AI FUNDAMENTALS · OVERVIEW

How AI Is Built and Implemented

The lifecycle, the mechanism inside the model, the scale economics, and the software layer that turns a model into a product.

REV 1.2 | 19 September 2026 | Confidence-tagged · provenance at end

CONTENTS

1. The lifecycle: two phases with different economics

"AI" as shipped today is a large neural network whose behavior is set in a one-time **training** phase and then used in a per-request **inference** phase. Training produces a frozen file of numbers called weights. Inference is that file being run, over and over, for every request. Every later cost and performance question traces back to this split: building is a capital expense paid once, implementing is an operating expense paid per token.

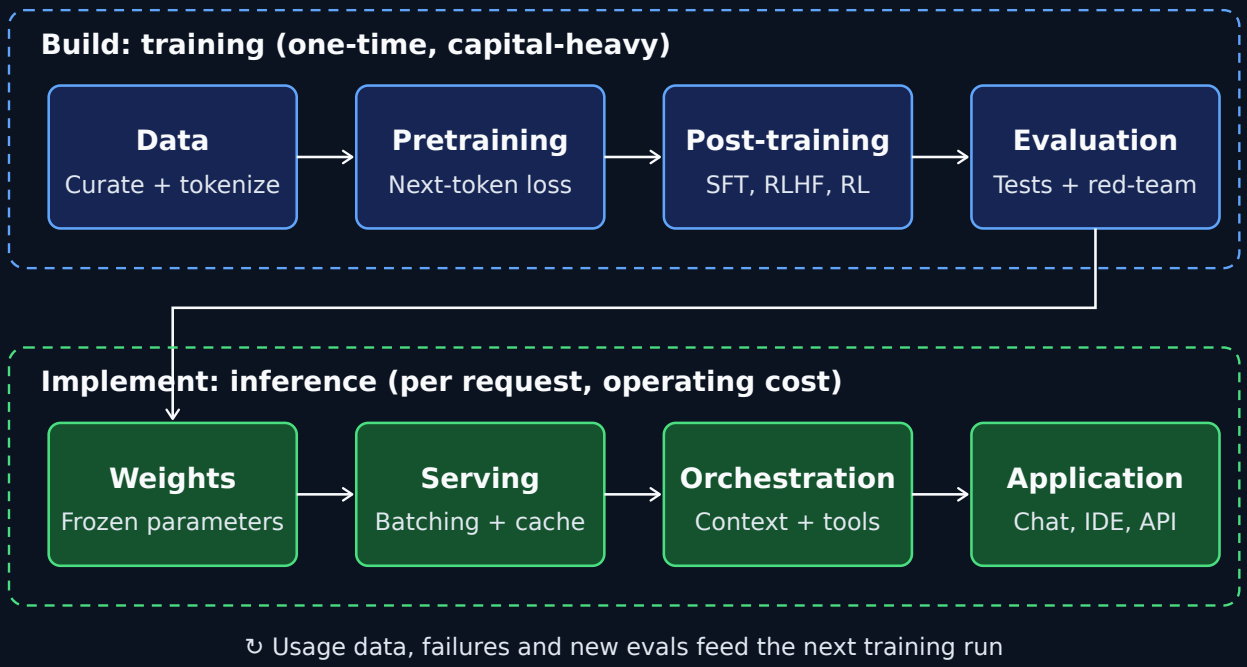


Figure 1. The build phase (blue) ends in a frozen set of weights. The implement phase (green) runs those weights for every request.

What each build stage contributes

- **Data.** Trillions of tokens of text, code, images and licensed or synthetic material. It is deduplicated, quality-filtered, and converted to tokens, the sub-word integer IDs the model actually sees. Data quality is now as much a differentiator as quantity.
- **Pretraining.** A single objective: predict the next token. Gradient descent adjusts the parameters until prediction error stops falling. This is where nearly all the raw knowledge and the bulk of the compute goes. The output is a "base model" that completes text but does not follow instructions.
- **Post-training.** Supervised fine-tuning (SFT) on example dialogues, then reinforcement learning, either from human preference (RLHF) or from verifiable outcomes such as tests that pass or math that checks. This turns a text completer into an assistant and produces "reasoning" behavior. Distillation, training a smaller model on a larger model's outputs, also sits here.
- **Evaluation.** Benchmarks, red-teaming and safety testing gate release. A pass freezes the weights.

2. Inside the model: one pass per token

The frozen weights implement a **transformer**. The model does not look anything up or run a program in the usual sense. It converts text to numbers, pushes those numbers through a deep stack of identical layers, and produces a probability for every possible next token. One token is chosen, appended to the input, and the whole process runs again.

