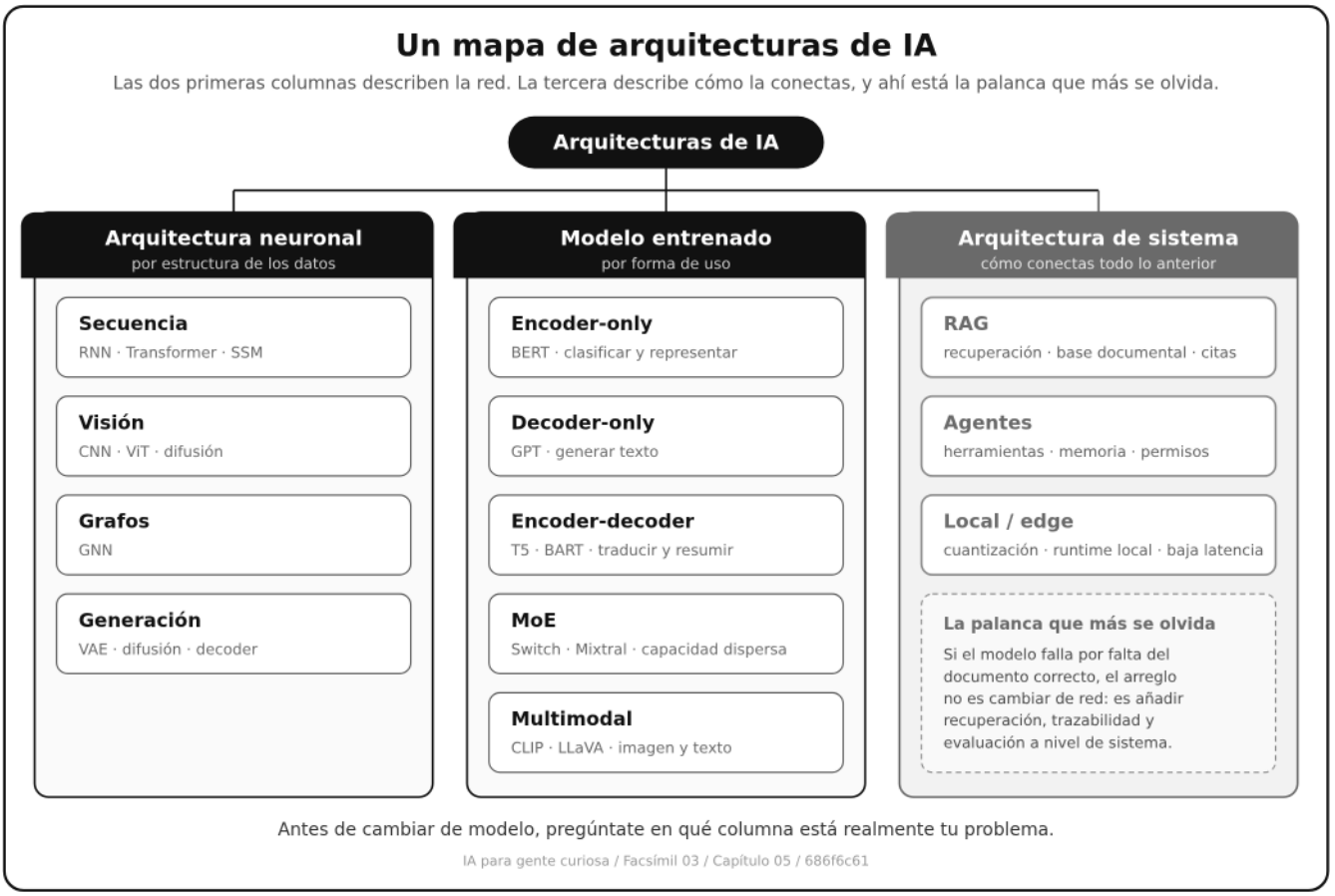
FAMILIA	CÓMO LEE	CÓMO GENERA	EJEMPLO	USO TÍPICO
Encoder-only	Mira el texto completo a ambos lados.	No está pensado principalmente para generar de izquierda a derecha.	BERT.[8]	Clasificación, búsqueda semántica, extracción, embeddings.
Decoder-only	Mira solo el pasado disponible.	Predice el siguiente token.	GPT-2.[9]	Chat, completado, código, generación libre.
Encoder-decoder	Un encoder lee la entrada; un decoder genera la salida.	Genera condicionándose en lo leído.	T5 y BART.[10] BART combina un encoder bidireccional y un decoder autoregresivo.[11]	Traducción, resumen, transformación de texto.
Vision Transformer	Divide imágenes en parches tratados como tokens.	Depende de la tarea: clasificar, detectar, generar en sistemas híbridos.	ViT.[12]	Visión y multimodalidad.
MoE Transformer	Usa atención y sustituye partes densas por expertos.	Igual que su base, pero con cómputo disperso.	Switch, Mixtral.	Más capacidad con menos parámetros activos por token.
Long-context / eficiente	Cambia atención, memoria o estado para manejar contextos largos.	Puede ser decoder, encoder o híbrido.	Mamba, variantes de atención eficiente.	Documentos largos, audio, series, agentes con memoria.

La pregunta útil no es «¿cuál es mejor?», sino «¿qué forma de leer y generar necesita mi problema?».

Arquitecturas de sistema que parecen arquitectura neuronal, pero no lo son

SISTEMA	QUÉ COMBINA	CUÁNDO APARECE
RAG	Recuperador, base documental, modelo generativo y citas o validadores.	Preguntas sobre conocimiento cambiante o privado.
Agente con herramientas	Modelo, herramientas, permisos, memoria, evaluadores y orquestación.	Tareas de varios pasos con acciones externas.
Pipeline multimodal	Codificadores de imagen/audio/documento, conectores y LLM.	Facturas, fotos, capturas, vídeo, voz.
Modelo local cuantizado	Pesos comprimidos, runtime local y límites de hardware.	Privacidad, coste, edge AI, baja latencia local.

Esto importa porque a veces se intenta resolver con arquitectura neuronal lo que en realidad es un problema de sistema. Si tu modelo falla porque no tiene el documento correcto, quizá no necesitas cambiar de LLM: necesitas recuperación, trazabilidad y evaluación.



Un ejemplo cotidiano: el ticket de soporte

Imagina que una tienda recibe este mensaje:

> Compré unos auriculares ayer. Han llegado rotos y necesito cambiarlos antes del viernes porque son para un regalo.

Un sistema sencillo podría mandar todo el texto al mismo modelo denso y pedir una respuesta. Eso funciona muchas veces. Pero una arquitectura moderna puede repartir mejor el trabajo:

PARTE DEL PROBLEMA	QUÉ NECESITA ENTENDER
«han llegado rotos»	Incidencia de producto o envío.
«cambiarlos»	Política de cambios y devoluciones.
«antes del viernes»	Urgencia temporal.
«son para un regalo»	Tono empático y prioridad percibida.

