Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Test-Time Improvement for Large Models: A Survey of Inference-Stage Mechanisms, Resource Control, and Research Opportunities

Yiming Tang
Beijing University of Posts and Telecommunications
Ziyu Liu
Beijing University of Posts and Telecommunications
Zihan Chen
Beijing University of Posts and Telecommunications
Jinghan Yang
Beijing University of Posts and Telecommunications
Liuyu Xiang
Beijing University of Posts and Telecommunications
Zhenbo Xu
xuzhenbo@bupt.edu.cn
Beijing University of Posts and Telecommunications
Zhaofeng He
Beijing University of Posts and Telecommunications

Research Article

Keywords: Test-Time Improvement, Large Models, Inference-Stage Intervention, Test-Time Adaptation, Test-Time Scaling, Resource Control

Posted Date: June 11th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-9875246/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Test-Time Improvement for Large Models: A Survey of Inference-Stage Mechanisms, Resource Control, and Research Opportunities

Yiming Tang[1†], Ziyu Liu[2†], Zihan Chen[3†], Jinghan Yang[4], Liuyu Xiang[4], Zhenbo Xu[4*], Zhaofeng He[4]

[1]School of Mathematical Sciences, Beijing University of Posts and Telecommunications, Beijing, 100876, China.
[2]International School, Beijing University of Posts and Telecommunications, Beijing, 100876, China.
[3]School of Future Talents, Beijing University of Posts and Telecommunications, Beijing, 100876, China.
[4*]School of Artificial Intelligence, Beijing University of Posts and Telecommunications, Beijing, 100876, China.

*Corresponding author(s). E-mail(s): xuzhenbo@bupt.edu.cn;
Contributing authors: tangyiming_0412@163.com;
liuziyu2023@bupt.edu.cn; 2024212719@bupt.cn;
jinghanyang@bupt.edu.cn; xiangly@bupt.edu.cn;
zhaofenghe@bupt.edu.cn;
†These authors contributed equally to this work.

Abstract

Large models are increasingly improved not only through training, but also through interventions applied during testing and inference. Prompting, retrieval augmentation, test-time adaptation, reasoning-time search, verification, memory, tool use, and agentic workflows have therefore become closely related, yet they are often reviewed as separate research lines. This survey introduces Test-Time Improvement (TTI) as a bounded inference-stage research space for studying how trained models improve their current task performance by using information, computation, feedback, state, or external actions available at inference time. To organize this heterogeneous field, we develop a Locus × Source research map. The Locus axis distinguishes whether an intervention acts on the input, inference-relevant state variables, or the inference process, while the Source axis

identifies whether the useful signal comes from internal model knowledge, static knowledge resources, live tools or environments, test streams, or self-generated and verifier-based feedback. Based on this map, we review input-level, state-level, and process-level interventions and trace how local interventions migrate into coupled TTI systems, such as context-to-memory, trace-to-search, tool-to-experience, and feedback-to-verifier trajectories. We further analyze how TTI methods convert capacity, context, and feedback bottlenecks into resource-control problems over inference-time computation, reusable state, and reliable evaluation. Finally, we analyze landscape-level research patterns and identify future directions for controllable, evaluable, and stable test-time improvement.

Keywords: Test-Time Improvement, Large Models, Inference-Stage Intervention, Test-Time Adaptation, Test-Time Scaling, Resource Control

1 Introduction

Large models are increasingly used not only as static predictors, but also as inference-time systems. In many applications, their behavior is shaped not only by parameters learned during training, but also by the information, computation, state, and feedback available during inference. Tasks may be reformulated through prompts or demonstrations, external evidence may be retrieved, test samples may trigger adaptation, and generation may be organized through reasoning, search, verification, tool use, or memory. These developments suggest that the test or inference stage has become an important site of model improvement. Although these directions differ in technical form, they share a common concern with how a trained model can improve inference-time task performance after training by intervening in the process through which it solves a current problem (Brown et al. 2020; Lewis et al. 2020; Guu et al. 2020; Sun et al. 2020; Schick and Schütze 2021; Wang et al. 2021; Wei et al. 2022; Wang et al. 2023b).

Figure 1 illustrates how representative inference-stage directions are coupled around a trained large model through input-side intervention, state-side adaptation or memory, and process-side control.

This shift creates an organizational challenge for reviewing the field. Prompting, retrieval augmentation, test-time adaptation, test-time scaling, verification, memory, tool use, and agentic systems are often discussed through separate lines of work. Recent systems, however, increasingly combine these components within the same inference procedure. A review organized only around method names cannot fully explain how these directions interact. This survey therefore uses Test-Time Improvement (TTI) as a bounded perspective for analyzing where inference-stage interventions act, what information they use, whether their effects persist, and what additional cost they introduce.

The survey starts from the coupling of inference-stage methods, introduces a bounded TTI perspective, develops a Locus $\times$ Source research map, analyzes mechanism development and cross-locus migration, examines bottleneck conversion