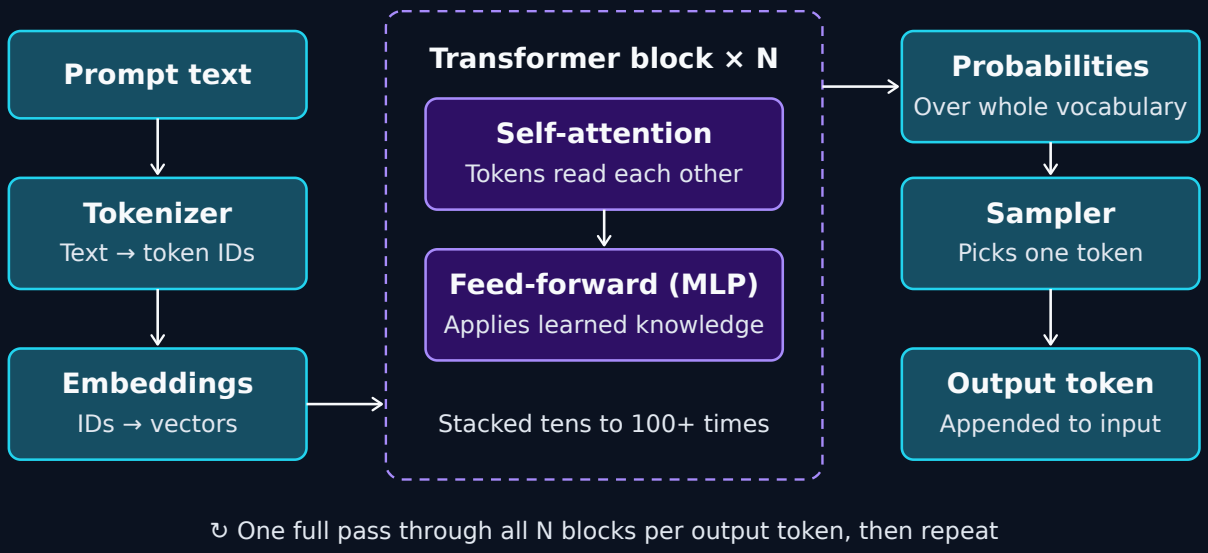


Figure 2. Input and output handling (cyan) surrounds the transformer stack (purple). Generation is a loop: every output token costs one full pass.

Three consequences of the loop

- **Input is cheap, output is expensive.** The whole prompt is processed in one parallel pass, called prefill. Each output token needs its own sequential pass, called decode. This is the mechanical reason output tokens are priced several times higher than input tokens, and why total token count tends to drive cost-per-task more than list price does.
- **The KV cache is the memory tax.** Attention stores intermediate results for every prior token so they are not recomputed. That cache grows with context length and consumes accelerator memory, which makes serving bound by memory bandwidth more than by arithmetic. Cached-prefix discounts exist because a provider can reuse this state.

- **"Reasoning" is more tokens through the same loop.** Effort tiers buy additional decode passes before the answer. The network itself does not change.

3. Scale: why building is a capital project

Capability has tracked training compute closely, so labs have scaled it relentlessly. Epoch AI's trend dashboard puts frontier language-model training compute at about 5× per year since 2020, a doubling roughly every 5.2 months, with training costs rising about 3.5× per year and power requirements doubling annually. HIGH