Un MoE intentaría activar rutas internas útiles para ese tipo de patrón. Un sistema con razonamiento podría separar pasos: identificar intención, comprobar restricciones, decidir respuesta, revisar si falta información. Un modelo multimodal entraría si el usuario adjunta una foto de los auriculares dañados o una captura del pedido.

Esta es la idea del capítulo: no nos interesan las siglas por sí mismas. Nos interesa qué problema resuelve cada arquitectura.

MoE: más capacidad sin activar todo

En el capítulo 04 vimos que cada bloque Transformer tiene un MLP. En muchos modelos MoE, esa parte se sustituye por varios MLPs llamados expertos. Para cada token, un router elige uno o varios expertos y solo esos se ejecutan.

La idea de activar partes del modelo según la entrada aparece en las capas sparsely-gated Mixture-of-Experts de Shazeer y coautores.[13]

Después, GShard llevó esta idea a escalado con sharding automático y cómputo condicional.[14] Switch Transformer simplificó el routing activando un experto por token.[15]

La fórmula básica:

$$s = W_r x$$

$$g = \mathrm{softmax}(s)$$

$$C_k = \mathrm{TopK}(g, k)$$

$$\mathrm{MoE}(x) = \sum_{i \in C_k} \frac{g_i}{\sum_{j \in C_k} g_j} E_i(x)$$

SÍMBOLO	SIGNIFICADO	EJEMPLO
x	Vector del token o posición que entra al bloque.	Representación de «rotos».
W_r	Matriz del router.	Produce puntuaciones para expertos.
s	Logits del router.	Una puntuación por experto.
g	Pesos tras softmax.	Probabilidad de usar cada experto.
C_k	Expertos seleccionados.	Si $k = 2$, se activan dos expertos.
$E_i(x)$	Salida del experto i.	MLP especializado por entrenamiento.
$\mathrm{MoE}(x)$	Mezcla final de expertos activados.	Sustituye o complementa al MLP denso.

Ejemplo pequeño con cuatro expertos:

EXPERTO	PESO DEL ROUTER G_I
E_1	0,62
E_2	0,24
E_3	0,09
E_4	0,05

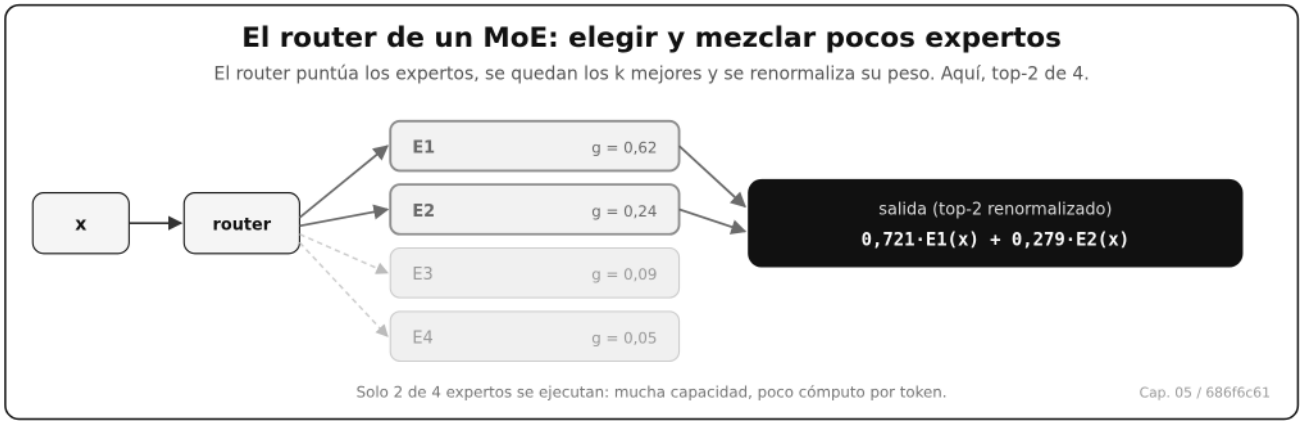
Si usamos top-2, conservamos E_1 y E_2. Renormalizamos:

$$\hat{g}_1 = \frac{0{,}62}{0{,}62 + 0{,}24} \approx 0{,}721$$

$$\hat{g}_2 = \frac{0{,}24}{0{,}62 + 0{,}24} \approx 0{,}279$$

La salida sería:

$$\mathrm{MoE}(x) \approx 0{,}721 E_1(x) + 0{,}279 E_2(x)$$



La clave: el modelo puede tener muchos parámetros, pero no los usa todos en cada token. Eso aumenta capacidad sin multiplicar de la misma forma el coste por token. Mixtral 8x7B es un ejemplo conocido de modelo de lenguaje sparse MoE: tiene varios expertos por capa y activa solo una parte para cada token.[16]

Parámetros totales frente a activos: los números reales

Aquí está la contabilidad que hace especial al MoE, y conviene verla con cifras. Mixtral 8x7B no son «ocho modelos de 7B»: comparte la atención y solo desdobla los MLP en ocho expertos, así que suma unos **47B de parámetros totales** pero, al activar dos expertos por token, usa solo unos **13B activos** en cada paso. La generación cuesta como un modelo de 13B, pero el modelo «sabe» como uno mucho mayor. El caso llevado al extremo es DeepSeek-V3: **671B de parámetros totales y apenas unos 37B activos por token**, alrededor del 5,5 %.[17] Por eso, al leer la ficha de un MoE, la pregunta no es «¿cuántos parámetros tiene?» sino **dos**: cuántos totales (cuánta memoria hace falta para cargarlo) y cuántos activos (cuánto cuesta cada token). A 2025-2026 esta separación es ya la norma: los modelos abiertos punteros convergieron en combinar MoE con una atención eficiente, del tipo GQA del capítulo 3, para mantener a raya tanto la memoria de pesos como la KV cache.[18]

La parte incómoda de MoE

MoE suena precioso en una frase: más capacidad, menos cómputo activo. Pero la ingeniería no sale gratis.

PROBLEMA	QUÉ OCURRE	POR QUÉ IMPORTA
Balanceo	El router puede mandar demasiados tokens a pocos expertos.	Algunos expertos se saturan y otros apenas aprenden.
Comunicación	En entrenamiento distribuido, los tokens viajan hacia expertos en distintos dispositivos.	Puede aumentar latencia y complejidad.
Servicio	No basta con contar parámetros totales.	Importa cuántos expertos se activan, dónde viven y cómo se cargan.
Interpretación	Un experto activado no siempre tiene una etiqueta humana clara.	No podemos decir sin prueba: «este experto sabe medicina».