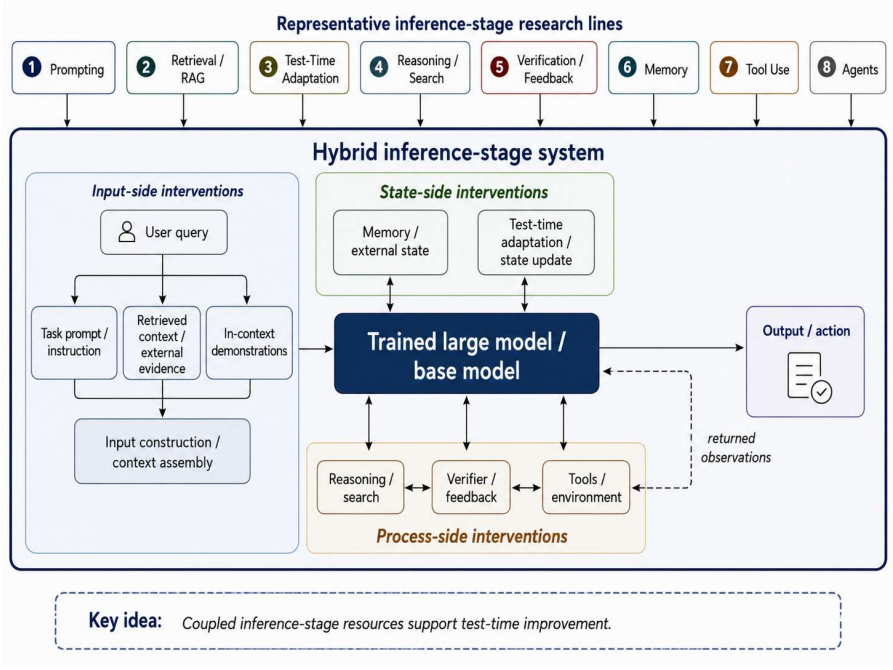


Fig. 1 A TTI system view of inference-stage resource coupling. Representative inference-stage directions are coupled around a trained large model through input-side intervention, state-side adaptation or memory, and process-side control, motivating the unified TTI perspective adopted in this survey.

and resource control, synthesizes landscape-level research patterns, and discusses limitations and future directions.

This survey uses TTI as a bounded inference-stage perspective on post-training methods or systems that directly affect a model's current solving behavior during testing or inference. The formal definition, boundary conditions, and inclusion criteria are developed in Section 2.

Existing surveys have examined important subregions such as test-time scaling, test-time adaptation, retrieval-augmented generation, tool learning, memory-augmented systems, and LLM agents (Zhang et al. 2025c; Alomrani et al. 2025; Venktesh et al. 2025; Liang et al. 2025; Wang et al. 2025b; Xiao and Snoek 2024; Gao et al. 2024; Fan et al. 2024; Ni et al. 2025; Zhou et al. 2024; Peng et al. 2026a; Qu et al. 2025; Chen et al. 2025b; Mialon et al. 2023; Wu et al. 2025a; Zhang et al. 2025d; Wu et al. 2025b; Wang et al. 2024b; Huang et al. 2024c; Li 2025). These surveys provide essential foundations, but they usually organize the field around specific method families or resource types. This motivates a cross-direction view that asks how these lines of work jointly form an inference-stage improvement space. Section 2.2 provides the detailed positioning against adjacent survey areas.

The core framework of this survey is a Locus-first research map. The Locus axis specifies where a test-time intervention acts on three parts of the inference pipeline, including the input presented to the model, state variables that are accessed, updated, or preserved during inference, and the process through which outputs and actions

are produced. The Source axis complements this axis by specifying where the useful test-time information comes from. Temporal persistence and inference-time cost are retained as auxiliary descriptors rather than additional primary axes. Together, these descriptors support method positioning and help analyze how local inference-stage interventions become coupled system resources.

Figure 2 summarizes the organization of the survey.

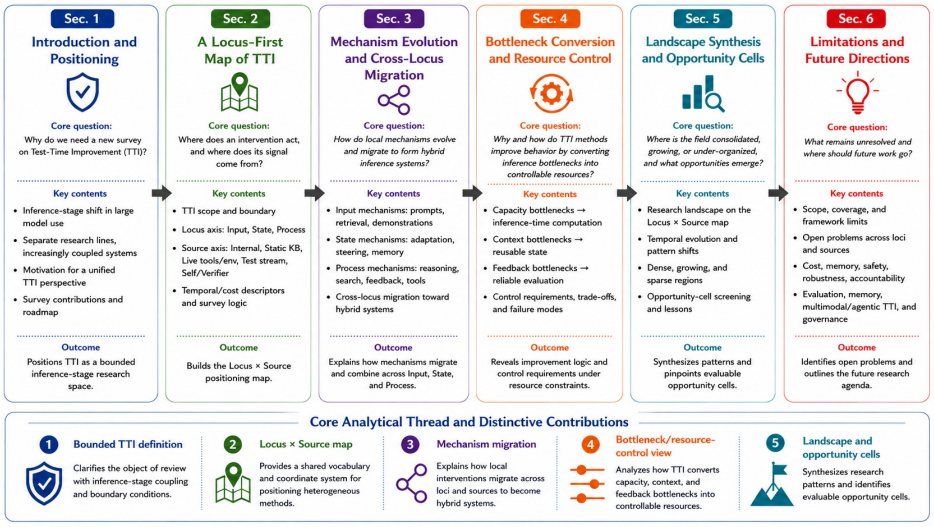


Fig. 2 Roadmap of the survey on Test-Time Improvement (TTI).

This survey makes four main contributions.

- **A bounded TTI object of review.** We formulate TTI as a bounded inference-stage research space. This allows prompting, retrieval, adaptation, reasoning, verification, memory, tools, and agents to be compared under a shared principle of inference-stage coupling, rather than being treated as unrelated techniques or as generic post-training improvements. This object of review complements specialized surveys by focusing on how these directions jointly form a broader inference-stage improvement space.
- **A Locus × Source positioning map.** We introduce a Locus-first map that positions heterogeneous methods by intervention locus and information source. This map helps distinguish superficially similar methods and connect technically different methods that play comparable roles during inference. It provides a common coordinate system for relating compute-, retrieval-, adaptation-, memory-, tool-, and agent-centered research lines.
- **A cross-locus analysis of mechanism migration.** We trace how local test-time interventions become components of coupled TTI systems, including context-to-memory, trace-to-search, tool-to-experience, and feedback-to-verifier or reward

trajectories. This analysis highlights how mechanisms initially studied within one area can migrate into another and become reusable system resources.

- **A resource-control and evaluability perspective on research opportunities.** We relate TTI gains to bottleneck conversion and resource-control requirements, and identify opportunity cells according to mechanism linkage, resource-control relevance, and evaluability rather than map gaps alone. This perspective emphasizes not only where new mechanisms may appear, but also whether they can be controlled, evaluated, and stabilized under realistic inference-time constraints.

The remainder of this survey is organized as follows. Section 2 establishes the scope of TTI and the Locus × Source map. Section 3 traces mechanism development and cross-locus migration across input, state, and process loci. Section 4 examines bottleneck conversion and the resulting resource-control requirements. Section 5 analyzes landscape-level research patterns and opportunity cells. Section 6 concludes with limitations and future directions.