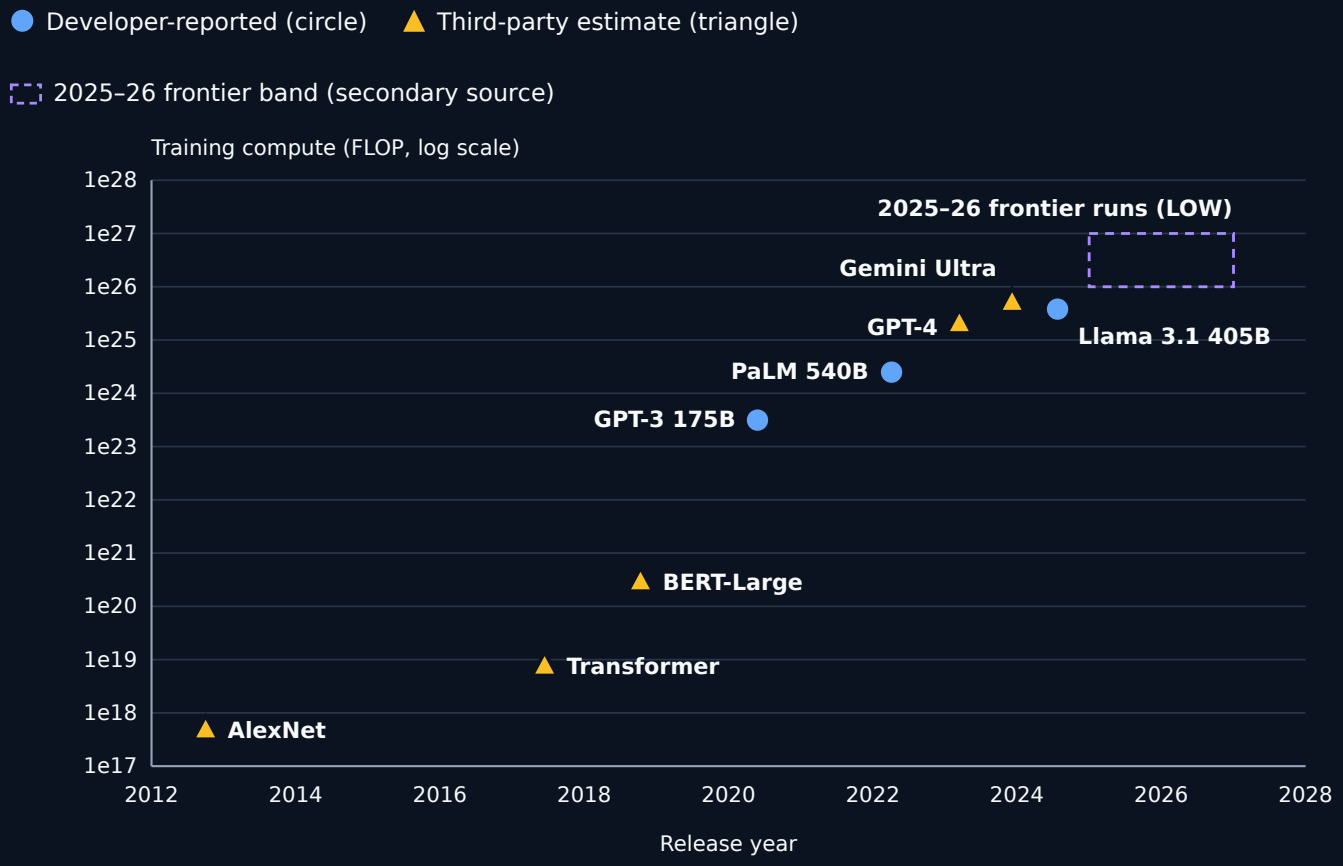


Figure 3. About eight orders of magnitude in twelve years. Marker shape encodes provenance: circles are developer-reported, triangles are third-party estimates. Hover a marker for its value. The 2025–26 band is drawn as a range, not as points, because it is not lab-disclosed.

Epoch's own estimates put GPT-4 at about 2e25 FLOP and Gemini Ultra at about 5e25, against GPT-3's roughly 3e23 in 2020. HIGH A secondary analysis places 2026 frontier runs in the 1e26 to 1e27 window. LOW

The physical stack, bottom to top

LAYER	WHAT IT IS	TYPICAL BINDING CONSTRAINT
Power and datacenter	Gigawatt-scale sites, cooling, grid interconnect	Energy availability, build time
Accelerators	GPUs and TPUs with high-bandwidth memory (HBM)	Chip and HBM supply, advanced packaging
Interconnect	NVLink, InfiniBand or Ethernet fabrics linking tens of thousands of chips	All-to-all bandwidth during training
Systems software	CUDA / ROCm, PyTorch or JAX, parallelism frameworks	Utilization, fault recovery at scale
Model	Architecture plus trained weights	Data quality, algorithmic efficiency
Serving	vLLM, TensorRT-LLM, llama.cpp, quantization	Memory bandwidth, KV cache size
Orchestration and app	Context assembly, tools, agents, user interface	Reliability, security, cost per task

The "binding constraint" column is synthesis, not a sourced ranking. MED

4. Implementation: the model is one component

A deployed AI product is mostly conventional software wrapped around the model call. The model is stateless and knows only what is in its **context window** on that pass. Everything that feels like knowledge of your files, memory of past conversations, or the ability to act is the orchestration layer putting text into that window and executing what comes out.