Una analogía útil: MoE no es una empresa con departamentos perfectamente etiquetados. Se parece más a un edificio con varias salas de trabajo y un recepcionista que aprende a enviar cada ficha a unas salas concretas. Algunas salas acaban especializándose, pero esa especialización no tiene por qué coincidir con nuestras categorías humanas.

¿Y cómo se evita que el recepcionista mande casi todo a dos salas y deje el resto vacías? Con un empujón durante el entrenamiento: una **pérdida auxiliar de balanceo** (*load-balancing loss*) que penaliza al router cuando reparte de forma muy desigual y lo anima a usar todos los expertos de manera parecida. Sin ese término, el MoE tiende a colapsar (unos pocos expertos acaparan el trabajo y los demás nunca aprenden); con él, la carga se distribuye y la capacidad del modelo se aprovecha. Es un detalle pequeño en la fórmula pero decisivo en la práctica: buena parte del arte de entrenar un MoE está en equilibrar la calidad de las respuestas con ese reparto de la carga.

Razonamiento: más proceso, no solo más parámetros

En el uso cotidiano decimos que un modelo razona cuando resuelve algo que parece necesitar pasos: un problema matemático, una planificación, una comparación de opciones, una depuración de código o una decisión con restricciones.

En términos técnicos, muchas mejoras de razonamiento vienen de cambiar **cómo se usa** el modelo durante inferencia o entrenamiento:

TÉCNICA	QUÉ HACE	EJEMPLO
Chain-of-thought	Usa ejemplos o instrucciones con pasos intermedios.	«Primero calcula el tiempo, luego comprueba la restricción».
Self-consistency	Genera varias soluciones y elige la respuesta más consistente.	Cinco rutas llegan a «14:30»; una ruta llega a «14:00».
Tree of Thoughts	Explora varias ramas de solución y evalúa avances parciales.	Planificar una agenda probando varias secuencias.
Verificación	Comprueba el resultado con reglas, código, tests o herramientas.	Ejecutar una consulta SQL o un test unitario.

El paper de chain-of-thought mostró que dar ejemplos con pasos intermedios podía mejorar tareas de razonamiento en modelos grandes.[19] Self-consistency propuso generar varias rutas y escoger la respuesta que aparece de forma más consistente.[20] Tree of Thoughts extendió esa idea hacia búsqueda sobre pasos intermedios.[21]

Podemos escribir self-consistency de forma simple:

$$r^{(m)} \sim p(r \mid x)$$

$$a^{(m)} = \text{extraer\_respuesta}(r^{(m)})$$

$$\hat{a} = \text{moda}(a^{(1)}, a^{(2)}, \ldots, a^{(M)})$$

SÍMBOLO	SIGNIFICADO	EJEMPLO
x	Problema de entrada.	«¿A qué hora debo salir?»
$r^{(m)}$	Ruta de solución número m.	Un intento de resolver paso a paso.
M	Número de rutas generadas.	$M = 5$.
$a^{(m)}$	Respuesta extraída de la ruta m.	«14:30».
$\hat{a}$	Respuesta final por consistencia.	La que más se repite o mejor verifica.

Ejemplo:

RUTA	RESPUESTA
1	14:30
2	14:30
3	14:00
4	14:30
5	14:45

La moda es 14:30. No garantiza verdad absoluta, pero reduce la dependencia de una sola muestra. En tareas profesionales, lo fuerte no es pedir «razona más» sin más. Lo fuerte es combinar pasos, herramientas, verificaciones y criterios de aceptación.

El giro de 2025: pensar más en inferencia

Estas técnicas apuntan a una idea que ha reorganizado el campo: el **escalado en tiempo de inferencia** (*test-time compute*). Durante años, mejorar un modelo significaba entrenarlo más grande. En 2025 se consolidó una vía complementaria: dejar fijos los pesos y **gastar más cómputo al generar la respuesta**, produciendo cadenas de razonamiento mucho más largas, explorando varias y verificándolas.[22] Los **modelos de razonamiento** (OpenAI o1 y o3, DeepSeek-R1 o Gemini con modo de pensamiento) nacen de aquí: generan una larga deliberación interna antes de responder. DeepSeek-R1 enseñó algo notable: esa capacidad puede incentivarse con **aprendizaje por refuerzo puro**, sin trayectorias de razonamiento escritas a mano, y alcanzó el nivel de o1 en matemáticas y código.[23] El matiz de ingeniería: no sale gratis. Más cómputo en inferencia es más latencia y más coste por respuesta, y la mejora **satura**: rinde mucho en problemas de varios pasos, como matemáticas o código, y bastante menos en preguntas de sentido común. La decisión, otra vez, es un compromiso medible, no un eslogan.

Modo ingeniero: routing y consistencia en código

Las dos primeras palancas caben en muy poco `numpy`. El router de un MoE es softmax, top-k y renormalizar; la self-consistency es muestrear y quedarse con la moda.

```
import numpy as np
from collections import Counter

def router_moe(x, W_r, k=2):
    s = W_r @ x                              # logits del router, uno por experto
    g = np.exp(s - s.max()); g = g / g.sum()  # softmax
    top = np.argsort(g)[::-1][:k]            # los k expertos de mayor peso
    pesos = g[top] / g[top].sum()            # renormalizar entre los elegidos
    return top, pesos                        # qué expertos y con qué peso mezclar

# con g = [0.62, 0.24, 0.09, 0.05] y k=2 -> expertos (0,1), pesos [0.721, 0.279]

def self_consistency(respuestas):
    return Counter(respuestas).most_common(1)[0][0]    # la moda

print(self_consistency(["14:30","14:30","14:00","14:30","14:45"]))   # 14:30
```

El router devuelve qué expertos se activan y con qué peso se mezclan sus salidas, exactamente la fórmula de MoE; y `self_consistency` se queda con la respuesta que más se repite entre varias rutas. Es lo que aísla el lab de este capítulo: las tres palancas con datos pequeños, para ver el mecanismo sin un modelo gigante delante.

Un ejemplo entendible: planificar una salida

Supongamos:

> La reunión empieza a las 16:00. Tardo 25 minutos en taxi. Quiero llegar 10 minutos antes. Necesito 15 minutos para preparar la mochila. ¿A qué hora empiezo a prepararme?

Un modelo que responde rápido podría saltar a una respuesta plausible. Un proceso más cuidadoso separa restricciones:

PASO	CÁLCULO	RESULTADO
Llegar 10 minutos antes	16:00 - 10 min	15:50
Restar taxi	15:50 - 25 min	15:25
Restar preparación	15:25 - 15 min	15:10

La respuesta es: empezar a prepararse a las **15:10**.

La lección no es que el modelo tenga una calculadora perfecta dentro. La lección es que algunas tareas mejoran cuando obligamos al sistema a separar pasos, comprobar restricciones y, si hace falta, usar una herramienta externa. Esto conecta directamente con facsímil 5, donde hablaremos de agentes y herramientas, y con facsímil 7, donde hablaremos de evaluación.

