

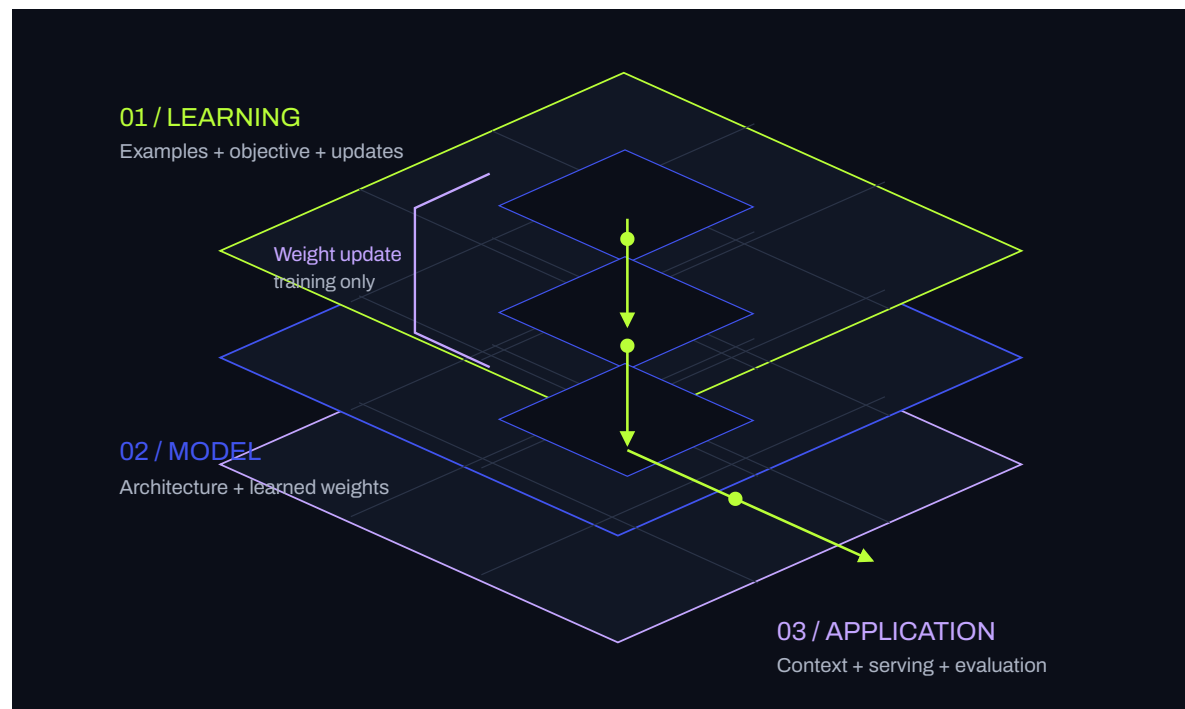
# Open models. Understood.

Llama, Gemma and DeepSeek, from the first training example to the application you run.

For engineers and business analysts. Read the seven guided lessons, explore the diagrams, then use the glossary and technical appendix.

A model is a learned numerical system.  
Understanding what it sees, changes and stores makes the business choices clearer.

Terms marked \* open a definition. Start with weights, tokens and inference.



A layered system: training creates weights; inference uses them. The application supplies the current context.

## Control

What does an open-weight release let you run, inspect and adapt?

## Mechanism

What happens inside attention, dense layers and routed experts?

## Evidence

Which data, evaluation and operating costs support your decision?

Release-focused primer · revised 4 October 2026 · covers Llama 3.1 / 4, Gemma 3 and original DeepSeek V3 / R1. These are named architectural examples, rather than a catalog of every later checkpoint.

# What is actually open?

The key distinction is whether you can obtain the learned weights. Code, data, permissions and reproducibility are separate questions.

| QUESTION                       | OPEN-WEIGHT RELEASE                                                                   | CLOSED-WEIGHT SERVICE                                                                      |
|--------------------------------|---------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------|
| Can I run it myself?           | Usually, under the checkpoint's license and hardware requirements.                    | The provider serves the model through its application or API.                              |
| Can I inspect / adapt weights? | Yes, if the release and license permit. Fine-tuning and quantization may be possible. | Weights remain private. Some providers offer managed fine-tuning.                          |
| Can I recreate training?       | Released weights do not imply released data, compute or the complete training recipe. | Generally unavailable; disclosure varies by provider.                                      |
| Who operates it?               | Your team or a hosting provider handles serving, monitoring and upgrades.             | The provider operates the model service; you still own the application and its evaluation. |

## Three different artifacts.

Weights contain the learned numerical values. Code defines how the network and training system work. Data supplies the examples. Opening one does not automatically open the other two.

Use the precise term open-weight. “Open source AI” is a stronger claim about the materials and permissions needed to study and modify the system.

**Business example:** an internal document assistant can use a self-hosted open-weight model or a managed API. Compare data location, task quality, uptime, staffing and total cost—not the word “open” alone.