2 Scope and Framework

2.1 Scope and Boundaries

Test-time improvement (TTI) refers to a family of methods that improve model behavior after the original training stage by changing the information presented to the model, adapting or steering the state variables that are accessed, updated, or preserved during inference, reorganizing the inference procedure, or introducing external evidence and feedback during testing. The defining property of TTI is not that the base model must remain completely unchanged, nor that the intervention must be limited to prompt manipulation. Rather, the improvement process is coupled to the test or inference stage because the method uses information, computation, feedback, or state resources available during inference to improve output quality, adaptation, reliability, or long-term usability.

The boundary of TTI is equally important. Conventional pretraining, standalone supervised fine-tuning, offline domain retraining, and independent post-training pipelines are not treated as TTI solely on the basis of post-training performance improvement. They fall within the scope only when their signals, objectives, or state updates are directly coupled to the testing process. For example, entropy-minimization adaptation on incoming test batches and prompt-parameter tuning at inference time are within scope, whereas a separately trained domain model is not. Similarly, a retrieval system is considered TTI when retrieval is invoked to supply inference-time evidence for a current query; a corpus construction paper or a benchmark paper is treated as context rather than as a core TTI mechanism.

Such a scope is necessary because inference-stage improvement methods have increasingly extended beyond their original subfield boundaries. Prompt engineering, retrieval augmentation, test-time adaptation, reasoning-time search, verification, memory, and tool-augmented agents were often developed in separate research communities, but recent systems combine them into hybrid inference architectures. Retrieval

pipelines may retrieve, critique, and iterate; agents may convert tool traces into reusable experience; state-level adaptation may use verifier or reward signals; and reasoning methods may scale through search, candidate selection, or execution feedback. A method-name taxonomy would obscure these mechanism-level connections. The locus-first map introduced next therefore organizes TTI by where an intervention acts in the inference pipeline, while source, temporal persistence, and cost describe how the intervention obtains information, persists over time, and consumes inference-stage resources.

2.2 Related Survey Areas

Several adjacent survey areas have examined important subregions of inference-stage improvement, including test-time scaling, test-time adaptation, retrieval-augmented generation, tool learning, memory-augmented systems, and LLM agents. These surveys provide essential foundations for understanding specific method families, resource types, and system settings. However, because they are typically organized around particular technical directions or research communities, their primary focus is not to describe how these directions jointly operate within a broader inference-stage improvement space.

The present survey adopts a different level of analysis. Instead of treating these areas as separate lines of work, it uses Test-Time Improvement (TTI) as a bounded inference-stage perspective to position them within a common Locus × Source framework. Table 1 summarizes this relationship at a high level and clarifies the scope of the present survey relative to adjacent reviews.

Table 1: Positioning of adjacent survey areas relative to the present survey.

Adjacent survey area	**Representative surveys and related support**	**Main organizing focus**	**Positioning in this survey**
Test-time scaling and test-time compute	(Zhang et al. 2025c; Alomrani et al. 2025; Venktesh et al. 2025); (Snell et al. 2025) as analysis-oriented support	Additional inference-time computation for reasoning, search, verification, and budget allocation.	Positioned mainly as a process-side route of TTI and connected with retrieval, adaptation, memory, tools, and feedback.
Online test-time adaptation	(Liang et al. 2025; Wang et al. 2025b; Xiao and Snoek 2024)	Adaptation to test samples, batches, or streams under distribution shift.	Positioned as a state-side route of TTI driven mainly by test-stream information.

Adjacent survey area	Representative surveys and related support	Main organizing focus	Positioning in this survey
RAG, trustworthy RAG, and GraphRAG	(Gao et al. 2024; Fan et al. 2024; Ni et al. 2025; Zhou et al. 2024; Peng et al. 2026a)	Retrieval, grounding, evidence organization, and knowledge-augmented generation.	Positioned mainly as an input-side route that may also migrate into memory, verification, and search.
Tool learning	(Qu et al. 2025; Chen et al. 2025b; Mialon et al. 2023) as broader augmented-LM support	Tool selection, tool calling, external actions, and tool-use evaluation.	Positioned as a process-side route that links live external information with action and experience reuse.
Memory mechanisms and long-term interaction	(Zhang et al. 2025d; Wu et al. 2025a); (Wu et al. 2025b) as benchmark/evaluation-oriented support	Storage, retrieval, updating, and reuse of information across interactions or tasks.	Positioned mainly as a state-side route that connects retrieval, demonstrations, feedback, and agent experience.
LLM agents and agentic workflows	(Wang et al. 2024b; Huang et al. 2024c; Li 2025) as review-oriented support	Planning, role organization, tool use, memory, feedback, and environment interaction.	Positioned as a hybrid process-oriented setting in which multiple inference-stage resources are combined.
This survey	–	Bounded inference-stage improvement through information, computation, state, feedback, and action.	Provides a Locus × Source map with Temporal and Cost descriptors to compare these directions at a shared level.

Note. The table is intended for positioning rather than evaluation. It summarizes the main organizing focus of representative adjacent survey areas and clarifies how this survey uses TTI as a cross-direction inference-stage perspective. Related support works are included when they help position the scope of an adjacent area, even if they are not themselves survey papers.

This positioning clarifies how the present survey complements specialized reviews. Existing surveys usually focus on a specific method family or resource type, such as inference-time computation, online adaptation, retrieval, tool use, memory, or agentic workflows. In contrast, this survey examines how these directions function as inference-stage interventions that differ in intervention locus, information source, temporal persistence, and inference-time cost. From this perspective, prompting, retrieval,

adaptation, reasoning, verification, memory, tools, and agents are not treated as isolated topics, but as related intervention routes that may be combined within the same inference procedure. This positioning motivates the Locus × Source framework developed in the following sections.