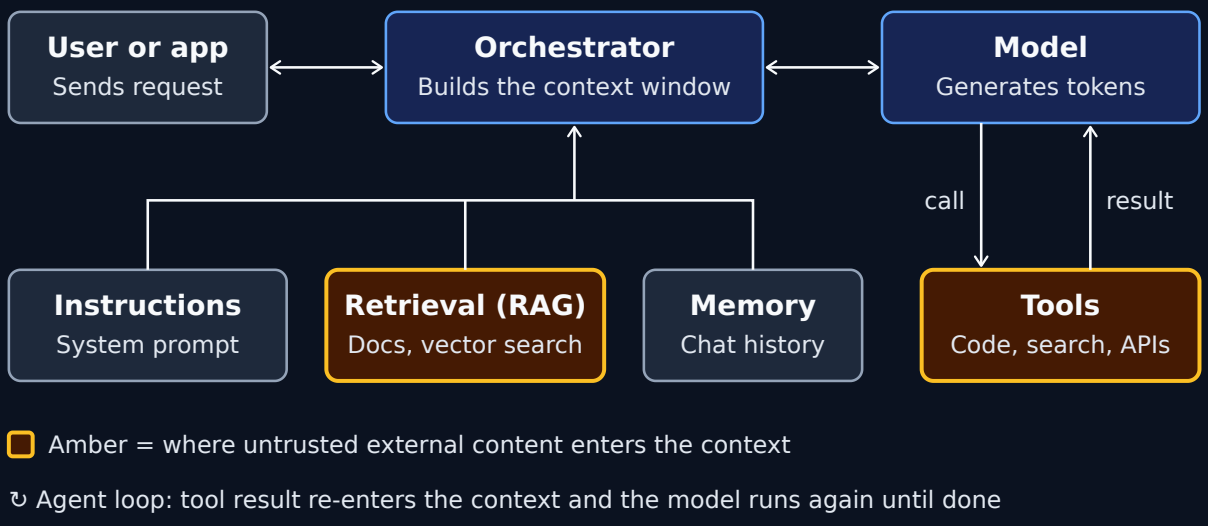


Figure 4. The orchestrator assembles instructions, retrieved documents and history into one context window. Tools close the loop that makes a model an agent.

Implementation patterns, in increasing order of autonomy and risk

- **Prompting.** Instructions plus the request. It is the cheapest lever and is often underused.
- **Retrieval-augmented generation (RAG).** Relevant documents are fetched and pasted into the context, so answers are grounded in current or private data without retraining.
- **Fine-tuning.** Small additional training, such as LoRA adapters. It changes format, tone or a narrow skill, and is a poor fit for injecting facts.
- **Tool use.** The model emits a structured call, the orchestrator executes it and returns the result. The Model Context Protocol (MCP) standardizes this interface.
- **Agents.** Tool use in a loop, with the model deciding the next step. Per-step errors compound and cost scales with loop length. If each step succeeds with probability p, a chain of N independent steps succeeds at most p^N of the time, which is an upper bound on decay rather than a forecast.

Security note: the lethal trifecta. An agent becomes dangerous when three things coincide: access to private data, exposure to untrusted content, and the ability to act autonomously. In Figure 4, retrieval and tool results are the entry points for untrusted content, and tools are also the action surface. The risk therefore concentrates in the orchestration layer, not in the model. Section 5 treats this as a layer in its own right.

Where the weights run: three deployment modes

MODE	WHO HOLDS THE WEIGHTS	COST SHAPE	TRADE-OFF	ROUTING LANE
Local open weights	You (llama.cpp, Ollama, quantized to fit consumer VRAM)	Hardware capex, near-zero marginal cost	Private and free per token; capability capped by memory	Lane 1
Hosted open weights	A cloud host behind a router	Low per-token opex	Cheap and capable; endpoint reliability varies	Lane 2
Closed frontier API	The lab only	Highest per-token opex	Top capability; no weight access	Lane 3

The routing decision across these modes is an intelligence-per-dollar question: cost per unit of completed task, including endpoint reliability, rather than headline capability.

5. The security layer: prominent, specified, not yet deployed

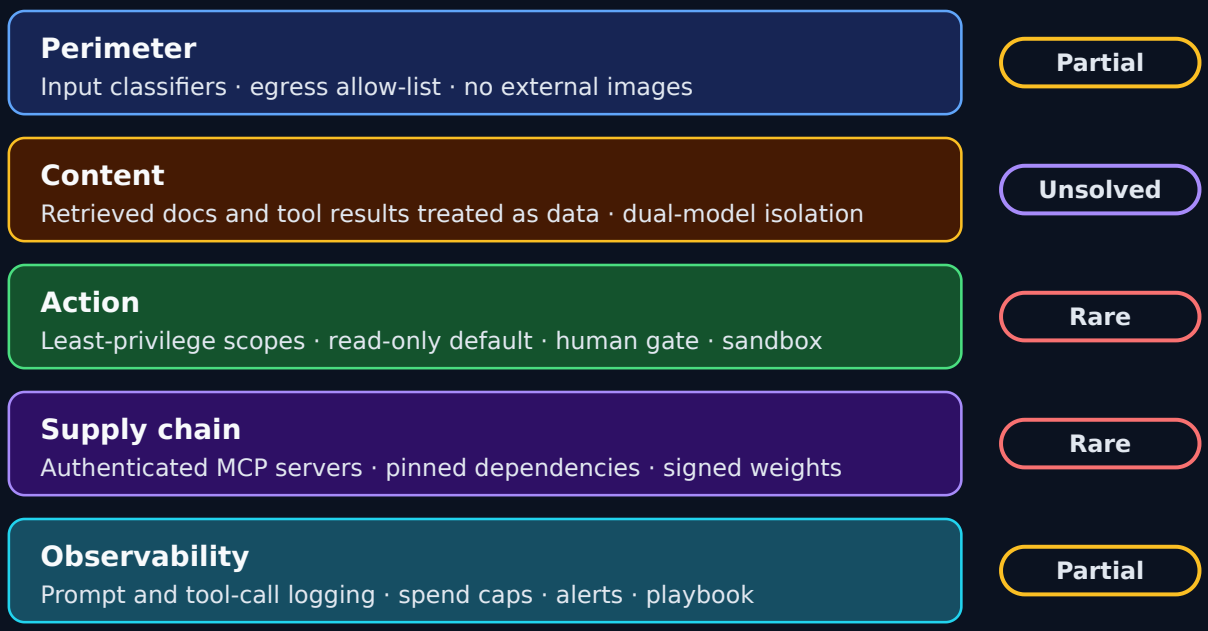
Figures 1 and 4 have no security component, and that omission mirrors the field. As of September 2026 the controls for AI systems are well specified in OWASP's Top 10 for LLM and Agentic Applications, MITRE ATLAS and the NIST AI Risk Management Framework, yet the incident record shows them deployed unevenly, and one class of threat has no complete fix at any layer. This is now a prominent issue that needs to be addressed, not a footnote to the architecture. The evidence divides into three tiers, and each carries a different status.