## Release-specific terms.

Llama uses a custom Community License. Gemma weights use Google’s Gemma terms, distinct from its implementation license. Original DeepSeek V3 weights use its Model License; R1 weights use MIT, with upstream obligations for some distillations. [1–6]

Evaluate the exact checkpoint, redistribution rules and use policy. A hosted open-weight API still sends your request to that host.

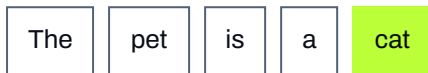
## 03 / FROM EXAMPLES TO WEIGHTS

# A wrong guess. A useful signal.

Training compares a prediction with an observed example. Loss turns that difference into a number; an optimizer uses gradients to guide weight updates.

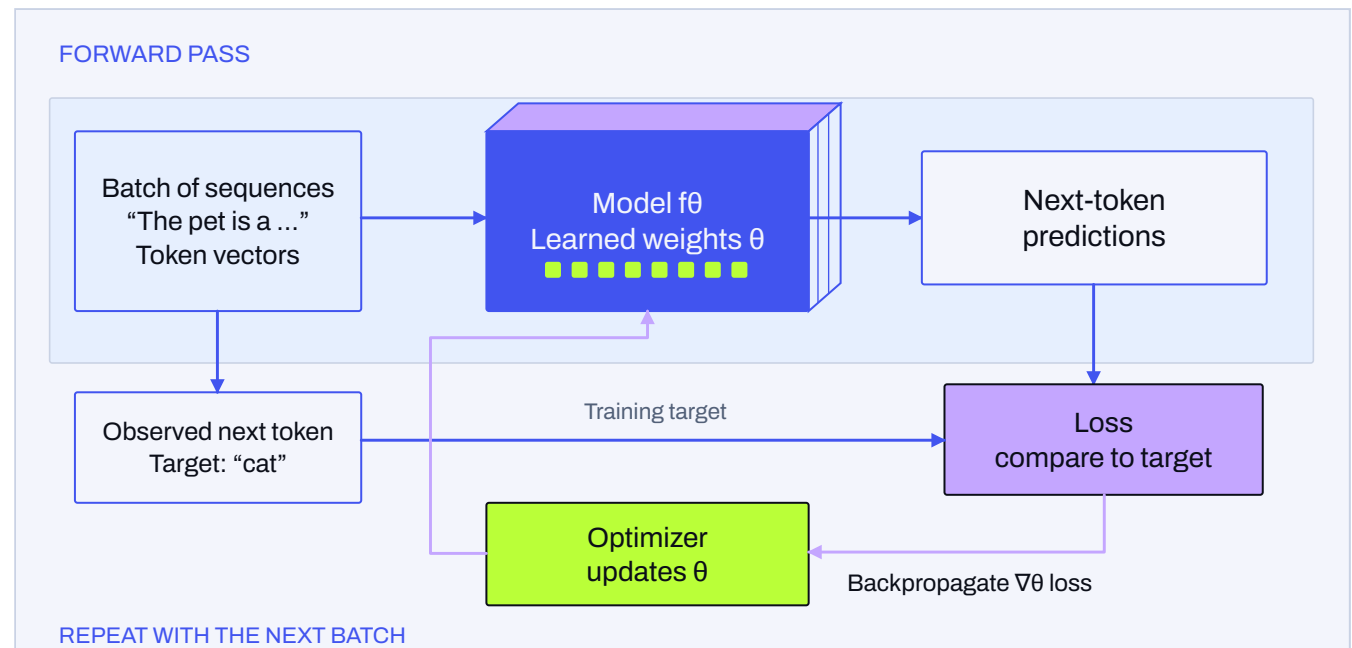
## The target already exists.

Imagine the recorded training text is “The pet is a cat.” At this position, the prefix is the input and cat is the observed next token.



The model assigns probabilities to all tokens in its vocabulary. If it assigns only 10% to “cat”, the numerical loss is 2.303. At 80%, loss falls to 0.223.

A simplified teaching example. Real tokenizers may split words differently. Other continuations can be sensible; the target is what this example actually contains.



— Processing — Active signal — Return / parallel path

Schematic training loop. Gradients and weight updates happen during training; ordinary inference uses fixed weights.

TRY IT / PROBABILITY &amp; LOSS

# What the error score does.

A lower probability for the observed token produces higher loss. Gradients connect that score to possible weight changes.

TRY IT / PROBABILITY ASSIGNED TO “CAT”

1% 10% 99%



Lime = “cat”; blue = all other tokens combined. Probabilities still sum to 100%.

NUMERICAL LOSS

## 2.303

Cross-entropy score  
Lower is better

### A score, then an update.

A low probability for the observed token gives a high error score. “Penalty” means this numerical score; nobody is being punished.

Backpropagation computes sensitivities. The optimizer changes trainable weights using those gradients. Loss alone does not determine the size of the update.

