OPEN ACCESS

REVIEW

INNOVATION PRESS

Journal homepage: www.the-innovation.org/ai-plus

AI Plus

AI Plus 1 (2026):100010

Contents lists available at INNOVATION PRESS

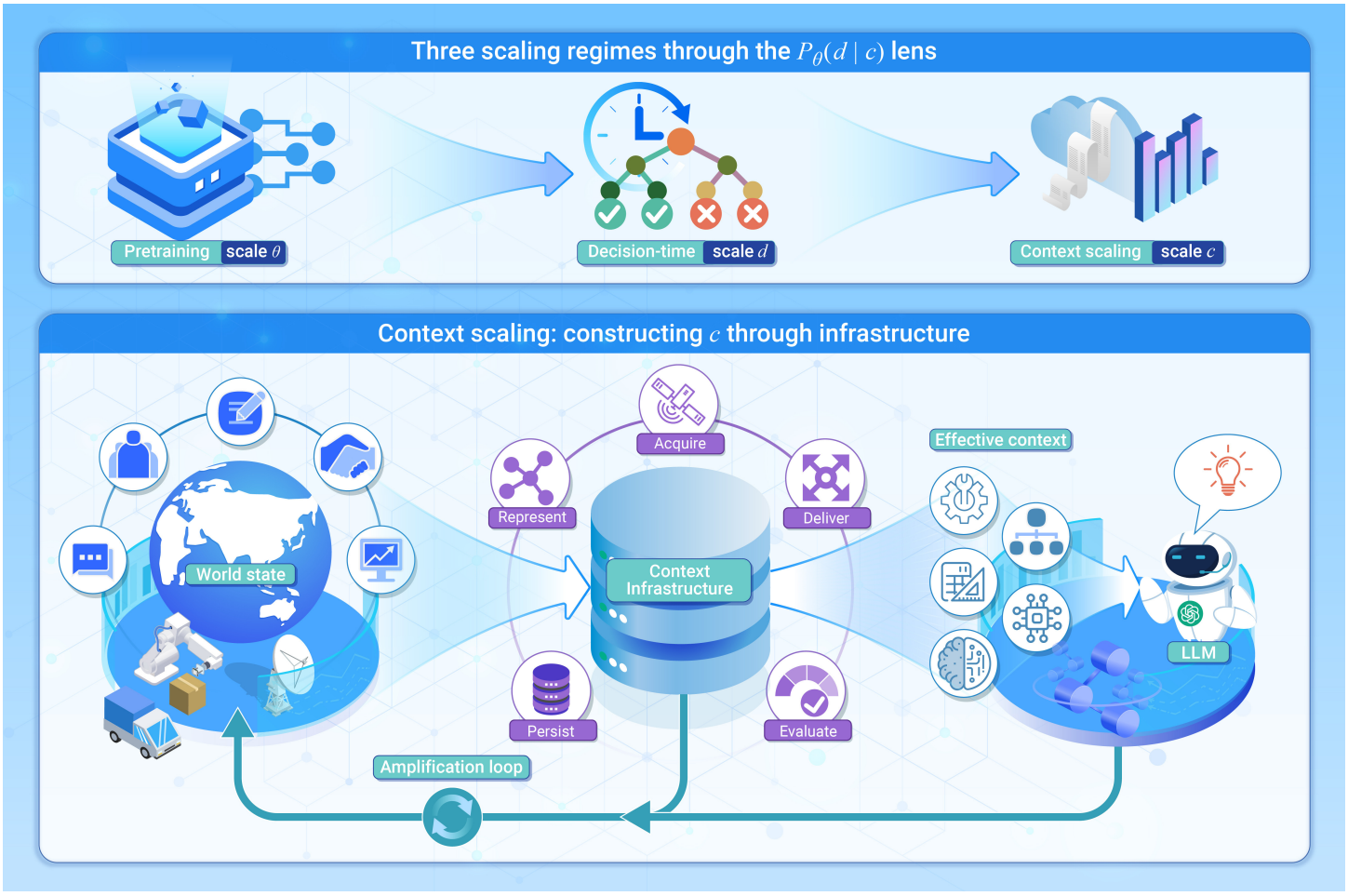
Context Scaling: A New Frontier of AI Scaling

Xipeng Qiu[1,2,*]

*Correspondence: xpqiu@fudan.edu.cn

Received: June 4, 2026; Accepted: August 15, 2026; Published Online: August 18, 2026; https://doi.org/10.59717/ipj.aiplus.2026.100010

GRAPHICAL ABSTRACT



PUBLIC SUMMARY

- CGS Some deployed failures arise because current task state is unavailable, distorted, poorly selected, or unused.
- Context scaling is a research program for measuring returns on context-infrastructure budget B_c.
- Training, decision-time, and context budgets interact and should be reported separately.
- Effective-context value is measured after intervention against a declared task and baseline.
- Utility-cost curves, not window length alone, are the primary measurement target.

REVIEW

OPEN ACCESS

INNOVATION PRESS

Journal homepage: www.the-innovation.org/ai-plus

AI Plus

AI Plus 1 (2026):100010

Contents lists available at INNOVATION PRESS

Context Scaling: A New Frontier of AI Scaling

Xipeng Qiu[1,2,*]

[1]OpenMOSS Team, Fudan University, Shanghai, China

[2]OpenMOSS Team, Shanghai Innovation Institute, Shanghai, China

*Correspondence: xpqiu@fudan.edu.cn

Received: June 4, 2026; Accepted: August 15, 2026; Published Online: August 18, 2026; https://doi.org/10.59717/ipj.aiplus.2026.100010

Citation: Qiu X. (2026). Context Scaling: A New Frontier of AI Scaling. *AI Plus* **1**:100010.

A recurring failure mode in deployed AI lies at the boundary between the state a task depends on and the model-visible context available when the system acts. In such cases, what looks like weak reasoning can instead reflect failed access: relevant user state, repository state, tool traces, sensor readings, or environment changes are absent, distorted, or poorly selected at decision time. Viewing a model as a conditional $P_\theta(d|c)$ embedded in a system with three budgets (training budget B_θ, decision-time budget B_d, and context-infrastructure budget B_c), we read recent progress as a sequence of bottleneck shifts. Pretraining scaling spends B_θ to improve the base model; decision-time scaling spends B_d to search, refine, or verify candidate decisions under a supplied context; context scaling spends B_c to improve what the system can make available before a decision. The three programs interact. We treat context infrastructure as a distinct investment and measurement target, while leaving the form of any general scaling law open. Reporting only the model and per-query inference compute obscures the contribution of context infrastructure. The relevant unit of analysis is the model--infrastructure pair. Stronger models can exploit richer decision-time information states; stronger context infrastructure can acquire, represent, deliver, evaluate, and persist those states; infrastructure-mediated deployment traces can improve both. Effective-context value is a task- and intervention-dependent attribution measured after observing the utility of model-visible context. Availability, fidelity, selection, and utilization failures separate potentially relevant deployment state from information that improves a decision. The paper closes with three problem clusters: grounded evaluation and reporting, interaction-data supply and full-cost infrastructure, and safety and governance for persistent and adaptive systems.

INTRODUCTION

Consider one recurring class of failure when a large language model (LLM) is deployed into a real task. A capable base model with a reasonable decision policy can still fail because task-relevant state belongs to the current user, repository revision, tool trace, sensor stream, or environment and is not available in usable form at decision time. We call this the context bottleneck. It is one possible bottleneck alongside model knowledge, reasoning, control, and objective specification; the empirical question is when it becomes the binding one.

The state that deployed systems must reason about is hard to handle: it evolves, has structure, arrives through non-textual channels, and can include tacit or permissioned information that resists direct serialization. Context scaling names the work of reducing this bottleneck by increasing usable decision-time access to deployment state, which takes more than lengthening a text buffer. Its target is higher utility from budgeted context infrastructure.

The dominant accounting of progress emphasizes training investment and, more recently, per-query inference compute. Current external state enters the decision through the infrastructure that constructs model-visible context; the budget for that infrastructure is largely absent from this accounting.

Contributions.

1. We augment the $P_\theta(d \mid c)$ lens with explicit budgets (B_θ, B_d, B_c), distinguishing training, decision-time, and context-infrastructure investments while retaining d as the output variable.