2.3 The Locus Axis: Intervention Positions

The primary axis of this survey is locus, defined as the position at which a test-time method intervenes in the inference pipeline. Locus is not a taxonomy of model architectures or application domains. It identifies the component of the test-time system that is primarily changed to alter or improve the model's inference-time outputs. At the level of abstraction needed for this survey, three loci provide a useful organizing structure. L1 Input methods change the information presented to the model. L2 State methods update, steer, extend, or preserve the state variables that are accessed, updated, or preserved during inference. L3 Process methods reorganize how reasoning, search, verification, tool use, action, or collaboration unfolds.

Table 2 summarizes the three intervention loci.

Table 2: Locus axis for test-time improvement.

Locus	**Intervention object**	**Typical mechanisms**	**Representative evidence**
L1 Input	Information presented to the model	Prompt rewriting, task expression, context assembly, retrieval evidence, ICL demonstrations, input or view augmentation	Few-shot and instruction prompting Brown et al. (2020); Schick and Schütze (2021); RAG and REALM Lewis et al. (2020); Guu et al. (2020); demonstration selection Lee et al. (2026).
L2 State	Internal or external states accessed, updated, or preserved during inference	Parameter or statistic updates, activation and hidden-state steering, external memory, experience routing, continual adaptation	TTT, TENT, MEMO and TPT Sun et al. (2020); Wang et al. (2021); Zhang et al. (2022); Shu et al. (2022); representation steering Zou et al. (2023); Li et al. (2023); long-term memory and agents Zhong et al. (2024); Packer et al. (2023); Wang et al. (2023c).
L3 Process	Inference procedure itself	CoT, decomposition, multi-path search, verifier-guided decoding, self-refinement, tool use, agents, multi-agent workflows	CoT and self-consistency Wei et al. (2022); Wang et al. (2023b); tree and graph search Besta et al. (2024); self-refinement and tools Madaan et al. (2023); Schick et al. (2023).

2.3.1 Input Intervention

L1 Input methods improve inference by changing the form, content, or organization of information presented to the model. These methods operate on the input side by rewriting the prompt, specifying an output schema, retrieving documents, selecting demonstrations, producing formalized representations of visual information, or assembling context that the model cannot reliably access from parametric knowledge alone. Input-level intervention should therefore be understood functionally rather than temporally. In many systems, input construction is iterative because a query may be rewritten after retrieval feedback, a context may be expanded after partial reasoning, or demonstrations may be selected adaptively. Even when such construction is iterative, the central intervention remains the information supplied to the model.

The literature illustrates three recurring input-level families. Prompting and task-expression methods exploit the sensitivity of large models to instructions, templates, roles, query formulations, and output formats (Bsharat et al. 2023; Zheng et al. 2024b; Nori et al. 2023). Context assembly and RAG methods address the limits of parametric knowledge by injecting retrieved documents, graph facts, domain evidence, or other external observations into the current inference context (Asai et al. 2024; Jiang et al. 2023a; Trivedi et al. 2023). Demonstration-based in-context learning methods use few-shot examples as a temporary task program, where the selection, ordering, and construction of demonstrations can substantially alter model behavior (Fang et al. 2025; Tian and Peng 2025). These families are not mutually exclusive in modern systems, but they clarify three major ways in which input-side interventions improve inference by shaping task expression, supplying evidence, or inducing behavior from examples.

2.3.2 State Intervention

L2 State methods act on the state variables that are accessed, updated, or preserved during inference. These states include model parameters, normalization statistics, adapters, learnable prompt parameters, hidden states, activation directions, external memory stores, tool histories, and long-lived experience buffers. The term state is therefore broader than model weights. The unifying criterion is not whether the base weights are changed, but whether an inference-relevant state variable is updated, steered, extended, retrieved, or preserved so that the system behaves differently at test time.

The state locus is central to classic test-time adaptation, where unlabeled test samples or test streams provide signals for parameter, statistic, or prompt-parameter updates. Test-time training, entropy minimization, augmentation-consistency adaptation, and test-time prompt tuning are anchor examples (Sun et al. 2020; Wang et al. 2021; Zhang et al. 2022; Shu et al. 2022). The same locus also covers representation-level interventions that steer hidden states without full test-time adaptation, including activation addition, function vectors, representation engineering, truthfulness- or factuality-oriented interventions, and contrastive layer decoding (Turner et al. 2023; Zou et al. 2023; Li et al. 2023; Todd et al. 2024; Chuang et al. 2024; Zhang et al. 2024b).

External memories and experience stores become state-level when they allow information, interaction history, or tool experience to be written, retrieved, and reused beyond a single context window, as in memory-augmented transformers, long-term chat memory, generative agents, and tool-experience systems (Zhong et al. 2024; Packer et al. 2023; Wang et al. 2023c; Wu et al. 2022; Wang et al. 2023a; Park et al. 2023; Xiao et al. 2025).

2.3.3 Process Intervention

L3 Process methods improve inference by reorganizing the procedure through which an output, action, or decision is produced. Their intervention lies in the control structure of inference, including how intermediate steps are generated, how alternative paths are explored, how feedback is incorporated, how tools or environments are used, and how roles are coordinated. The defining feature of this locus is therefore procedural organization rather than any particular method name.

Process-level methods can be characterized as different ways of organizing additional inference-time computation. Some methods make the computation more explicit by externalizing or decomposing reasoning into intermediate steps, as in chain-of-thought and least-to-most prompting (Wei et al. 2022; Zhou et al. 2023). Others increase diversity and robustness by sampling multiple reasoning traces and aggregating answers, as in self-consistency (Wang et al. 2023b). Search-based methods extend this approach by transforming a single generation into candidate-space exploration over trees or graphs (Yao et al. 2023b; Besta et al. 2024). Feedback and verification methods add an evaluative loop, allowing outputs to be judged, selected, or revised before a final response is produced (Madaan et al. 2023; Shinn et al. 2023; Hosseini et al. 2024). Tool and agent methods extend the procedure beyond the language model itself, allowing the system to search the web, execute code, interact with environments, or delegate subtasks (Yao et al. 2023a; Schick et al. 2023; Surís et al. 2023; Zheng et al. 2024a). Across these variants, the central contribution of L3 is not limited to increasing test-time computation, but lies in making that computation procedurally organized and controllable.