## Pretrain

Learn broad patterns by predicting tokens across many examples. Quality, mixture and filtering shape what is learned.

## Post-train

Supervised fine-tuning teaches response examples. Preferences and verifiable rewards can guide further optimization.

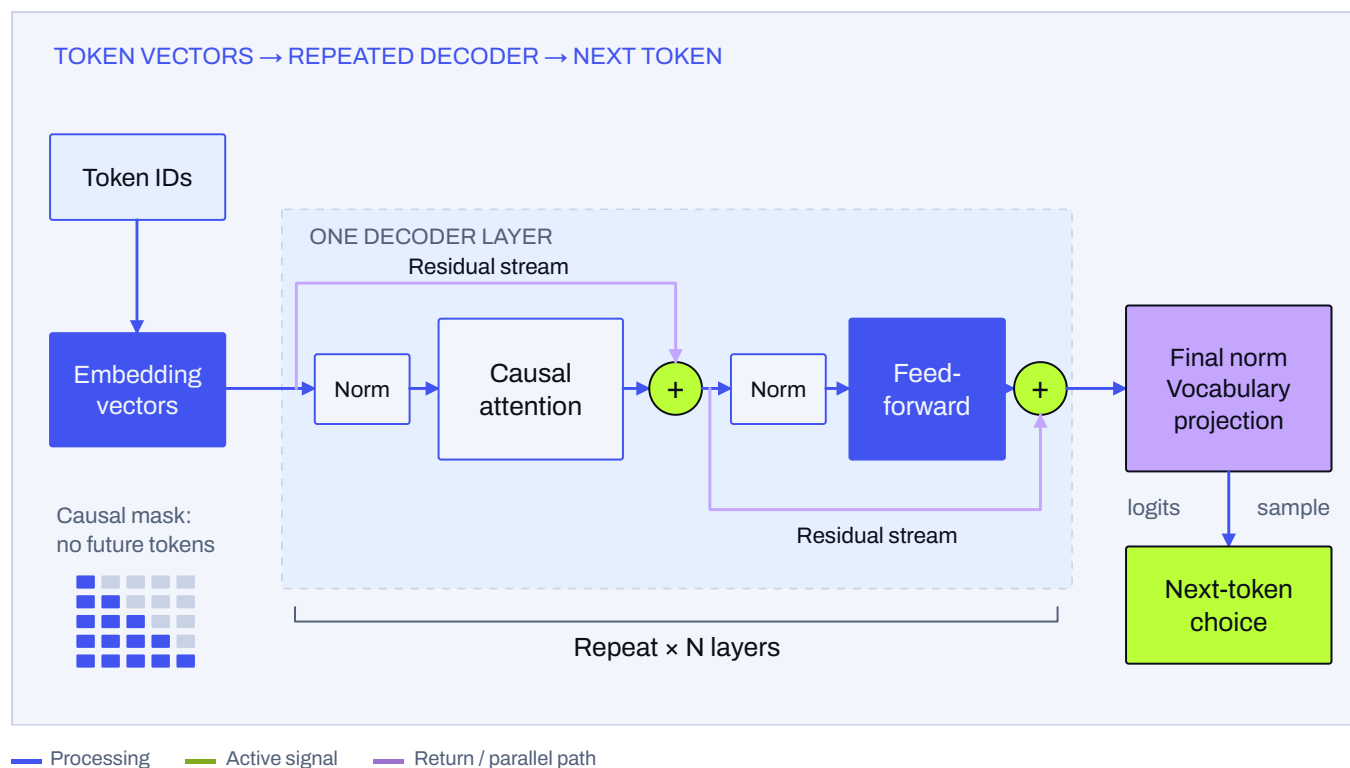
## Evaluate

Measure performance on held-out tasks. Lower training loss does not guarantee factual accuracy or business usefulness.

Training changes weights. Ordinary inference uses fixed weights to generate. The reference appendix works through the gradient calculation.

# Context in. One token out.

A decoder repeatedly turns the available context into a next-token distribution. Attention mixes eligible token information; a feed-forward network transforms each position.



Generic decoder schematic: embedding → repeated attention / feed-forward blocks → vocabulary probabilities. Position encoding and exact layer details vary by release.

# From context to generation.

## Attention mixes context.

Each token representation creates a query, key and value. Queries compare with keys; their scores become weights that mix the values. A causal mask blocks future positions.

Residual connections carry the earlier representation around a sublayer. The feed-forward network adds another learned transformation.

## Context is the current input.

The context window is the token budget available to a request. It can include instructions, a question, retrieved passages, conversation history and output tokens.

It is different from learned weights. Supplying a document changes the input to this request; it does not automatically retrain the model or make the document permanent memory.

## Watch inference.

Inference uses trained weights to generate. Ordinary autoregressive generation adds a token, then uses the extended context to choose the next.