2. We define effective-context value as a task-conditional, counterfactual attribution, separate it from raw state and model-visible context, and identify the availability, fidelity, selection, and utilization failures that reduce decision value.

3. We frame context infrastructure, with agent harnesses as its best-developed runtime form, as the system layer that converts deployment state into model-visible context; four descriptive dimensions identify where different mechanisms are stressed.

4. We organize the open problems into grounded evaluation and reporting, interaction-data supply and full-cost infrastructure, and safety and governance for persistent and adaptive deployments.

This article is a position paper with a narrative, non-exhaustive survey; it reports no new model or controlled experiment. Context scaling complements pretraining, alignment, and decision-time scaling. Runtime infrastructure remains responsible for acquiring and maintaining deployment-specific state such as the current repository revision, a live sensor reading, or an authorized user history. The survey prioritizes work that isolates a budget in (B_θ, B_d, B_c), implements a context-lifecycle mechanism, or exposes a measurement, data-supply, safety, or governance constraint; peer-reviewed studies provide the main evidence base, while recent arXiv papers, model reports, and practitioner documentation describe frontier practice.

Testable predictions. The program predicts positive returns on B_c in controlled comparisons at practically relevant model scales, some transfer of infrastructure improvements within declared task families, and context-value proxies that predict held-out utility better than raw token length or context cost does.

Compact glossary. We use *deployment state* for the external and persistent state that could inform a task; *model-visible context* for the finite input actually supplied at one invocation; *context infrastructure* for the runtime layer that maps the former into the latter; *context policy* π_c for the policy controlling that mapping; *context budget* B_c for the costed capacity spent on acquisition, representation, selection, and persistence; and *effective-context value* for the utility attributed to a context intervention under a declared baseline. Each term names a distinct object.

A UNIFIED VIEW OF AI SCALING

At the level of one model invocation, an AI model can be described as a conditional distribution over decisions given a model-visible context,

$$P_\theta(d \mid c), \tag{1}$$

where θ denotes the model's parameters, c the conditioning context (a prompt, prior observations, retrieved documents, a tool-call history, an image, an audio stream, ...), and d the decision: a token, a reasoning chain, an action in an environment, or another output the system commits to. Training compute, search, retrieval, and verification enter through the surrounding system, which induces the decision distribution

$$d \sim Q_{\theta,\pi}(d \mid x; B_\theta, B_d, B_c), \tag{2}$$

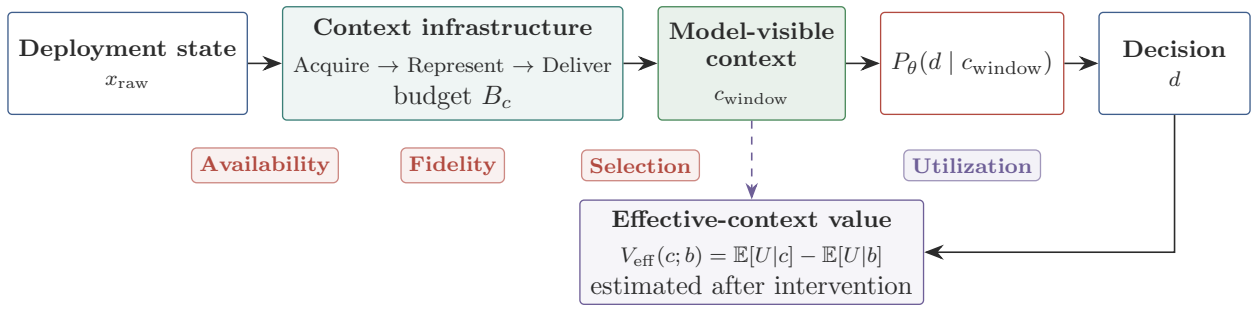


Figure 1. Deployment state x_{raw} is transformed by context infrastructure into model-visible context c_{window}, on which P_θ actually conditions Effective-context value V_{eff} is estimated afterward from utility under declared interventions and baselines. Availability, fidelity, and selection failures occur before the model invocation; utilization failure occurs when faithfully delivered evidence fails to improve the model's decision.

where x is deployment state, π denotes the policies and interfaces surrounding the model, B_θ is the training investment embodied in θ, B_d is per-decision compute for proposing, searching, and verifying outputs, and B_c is the budget for acquiring, representing, selecting, and persisting model-visible context. The three budgets can interact, and many systems jointly vary more than one.

The system-level distribution still leaves its two context-side objects informal: the deployment state x that a system can draw on, and the value created when part of that state reaches the model as usable context. Defining both makes the return on B_c measurable before the three investment programs are compared.

From world state to effective context

Context should be distinguished from environment. The environment is the external task world in which the system acts: a repository and terminal for a coding agent, a physical scene for a robot, or a user's evolving work state for an assistant. We treat deployment as a sequence of decision steps indexed by t; at each step the model commits a decision d_t conditioned on model-visible context c_t, produced from deployment state by the surrounding infrastructure. The environment is larger than this representation, dynamic, and only partially observed. The distinction between c and d is local to a model invocation: across a trajectory, prior decisions, tool results, and execution traces become part of later context.

Context infrastructure. The model-level c_t is the output of context infrastructure: the system layer that decides what state reaches the model at decision time. Increasing B_c expands the budgeted capacity of this mapping.

Model-visible context varies in utility. Four failures can intervene between deployment state and task utility. *Availability failure* leaves relevant state unreachable; *fidelity failure* distorts it during encoding or summarization; *selection failure* excludes, delays, or poorly places represented state under the read budget; and *utilization failure* occurs when the model fails to use faithfully delivered evidence. The first three are primarily infrastructure-side failures, while utilization depends strongly on the model; in practice the boundary is joint because selection and presentation can change what a fixed model uses.

Deployment state, model-visible context, and effective-context value. Three objects should be separated. Let x_{raw} denote the deployment state and affordances that could in principle inform a decision: retrieved passages, tool outputs, observations, memory entries, logs, interaction histories, tool schemas, and handler states. A context-production mapping $\mathcal{I}_{\pi_c}$ acquires and represents parts of this heterogeneous state, then selects a finite model-visible input c_{window} under budget B_c:

$$c_{\text{window}} = \mathcal{I}_{\pi_c}(x_{\text{raw}}; B_c).$$

The mapping transforms deployment state: a sensor stream may become an embedding, a tool trace may become a structured record, and several sources may become one summary. The model conditions on c_{window}. We reserve *effective-context value* for the utility attributed to that input after fixing a task distribution, model, decision procedure, utility function, and intervention baseline. Write $U(\tau)$ for trajectory utility and b for a declared baseline context. The effective-context value of an intervention producing c is

$$V_{\text{eff}}(c;b) = \mathbb{E}[U(\tau) \mid c] - \mathbb{E}[U(\tau) \mid b].$$

It is therefore a task-conditional causal estimand defined over interventions on the model-visible input. When a declared unit i (a passage, source, memory slot, sensor channel, or structured field) is ablated or replaced, its marginal attribution is

$$\Delta U_i = \mathbb{E}\big[U(\tau) \mid c\big] - \mathbb{E}\big[U(\tau) \mid c \setminus \{i\}\big],$$

where $c \setminus \{i\}$ denotes the pre-specified replacement intervention, which may replace a source or representation instead of deleting tokens.

Synergy can be represented by averaging marginal contributions over entry orders, following a Shapley-style allocation;[1,2] the result still depends on the chosen units, replacement baseline, and sampled coalition distribution. Interventions on latent, compressed, or multimodal state should target a source, channel, memory slot, or representation directly. Negative contributions and redundancy remain part of the estimate. Figure 1 summarizes the transformation and the four failure points.

THREE SCALING PROGRAMS

The three-budget lens sorts recent progress into three interacting investment programs. Pretraining scaling spends B_θ to improve the base conditional over a wide range of (c,d) pairs. Decision-time scaling spends B_d to explore, refine, or verify candidate decisions under model-visible context held fixed by protocol. Context scaling spends B_c to change the mapping from deployment state x to model-visible context c. Figure 2 places the three programs around the same model invocation.