2026 evidence. Figures are as reported by the cited source; most sources sell security products or research, so treat counts as indicative.

- HIGH Prompt injection maps to six of the ten OWASP agentic categories and is described by OWASP's own researchers as unsolved. Root cause: the model receives system prompt, user text and retrieved content as one token stream with no channel that separates command from data.
- MED MCP authentication is optional in the protocol; the July 2026 spec revision leaves permission models, rate limiting and response integrity to the ecosystem. About 7,000 MCP servers were found reachable on the public internet; 14 CVEs by Q3 2026, over 30 remote-code-execution issues sharing one root cause.
- HIGH Supply chain: a backdoored postmark-mcp package reached roughly 300 organizations; a LiteLLM PyPI backdoor was downloaded about 47,000 times in three hours.
- HIGH Insecure output handling: Microsoft's Semantic Kernel shipped two remote-code-execution flaws (CVE-2026-26030, CVE-2026-25592) where model-influenced strings reached an interpreter. Microsoft's remediation guidance is the textbook control: treat any model-influenced tool parameter as attacker-controlled input.
- MED Governance: only 37% of organizations surveyed by OWASP contributors have policies to detect shadow AI deployments.
- LOW An estimate of over 200,000 MCP servers potentially compromisable through SDK flaws is an extrapolation, not a scan.

Three tiers of status

- **Tier 1: specified, solvable, not deployed at scale.** Authentication, least privilege, egress control, sandboxing, dependency pinning, spend caps and audit logging. These are ordinary security engineering. The gap is adoption, not invention.
- **Tier 2: unsolved at the model level.** Prompt injection. Filters and classifiers reduce it; nothing eliminates it, because the weakness is architectural (Section 6). The only structural defense is to separate the model that reads untrusted content from the model that can act, and that pattern is rare in production.
- **Tier 3: deployed but vendor-reported.** Build-side controls: safety evaluations, red-teaming, refusal training. Frontier labs report these in place and some independent testing exists. They belong in a separate column from the deploy-side gap and are not independently verifiable at the level of detail a buyer would want.



Status = author's assessment of typical 2026 production deployments, from the evidence above. Partial: common but inconsistent. Rare: specified, seldom deployed. Unsolved: no complete control exists.

Figure 5. Defense in depth, outside in. Every band is specified in OWASP, ATLAS or NIST guidance; the badges record how often it is found in practice.

Build-side threats and controls

THREAT	WHERE IT ENTERS	CONTROL	STATUS
Data poisoning, backdoors	Pretraining and fine-tuning data	Provenance filtering, dedup, canary strings, held-out backdoor evals	PARTIAL
Malicious model files	Weight downloads; pickle-based formats can execute code on load	safetensors or GGUF only, checksum and signature verification, trusted hubs	PARTIAL
Model theft, distillation	Inference endpoint	Rate limits, output watermarking, terms of service	WEAK
Evaluation gaming	Benchmark contamination	Held-out and private evals, contamination checks	PARTIAL
Unsafe capability	Post-training	Refusal training, constitutional methods, red-teaming, independent safety evals	DEPLOYED vendor-reported

Implement-side threats and controls

THREAT	WHERE IT ENTERS	CONTROL	STATUS
Direct prompt injection, jailbreaks	User input	Input classifiers, system-prompt hardening, output filtering	PARTIAL
Indirect prompt injection	Retrieved documents, web pages, tool results, email	Treat fetched content as data; separate the model that reads from the model that acts	UNSOLVED
Data exfiltration	Tool calls, markdown image URLs, outbound requests	Egress allow-lists, no auto-rendered external images, tool-result redaction	RARE
Excessive agency	Agent loop	Least-privilege tool scopes, read-only default, human approval on irreversible or financial actions	RARE
Insecure output handling	Downstream code	Never eval or shell model output unsandboxed;	PARTIAL