This fixed sentence illustrates the sequence. It is an animation, rather than a live model or measured timing.

### ILLUSTRATIVE GENERATION

The pet is a **cat**

Step 1: predict “cat” from the prefix. The model’s weights remain fixed.

6,000 input

1,000 output

1,192 free

A hypothetical 8,192-token budget. Actual accounting and maximum input/output lengths depend on the runtime and checkpoint.

# Llama 3.1 and Llama 4.

Different generations use different decoder and input designs.

## Two Llama generations. Two architectures.

Llama 3.1 is a text-only dense decoder with grouped-query attention. Its released sizes are 8B, 70B and 405B parameters. [1]

Llama 4 Scout / Maverick combine text with encoded image features through early fusion. MoE layers use one routed expert plus a parallel shared expert. Output is text or code. [2]

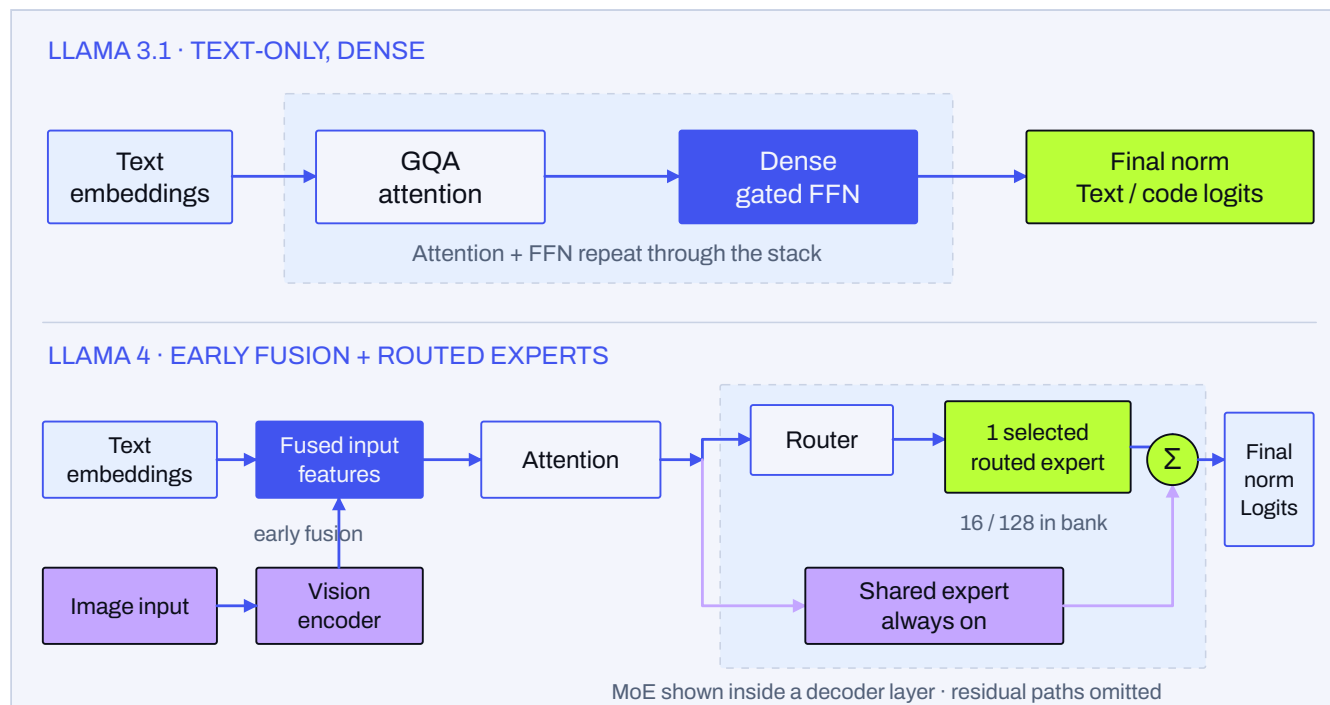
**109B / 17B**

Scout total / active  
16 routed experts

**400B / 17B**

Maverick total / active  
128 routed experts

Active parameters describe participation in computation. The full checkpoint still has to be stored and served. Experts are sublayers, rather than separate chatbots.



— Processing — Active signal — Return / parallel path

Llama 3.1's dense path is separate from Llama 4's multimodal MoE path. The shared expert bypasses the router and joins the routed contribution. [1,2]

# Gemma 3.

Dense feed-forward layers with local attention and periodic global connections.