Demarcation by counterfactual budget. For an intervention that touches several components, classification is secondary to attribution. We first report which of (B_θ, B_d, B_c) changed and estimate a response surface $U(B_\theta, B_d, B_c)$ on a declared task family. A single-axis label is used only when two budgets are held fixed by design and the third is the manipulated resource.

Take the two cases that most often blur the boundary. Reasoning reinforcement learning (RL; o1- and R1-style[3,4]) changes B_θ during training and often enables a larger effective B_d at deployment; its total gain should therefore be decomposed with both budgets reported. Tool-use and retrieval-policy RL (e.g. Search-R1[5]) changes B_θ, uses B_d to choose queries or calls, and consumes B_c to return evidence. Withholding evidence establishes the value of external state; budget attribution additionally requires varying retrieval capacity or evidence quality at matched model and decision budgets and reporting interaction effects.

Pretraining scaling

Empirical scaling laws describe how pretraining loss decays as a joint function of parameters and tokens under a fixed compute budget;[6,7] much generic model-improvement investment has therefore gone into larger models, datasets, and training runs. Data-constrained scaling work suggests that high-quality natural data is becoming a tighter resource, while filtering, synthetic augmentation, and multi-epoch training alter that constraint.[8,9] Situated deployments also require current interaction history, tool feedback, user-authorized state, or post-cutoff information beyond the pretrained prior.

Decision-time scaling

Decision-time scaling varies the inference compute spent per decision

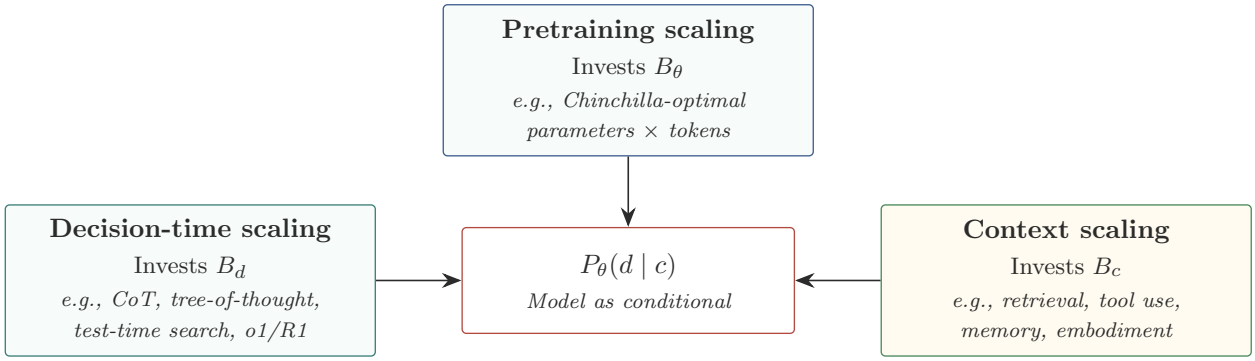


Figure 2. Three coupled investment programs around a model invocation $P_\theta(d \mid c)$ Pretraining spends B_θ on the base model; decision-time scaling spends B_d on proposing, searching, or verifying decisions under a protocol-fixed context; context scaling spends B_c on the infrastructure that maps deployment state into model-visible context. The budgets are reported separately because their returns can interact.

Demarcation rule

To attribute a gain, hold two budgets fixed and vary the third. Report mixed interventions by the budgets they change:

- vary training data, parameters, or optimization while holding the runtime protocol fixed ⇒ estimate the marginal return on B_θ;
- vary per-decision search, sampling, rollout, or verification while holding model-visible context fixed ⇒ estimate the marginal return on B_d;
- vary acquisition, representation, selection, or persistence capacity while holding the model and decision procedure fixed ⇒ estimate the marginal return on B_c.

under a protocol that holds the model and supplied external evidence fixed. Chain-of-thought prompting unrolls an intermediate reasoning trajectory;[10] self-consistency, tree-of-thought search, and verifier- or process-reward-guided decoding sample, search over, and score multiple trajectories;[11-14] reasoning-specialized models can allocate more test-time computation per query,[3,4] with empirical compute–performance curves in restricted settings.[15] Current repository changes, sensor readings, and user actions enter the decision only through acquisition, which spends B_c and often B_d on information-seeking actions.

Context scaling

The third program, the subject of this paper, invests the infrastructure budget B_c in constructing the conditioning variable c.

Context scaling covers budgeted investments that change the information state on which decisions are conditioned. Its scope includes long-context mechanisms and extends to retrieval, memory, tools, and grounded observation; a million-token window filled with static documentation can increase token count while adding little task utility. Agentic decision-making has a broader scope that also includes goals, planning, and action selection.

We therefore define context scaling as a budgeted intervention on the capacity to acquire, represent, select, or persist deployment state for model use.

Measuring and optimizing context value. The counterfactual definition yields intervention-specific estimates of context value. *Declare* the task distribution, utility horizon, model checkpoint, context units, replacement baseline, evaluator, and all three budgets. *Intervene* with a randomized or matched design on one context-infrastructure capacity at several B_c levels; when the intervention also changes tool calls, rollout length, or verifier use, report the corresponding change in B_d and estimate the interaction. *Measure* a utility–cost curve and uncertainty intervals, using grouped or randomized subset interventions when exact Shapley attribution is infeasible. *Report* the marginal utility per unit context cost and, optionally, a decision-relevance density (the fraction of declared units whose specified replacement lowers utility beyond a stated tolerance). Density is comparable within protocols that share the same units, tokenization or representation, replacement baseline, task distribution, and tolerance; compression, duplication, or re-partitioning can change density while preserving information. Cross-domain comparison should therefore use disclosed cost and utility curves.

The context policy π_c chooses infrastructure actions to maximize expected task utility minus deployment-specific context costs. Retrieval, tool-use, memory-write, active-sensing, and context-budget policies expose different learning signals;[5,16] sparse terminal rewards can discourage informative questions because they fail to distinguish useful queries along the trajectory.[17] Objective design also faces a substantive tension between answer accuracy, which rewards resurfacing known content, and information gain, which rewards reaching new material.

What kind of scaling law exists?. Mechanism-specific response curves currently provide the closest counterpart to the Kaplan or Chinchilla scaling laws. B_c purchases heterogeneous capacities, and c is a structured information state. Here, a scaling program means the systematic estimation of marginal utility as B_c varies at matched B_θ and B_d.

Restricted scaling laws. The context-aware scaling law of Montgomery et al.[18] models downstream task performance as the product of two saturating power laws (one in training compute, one in prompt context length) times a penalty when the prompt exceeds the model's context window. The form fits in-distribution downstream performance across three orders of magnitude of training compute on three tasks, with a task-dependent characteristic context length. A complementary line[19] argues from an intrinsic-entropy framework that training-data volume bounds the optimal context length. Together they characterize restricted context coordinates and their interaction with training scale.

Empirical patterns. Dense retrieval exhibits regular scaling with model size and annotation budget.[20] LongPPL predicts long-context benchmark scores more reliably than standard perplexity in the studied setting.[21] Frontier technical reports document growing context windows and agent-oriented context training,[22,23,24,25] and benchmarks such as RULER and LongBench v2 separate window capacity from evidence utilization.[26,27]

Mixed budget interventions. The budget view reports mixed cases by expenditure. Domain-adaptive continued pretraining varies B_θ under transfer-style laws.[30] In-context learning changes model-visible context and may also change generation cost. Retrieval and tool use spend B_d on query or action selection and B_c on interfaces, returned evidence, and state handling. Chain-of-thought primarily varies B_d when no external observation is acquired; a world-model rollout combines decision-time simulation with a learned proxy for environment transitions. Persistent memory spends B_c on storage, validation, and recall, while continual weight adaptation also changes B_θ.

Complementarity and interaction. Returns on B_c are coupled to both B_θ and B_d: richer context pays when the model and decision procedure can use it, and some context gains should shrink as models improve. The resulting object is a joint response surface whose B_c slope can vary with model scale and decision-time compute. This coupling also yields a falsification condition: if controlled returns on B_c vanish at matched B_θ and B_d, a separate context program loses its motivation.