THREAT	WHERE IT ENTERS	CONTROL	STATUS
		parameterize; sandbox code execution	
Supply chain: MCP servers, plugins, skills	Orchestration layer	Authentication on, pinned versions, third-party review, scoped credentials per server	RARE
Secrets and billing exposure	API keys	Per-use-case keys, hard spend caps, no keys in prompts or repositories	PARTIAL
Privacy and residency	Context window, logs	Data minimization, retention policy, zero-retention endpoints where offered	PARTIAL
Observability	Everything	Log prompts, tool calls and outputs; anomaly alerts; incident playbook	PARTIAL

The architectural answer to Tier 2

Because no filter reliably separates instructions from data inside one context window, the defense that works is structural: two models with different privileges. A quarantined reader sees untrusted content but has no tools. A privileged actor has tools but never

sees raw untrusted text, only structured output from the reader that the orchestrator validates against a schema. Injected instructions cannot reach anything that can act.

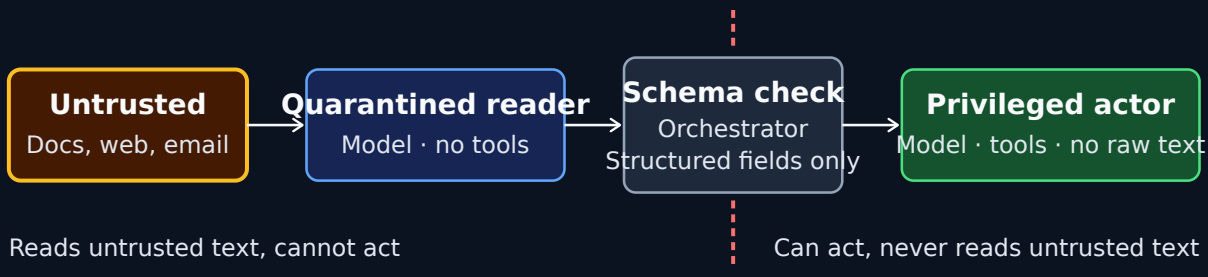


Figure 6. The dual-model pattern (also called dual LLM, or CaMeL in the research literature). It is the one control that addresses indirect prompt injection structurally rather than probabilistically.

The closing point is about adoption, not invention. Nearly everything in the two tables is conventional security engineering applied to a new component: authenticate, scope, log, cap, verify. A single operator can apply most of it in an afternoon: per-use-case API keys with hard credit limits, headless jobs restricted to read-only, dependency pinning, mandatory access control left enforcing. The controls are cheap and known. What is missing in 2026 is the expectation that they ship by default.

6. How a model reads one token, and where backpropagation fits

A common mental model is that a language model "reads" a character such as **A**, tries several interpretations, and backtracks when one fails. The real mechanism is different in two ways. First, the model never sees a shape: the tokenizer hands it an integer ID, and the work is deciding what that ID means *here* (the article "a", the letter's name, a grade, a blood type, a musical note, a variable). Second, within a single forward pass nothing backtracks. Each layer rewrites the token's vector, so an early, generic reading is overwritten by a context-specific one. Interpretability researchers call reading those intermediate states out the **logit lens**; the process itself is **iterative refinement in the residual stream**.