## Local detail. Global connections.

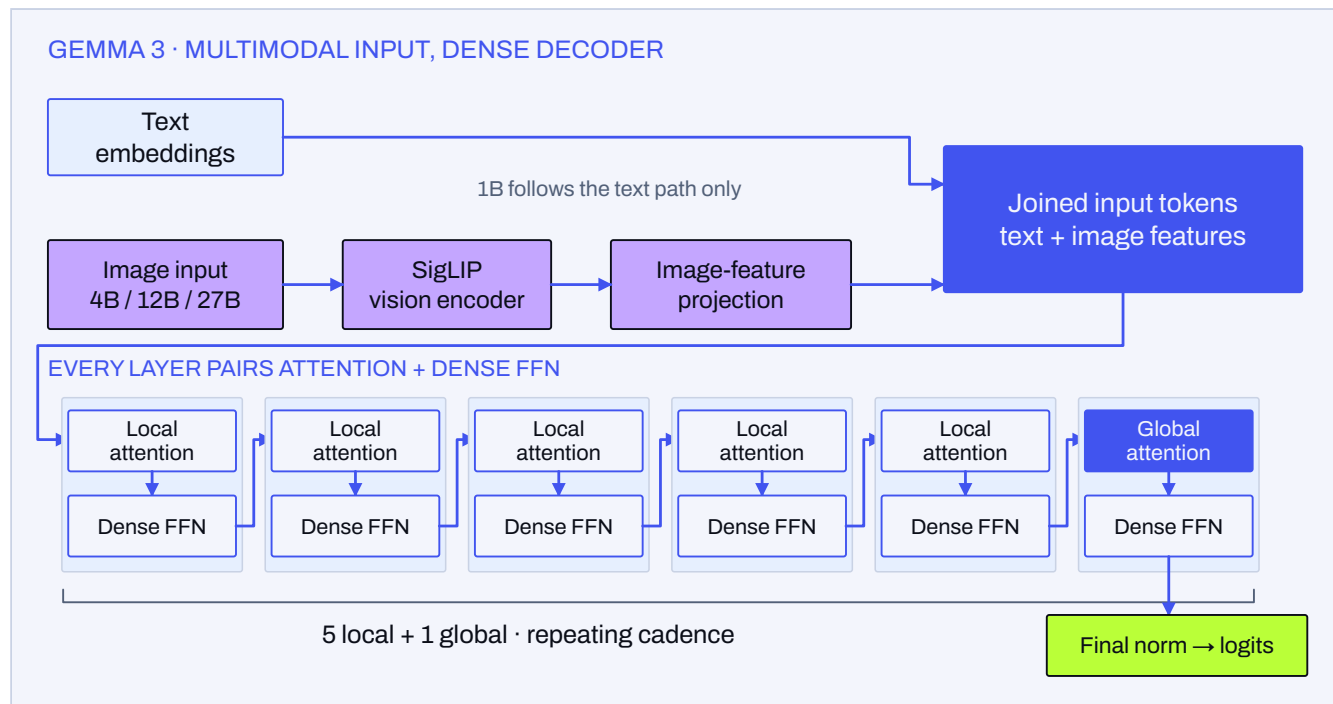
Gemma 3 keeps a dense feed-forward network in every decoder layer. Five local-attention layers are followed by one global-attention layer; the pattern repeats. [5,6]

The 1B variant is text-only. The 4B, 12B and 27B variants add a SigLIP image encoder and projected visual features. Pan and scan can create extra image crops.

**1 / 4 / 12 / 27B**    **5 local : 1**  
Dense parameter sizes    **global**

Repeated layer schedule

Local windows: 512 tokens for 1B; 1,024 for larger variants. Whole-request context support is a different limit; check the exact checkpoint and runtime. [5]



— Processing    — Active signal    — Return / parallel path

Each of the six depicted layers contains both attention and a dense FFN. The image branch applies to 4B / 12B / 27B, rather than the text-only 1B variant. [5,6]

# Original V3 and R1.

A compressed attention cache and parallel expert computation.

## Compress attention. Route computation.

Original DeepSeek V3 uses multi-head latent attention and mixture-of-experts feed-forward layers. Its first three layers have dense FFNs; the remaining 58 use MoE. [3]

Each MoE layer selects eight of 256 routed experts, alongside one always-on shared expert. MLA reduces cached attention state; it is separate from expert routing and weight quantization.