Model scale and infrastructure value. The bitter lesson[31] predicts that

Observed returns on context investment

- **Datastore capacity.** Shao et al.[28] report monotonic gains as a retrieval datastore grows, including settings where a smaller retrieval-augmented model outperforms a larger standalone model at matched training compute. Runtime retrieval and storage costs determine the resulting budget substitution.
- **Learned retrieval policy.** Jin et al.[5] report 20–41% relative gains over retrieval-augmented baselines on factual QA. The intervention spans (B_θ, B_d, B_c).
- **Retrieval and generation allocation.** Yue et al.[29] hold the base model fixed and vary a budget distributed across retrieved documents and generation, tracing a coupled (B_c, B_d) frontier.

learned policies will replace many hand-tuned chunking rules, ordering workarounds, and fixed summarization heuristics. Proposals for *agent internalization* extend this prediction to goals, identity, and deliberation.[32] Runtime acquisition and persistence serve tasks that depend on exact current repository state, sensor observations, or authorized interaction history. This yields a testable trend: as B_θ increases, representation and selection interventions should often lose value, while acquisition and persistence should retain value in proportion to a task's dependence on current external state.

Why context matters now

Two developments make c increasingly important.

On the practical side, state created after training and private authorized state must enter through runtime infrastructure. End-to-end browsing agents combine reinforcement learning, search, tools, and evidence access.[33] Their performance on GAIA[34] and BrowseComp[35] exposes the model–infrastructure pair as the unit of performance.

On the theoretical side, Silver and Sutton[36] describe a shift from static human-generated data toward *experience*: trajectories that agents generate through interaction with an environment. Experience updates θ as training signal and populates c as grounded conditioning state at decision time. Both uses depend on infrastructure that acquires, persists, and evaluates interaction traces. The same infrastructure pattern applies to assistants, scientific tools, and knowledge-work systems outside an RL-agent setting.

An asymmetry with pretraining. A web-scale corpus can support a broadly reusable base model, whereas context infrastructure is partly domain-specific: a coding repository, clinical workflow, and robot expose different acquisition surfaces, permissions, representations, and evaluators. Transferable substrate includes trace schemas, memory and retrieval primitives, access control, replay, and reporting conventions. Per-domain rebuild cost belongs in B_c and shapes the economics of context investment relative to model scaling.

Relation to adjacent framings. Table 1 sets out labels in the recent literature that overlap with context scaling: what each names, its measured object, and its role in context scaling. Here, *utility–cost evaluation* means measuring task utility across several B_c levels under declared controls and reporting the corresponding context-infrastructure cost and any interaction with B_d.

CONTEXT INFRASTRUCTURE

Context infrastructure is the runtime system that turns deployment state into model-visible context and model decisions into observable consequences. It is the operational layer on which B_c is spent. It includes stores, tool and execution interfaces, memory systems, and feedback channels that determine what the model can know, what it can do, and what the system can learn from the result.

Formally, the runtime can be written as a compact partially observed interaction in which model-visible context is constructed under a budget. Let $x_t = \{h_t, m_t, e_t\}$ denote infrastructure state: history h_t, memory m_t, and environment-facing state e_t (observations, tool affordances, permissions, routing). This is the step-t decomposition of the deployment state x_{raw} above, and $\mathcal{T}(x_t, a_t)$ is the action-conditioned form of $\mathcal{T}_{\pi_c}$. A context policy π_c chooses an infrastructure action a_t; the infrastructure produces model-visible context c_t; the model makes a decision d_t; and the execution interface returns outcome evidence o_t. Figure 3 separates the online prepare–act path from feedback that updates history and environment state, persistent memory, and model parameters.

Prior information-state traditions. This framing builds on partially observable Markov decision process (POMDP) belief-state planning, active perception, and information-gathering control, which couple actions to future observations and trade task reward against information value.[48,49] Interactive information retrieval studies search as an evolving dialogue,[50] while distributed-cognition work treats external representations and artifacts as part of a cognitive system.[51] In LLM systems, the input available to the base model is often a serialized, permissioned, and budgeted construction assembled from histories, memories, retrievals, tool interfaces, and multimodal encoders. The context policy π_c makes that engineering layer explicit and separately auditable from P_θ.

Context infrastructure across three update paths

Existing context-engineering practice organizes operations such as selecting, compressing, writing, and isolating context.[38] We group such operations under five lifecycle labels with distinct roles. Acquire, Represent, and Deliver form the online context-production path. Evaluate supplies measurement and credit assignment after outcomes are observed. Persist carries selected state across decisions, while evaluated traces can also feed a separate offline model-learning branch. Together they specify what state is updated, on what timescale, and under which budget.

The four failure points of the section "From world state to effective context" are addressed operation by operation in the subsections below. Evaluate's online compute belongs to B_d when it verifies a current decision, while its offline use belongs to evaluation or learning infrastructure and is reported separately. This accounting keeps improvements in grading or search distinct from B_c interventions.

Content dimensions. Context content stresses the lifecycle unevenly, and four overlapping descriptive dimensions help locate the pressure: form (verbal to experiential), temporal horizon (one invocation to many sessions), modality (text to structured records, sensor, and action), and acquisition channel (observed, acted, or generated). Static text pretraining most directly covers verbal, textual records, whereas experiential and multimodal entries stress Represent, long-horizon entries stress Persist, acted channels stress Acquire and Evaluate, and generated traces stress provenance handling in Represent and Persist.

Acquire: reaching candidate state. Acquisition asks whether the relevant deployment state can be made addressable before a decision; its primary failure mode is availability failure (the section "From world state to effective context"). A decisive repository file, sensor reading, tool result, user state, or environment transition remains unreachable however large the prompt. Increasing acquisition capacity broadens the candidate material from which model-visible context can be built. In agentic settings, the system may have to discover that material through a sequence of information-seeking actions under cost.

Retrieval-augmented generation and tool use bring external material into c.[41-43] Simulators expose situated observations and transitions through interaction;[52] learned world models generate approximate transitions that can be supplied as candidate context.[53,54] When the decision procedure generates or searches learned rollouts, that compute belongs to B_d; selecting and packaging resulting observations or rollouts into c consumes B_c. These channels offer different balances of controllability and fidelity; learned world-model rollouts can drift from the environment they approximate.

The most valuable sources are often privacy-sensitive and fragmented across schemas, including user histories, tool outputs, browser state, environment records, and deployment traces. Every probe also incurs latency, token, and permission costs, so wider coverage reduces the budget available to later stages of the lifecycle.

Represent: preserving decision-relevant fidelity. Acquired material must

Table 1. Context scaling versus adjacent framings.

Framing	What it names	Measured object	Role in context scaling
Context engineering[37,38]	Assembly and upkeep of model-visible context	Per-call prompt / window quality	Implements context construction; context scaling measures the return on the B_c spent on it
Harness engineering[39,40]	Governed runtime for action, state, recovery, observability	Runtime reliability, control, observability	Implements the runtime across the five context operations; its context costs belong to B_c
Retrieval-augmented generation (RAG) / tool use[41-43]	Fetching documents or tool results into c	Retrieval / tool-call accuracy, end-task gain	Implements Acquire; returned evidence uses B_c, while query or action selection uses B_d
Memory systems[44]	Persisting and recalling state across sessions	Recall accuracy over a memory horizon	Implements Persist; storage, validation, and recall are charged to B_c
Externalization[45]	Moving memory, skills, protocols outside the weights	Where capability lives (in-weights vs. external)	Places state outside θ; it contributes to context scaling when that state is made available through B_c
Experience-based learning[36,46]	Shift toward interaction-generated learning signal	Source and yield of the learning signal	Produces traces that may update θ or populate c; report the resulting B_θ and B_c changes separately
Long-context windows[47]	Enlarging the architectural read budget	Window length $\lvert c_{\text{window}}\rvert$ (tokens)	Raises the read limit; context scaling measures whether filling that capacity with usable state improves utility
Agent internalization[32]	Endogenous goals, identity, deliberation vs. external orchestration	Endogeneity of the agent's competence	Changes where competence resides, not the context budget itself; access to deployment state still uses B_c
Context scaling	*Investment in decision-time access to deployment state*	*Utility–cost response to B_c under declared controls*	*Compares returns on these mechanisms while reporting interactions with B_θ and B_d*