Multimodalidad: convertir otros mundos en tokens útiles

Un modelo de lenguaje trabaja con tokens. Para trabajar con imágenes, audio, vídeo o acciones, necesita convertir esas señales en representaciones que puedan relacionarse con el lenguaje.

¿Cómo se convierte una imagen en algo parecido a tokens? La receta dominante viene del **Vision Transformer** (ViT): se corta la imagen en una cuadrícula de **parches** (por ejemplo, recuadros de 16×16 píxeles), cada parche se aplana y se proyecta a un vector, y a partir de ahí esos vectores se tratan **como si fueran tokens** de una frase.[24] Una foto deja de ser píxeles y pasa a ser una secuencia de «palabras visuales» que un Transformer puede atender igual que atiende texto. Sobre esa idea se construye casi toda la visión moderna en LLMs.

Hay varias familias de aproximaciones:

FAMILIA	IDEA	EJEMPLO
Alineación contrastiva	Aprender que una imagen y su texto cercano deben quedar cerca en un espacio común.	CLIP.
Conectores visuales	Usar un codificador de imagen y proyectar sus salidas al LLM.	LLaVA.
Cross-attention	Dejar que el modelo de lenguaje atienda a representaciones visuales.	Flamingo.
Modelos incorporados en entornos	Combinar lenguaje, visión y señales de acción u observación.	PaLM-E.

CLIP entrenó un codificador de imagen y otro de texto para acercar pares imagen-texto correctos y separar pares incorrectos.[25] Una similitud típica es el coseno:

$$\mathrm{sim}(v, t) = \frac{v \cdot t}{\|v\|\|t\|}$$

SÍMBOLO	SIGNIFICADO	EJEMPLO
v	Vector de imagen.	Representación de una foto de un perro.
t	Vector de texto.	Representación de «foto de un perro».
$v \cdot t$	Producto punto.	Mide alineación entre vectores.
$\|v\|$, $\|t\|$	Normas de los vectores.	Sirven para normalizar magnitudes.
$\mathrm{sim}(v, t)$	Similitud coseno.	Cerca de 1 si encajan mucho.

Ejemplo:

TEXTO CANDIDATO	SIMILITUD CON LA IMAGEN
«auriculares rotos»	0,82
«caja de zapatos»	0,31
«perro en un parque»	0,08

El sistema no «ve» como una persona; compara representaciones. Esa comparación puede ser muy útil para buscar, clasificar o describir, pero hay que evaluarla en el dominio real.

Modelos posteriores conectaron visión y lenguaje de formas más generativas. Flamingo combinó modelos visuales y de lenguaje para manejar secuencias intercaladas de imágenes, vídeo y texto.[26] LLaVA conectó un codificador visual con un LLM y usó ajuste con instrucciones visuales para diálogo imagen-texto.[27] PaLM-E exploró modelos que integran lenguaje, visión y señales de entornos físicos.[28]

El patrón común: adaptar representaciones

Aunque las arquitecturas cambien, hay una receta recurrente:

```
entrada no textual -> codificador -> vectores -> conector -> modelo de lenguaje -> salida
```

Para una imagen:

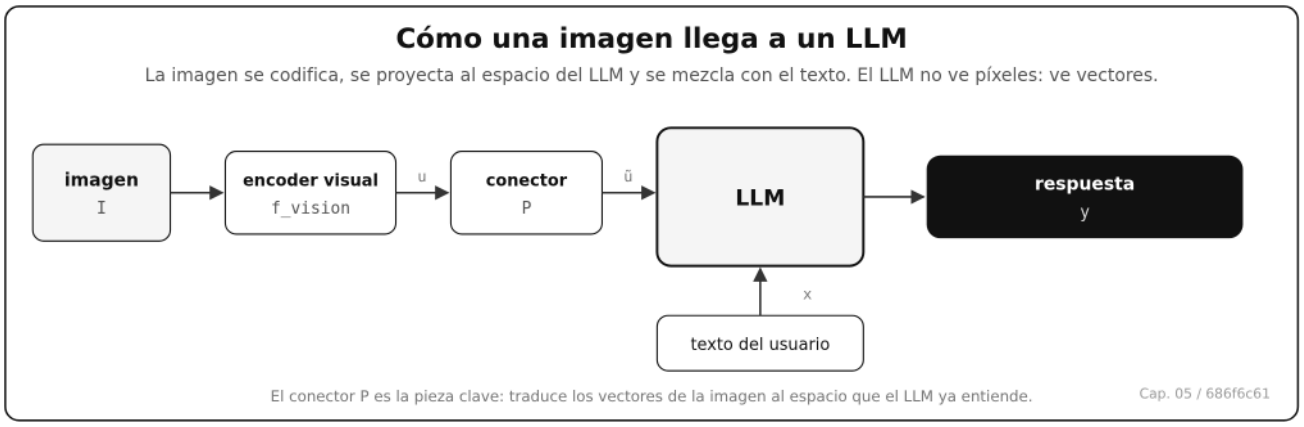
$$u = f_{\text{vision}}(I)$$

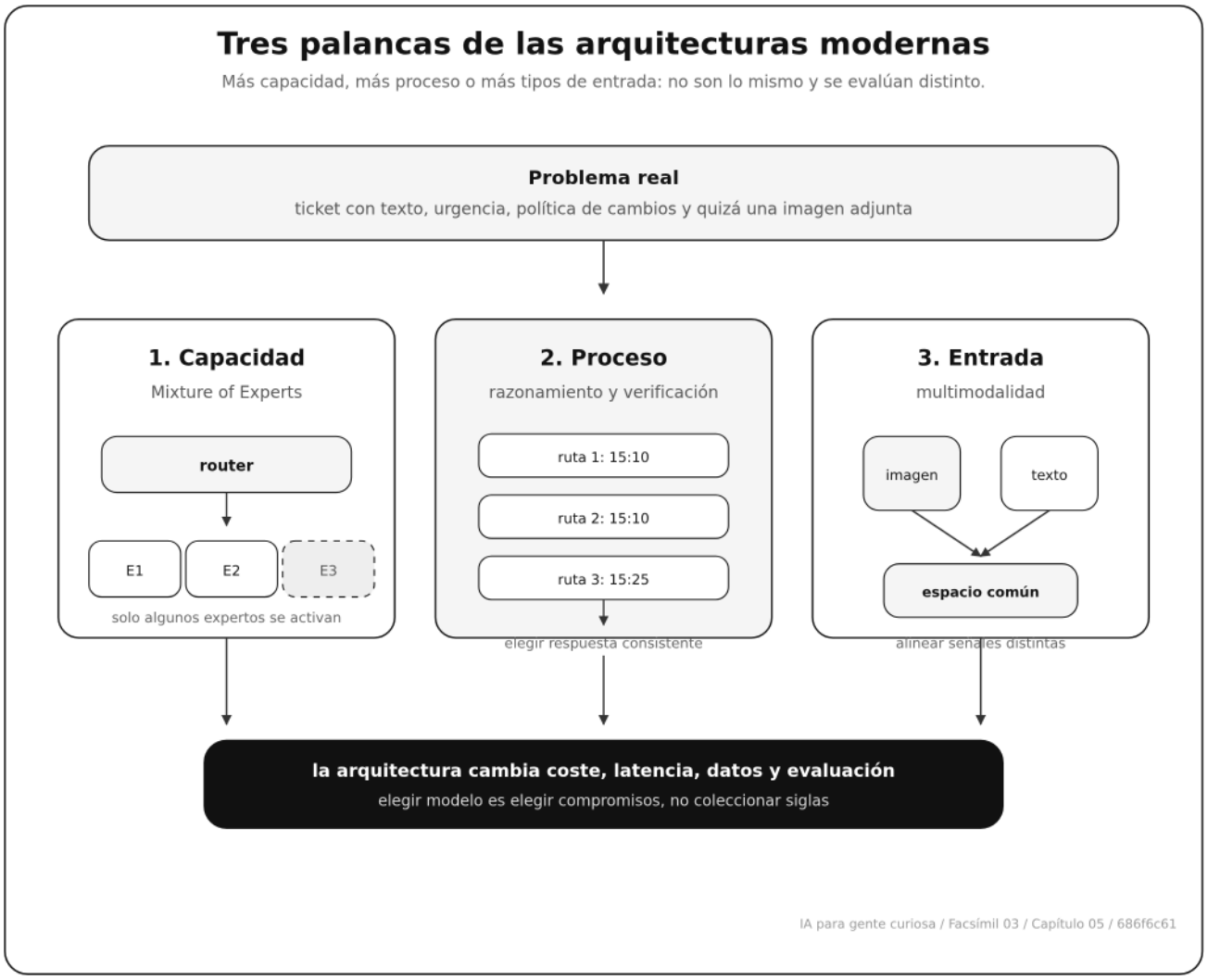
$$\tilde{u} = P(u)$$

$$y \sim p(y \mid \tilde{u}, x_{\text{text}})$$

SÍMBOLO	SIGNIFICADO	EJEMPLO
I	Imagen de entrada.	Foto del producto dañado.
f_{vision}	Codificador visual.	Convierte píxeles en vectores.
u	Tokens o vectores visuales.	Representaciones de regiones o parches.
P	Proyector o conector.	Adapta dimensión y formato al LLM.
$\tilde{u}$	Representación visual ya conectada.	Lo que el LLM puede consumir.
x_{text}	Texto del usuario.	«¿Me lo cambian?»
y	Respuesta generada.	Explicación o siguiente acción.

El punto importante: multimodalidad no es solo añadir una imagen al prompt. Hay que decidir cómo se codifica, cómo se alinea, en qué capas se integra, qué datos se usaron para entrenar y cómo se evalúa el resultado.





Hacia dónde va esto

Conviene cerrar mirando la dirección del campo, con la cautela de que esto cambia muy rápido (fecha de corte: 10 de junio de 2026). Las tres palancas de este capítulo no se han quedado quietas: se han combinado, y de esa combinación salen las tendencias que hoy dominan el estado del arte.

Convergencia hacia MoE disperso con atención eficiente. Los modelos abiertos punteros han convergido en una receta común: capas MoE para tener muchos parámetros sin pagarlos todos en cada token, más una atención de bajo coste de memoria (GQA o variantes como la *multi-head latent attention*). DeepSeek-V3, con 671B totales y unos 37B activos, ejemplifica hacia dónde va la escala: enorme en conocimiento, contenida en cómputo por token.[29]

Del entrenamiento a la inferencia: razonamiento por test-time compute. El segundo eje es gastar el cómputo donde más rinde. Tras DeepSeek-R1 y los modelos o1 y o3, el razonamiento por refuerzo y el escalado en inferencia se han vuelto centrales: en lugar de un

único modelo gigante, un modelo que delibera más cuando el problema lo merece. La pregunta abierta es hasta dónde escala esto antes de saturar y cómo controlar su coste.[30]

Multimodalidad nativa. La multimodalidad ha pasado de «pegar» un encoder a un LLM a entrenar un único modelo que procesa texto, imagen y audio en un espacio común desde el principio (*early fusion*). Familias como Qwen-VL, InternVL o Llama 4 apuntan a que aceptar varias modalidades sea la norma, no una extensión.

Alternativas y arquitecturas híbridas. El Transformer sigue siendo dominante, pero no es un dogma: los State Space Models y los diseños híbridos de atención más SSM buscan domar el coste del contexto largo, y la investigación en atención eficiente no se detiene.

La lectura de ingeniería no cambia con las modas: cada avance mueve un compromiso (memoria, latencia, datos, evaluación), y la pregunta sigue siendo cuál te conviene mover para tu problema, no cuál es el modelo más nuevo del mes.

En el día a día

Elegir un modelo para soporte. Si atiendes miles de tickets diarios, un MoE puede parecer atractivo por capacidad y coste activo. Pero tienes que medir latencia real, disponibilidad, memoria, precio por token y calidad en tus casos. No basta con leer «8x7B» o «experts» en una ficha.

Diseñar un flujo con razonamiento. Para una tarea legal, médica, financiera o de ingeniería, no deberías conformarte con una respuesta directa si hay pasos verificables. Puedes pedir estructura, usar herramientas, ejecutar comprobaciones y registrar evidencias. El razonamiento útil se diseña como proceso, no como decoración textual.

Añadir imágenes a un producto. Si un usuario sube una foto de un daño, una factura o una pizarra, el sistema debe saber qué espera extraer. No es lo mismo describir la imagen que localizar un número de serie, leer una tabla o decidir si falta información.

Evaluar de verdad. Cada palanca trae fallos distintos. MoE puede fallar por routing o servicio. Razonamiento puede fallar por pasos plausibles pero incorrectos. Multimodalidad puede fallar por lectura visual, alineación o detalle espacial. La evaluación debe cubrir esos modos, no solo una puntuación global.

Por qué debería importarte

Porque aquí empiezas a leer fichas técnicas con criterio. Cuando alguien dice «modelo MoE», ya no preguntas solo cuántos parámetros tiene: preguntas cuántos expertos se activan, cómo se sirve y qué latencia tiene. Cuando alguien dice «modelo de razonamiento», preguntas si

mejora en tus tareas, cuánto tarda y cómo se verifica. Cuando alguien dice «multimodal», preguntas qué modalidades entran, cómo se alinean y qué errores has medido.

También importa por coste. Una arquitectura moderna no es una pegatina comercial; cambia memoria, inferencia, datos, evaluación y producto. Elegir mal puede darte un sistema caro, lento o difícil de depurar. Elegir bien puede convertir una tarea imposible en una herramienta útil.