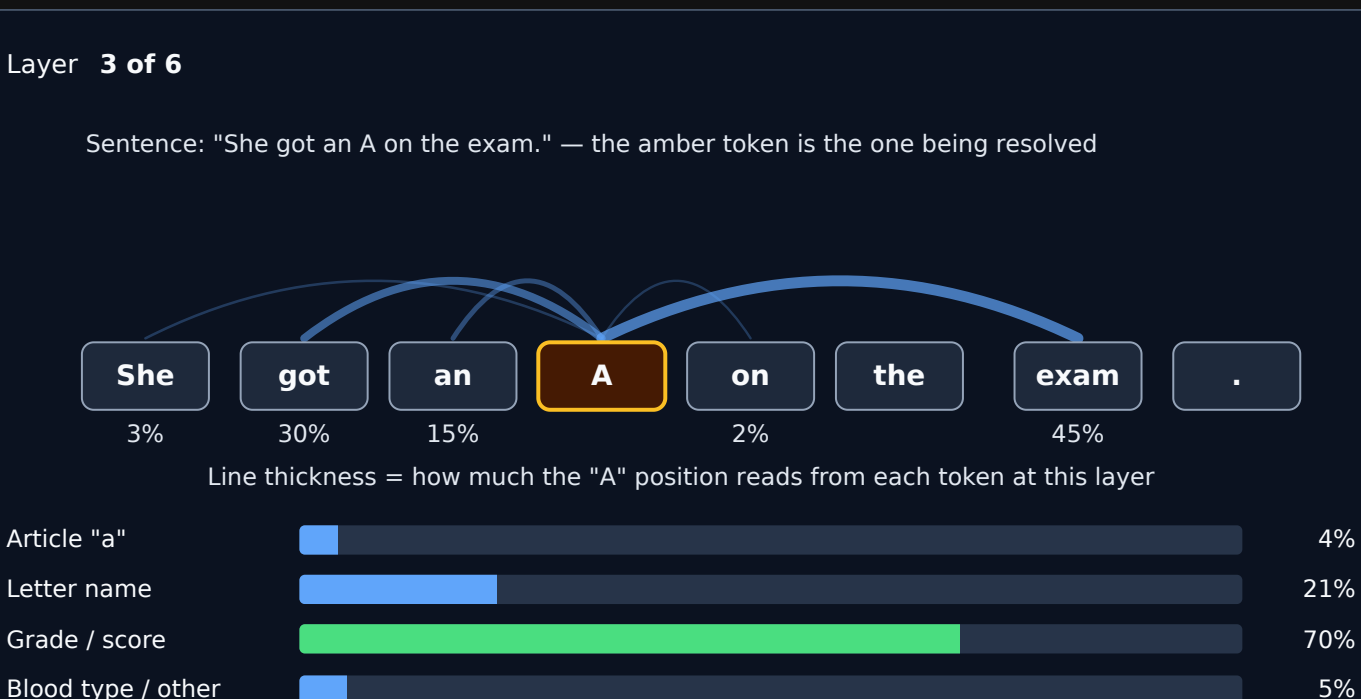


Figure 7. Per-layer resolution of one token. The web edition is interactive: drag the layer slider. The PDF edition shows layer 3, where the reading flips from "letter" to "grade". Values are illustrative, not measured from a real model.

Two things to notice. The "article" reading at layer 0 was never wrong: it was the correct prior with no context. And the flip at layer 3 is not a decision the model could revisit; it is arithmetic. Attention added the "exam" vector into the "A" position, and the sum now points in the "grade" direction.

Recognizing A as a *shape* is the job of a vision encoder in a multimodal model. It is also a single forward pass, but the hierarchy is spatial rather than contextual, and it is the one place where the picture of competing hypotheses is closest to literal.

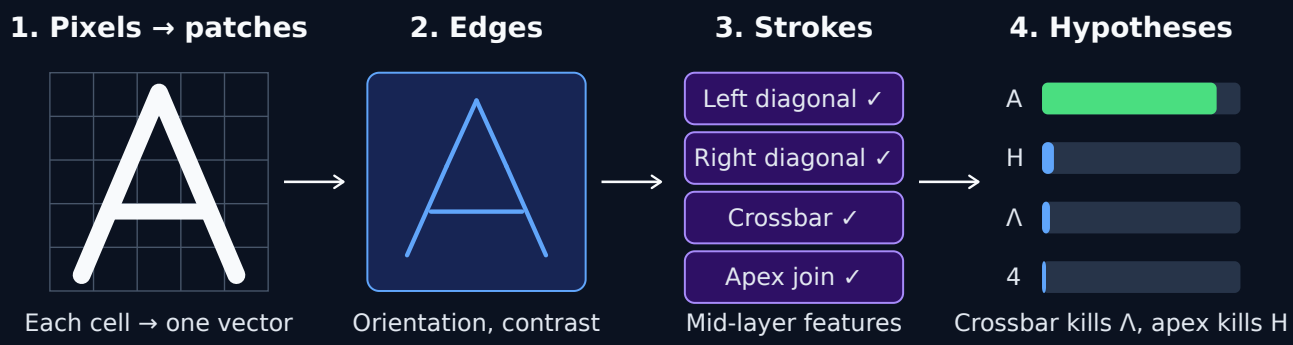


Figure 8. Spatial hierarchy in a vision encoder. Λ and H are not tried and abandoned; their evidence is simply outweighed in the same forward sum.

The correct terms

WHAT IS HAPPENING	TERM	WHERE IT RUNS
Token meaning refined layer by layer	Iterative refinement in the residual stream; read out with the logit lens or tuned lens	Inference, inside one forward pass
Competing readings coexisting until one dominates	Superposition of features, resolved by contextualization	Inference, inside one forward pass
Model writes a hypothesis, then corrects it	Self-correction, self-verification, reflection	Inference, in the output tokens
Several candidate continuations kept, losers dropped	Beam search; search over reasoning paths (tree of thoughts, best-of-N)	Inference, across candidate outputs
Error pushed back through the layers to adjust weights	**Backpropagation**, with gradient descent as the update rule	Training only

So the process that resembles backtracking through the layers is **backpropagation**, and it happens only during training. After a wrong prediction, the backward pass computes how much each weight contributed to the error, layer by layer from the output back to the input, and gradient descent nudges each one. It is how the stroke detectors in Figure 8 were learned. Once the weights are frozen for deployment it never runs again.