The three loci above define the main intervention positions used throughout this survey. For later discussion, Table 3 summarizes the recurring branches that appear under each locus. These branches should be read as compact descriptors of common intervention patterns, not as additional Section-level axes.

Table 3: Recurring branches under the locus axis.

Locus	Branch	Core function
L1 Input	L1.A Prompt and task expression	Shapes task interpretation and response formation through the input surface.
	L1.B Context assembly and retrieval augmentation	Supplies evidence or domain information through the current input context.
	L1.C Demonstration-based in-context learning	Conditions local task behavior through examples placed in context.

Locus	Branch	Core function
L2 State	L2.A Test-time parameter/statistic adaptation	Updates inference-relevant variables using current samples, batches, or streams.
	L2.B Activation and representation steering	Steers hidden states or representation trajectories without full test-time adaptation.
	L2.C External memory and experience state	Extends state beyond a single context window through persistent storage and reuse.
	L2.D Continual adaptation and state stabilization	Maintains useful state under non-stationary streams or repeated updates.
L3 Process	L3.A Reasoning decomposition	Externalizes or decomposes inference into intermediate reasoning steps.
	L3.B Search and candidate exploration	Explores multiple paths or solution candidates before producing a final answer.
	L3.C Feedback, verification, and refinement	Adds evaluative loops for judging, selecting, or revising outputs.
	L3.D Tool use and agentic action	Extends inference through external actions, tools, or environments.
	L3.E Multi-agent and role-based inference	Coordinates multiple roles or agents for planning, solving, checking, or execution.

2.4 Sources of Inference-Time Information

Locus describes where a TTI method intervenes in the inference pipeline; source describes where the information used by that intervention originates. This distinction is necessary because the same intervention locus can rely on different kinds of information. An input-side method may use only the model's internal knowledge, retrieve evidence from a static corpus, query a live tool, adapt to a test sample, or incorporate feedback from a verifier. Likewise, the same information source may be used at different loci: retrieved evidence may be inserted into the prompt, written into memory, or used to guide a multi-step reasoning process.

Table 4 summarizes the five information-source categories.

Table 4: Source axis for test-time improvement.

Source	Meaning	Typical directions	Examples in the reviewed literature
S0 Internal-only	No new external information is introduced during the inference episode; the system reorganizes existing model knowledge, samples, hidden states, or generated traces.	CoT, self-consistency, latent reasoning, activation steering	CoT, self-consistency, activation steering and representation engineering Wei et al. (2022); Wang et al. (2023b); Turner et al. (2023); Zou et al. (2023).
S1 Static-KB	Information comes from a static corpus, knowledge base, knowledge graph, retriever index, or prebuilt resource.	RAG, KG-RAG, domain KB, procedural KB, memory-style RAG	RAG, REALM, Atlas, graph and medical RAG Lewis et al. (2020); Guu et al. (2020); Izacard et al. (2023); Mavromatis and Karypis (2025); Wu et al. (2024c).
S2 Live-tool/env	Information comes from real-time tools, web pages, APIs, code execution, interactive environments, or other external observations.	ReAct, web agents, code agents, tool learning, visual tools	ReAct, Toolformer, ViperGPT, GPT-4V web agents and ToolSandbox Yao et al. (2023a); Schick et al. (2023); Surís et al. (2023); Lu et al. (2025b).
S3 Test-stream TTA	Information comes from the current test sample, test batch, or non-stationary test stream.	Test-time training, entropy minimization, continual TTA, black-box or VLM TTA	TTT, TENT, CoTTA, EATA and SAR Sun et al. (2020); Wang et al. (2021, 2022); Niu et al. (2022, 2023).
S4 Self-/Verifier	Information comes from model outputs, self-generated feedback, critics, verifiers, reward models, execution-based checks, or other evaluative signals.	Self-refine, Reflexion, verifier decoding, reward-driven reasoning, self-evolution	Self-Refine, Reflexion, verifier-guided decoding and verifiable self-evolution Madaan et al. (2023); Shinn et al. (2023); Heng et al. (2026).

Source should be read as an information-origin descriptor rather than as a method class. Its role is to refine the locus axis by distinguishing methods that intervene at a similar position but rely on different inference-stage information. An input-level intervention, for instance, may be driven by internal model knowledge, static retrieved evidence, live tool observations, test-sample signals, or verifier feedback. These methods share the same locus, but they differ in what information is brought into the inference episode and what role that information plays. Conversely, the same source may support different loci: retrieved evidence may be inserted into a prompt, stored as reusable memory, or used inside a multi-step reasoning process. Locus therefore

specifies where an intervention acts, while Source specifies where the information used by that intervention comes from. The next section combines the two axes into a Locus × Source research map.

2.5 Locus × Source Map

Combining the locus and source axes gives the survey a compact positioning map for test-time improvement. Figure 3 visualizes this space as a qualitative heatmap: rows indicate where an intervention acts in the inference pipeline, columns indicate where the information used by the intervention originates, and color intensity indicates relative research density and consolidation. The cell labels identify representative configurations rather than exhaustive method classes. The heatmap should therefore be read as a positioning device, not as a performance ranking. The search, screening, coding, and per-cell count protocol used to construct this heatmap is reported in Appendix A.

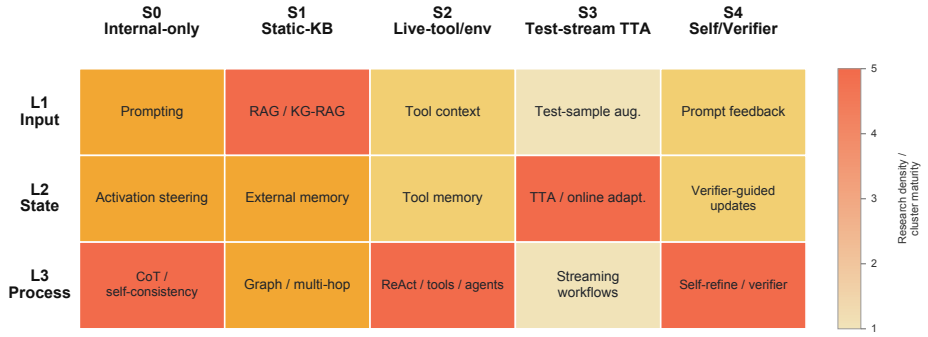


Fig. 3 Locus × Source heatmap for test-time improvement. Rows represent intervention loci and columns represent information sources. Color intensity is an ordinal synthesis based on the screened core-method corpus and coding protocol in Appendix A. The heatmap is intended as a positioning device rather than a performance ranking.