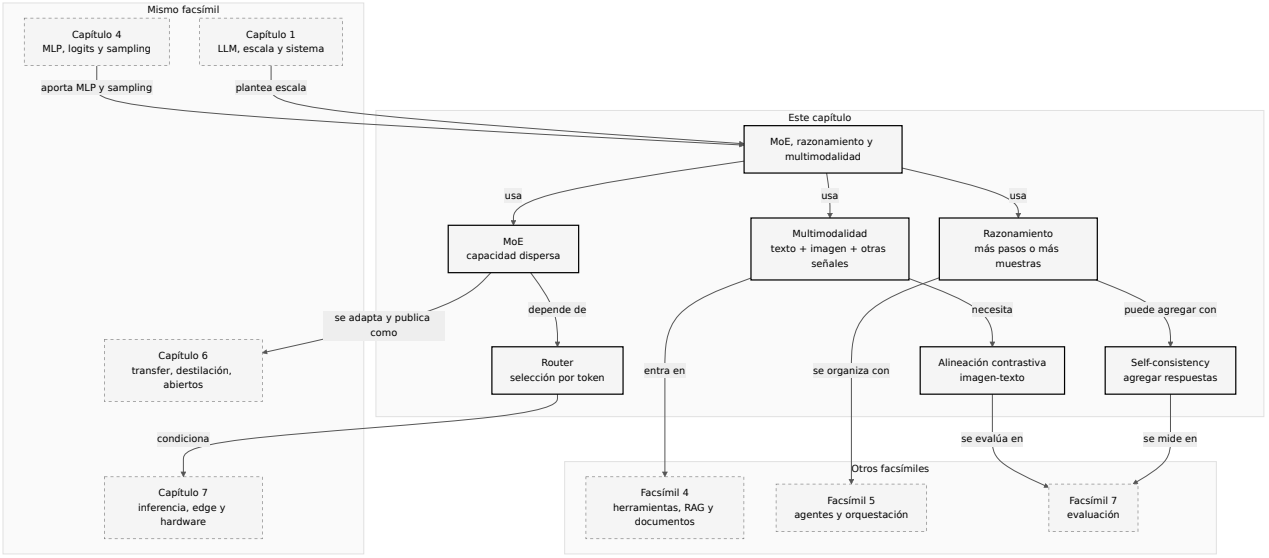
Dónde solía tropezar yo

ERROR	POR QUÉ ES UN ERROR	ANTÍDOTO
Pensar que los expertos tienen profesiones humanas	Un experto MoE es una subred aprendida, no un departamento etiquetado.	Habla de routing y activación, no de «el experto de derecho» sin evidencia.
Comparar parámetros totales sin mirar parámetros activos	En MoE, no todos los parámetros se usan en cada token.	Pregunta cuántos expertos se activan y cuál es el coste por token.
Creer que razonar es escribir más texto	Un texto largo puede sonar convincente y estar mal.	Diseña verificación: cálculo, tests, fuentes o consistencia.
Tratar multimodalidad como OCR con esteroides	Una imagen no siempre se reduce a leer texto. Puede implicar objetos, posición, contexto y ambigüedad.	Define qué tarea visual necesitas y evalúala con ejemplos reales.
Meter una arquitectura moderna sin cambiar evaluación	Cada arquitectura trae fallos propios.	Añade pruebas específicas para routing, latencia, pasos y entrada visual.

Cómo encaja todo

Este mapa relaciona las familias modernas con lo aprendido antes y con lo que vendrá después. MoE hereda el MLP y cambia coste activo; razonamiento hereda sampling y añade proceso; multimodalidad hereda embeddings y exige alineación.

La decisión de ingeniería no es “usar lo más moderno”, sino saber qué compromiso estás moviendo: memoria, latencia, datos, evaluación, herramientas o UX.


Mismo facsímil
Capítulo 4
MLP, logits y sampling
Capítulo 1
LLM, escala y sistema
aporta MLP y sampling
plantea escala
Este capítulo
MoE, razonamiento y multimodalidad
usa
usa
usa
MoE
capacidad dispersa
Multimodalidad
texto + imagen + otras señales
Razonamiento
más pasos o más muestras
se adapta y publica como
depende de
necesita
puede agregar con
Capítulo 6
transfer, destilación, abiertos
Router
selección por token
entra en
se organiza con
Alineación contrastiva
imagen-texto
Self-consistency
agregar respuestas
condiciona
se evalúa en
se mide en
Otros facsímiles
Capítulo 7
inferencia, edge y hardware
Facsímil 4
herramientas, RAG y documentos
Facsímil 5
agentes y orquestación
Facsímil 7
evaluación

Vocabulario aprendido

TÉRMINO	DEFINICIÓN
Mixture of Experts	Arquitectura con varios expertos donde solo algunos se activan por token o ejemplo.
Router	Módulo que asigna tokens a expertos mediante puntuaciones aprendidas.
Cómputo disperso	Uso de una parte del modelo en cada paso, no de todo el modelo.
Parámetros activos	Parámetros que participan realmente en una pasada concreta.
Encoder-only	Transformer orientado a comprender una entrada completa y producir representaciones.
Decoder-only	Transformer causal orientado a generar el siguiente token.
Encoder-decoder	Arquitectura que lee una entrada con encoder y genera salida con decoder.
State Space Model	Modelo de secuencia que mantiene un estado interno y puede escalar bien en secuencias largas.
Diffusion model	Modelo generativo que aprende a quitar ruido progresivamente.
Graph Neural Network	Red que opera sobre nodos y aristas de un grafo.
Arquitectura de sistema	Diseño que combina modelos, datos, herramientas, memoria, recuperación y validadores.
Chain-of-thought	Técnica que usa pasos intermedios para resolver tareas de varios pasos.
Self-consistency	Generar varias soluciones y escoger la respuesta más consistente.
Test-time compute	Escalado del cómputo en inferencia (más muestras o pasos) para mejorar la respuesta sin cambiar los pesos.
Multimodalidad	Integración de texto, imagen, audio, vídeo u otras señales.
Vision Transformer	Transformer que parte una imagen en parches y los trata como tokens.
Alineación contrastiva	Entrenar representaciones para acercar pares relacionados y separar pares no relacionados.
Conector multimodal	Proyección que traduce representaciones visuales u otras señales al espacio que usa el LLM.