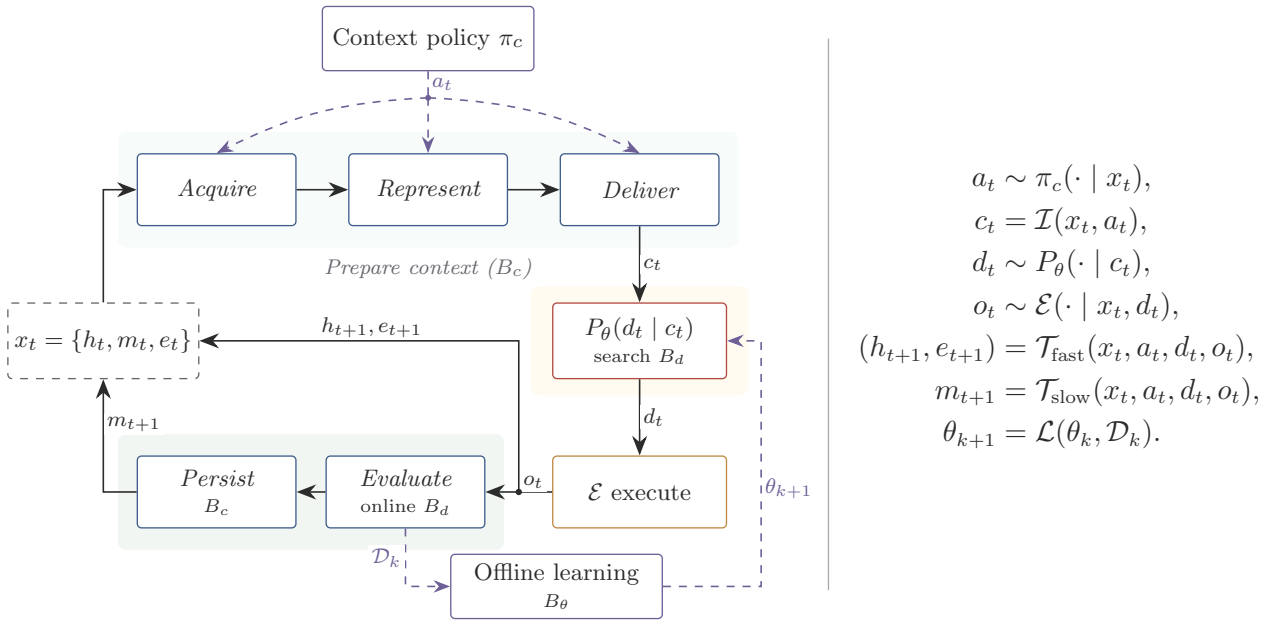


Figure 3. Context infrastructure and model learning. Feedback updates fast history and environment state (h_{t+1}, e_{t+1}), durable memory m_{t+1}, or model parameters θ_{k+1} from evaluated trace batches $\mathcal{D}_k$. Context preparation and persistence consume B_c, model-side search and online verification consume B_d, and offline learning consumes B_θ; environment execution and offline evaluation costs are reported separately

still be encoded in a form that preserves what makes it useful; failures here are fidelity failures (the section "From world state to effective context"), and they occur independently of coverage. A document, video, or sensor stream may be available yet unusable because it is coarse, stale, lossy, ungrounded, or stripped of structure.

Multimodal encoders for speech, vision, video, documents, and embodied sensing determine which signals the model can absorb from raw input.[55-58] Provenance-aware refinement, faithful summarization, and structured trace formats determine what survives the infrastructure path into c. A useful trace records arguments, outputs, timestamps, tool state, source authority, and permissions; a flat transcript may discard evidence needed later.

Representation loses fidelity when image downsampling removes small objects, temporal subsampling removes event order, or document linearization discards layout and relations. Text pipelines have analogous failure modes: summaries erase exceptions, refinement removes provenance, and flat transcripts lose the structure of actions and observations.

Deliver: making state decision-available. Delivery asks whether represented information reaches the model at the relevant decision point. Exclusion under the read budget is a selection failure; poor ordering, burial, or crowding can lead to an observed utilization failure. Since presentation changes the behavior of a fixed model, utilization reflects an interaction between the model and its infrastructure.

Delivery has two infrastructure decisions. The context policy determines what enters c, when, and in what form; a partitioning policy assigns material to context spaces. Agent harnesses rank, order, summarize, and prune material, and allocate token budgets turn by turn.[39,40,59,60] Tool namespaces and sandboxes isolate sources, while sub-agents and private memories partition unrelated trajectories.[39,40,61] Context-Folding treats branch-and-return delivery as a learned action.[62] Long-context architectures and key–value (KV) cache management set the feasible read budget.[63-65]

Access still degrades with position, length, distractors, and compression, and additional material can lower task utility.[21,26,66] Selection value is counter-

Table 2. Context-infrastructure mechanisms by lifecycle operation. Columns abbreviate Acquire (Acq), Represent (Rep), Deliver (Del), Evaluate (Eval), and Persist (Per). Checkmarks indicate direct implementation; an agent harness can implement all five operations.

Mechanism	Acq	Rep	Del	Eval	Per
Retrieval-augmented generation[41]	✓		✓		
Tool-mediated observations[42,43]	✓				
Simulator observations and selected world-model rollouts[53,54]	✓				
Multimodal encoders and document parsers[55-57]		✓			
Provenance-aware summaries and structured traces[59,60]		✓	✓		✓
Context folding and branch–return policies[62]			✓		✓
Episodic / external memory[70,71]			✓		✓
Memory operations as policy actions[44,72,85]					✓
Process / verifier reward models[13,14]				✓	
Agent harness (all five operations)[39,40]	✓	✓	✓	✓	✓

factual, which makes it difficult to estimate before execution.

Evaluate: closing the feedback loop. Evaluation closes the loop after execution produces evidence. Without corrective signals, acquisition accumulates noise, representation drifts, delivery admits distractors, and persistence becomes an unbounded cache. Final outcomes such as a useful answer or a passing test are coarse; signals attached to component operations can identify which acquisition, representation, delivery, or persistence decision helped.[13,14]

Verifier, process-reward, and critique models assess intermediate steps. Infrastructure-side signals include test results, type checks, diff correctness, retrieval-faithfulness checks, user corrections, and simulator reward. These signals are most useful when they can be attributed to a specific infrastructure decision: harness-optimization systems record the expected effect of each automatic edit and compare it with later outcomes.[67] Stronger verifiers permit more autonomous evaluation, while infrastructure feedback supplies data for improving the verifiers themselves.

Fine-grained credit remains expensive because it depends on labeled trajectories, simulator rewards, or detailed verifier outputs. Process rewards also invite Goodhart effects and may suppress the diversity of decisions they were intended to guide.[68,69]

Persist: keeping state across time. Persistence determines which state survives beyond the current decision and how that state remains valid, recoverable, revisable, and forgettable. Missing durable state is an availability failure; stale or incorrectly summarized memory is a fidelity failure. Without persistence, the context policy must rediscover relevant state on every invocation. With it, externalized traces can shape the context and infrastructure policy used in later decisions.

External memory systems and trajectory summaries keep selected state outside the current window.[70,71] Recent work treats memory operations themselves as policy actions.[44,72] Staleness checks, versioning, and invalidation determine when an entry should be refreshed, superseded, or retired.

Operational data stored in persistent context may be private, biased, or product-specific. Persistent context needs authorization, provenance, audit, rollback, and forgetting hooks when state is written.[73,74]

Coding harness as a worked example

Production coding harnesses, including Claude Code, ChatGPT Codex, and Cursor, run all five operations in a single runtime over realistic long-horizon tasks. They are also among the few systems in which the model and surrounding infrastructure have been iterated together by the same teams; this runtime form is now formalized in engineering studies.[75,76]

One runtime, five context operations.