The heatmap highlights two observations. First, TTI research has developed unevenly across the Locus × Source space. Some pairings have formed relatively stable research clusters, including input-level methods grounded in static knowledge sources, state-level methods driven by test-stream adaptation, and process-level methods organized around internal reasoning, live tools, or self/verifier feedback. These darker regions indicate where recurring mechanisms, representative methods, and evaluation practices have become more consolidated.

Second, lighter or intermediate cells are not simply irrelevant gaps. They indicate pairings that are less consolidated, emerging, or only partially connected to existing research lines. Their value is mainly diagnostic: they show where the current landscape is thinner, where hybrid configurations may be harder to position, and where additional work may be needed to clarify mechanisms, evaluation settings, or system

roles. In this sense, the heatmap positions TTI as a structured design space rather than a list of isolated techniques. Locus identifies the part of the inference system being intervened on, while source identifies the origin of the information that enables the intervention.

2.6 Persistence and Cost

Temporal persistence and inference-time cost are retained as auxiliary descriptors rather than Section-level axes. Temporal persistence describes the time scale over which a test-time intervention operates or remains effective, ranging from a single inference episode to within-sample interaction, cross-sample online accumulation, and persistent state across sessions or tasks. Inference-time cost describes the additional resources required to realize the intervention, including extra decoding, online gradients, explicit search, tool calls, environment interaction, or agent coordination.

Methods in the same Locus $\times$ Source region may still differ substantially in persistence and resource demand. These descriptors capture that variation and provide the vocabulary for the trade-off analysis in Section 4; Table 5 summarizes the corresponding labels.

Table 5: Auxiliary descriptors for TTI.

Descriptor	Label	Meaning	Typical examples
Temporal	T0 Single episode	Affects one inference episode without multi-round interaction or persistent state.	Simple prompt reformulation (Brown et al. 2020; Schick and Schütze 2021), single retrieval (Lewis et al. 2020; Guu et al. 2020), single activation steering (Turner et al. 2023).
Temporal	T1 Within-sample multi-round	Uses multiple rounds within one sample or task.	Self-consistency (Wang et al. 2023b), self-refine (Madaan et al. 2023), tool loops (Yao et al. 2023a), tree search (Yao et al. 2023b).
Temporal	T2 Cross-sample online	Accumulates information across a test batch or stream.	TENT (Wang et al. 2021), CoTTA (Wang et al. 2022), EATA (Niu et al. 2022), online TTA (Wang et al. 2025b).
Temporal	T3 Persistent	Maintains reusable state across sessions, tasks, users, or agents.	MemoryBank (Zhong et al. 2024), MemGPT (Packer et al. 2023), Voyager (Wang et al. 2023c), tool memory (Xiao et al. 2025).

Descriptor	Label	Meaning	Typical examples
Cost	C0 Single forward	Adds little additional inference procedure beyond the base forward process.	Simple prompting (Brown et al. 2020) or representation steering (Turner et al. 2023; Zou et al. 2023).
Cost	C1 Multi-decoding / iterative generation	Multiple generations, candidate sampling, or iterative refinement.	Self-consistency (Wang et al. 2023b), Best-of-N or repeated sampling (Brown et al. 2024), self-refine (Madaan et al. 2023).
Cost	C2 Online gradient	Gradient update during testing or inference.	TTT (Sun et al. 2020), TENT (Wang et al. 2021), TPT (Shu et al. 2022), MEMO (Zhang et al. 2022).
Cost	C3 Search/MCTS	Explicit tree, graph, or value-guided search.	Tree of Thoughts (Yao et al. 2023b), Graph of Thoughts (Besta et al. 2024), compute-optimal TTS (Snell et al. 2025; Liu et al. 2025a).
Cost	C4 External tools / agents	Tool calls, external environments, or agent coordination.	ReAct (Yao et al. 2023a), Toolformer (Schick et al. 2023), web agents (Zheng et al. 2024a), multi-agent systems (Wu et al. 2024a; Hong et al. 2024).

2.7 Survey Logic

The framework developed in this section provides the organizing basis for the rest of the survey. Locus identifies the intervention position, Source identifies the origin of inference-stage information, and the Temporal and Cost descriptors record persistence and resource demand. Together, these dimensions allow TTI methods to be analyzed without reducing them to method names or isolated subfields.

Within this framework, the core discussion focuses on works whose test- or inference-stage mechanisms directly shape the current solving behavior of a trained model. Works on pretraining, offline fine-tuning, dataset construction, or benchmarks are cited when they clarify the development, evaluation, or boundary of inference-stage improvement. Each reviewed method is described through the Locus, Source, Temporal, and Cost descriptors introduced above. For hybrid systems, the analysis follows the functional role of each component in the inference pipeline rather than assigning the whole system to a single method category. This same logic also supports the later landscape analysis: heatmap intensity summarizes the relative concentration and consolidation of recurring mechanisms within the Locus $\times$ Source map, while opportunity regions are discussed in relation to mechanism trajectories, resource-control questions, and evaluable settings.

The following sections use this map in two steps. Section 3 follows the Locus axis to describe mechanism development and cross-locus migration. Section 4 then shifts from mechanism description to improvement logic, asking how these developments transform bottlenecks into inference-time computation, reusable state, or verifiable supervision.

3 Test-Time Interventions and Hybrid Systems

Building on the locus-first map introduced in Section 2, this section examines how test-time improvement mechanisms have evolved across input, state, and process loci. Rather than adding another layer of classification, the discussion traces how local inference-stage interventions develop into more coupled mechanisms. The following sections therefore organize the discussion around input-level, state-level, and process-level mechanisms before analyzing their migration across loci into hybrid inference systems.