671B / 37B

Total / active parameters

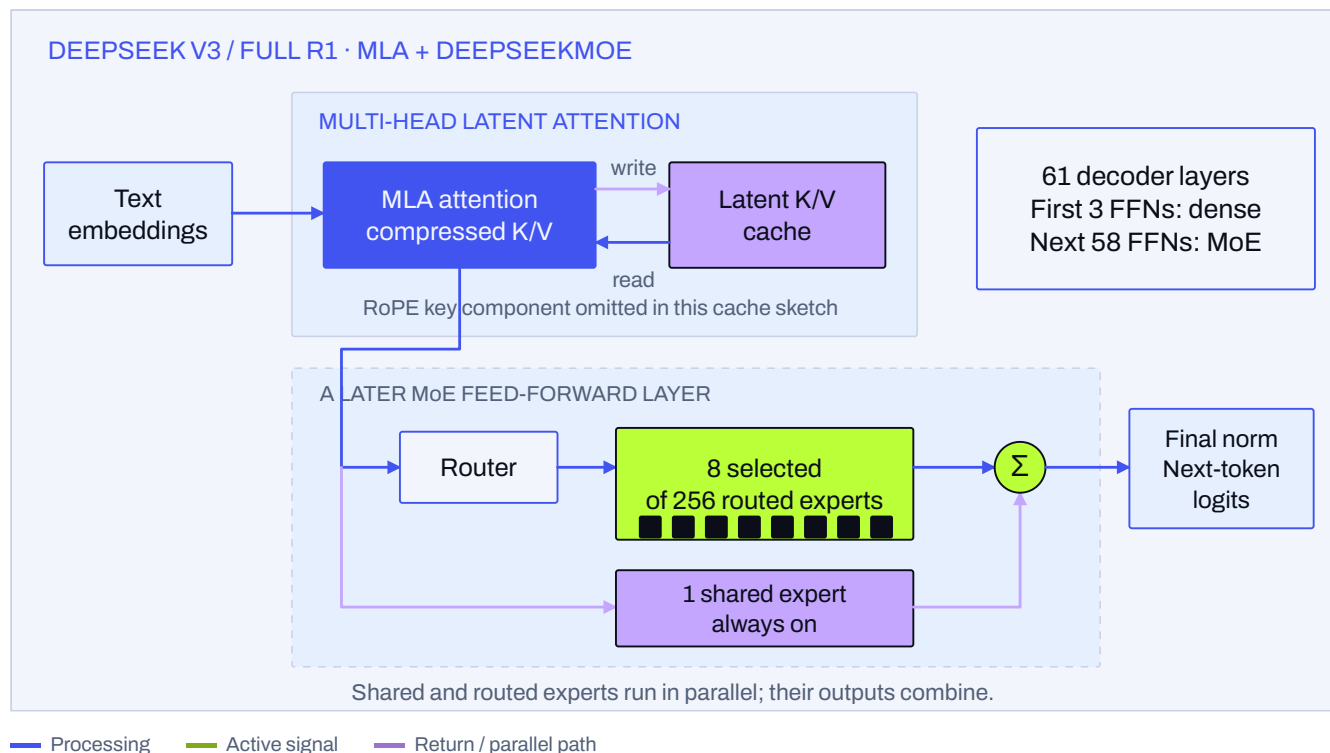
8 of 256 + 1

Selected routed + shared

Original R1 post-trains V3-Base for reasoning. R1-Zero used RL without preliminary SFT; R1 added cold-start examples and further SFT/RL. [4]

R1 distillations are smaller Qwen/Llama students. They retain their student architecture, rather than becoming miniature V3 MoE models.

**Engineering implication:** total weights, active computation and request cache are three different memory/cost questions. “17B active” or “37B active” is not the checkpoint’s storage size.



Full V3 / original R1 backbone. The shared expert is parallel to the routed experts. MLA sits inside attention; the first three FFNs are dense. [3,4]

# A dataset is a recipe.

## Not just a size.

Collection, permission, filtering, duplication, language mix and held-out evaluation all matter. Token counts describe volume; they do not establish quality or complete reproducibility.

| NAMED RELEASE                          | DISCLOSED TRAINING EVIDENCE                                                                                                                                   | WHAT REMAINS UNKNOWN HERE                                                                                           |
|----------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------|
| Llama 3.1 <a href="#">[1]</a>          | About 15T pretraining tokens from publicly available sources. Fine-tuning includes over 25M synthetic examples.                                               | Exact corpus membership, URLs, per-source mixture and the complete training recipe.                                 |
| Llama 4 <a href="#">[2]</a>            | Scout about 40T; Maverick about 22T tokens. Public, licensed and Meta-product sources, including public Facebook/Instagram material and Meta AI interactions. | The full corpus and exact source-level proportions. Token counts are pretraining volume, not unique documents.      |
| Gemma 3 <a href="#">[5,6]</a>          | The inspected implementation establishes the architecture. Google’s report and model card provide further data discussion.                                    | No precise corpus total or mixture is asserted from the locally verified evidence in this brief.                    |
| Original V3 / R1 <a href="#">[3,4]</a> | V3: 14.8T pretraining tokens. Qwen-based R1 distillations: 800K curated training examples.                                                                    | Full pretraining corpus and membership. Distillation samples and pretraining tokens are different stages and units. |

### Collect & prepare

Acquire allowed material, normalize formats, deduplicate, filter quality and sensitive content, and set a mixture. These choices affect both capability and risk.

### Learn & adapt

Pretraining fits broad token patterns. Post-training uses instructions, preferences or rewards. Distillation trains a student on teacher-generated examples.