- *Acquire* (the Acquire subsection). The harness exposes file-system reads, shell commands, repository search, version-control state, and build and test outputs, so deployment state is directly addressable.
- *Represent* (the Represent subsection). Harnesses avoid flat text dumps: they chunk files by abstract syntax tree (AST) or symbol structure and preserve tool returns with arguments, exit codes, and provenance.[39,40]
- *Deliver* (the Deliver subsection). A context manager decides, turn by turn, which files, tool returns, and prior reasoning enter the next prompt under a fixed token budget; Cursor's self-summarization policy, for instance, compresses earlier turns under a task-completion reward,[59] and sub-agent partitions keep code-review or debugging sub-tasks from contaminating the main thread.
- *Evaluate* (the Evaluate subsection). The test suite, type checker, and runtime errors provide a verifier signal that is cheap, fast, and tied to the operation that produced the artifact; user accept/reject of a diff supplies a rarer, higher-quality complement, and harness-optimization work back-attributes the combined reward to component decisions.[67]
- *Persist* (the Persist subsection). Long sessions outlast the model's context window; OpenClaw, a personal-assistant harness, uses a pre-compaction memory flush[77] to write durable notes before compaction, and project-level files such as CLAUDE.md carry conventions across sessions while θ remains fixed.

Model–infrastructure co-evolution as a visible loop. Coding is where this co-evolution is currently easiest to watch. Engineering studies and practitioner reports attribute progress on SWE-bench Verified over 2024–2026 to two axes simultaneously: stronger base models, and harness changes that alter what reaches the model in the first place.[39,67,75,78] The qualitative shape is what the Evaluate subsection predicts: consented, replayable deployment traces become training signal for both the next model and the next harness.

The example is a system-level illustration of a favorable digital, replayable domain; it does not provide controlled evidence for a general response curve. Coding is also a best case along a second dimension, verification. Many economically significant deployments (creative work, mental-health support, negotiation, strategic decision-making) lack a cheap automated verifier; Evaluate becomes the bottleneck, and context interventions may be useful yet difficult to certify. Evaluation then relies on human and process signals such as interactive social-simulation outcomes,[79] self-play and negotiation success,[80] and turn-taking and interaction quality,[81] with evaluator uncertainty reported explicitly.

Beyond coding, recent AI-for-science systems run the same loop with slower evaluators and higher demands on provenance and persistence across research cycles.[82-84]

The Loop as a Classifier of Mechanisms

Table 2 maps the representative mechanisms surveyed in this section to the operations they directly implement.

CHALLENGES AND OPEN PROBLEMS

The next engineering tasks fall into three clusters: *grounded evaluation and reporting*, *interaction data and full-cost infrastructure*, and *safety and governance for persistent and adaptive systems*.

Measurement: evaluation infrastructure and proxy metrics

Evaluation infrastructure. Static prompt–response benchmarks usually

Context-scaling reporting checklist (per result)

Field	What to report
Model-visible context	tokens or representation units actually supplied, separate from the architectural window limit
B_c components	storage, retrieval, tool-interface, representation, selection, persistence, latency, and maintenance costs
B_d components	generation, search, tool-selection actions, rollouts, and online verifier compute per decision
Interaction depth	average actions/retrievals per committed decision
Memory horizon	longest span over which conditioning stays coherent
Decision-relevance density	optional fraction of declared units whose specified replacement lowers U beyond a tolerance; report units and baseline
Provenance quality	source, licensing, and trust grade of admitted content
Recency	lag between admitted state and the current world
Utility–cost curve	task utility with uncertainty across several B_c levels, including interactions with B_d

hold context construction outside the evaluated system. Interactive benchmarks such as SWE-bench, WebArena, and AgentBench instead require action, observation, and adaptation, though their coverage and reproducibility remain limited.[78,86,87] Longitudinal benchmarks test memory and user state accumulated over longer horizons;[88,89] embodied evaluations additionally require faithful sensor and action streams.

Reporting. Results should disclose the model, the infrastructure that produced c, separate B_c and B_d components, interaction and memory scale, context-quality proxies, and a utility–cost curve.[26,39,66] These fields should remain separate rather than being collapsed into a composite score. The checklist below defines the per-result disclosure.

Interaction data and compute at scale

Data supply. Interaction traces grounded in environments, tools, and multimodal streams are usually produced for a specific system and tied to an environment version, unlike web text, whose original production cost was largely external to model developers. Three sources dominate. Simulation environments offer controllable volume but incur environment-authoring costs and sim-to-real gaps.[90,91] Consented product telemetry offers deployment fidelity but raises privacy and legal constraints.[92] Synthetic and self-generated traces scale more readily but can narrow the data distribution as models learn from their own outputs.[93]

Full-cost infrastructure. Closed-loop training couples policy rollouts, environment steps, evaluation, and parameter updates, and long-horizon episodes compound the cost.[94] Accounting must therefore include environment simulation, evaluator queries, and trace storage alongside GPU throughput. A reusable stack needs versioned environments, structured replayable traces, privacy-preserving filters, and evaluators that can replay trajectories.[86,87,92] Because the data is stateful and environment-dependent, the practical shared substrate is a governed collection of simulators, consented traces, and replay tools rather than a public dump.

Safety and governance

Safety risks. Persistent context and adaptive policies can change deployment behavior after a model checkpoint has passed safety evaluation. Indirect prompt injection or unauthorized tool output can enter c;[95] poisoned summaries and memories can then propagate errors across sessions; weak tenant partitioning can leak private state; and repeated state or policy updates can shift refusal behavior, deference, or norm compliance.[74,96] These failures can accumulate while θ remains fixed, and parameter updates add a separate source of alignment drift.

Governance requirements. Governance must cover both external-state changes and model updates. A deployment should record consent and source permissions, preserve provenance for admitted and written state, version memories and model checkpoints, evaluate behavior after updates, and support rollback and forgetting.[73,97,98] Incident records should attribute harmful decisions to the retrieved passage, tool output, memory entry, policy change, or model update that preceded them. External-state rollback and model-checkpoint rollback should remain separately auditable, and material changes in behavior or data use should be disclosed to affected users and stakeholders.

A short research agenda

Each direction below is empirically separable: progress on one does not wait on the others.

- *Build grounded evaluation infrastructure.* The field needs reproducible, cost-controlled benchmarks for interactive and longitudinal behavior beyond static prompt–response grading.
- *Derive principled proxies for context value.* Protocol-bound metrics such as decision-relevance density should be tested for whether they predict held-out utility better than raw context length and cost within declared task families; cross-domain comparison requires separate calibration.
- *Develop shared interaction-data infrastructure.* Useful shared infrastructure would include versioned simulators, consented trace collection with privacy controls, synthetic-data diversity audits, replayable storage, and full-cost accounting.
- *Establish safety and governance conventions for persistent and adaptive deployments.* Systems need tests for context-induced behavioral drift, consent and source permissions, separate audit and rollback for external state and model updates, incident attribution, and disclosure of material changes.
- *Pursue model and context-infrastructure co-evolution.* A convincing study would estimate $U(B_\theta, B_d, B_c)$ and show how infrastructure reduces availability, fidelity, and selection failures while model improvements reduce utilization failures under matched context inputs.

CONCLUSION

We opened with the context bottleneck: task-relevant deployment state exists but is unavailable, distorted, poorly selected, or unused at decision time. Embedding $P_\theta(d \mid c)$ in the budgeted system distribution $Q_{\theta,\pi}(d \mid x; B_\theta, B_d, B_c)$ makes the interactions and accounting of the three budgets explicit.

Context scaling is a distinct engineering and measurement program (the research agenda), evaluated through controlled budget interventions and utility–cost curves. Its engineering target is the model–infrastructure pair. The immediate work is to make B_c visible in published results and to estimate $U(B_\theta, B_d, B_c)$ on declared task families; the reporting conventions, data infrastructure, and governance hooks of the open-problems section are the instruments for doing so.

DECLARATION OF GENERATIVE AI AND AI-ASSISTED TECHNOLOGIES IN THE WRITING PROCESS

During the preparation of this work, the author used Claude (Anthropic) and Codex (OpenAI) in order to improve language, structure, and readability and to perform consistency checks. After using these tools, the author reviewed and edited the content as needed and takes full responsibility for the content of the publication.