3.1 Input-Level Interventions

At the input locus, TTI starts from a basic property of model behavior: the same trained model can produce different outputs when a task is formulated differently, when retrieved documents or other external information are added to the context, or when in-context examples specify a local task pattern. Input-level methods can therefore be understood through three recurring mechanisms: task expression, retrieval-augmented context construction, and demonstration-based conditioning. Across these mechanisms, the field has moved from relatively static input specification toward adaptive construction of the information made available to the model at inference time; Table 6 summarizes this movement by comparing their intervention objects, representative methods, and technical trajectories.

These trajectories show that input-level TTI is no longer limited to prompt wording alone; it increasingly treats the inference context as an object to be formulated, populated, and organized before generation. The following subsections examine the three mechanisms in turn.

3.1.1 Task Expression and Prompting

Task expression methods intervene before generation by changing how a problem is stated, constrained, or mapped onto the model's prediction interface. GPT-3 made this mechanism visible at scale by showing that language models can perform zero-shot, one-shot, and few-shot tasks from natural-language prompts without task-specific parameter updates (Brown et al. 2020). PET/iPET develops a complementary formulation for masked language models: classification and inference tasks are converted into cloze-style language modeling problems through pattern-verbalizer pairs, so that label words become the interface through which the model makes a decision (Schick and Schütze 2021). In both cases, the input does more than carry the query; it defines the operational form in which the model encounters the task.

Later work makes task expression more systematic and evaluation-sensitive. Instruction-design studies examine how prompt components influence model behavior,

Table 6 Mechanism trajectories in input-level TTI.

Input-level mechanism	Intervention object	Representative methods	Technical trajectory
L1.A Task expression	Task formulation, instructions, verbalizers, role framing, output schemas, query rewriting, and cross-modal formalization.	GPT-3 few-shot prompting; PET/iPET; instruction-design studies; Medprompt; persona-prompt analysis; MaFeRw; R1-Onevision Brown et al. (2020); Schick and Schütze (2021); Bsharat et al. (2023); Zheng et al. (2024b); Nori et al. (2023); Wang et al. (2025c); Yang et al. (2025b).	From manually designed task formulations and few-shot examples toward more systematic instruction design, feedback-conditioned query rewriting, structured output control, and cross-modal formalization.
L1.B Retrieval-augmented context construction	Retrieved documents, knowledge passages, graph facts, domain information, rankings, and structured context supplied to the current inference episode.	RAG; REALM; Atlas; Self-RAG; FLARE; IRCoT; RankRAG; RVR; graph-based and medical RAG; RAG evaluation studies Lewis et al. (2020); Guu et al. (2020); Asai et al. (2024); Jiang et al. (2023a); Trivedi et al. (2023); Izacard et al. (2023); Mavromatis and Karypis (2025); Wu et al. (2024c); Yu et al. (2024); Qian et al. (2026); Gutiérrez et al. (2024); Li et al. (2025b); Tang and Yang (2024); Yang et al. (2024); Wang et al. (2024a); Ni et al. (2025); Peng et al. (2026a).	From retrieve-then-read pipelines toward adaptive retrieval, interleaved retrieval and reasoning, context ranking, graph-aware or domain-specific grounding, and dedicated evaluation of retrieval quality.
L1.C Demonstration-based conditioning	In-context examples that specify label spaces, input-output mappings, reasoning formats, ordering effects, or multimodal alignments.	GPT-3 in-context learning; visual demonstration selection; invariance-oriented ICL; multimodal few-shot relation extraction Brown et al. (2020); Lee et al. (2026); Fang et al. (2025); Tian and Peng (2025).	From simple example concatenation toward similarity-, diversity-, order-, structure-, and modality-aware demonstration selection, with later movement toward retrievable examples, memory, and search.

while Medprompt shows that organized prompting strategies can support specialized medical question answering without modifying the base model (Bsharat et al. 2023; Nori et al. 2023). Persona-prompt analyses add an important constraint: role descriptions may change behavior, but they do not guarantee improvement across tasks or evaluation settings (Zheng et al. 2024b). These findings position task expression as a controllable but conditional mechanism, whose effect depends on the task, model, prompt structure, and baseline.

Recent work further extends task expression from hand-written prompts to adaptive and cross-modal input construction. MaFeRw rewrites conversational RAG queries with multi-aspect feedback, so that the query supplied to the retriever and generator reflects retrieval and answer-quality signals rather than only a static template (Wang et al. 2025c). R1-Onevision broadens task expression across modalities by converting visual information into formalized representations that are more compatible with language-based reasoning (Yang et al. 2025b). In these cases, the immediate intervention is still the form of the input through which the model receives the task, even when that input is produced by feedback, retrieval, or multimodal preprocessing.

3.1.2 Retrieval-Augmented Context Construction

Retrieval-augmented methods address an input-level limitation that prompting alone cannot resolve: the model may not have access to the knowledge required for the current query, or the relevant information may be stale, domain-specific, or too precise to recover reliably from parametric memory. RAG combines a pretrained generator with a non-parametric memory implemented as a dense vector index of Wikipedia, allowing generation to condition on retrieved passages (Lewis et al. 2020). REALM similarly learns a retriever that retrieves documents from a large corpus and provides them to the language model during pretraining, fine-tuning, and inference (Guu et al. 2020). Atlas extends this retrieval-augmented language modeling line to few-shot knowledge-intensive tasks (Izacard et al. 2023). The central intervention is contextual availability: retrieval changes what information is available to the generator in the current inference episode.