### Hold out & test

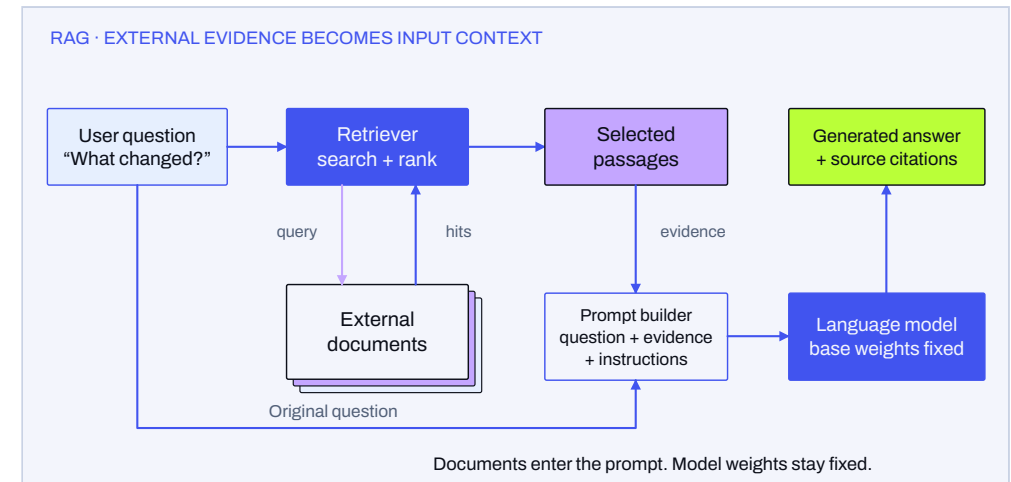
A holdout set should test generalization. Benchmark contamination can inflate scores when evaluation examples enter training.

# Use evidence in the current context.

## Your documents can stay outside the weights.

Retrieval-augmented generation finds relevant passages and adds them to the prompt. It helps supply current evidence and citations; it does not guarantee a supported answer.

Fine-tuning changes behavior through weight updates. RAG changes the supplied context. They solve different problems and can be combined.



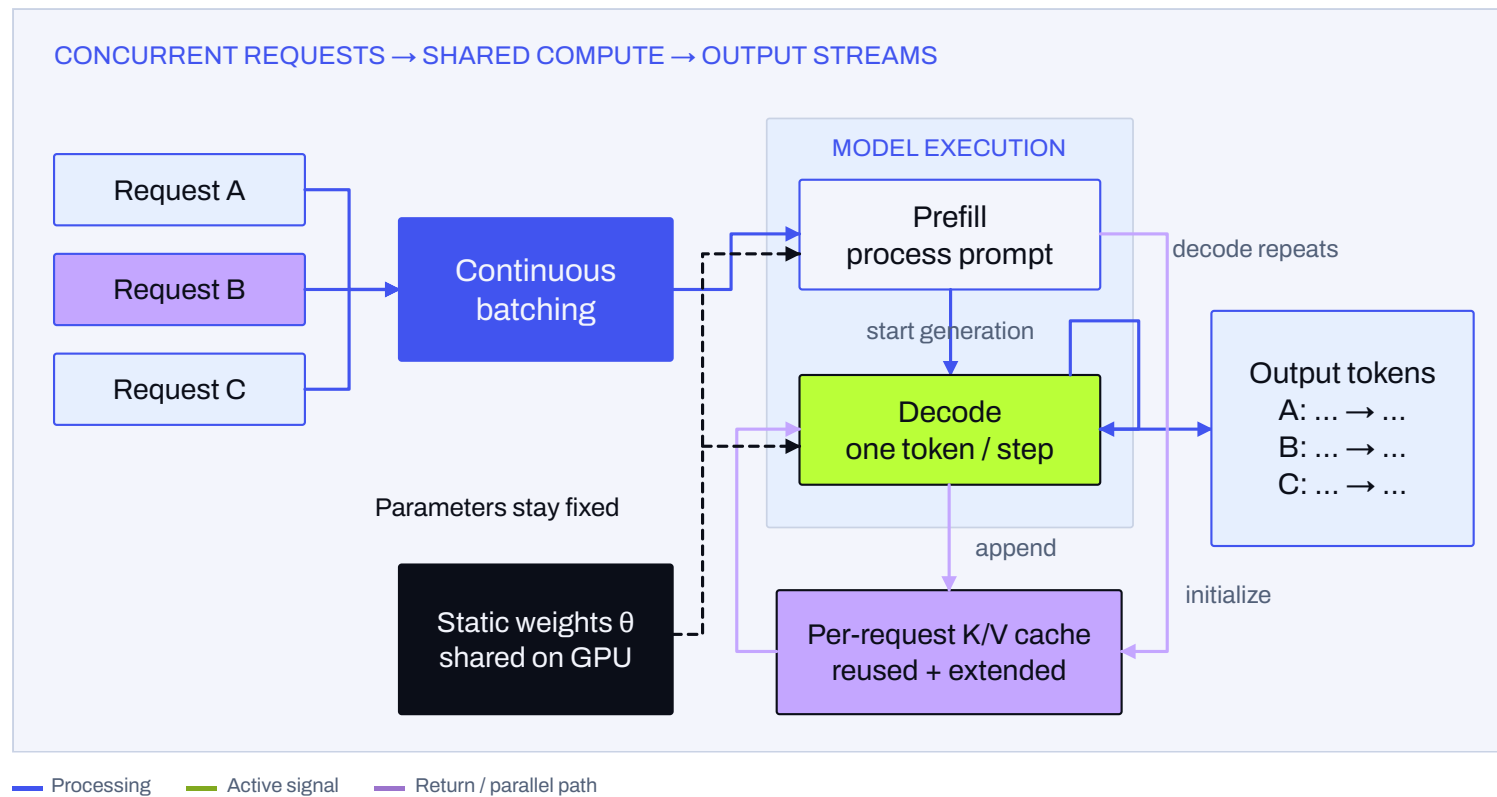
— Processing — Active signal — Return / parallel path