REFERENCES

1. Shapley L. S.. (1953). A Value for n-Person Games. *Contributions to the Theory of Games* **2**:307–17. DOI:10.7249/p0295
2. Lundberg S. M. and Lee S. I.. (2017). A unified approach to interpreting model predictions. *Advances in neural information processing systems* 4765–74.
3. OpenAI (2024). Learning to reason with LLMs. (*OpenAI public report; https://openai.com/index/learning-to-reason-with-llms/*).
4. DeepSeek-AI, Guo D., Yang D., et al. (2025). DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature* **645**:633–8. DOI:10.1038/s41586-025-09422-z
5. Jin B., Zeng H., Yue Z., et al. (2025). Search-R1: Training LLMs to reason and leverage search engines with reinforcement learning. *Conference on language modeling.*
6. Kaplan J., McCandlish S., Henighan T., et al. (2020). Scaling laws for neural language models. *arXiv preprint arXiv: 2001.08361.*
7. Hoffmann J., Borgeaud S., Mensch A., et al. (2022). An empirical analysis of Compute-Optimal large language model training. *Advances in neural information processing systems.*
8. Muennighoff N., Rush A. M., Barak B., et al. (2023). Scaling Data-Constrained language models. *Advances in neural information processing systems.*
9. Sorscher B., Geirhos R., Shekhar S., et al. (2022). Beyond neural scaling laws: Beating power law scaling via data pruning. *Advances in neural information processing systems.*
10. Wei J., Wang X., Schuurmans D., et al. (2022). Chain-of-Thought prompting elicits reasoning in large language models. *Advances in neural information processing systems.*
11. Wang X., Wei J., Schuurmans D., et al. (2023). Self-Consistency improves chain of thought reasoning in language models. *International conference on learning representations.*
12. Yao S., Yu D., Zhao J., et al. (2023). Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems.*
13. Lightman H., Kosaraju V., Burda Y., et al. (2024). Let's verify step by step. *International conference on learning representations.*
14. Cobbe K., Kosaraju V., Bavarian M., et al. (2021). Training verifiers to solve math word problems. *arXiv preprint arXiv: 2110.14168.*
15. Snell C., Lee J., Xu K., et al. (2025). Scaling LLM Test-Time Compute Optimally Can be More Effective than Scaling Parameters for Reasoning. *International conference on learning representations.*
16. Wei Z., Yao W., Liu Y., et al. (2025). WebAgent-R1: Training web agents via End-to-End Multi-Turn reinforcement learning. *Conference on empirical methods in natural language processing* 7909–28. DOI:10.18653/v1/2025.emnlp-main.401
17. Zou D. and others (2026). On Information Self-Locking in Reinforcement Learning for Active Reasoning of LLM agents. *arXiv preprint arXiv: 2603.12109.*
18. Montgomery K., Park D., Tu J., et al. (2025). Predicting Task Performance with Context-aware Scaling Laws. *arXiv preprint arXiv: 2510.14919.*
19. Shi J., Ma Q., Liu H., et al. (2025). Intrinsic entropy of context length scaling in LLMs. *arXiv preprint arXiv: 2502.01481.*
20. Fang Y., Zhan J., Ai Q., et al. (2024). Scaling laws for dense retrieval. *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval.*
21. Fang L., Wang Y., Liu Z., et al. (2025). What is Wrong with Perplexity for Long-context Language Modeling? *International conference on learning representations.*
22. Kimi Team, Du A., Gao B., et al. (2025). Kimi k1.5: Scaling reinforcement learning with LLMs. *arXiv preprint arXiv: 2501.12599.*
23. DeepSeek-AI (2026). DeepSeek-V4: Towards highly efficient Million-Token context intelligence. (*Hugging Face model card / technical report; https://huggingface.co/deepseek-ai/DeepSeek-V4-Pro*).
24. GLM-4.5 Team, Zeng A., Lv X., et al. (2025). GLM-4.5: Agentic, reasoning, and coding (ARC) foundation models. *arXiv preprint arXiv: 2508.06471.*
25. Z.ai (2026). GLM-4.7 & GLM-4.6 & GLM-4.5. (*GitHub repository; https://github.com/zai-org/GLM-4.5*).
26. Hsieh C. P., Sun S., Kriman S., et al. (2024). RULER: What's the real context size of your Long-Context language models? *arXiv preprint arXiv: 2404.06654.*
27. Bai Y., Tu S., Zhang J., et al. (2025). LongBench v2: Towards Deeper Understanding and Reasoning on Realistic Long-context Multitasks. *Annual meeting of the association for computational linguistics* 3639–64. DOI:10.18653/v1/2025.acl-long.183
28. Shao R., He J., Asai A., et al. (2024). Scaling Retrieval-Based language models with a Trillion-Token datastore. *Advances in neural information processing systems.*
29. Yue Z., Zhuang H., Bai A., et al. (2025). Inference scaling for Long-Context retrieval augmented generation. *International conference on learning representations.*
30. Hernandez D., Kaplan J., Henighan T., et al. (2021). Scaling laws for transfer. *arXiv preprint arXiv: 2102.01293.*
31. Sutton R. S. (2019). The bitter lesson. (*Essay; http://www.incompleteideas.net/IncIdeas/BitterLesson.html*).
32. Xing E., Deng M. and Hou J. (2026). Critique of agent model. *arXiv preprint arXiv: 2606.23991.*
33. OpenAI (2025). Introducing deep research. (*OpenAI product announcement; https://openai.com/index/introducing-deep-research/*).
34. Mialon G., Fourrier C., Swift C., et al. (2024). GAIA: A benchmark for general AI assistants. *International conference on learning representations.*
35. Wei J., Sun Z., Papay S., et al. (2025). BrowseComp: A simple yet challenging benchmark for browsing agents. (*OpenAI technical report; https://openai.com/index/browsecomp/*).
36. Silver D. and Sutton R. S. (2025). Welcome to the era of experience. (*Preprint of a chapter in* Designing an Intelligence*; https://storage.googleapis.com/deepmind-media/Era-of-Experience%20/The%20Era%20of%20Experience%20Paper.pdf*).
37. Karpathy A. (2025). On context engineering over prompt engineering. (*X (Twitter) post; https://x.com/karpathy/status/1937902205765607626*).
38. LangChain (2025). Context engineering for agents. (*LangChain blog post; https://blog.langchain.com/context-engineering-for-agents/*).
39. He C., Zhou X., Wang D., et al. (2026). Harness engineering for language agents: The harness layer as control, agency, and runtime. *Preprints.* DOI: 10.20944/preprints202603.1756.v2
40. Meng Q., Wang Y., Chen L., et al. (2026). Agent harness for large language model agents: A survey. *Preprints.* DOI: 10.20944/preprints202604.0428.v3
41. Lewis P., Perez E., Piktus A., et al. (2020). Retrieval-Augmented generation for Knowledge-Intensive NLP tasks. *Advances in neural information processing systems.*
42. Schick T., Dwivedi-Yu J., Dessì R., et al. (2023). Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems.*
43. Yao S., Zhao J., Yu D., et al. (2023). ReAct: Synergizing reasoning and acting in language models. *International conference on learning representations.*
44. Yan S., Yang X., Huang Z., et al. (2026). Memory-R1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. *Annual meeting of the association for computational linguistics* 12805–25.
45. Zhou C., Chai H., Chen W., et al. (2026). Externalization in LLM agents: A unified review of memory, skills, protocols and harness engineering. *arXiv preprint arXiv: 2604.08224.*
46. Sutton R. S. and Barto A. G. (2018). Reinforcement learning: An introduction. (*MIT Press*).
47. Liu X., Li R., Huang M., et al. (2025). Thus spake Long-Context large language model. *arXiv preprint arXiv: 2502.17129.*
48. Kaelbling L. P., Littman M. L. and Cassandra A. R.. (1998). Planning and acting in partially observable stochastic domains. *Artif Intell* **101**:99–134. DOI:10.1016/S0004-3702(98)00023-X
49. Bajcsy R.. (1988). Active perception. *Proc. IEEE* **76**:996–1005. DOI:10.1109/5.5968
50. Marchionini G.. (2006). Exploratory search: From finding to understanding. *Commun ACM* **49**:41–6. DOI:10.1145/1121949.1121979
51. Hollan J., Hutchins E. and Kirsh D.. (2000). Distributed cognition: Toward a new foundation for Human-Computer interaction research. *ACM Trans. Comput.-Hum. Interact.* **7**:174–96. DOI:10.1145/353485.353487
52. Todorov E., Erez T. and Tassa Y. (2012). MuJoCo: A physics engine for model-based control. *2012 IEEE/RSJ international conference on intelligent robots and systems* (*IEEE*): 5026–33.
53. Ha D. and Schmidhuber J. (2018). World models. *arXiv preprint arXiv: 1803.10122.*
54. Hafner D., Pasukonis J., Ba J., et al. (2025). Mastering diverse control tasks through world models. *Nature* **640**:647–53. DOI:10.1038/s41586-025-08744-2
55. Défossez A., Mazaré L., Orsini M., et al. (2024). Moshi: a speech-text foundation model for real-time dialogue. *arXiv preprint arXiv: 2410.00037.*
56. Liu H., Li C., Wu Q., et al. (2023). Visual instruction tuning. *Advances in neural information processing systems.*
57. Xu Y., Li M., Cui L., et al. (2020). LayoutLM: Pre-training of Text and Layout for Document Image Understanding. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.*
58. Kim M. J., Pertsch K., Karamcheti S., et al. (2024). OpenVLA: An Open-Source Vision-Language-Action model. *Conference on robot learning* 2679–713.
59. Cursor (2026). Training composer for longer horizons. (*https://cursor.com/blog/self-summarization*).
60. Chroma (2026). Chroma context-1: Training a Self-Editing search agent. (*https://www.trychroma.com/research/context-1*).
61. Wu Q., Bansal G., Zhang J., et al. (2024). AutoGen: Enabling Next-Gen LLM applications via Multi-Agent conversation. *Proceedings of the first conference on language modeling.*
62. Sun W., Lu M., Ling Z., et al. (2025). Scaling Long-Horizon LLM agent via Context-Folding. *arXiv preprint arXiv: 2510.11967.*
63. Liu X., Yan H., An C., et al. (2024). Scaling Laws of RoPE-based Extrapolation. *International conference on learning representations.*
64. Shi L., Zhang H., Yao Y., et al. (2024). Keep the cost down: A review on methods to optimize LLM's KV-Cache consumption. *arXiv preprint arXiv: 2407.18003.*
65. Xiao G., Tian Y., Chen B., et al. (2024). Efficient streaming language models with attention sinks. *International conference on learning representations.*
66. Liu N. F., Lin K., Hewitt J., et al. (2024). Lost in the middle: How language models use long contexts. *Trans Assoc Comput Linguist.*
67. Lin J., Liu S., Pan C., et al. (2026). Agentic harness engineering: Observability-Driven automatic evolution of Coding-Agent harnesses. *arXiv preprint arXiv: 2604.25850.*
68. Amodei D., Olah C., Steinhardt J., et al. (2016). Concrete problems in AI safety. *arXiv*