Antes de pasar página

- [ ] ¿Puedo explicar MoE sin decir que los expertos son humanos en miniatura? (Si no, vuelve a «MoE».)
- [ ] ¿Puedo distinguir arquitectura neuronal, modelo entrenado y arquitectura de sistema? (Si no, vuelve a «El mapa amplio de arquitecturas».)
- [ ] ¿Sé diferenciar encoder-only, decoder-only y encoder-decoder con un ejemplo de uso? (Si no, vuelve a «Familias Transformer en lenguaje».)
- [ ] ¿Sé calcular una ruta top-2 con pesos de router? (Si no, vuelve a «MoE: más capacidad sin activar todo».)
- [ ] ¿Puedo distinguir parámetros totales de parámetros activos con un ejemplo (Mixtral, DeepSeek-V3)? (Si no, vuelve a «Parámetros totales frente a activos».)
- [ ] ¿Sé por qué un SSM escala $O(n)$ y la atención $O(n^2)$? (Si no, vuelve a «El mapa amplio de arquitecturas».)
- [ ] ¿Entiendo por qué razonamiento no es solo escribir más texto, y qué es el test-time compute? (Si no, vuelve a «Razonamiento» y «El giro de 2025».)
- [ ] ¿Puedo resolver el ejemplo de la salida a las 15:10 paso a paso? (Si no, vuelve a «Un ejemplo entendible».)
- [ ] ¿Entiendo cómo una imagen se convierte en tokens con un Vision Transformer (parches)? (Si no, vuelve a «Multimodalidad».)
- [ ] ¿Sé explicar cómo una imagen llega a un modelo de lenguaje? (Si no, vuelve a «El patrón común».)
- [ ] ¿Puedo nombrar hacia dónde va el campo (MoE disperso, test-time compute, multimodalidad nativa)? (Si no, vuelve a «Hacia dónde va esto».)
- [ ] ¿Sé cómo cambiaría el resultado al variar un ejemplo de routing o de alineación? (Si no, vuelve a «MoE: más capacidad sin activar todo» y «El patrón común».)
- [ ] ¿Puedo relacionar MoE, razonamiento y multimodalidad con coste, latencia y evaluación? (Si no, vuelve a «Por qué debería importarte».)

En resumen

IDEA FUERZA	DETALLE
No hay una sola familia de arquitectura.	MLP, CNN, RNN, Transformer, GNN, difusión, SSM y sistemas RAG resuelven problemas distintos.
Hay que separar arquitectura neuronal, modelo y sistema.	BERT, GPT, T5 o RAG no son el mismo tipo de cosa aunque aparezcan juntos en conversaciones de producto.
MoE aumenta capacidad activando solo parte del modelo.	El router decide qué expertos procesan cada token, pero eso añade retos de balanceo y servicio.
Razonamiento es proceso, no eslogan.	Más pasos, varias muestras y verificación pueden mejorar tareas complejas si se evalúan bien.
Multimodalidad convierte señales distintas en representaciones compatibles.	Imagen y texto deben alinearse antes de que el LLM pueda usarlos de forma útil.
Las arquitecturas modernas cambian compromisos.	Calidad, coste, latencia, memoria y evaluación se mueven a la vez.

Para saber más

Alayrac, J.-B. y otros (2022). Flamingo: a visual language model for few-shot learning. *Advances in Neural Information Processing Systems 35*, 23716-23736. https://arxiv.org/abs/2204.14198

Devlin, J., Chang, M.-W., Lee, K. y Toutanova, K. (2019). BERT: pre-training of deep bidirectional transformers for language understanding. *Proceedings of NAACL-HLT*, 4171-4186. https://doi.org/10.18653/v1/N19-1423

Dosovitskiy, A. y otros (2021). An image is worth 16x16 words: Transformers for image recognition at scale. *International Conference on Learning Representations*. https://arxiv.org/abs/2010.11929

Driess, D. y otros (2023). *PaLM-E: An embodied multimodal language model*. https://arxiv.org/abs/2303.03378

Fedus, W., Zoph, B. y Shazeer, N. (2022). Switch Transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research, 23*(120), 1-39. https://jmlr.org/papers/v23/21-0998.html

Gu, A. y Dao, T. (2023). *Mamba: Linear-time sequence modeling with selective state spaces*. https://arxiv.org/abs/2312.00752

Ho, J., Jain, A. y Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems 33*, 6840-6851. https://papers.nips.cc/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html

Jiang, A. Q. y otros (2024). *Mixtral of experts*. https://arxiv.org/abs/2401.04088

Kingma, D. P. y Welling, M. (2014). Auto-encoding variational Bayes. *International Conference on Learning Representations*. https://arxiv.org/abs/1312.6114

Kipf, T. N. y Welling, M. (2017). Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations*. https://arxiv.org/abs/1609.02907

Lepikhin, D. y otros (2021). GShard: Scaling giant models with conditional computation and automatic sharding. *International Conference on Learning Representations*. https://arxiv.org/abs/2006.16668

Lewis, M. y otros (2020). BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *Proceedings of ACL*, 7871-7880. https://doi.org/10.18653/v1/2020.acl-main.703

Lewis, P. y otros (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems 33*, 9459-9474. https://papers.nips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html

Liu, H., Li, C., Wu, Q. y Lee, Y. J. (2023). Visual instruction tuning. *Advances in Neural Information Processing Systems 36*. https://arxiv.org/abs/2304.08485

Radford, A. y otros (2021). Learning transferable visual models from natural language supervision. *Proceedings of the 38th International Conference on Machine Learning*, 8748-8763. https://arxiv.org/abs/2103.00020

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. y Sutskever, I. (2019). *Language models are unsupervised multitask learners*. https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

Raffel, C. y otros (2020). Exploring the limits of transfer learning with a unified text-to-text Transformer. *Journal of Machine Learning Research, 21*(140), 1-67. https://www.jmlr.org/papers/v21/20-074.html

Shazeer, N. y otros (2017). Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *International Conference on Learning Representations*. https://arxiv.org/abs/1701.06538

Sutskever, I., Vinyals, O. y Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems 27*, 3104-3112. https://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks

Wang, X. y otros (2023). Self-consistency improves chain of thought reasoning in language models. *International Conference on Learning Representations*. https://arxiv.org/abs/2203.11171

Wei, J. y otros (2022). Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems 35*, 24824-24837. https://arxiv.org/abs/2201.11903

Yao, S. y otros (2023). *Tree of thoughts: Deliberate problem solving with large language models*. https://arxiv.org/abs/2305.10601

Notas

1. Lewis, P. y otros (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems 33*, 9459-9474. https://papers.nips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html. RAG combina memoria paramétrica del modelo con recuperación externa de documentos. ↩

2. LeCun, Y., Bengio, Y. y Hinton, G. (2015). Deep learning. *Nature, 521*(7553), 436-444. https://doi.org/10.1038/nature14539. El artículo revisa CNNs, redes recurrentes y aprendizaje profundo moderno. ↩

3. Sutskever, I., Vinyals, O. y Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems 27*, 3104-3112. https://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks. El trabajo mostró modelos secuencia-a-secuencia basados en redes recurrentes para traducción. ↩