Application pattern: retrieve → assemble context → generate → verify. The document collection is separate from the model weights.

RAG changes the supplied context. Fine-tuning changes learned weights. Retrieval alone does not guarantee that an answer is supported.

# One request. A whole service.

A model becomes a product through scheduling, memory, retrieval, monitoring and evaluation. Concurrency means requests in progress; throughput means work completed per unit time.



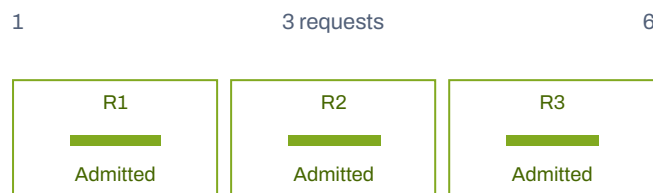
Schematic serving system. Prefill and decode are distinct phases; batching can share hardware across requests. This is not a benchmark.

## SERVICE METRICS / OPERATING COST

# Capacity is a service question.

## Watch a scheduler.

## ILLUSTRATIVE SIMULTANEOUS REQUESTS



3 admitted in a hypothetical four-slot scheduler. Real scheduling and capacity depend on memory, workloads and runtime; this is not a performance forecast.

## Measure the experience.

Time to first token includes waiting and prompt processing. Time per output token describes the later generation pace. Latency is elapsed time for a defined event.

Example: first token at 0.40s, final token at 2.38s, 100 output tokens. TPOT =  $(2.38 - 0.40) / 99 = 20\text{ms}$ ; decode rate  $\approx 50$  tokens/s.

## Budget the memory.

Weights, attention cache, temporary activations and runtime allocations all consume memory. More admitted requests and longer retained sequences increase cache memory.

Raw weight storage: parameters  $\times$  bits / 8. A 671B model at four bits is about 335.5GB before scales, cache and overhead. “Active” parameters do not replace total weights.

## Choose a deployment.

Self-host when control and predictable workloads justify operating effort. Use a managed service when its reliability, capability and operating model fit. Test the actual hardware, precision and runtime.

## Measure accepted work.

Use representative tasks, human review, p95 latency and cost per accepted task. Report prompt/output lengths and concurrency beside performance. More aggregate throughput can coexist with slower individual responses.

Compare accepted task quality, p95 latency and total operating cost using representative requests. Record model revision, hardware, precision and serving settings.

# Inspect the claim. Pin the release.

Architecture and licensing change between releases. This guide identifies its reference checkpoints and distinguishes verified implementation details from illustrative explanations.

01

## Meta / Llama 3.1

Model card: dense architecture, sizes, context and data disclosures. Community license is checkpoint-specific.

[Model card](#) · [License](#)

02

## Meta / Llama 4

Scout / Maverick card and reference implementation: early fusion, sizes, data and one routed plus parallel shared expert.

[Model card](#) · [Code & license](#)

03

## DeepSeek / original V3

Repository and 671B configuration: MLA, three dense / 58 MoE layers, 8 of 256 routed plus one shared, 14.8T tokens and original model license.

[Repository](#) · [Configuration](#)

04

## DeepSeek / original R1

R1 / R1-Zero post-training, V3-Base backbone, distillation, generated samples and upstream licensing obligations.

[Repository & license](#)

05

## Google / Gemma implementation

Configurations and code verify 1B / 4B / 12B / 27B variants, GQA, local/global cadence, windows and vision path. Whole-request context support must be checked against the specific checkpoint.

[Configurations](#) · [Model code](#)

06

## Gemma 3 / documentation

Transformers documentation describes image pan and scan and post-training. Google's report, card and weight terms are further reading; precise corpus statistics are not asserted here.

[Implementation docs](#) · [Report](#) · [Card](#) · [Terms](#)

KAF World Digital · revision 02 · 4 October 2026. Brand references: v10 guide, website consulting systems and portal reading patterns.