preprint arXiv: 1606.06565.
69. Pan A., Bhatia K. and Steinhardt J. (2022). The effects of reward misspecification: Mapping and mitigating misaligned models. *International conference on learning representations*.
70. Packer C., Wooders S., Lin K., et al. (2023). MemGPT: Towards LLMs as operating systems. *arXiv preprint arXiv: 2310.08560*.
71. Zhang Z., Bo X., Ma C., et al. (2025). A Survey on the Memory Mechanism of Large Language Model based Agents. *ACM Trans Inf Syst* **43**:155:1–155:47. DOI:10.1145/3748302
72. Yu Y., Yao L., Xie Y., et al. (2026). Agentic memory: Learning unified Long-Term and Short-Term memory management for large language model agents. *Annual meeting of the association for computational linguistics* 21457–83.
73. Shavit Y., Agarwal S., Brundage M., et al. (2023). Practices for governing agentic AI systems. (*OpenAI research paper; https://cdn.openai.com/papers/practices-for-governing-agentic-ai-systems.pdf*).
74. Casper S., Ezell C., Siegmann C., et al. (2024). Black-Box Access is Insufficient for Rigorous AI Audits. *Proceedings of the 2024 ACM conference on fairness, accountability, and transparency*.
75. Ning X., Tieu K., Fu D., et al. (2026). Code as agent harness. *arXiv preprint arXiv: 2605.18747*.
76. Pan L., Zou L., Guo S., et al. (2026). Natural-Language agent harnesses. *arXiv preprint arXiv: 2603.25723*.
77. OpenClaw (2026). Session management: Compaction. (*https://openclaw.bot/docs/cli/reference/session-management-compaction/*).
78. Jimenez C. E., Yang J., Wettig A., et al. (2024). SWE-bench: Can Language Models Resolve Real-World GitHub Issues? *International conference on learning representations*.
79. Zhou X., Zhu H., Mathur L., et al. (2024). SOTOPIA: Interactive evaluation for social intelligence in language agents. *International conference on learning representations*.
80. Fu Y., Peng H., Khot T., et al. (2023). Improving language model negotiation with Self-Play and In-Context learning from AI feedback. *arXiv preprint arXiv: 2305.10142*.
81. Arora S., Lu Z., Chiu C. C., et al. (2025). Talking turns: Benchmarking audio foundation models on Turn-Taking dynamics. *International conference on learning representations*.
82. Google (2026). New AI tools for the future of science. (*https://blog.google/innovation-and-ai/technology/research/gemini-for-science-io-2026/*).
83. Gottweis J., Weng W. H., Daryin A., et al. (2026). Accelerating scientific discovery with Co-Scientist. *Nature*. DOI: 10.1038/s41586-026-10644-y
84. Aygün E., Belyaeva A., Comanici G., et al. (2026). An AI system to help scientists write expert-level empirical software. *Nature*. DOI: 10.1038/s41586-026-10658-6
85. Li R., Zhang X., Yu H., et al. (2026). MemPO: Self-Memory policy optimization for Long-Horizon agents. *arXiv preprint arXiv: 2603.00680*.
86. Zhou S., Xu F. F., Zhu H., et al. (2024). WebArena: A realistic web environment for building autonomous agents. *International conference on learning representations*.
87. Liu X., Yu H., Zhang H., et al. (2024). AgentBench: Evaluating LLMs as agents. *International conference on learning representations*.
88. Maharana A., Lee D. H., Tulyakov S., et al. (2024). Evaluating very Long-Term conversational memory of LLM agents. *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*.
89. Zhong W., Guo L., Gao Q., et al. (2024). MemoryBank: Enhancing large language models with Long-Term memory. *AAAI conference on artificial intelligence* 19724–31. DOI:10.1609/aaai.v38i17.29946
90. Zhao W., Queralta J. P. and Westerlund T. (2020). Sim-to-Real transfer in deep reinforcement learning for robotics: a survey. *2020 IEEE symposium series on computational intelligence (SSCI)*.
91. Li X., Hsu K., Gu J., et al. (2024). Evaluating Real-World robot manipulation policies in simulation. *Conference on robot learning* 3705–28.
92. Abadi M., Chu A., Goodfellow I., et al. (2016). Deep learning with differential privacy. *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*.
93. Shumailov I., Shumaylov Z., Zhao Y., et al. (2024). AI models collapse when trained on recursively generated data. *Nature* **631**:755–9. DOI:10.1038/s41586-024-07566-y
94. Sardana N., Portes J., Doubov S., et al. (2024). Beyond Chinchilla-Optimal: Accounting for inference in language model scaling laws. *Proceedings of the 41st International Conference on Machine Learning*.
95. Greshake K., Abdelnabi S., Mishra S., et al. (2023). Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security (AISec)* 79–90.
96. Perez E., Huang S., Song F., et al. (2022). Red teaming language models with language models. *Proceedings of the 2022 conference on empirical methods in natural language processing*.
97. Ilyas A., Park S. M., Engstrom L., et al. (2022). Datamodels: Understanding predictions with data and data with predictions. *International conference on machine learning* 9525–87.
98. Coalition for Content Provenance and Authenticity (C2PA) (2024). C2PA technical specifications. (*Open technical standard; https://spec.c2pa.org/specifications/specifications/2.2/index.html*).

FUNDING AND ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (No. 62525602). The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

AUTHOR CONTRIBUTIONS

X.Q. is the sole author and is responsible for the entire work: the position, the framework, and the manuscript.

DECLARATION OF INTERESTS

The author declares no competing interests.

ETHICAL STATEMENT AND PATIENT CONSENT

Not applicable.

DATA AND CODE AVAILABILITY

No new data were created or analyzed in this study.

LEAD CONTACT WEBSITE

Xipeng Qiu, https://xpqiu.github.io/