4. Kipf, T. N. y Welling, M. (2017). Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations*. https://arxiv.org/abs/1609.02907. El trabajo presenta una forma eficiente de aplicar redes convolucionales a datos con estructura de grafo. ↩

5. Kingma, D. P. y Welling, M. (2014). Auto-encoding variational Bayes. *International Conference on Learning Representations*. https://arxiv.org/abs/1312.6114. El VAE aprende una representación latente probabilística que permite generación y reconstrucción. ↩

6. Ho, J., Jain, A. y Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems 33*, 6840-6851. https://papers.nips.cc/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html. DDPM formaliza generación como un proceso de denoising iterativo. ↩

7. Gu, A. y Dao, T. (2023). *Mamba: Linear-time sequence modeling with selective state spaces*. https://arxiv.org/abs/2312.00752. Mamba propone modelos de espacio de estados selectivos para modelar secuencias en tiempo lineal. ↩

8. Devlin, J., Chang, M.-W., Lee, K. y Toutanova, K. (2019). BERT: pre-training of deep bidirectional transformers for language understanding. *Proceedings of NAACL-HLT*, 4171-4186. https://doi.org/10.18653/v1/N19-1423. BERT usa una arquitectura encoder bidireccional orientada a comprensión. ↩

9. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. y Sutskever, I. (2019). *Language models are unsupervised multitask learners*. https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf. GPT-2 popularizó el Transformer decoder autoregresivo a escala. ↩

.0. Raffel, C. y otros (2020). Exploring the limits of transfer learning with a unified text-to-text Transformer. *Journal of Machine Learning Research, 21*(140), 1-67. https://www.jmlr.org/papers/v21/20-074.html. T5 convierte tareas de NLP a un formato texto-a-texto con arquitectura encoder-decoder. ↩

.1. Lewis, M. y otros (2020). BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *Proceedings of ACL*, 7871-7880. https://doi.org/10.18653/v1/2020.acl-main.703. ↩

.2. Dosovitskiy, A. y otros (2021). An image is worth 16x16 words: Transformers for image recognition at scale. *International Conference on Learning Representations*. https://arxiv.org/abs/2010.11929. ViT aplica Transformers a imágenes dividiéndolas en parches. ↩

.3. Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G. y Dean, J. (2017). Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *International Conference on Learning Representations*. https://arxiv.org/abs/1701.06538. El trabajo introduce una capa MoE con una red de gating que selecciona una combinación escasa de expertos por ejemplo. ↩

.4. Lepikhin, D. y otros (2021). GShard: Scaling giant models with conditional computation and automatic sharding. *International Conference on Learning Representations*. https://arxiv.org/abs/2006.16668. ↩

.5. Fedus, W., Zoph, B. y Shazeer, N. (2022). Switch Transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research, 23*(120), 1-39. https://jmlr.org/papers/v23/21-0998.html. ↩

.6. Jiang, A. Q. y otros (2024). *Mixtral of experts*. https://arxiv.org/abs/2401.04088. El artículo presenta Mixtral 8x7B como un modelo sparse MoE donde cada capa usa varios expertos feed-forward y selecciona algunos por token. ↩

.7. DeepSeek-AI. (2024). *DeepSeek-V3 Technical Report*. https://arxiv.org/abs/2412.19437. DeepSeek-V3 es un modelo MoE de 671B de parámetros con unos 37B activos por token. ↩

.8. Raschka, S. (2025). *The State of LLMs 2025*. https://magazine.sebastianraschka.com/p/state-of-llms-2025. Repaso del año en arquitecturas de LLM, con la convergencia hacia MoE y atención eficiente. ↩

19. Wei, J. y otros (2022). Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems 35*, 24824-24837. https://arxiv.org/abs/2201.11903. El trabajo estudia cómo ejemplos con razonamiento paso a paso mejoran el rendimiento en tareas que requieren varios pasos. ↩

20. Wang, X. y otros (2023). Self-consistency improves chain of thought reasoning in language models. *International Conference on Learning Representations*. https://arxiv.org/abs/2203.11171. El método muestrea múltiples cadenas de solución y agrega respuestas para mejorar robustez. ↩

21. Yao, S. y otros (2023). *Tree of thoughts: Deliberate problem solving with large language models*. https://arxiv.org/abs/2305.10601. El marco explora unidades de pensamiento como nodos de búsqueda y permite evaluar caminos parciales. ↩

22. Snell, C., Lee, J., Xu, K. y Kumar, A. (2024). *Scaling LLM Test-Time Compute Optimally Can Be More Effective Than Scaling Model Parameters*. https://arxiv.org/abs/2408.03314. El trabajo muestra que repartir más cómputo en inferencia puede rendir más que aumentar los parámetros. ↩

23. Guo, D. y otros (DeepSeek-AI). (2025). *DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning*. https://arxiv.org/abs/2501.12948. R1 muestra que el razonamiento paso a paso puede emerger de refuerzo puro y escalar con el cómputo de inferencia. ↩

24. Dosovitskiy, A. y otros (2021). An image is worth 16x16 words: Transformers for image recognition at scale. *International Conference on Learning Representations*. https://arxiv.org/abs/2010.11929. El título lo dice literalmente: una imagen vale 16x16 palabras, los parches que se tratan como tokens. ↩

25. Radford, A. y otros (2021). Learning transferable visual models from natural language supervision. *Proceedings of the 38th International Conference on Machine Learning*, 8748-8763. https://arxiv.org/abs/2103.00020. CLIP aprende representaciones visuales usando supervisión en lenguaje natural y permite transferencia zero-shot. ↩

26. Alayrac, J.-B. y otros (2022). Flamingo: a visual language model for few-shot learning. *Advances in Neural Information Processing Systems 35*, 23716-23736. https://arxiv.org/abs/2204.14198. Flamingo integra información visual y textual con mecanismos que permiten few-shot learning multimodal. ↩

27. Liu, H., Li, C., Wu, Q. y Lee, Y. J. (2023). Visual instruction tuning. *Advances in Neural Information Processing Systems 36*. https://arxiv.org/abs/2304.08485. LLaVA conecta un encoder visual y un LLM, y ajusta el sistema con datos de instrucciones multimodales. ↩

28. Driess, D. y otros (2023). *PaLM-E: An embodied multimodal language model*. https://arxiv.org/abs/2303.03378. PaLM-E integra observaciones multimodales en un modelo de lenguaje para tareas encarnadas. ↩

29. Raschka, S. (2025). *The State of LLMs 2025*. https://magazine.sebastianraschka.com/p/state-of-llms-2025. El repaso del año describe la convergencia de los modelos abiertos hacia MoE más atención eficiente. ↩

30. Snell, C., Lee, J., Xu, K. y Kumar, A. (2024). *Scaling LLM Test-Time Compute Optimally Can Be More Effective Than Scaling Model Parameters*. https://arxiv.org/abs/2408.03314. ↩