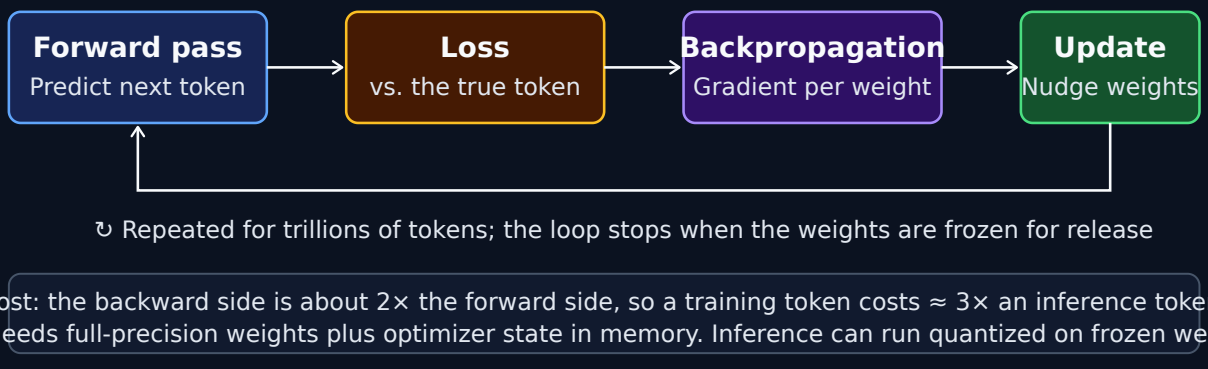


Figure 9. Backpropagation is the training-time loop. Its roughly 3× per-token cost and memory footprint are the mechanical root of the capex/opex split in Figure 1.

7. Provenance and confidence

- HIGH Lifecycle structure, transformer loop, prefill/decode asymmetry, KV cache role, implementation patterns. Stable, widely documented engineering.
- HIGH The 5× per year compute trend and the GPT-3, GPT-4 and Gemini Ultra compute figures, checked against Epoch AI pages on 19 Sep 2026. The GPT-4 and Gemini Ultra figures are Epoch estimates, not lab disclosures.
- MED The AlexNet, Transformer, BERT-Large, PaLM and Llama 3.1 405B chart points are from recall of the source papers and Epoch's dataset and were not re-pulled for this revision. PaLM and Llama are developer-reported values; the other three are third-party estimates.
- MED The stack table's "binding constraint" column is the author's synthesis.
- HIGH Section 6 mechanism: one-directional forward pass, residual-stream refinement, logit lens as a read-out, and the placement of backpropagation, self-correction and search. Standard usage in the ML and interpretability literature.
- MED The ≈3× training-to-inference cost per token is the standard rule of thumb (forward ≈ 2·N·D FLOP, full step ≈ 6·N·D), not a measured figure for any named model.
- LOW The per-layer percentages and attention weights in Figure 7 and the hypothesis bars in Figure 8 are illustrative constructions, not measurements.

- HIGH Section 5 threat classes and controls follow the OWASP Top 10 for LLM and Agentic Applications, MITRE ATLAS and NIST AI RMF categories. Prompt injection's unsolved status is OWASP's own characterization.
- MED Section 5 status badges (Partial / Rare / Unsolved) are the author's assessment of typical 2026 production deployments, inferred from the incident record; they are not survey data. Counts of exposed MCP servers, CVEs and downloads are as reported by security vendors and researchers with a commercial interest in the topic.
- LOW The "over 200,000 potentially compromisable MCP servers" figure is an extrapolation by its source, not a scan.
- LOW The 2025–26 band of 1e26 to 1e27 FLOP comes from a single secondary consultancy write-up citing Epoch, not from lab disclosure.

Sources

- Epoch AI — Trends in Artificial Intelligence (compute growth rate, doubling time, cost and power growth)
- Epoch AI — Training compute of frontier AI models grows by 4–5× per year (GPT-3, GPT-4, Gemini Ultra estimates)
- Epoch AI — Data on AI Models (notable models dataset)
- Help Net Security — Prompt injection still drives most agentic AI security failures in production (OWASP 2026 report coverage)
- The Agent Report — MCP Security in Q3 2026: CVEs, exposed servers, mitigations
- Cloud Security Alliance — MCP Security Crisis: systemic design flaws in AI agent infrastructure
- Microsoft Security Blog — When prompts become shells: RCE vulnerabilities in AI agent frameworks
- Infosecurity Magazine — Prompt injection remains unsolved, OWASP researcher warns
- The Hacker News — MCP design vulnerability enables RCE
- OWASP GenAI Security Project — Top 10 for LLM and Agentic Applications · MITRE ATLAS · NIST AI RMF
- Deluair Consultancy — Frontier AI training cost trajectory 2026 (secondary source for the 2025–26 band)