This mechanism becomes more active when retrieval is no longer a single preliminary step. Self-RAG trains a model to decide when to retrieve and to critique both retrieved passages and generated content through reflection tokens, thereby coupling retrieval, generation, and self-evaluation within the same inference behavior (Asai et al. 2024). FLARE anticipates future content by generating a provisional next sentence and using it as a retrieval query when low-confidence tokens suggest that additional information is needed (Jiang et al. 2023a). IRCoT interleaves retrieval with chain-of-thought reasoning for multi-step question answering, because later retrieval needs depend on intermediate reasoning states (Trivedi et al. 2023). RankRAG adds an explicit ranking function by instruction-tuning a single LLM for both context ranking and answer generation (Yu et al. 2024). RVR adds a verifier-conditioned retrieval loop: verified documents from earlier rounds are used to expand later queries, making retrieval coverage itself a feedback-controlled input-construction

process (Qian et al. 2026). Together, these methods shift retrieval-augmented generation from retrieve-then-read pipelines toward adaptive construction of the context supplied to the model.

The same trajectory continues when retrieval enters graph-structured and domain-specific settings. Knowledge-graph and medical RAG systems do not merely retrieve more passages; they attempt to organize retrieved information around relations, multi-hop structure, domain constraints, or clinically relevant grounding (Mavromatis and Karypis 2025; Wu et al. 2024c; Li et al. 2025b). As retrieval-augmented systems move into multi-hop, graph-based, domain-specific, and safety-sensitive scenarios, context quality becomes a central research object. Benchmarks and surveys such as MultiHop-RAG, CRAG, RAG best-practice studies, trustworthy RAG surveys, and GraphRAG surveys make this shift explicit by focusing on retrieval quality, ranking, grounding, robustness, and evaluation (Tang and Yang 2024; Yang et al. 2024; Wang et al. 2024a; Ni et al. 2025; Peng et al. 2026a). The resulting question is no longer only whether external information is added to the prompt, but how the system selects, orders, structures, and validates the information made available during inference.

3.1.3 Demonstration Conditioning and Selection

Demonstration-based conditioning uses examples in the current context to specify a local task pattern for the present query. GPT-3 made this mechanism visible by showing that few-shot examples can induce task behavior without task-specific parameter updates (Brown et al. 2020). Later work clarifies what such examples contribute: they may indicate the label space, input-output relation, reasoning format, or multimodal mapping that the model should follow. FewRelEx illustrates this point in multimodal few-shot relation extraction, where visual and textual features are aligned to support relation prediction under limited examples (Tian and Peng 2025). Analyses of multimodal in-context learning further show that demonstrations are not automatically useful simply because they are included in the prompt; their effect depends on how examples are retrieved, ordered, and constructed for the target task (Baldassini et al. 2024; Qin et al. 2024). Demonstrations are therefore best understood as temporary task specifications whose usefulness depends on their content and organization.

The mechanism becomes more explicit when demonstration choice is treated as part of input construction rather than as simple example concatenation. Visual demonstration selection shows that nearest-neighbor similarity can be insufficient in multimodal settings, where selected examples may need to balance visual relevance and diversity (Lee et al. 2026). Se2 extends this idea by formulating example construction as sequential selection, so that inter-example relations and ordering are considered during selection (Liu et al. 2024c). Recent work then pushes the mechanism toward scalable and structured demonstrations: UniICL combines selection with compression through virtual tokens, and Multimodal Task Vectors compress many multimodal examples into compact task representations for many-shot multimodal ICL (Gao et al. 2025; Huang et al. 2024b). Long-context and domain-specific studies further indicate that larger context windows or larger candidate pools do not remove the need for careful example choice: Refract ICL revisits selection under million-token contexts, while RAICL, CoDR, and GraphWalker adapt demonstration retrieval or organization

to medical multimodal classification, LVLM demonstration selection, and EHR-based clinical reasoning (Akula et al. 2025; Zhan et al. 2026; Wang et al. 2026; Fang et al. 2026a). Overall, L1.C evolves from placing examples in the prompt toward selecting, ordering, compressing, and structuring examples as input resources; this also prepares its later migration into reusable example libraries, retrieval targets, or memory-like resources.

3.2 State-Level Interventions

State-level TTI acts on inference-relevant variables and can be updated, steered, stored, or stabilized at test time. These variables include model parameters, normalization statistics, prompt parameters, adapters, hidden activations, memory entries, tool histories, and skill libraries. The relevant unit of analysis is therefore the state that changes, guides, or persists during inference, rather than the visible input alone. This section follows four state-level trajectories: test-time updating of parameters or statistics, activation and representation steering, external memory and experience state, and continual stabilization under repeated or non-stationary test-time updates. Table 7 summarizes these trajectories by comparing the state variables emphasized, representative methods, and technical movements.

3.2.1 Test-Time State Updating

Classical test-time adaptation updates model variables using information from the current test sample, batch, or stream. TTT turns a single unlabeled test sample into a self-supervised learning problem and updates model parameters before making the prediction (Sun et al. 2020). TENT considers a fully test-time setting in which only the trained model and test data are available; it minimizes prediction entropy while estimating normalization statistics and optimizing channel-wise affine parameters online (Wang et al. 2021). MEMO combines test-time adaptation with augmentation by minimizing the entropy of the marginal prediction over augmented views of a single test input, thereby encouraging predictions that are stable under input transformations (Zhang et al. 2022). TPT extends test-time updating to CLIP-style vision-language models by learning adaptive prompts for each test sample, using augmented views and confidence selection to guide the update (Shu et al. 2022). These methods establish the basic state-level pattern: test-time information is used not only to produce a prediction, but also to adjust inference-relevant variables.

Later work refines this pattern by making the update target and update signal more selective. T3A adjusts the classifier at test time rather than updating the whole model, showing that local decision modules can be adapted using target-domain features (Iwasawa and Matsuo 2021). AdaContrast introduces contrastive objectives into test-time adaptation, shifting the signal from entropy alone toward representation structure under distribution shift (Chen et al. 2022). Test-time training on nearest neighbors further links adaptation to retrieved neighbor examples, so that the update is conditioned by local similarity structure rather than by the test sample in isolation (Hardt and Sun 2024). In vision-language settings, CLIPTTA, ClipTTT, and Ramen continue this movement toward lightweight and sample-aware adaptation: they focus

Table 7 Mechanism trajectories in state-level TTI.

State-level mechanism	State variable emphasized	Representative methods	Technical trajectory
L2.A Test-time parameter/statistic adaptation	Model weights, normalization statistics, prompt parameters, classifiers, adapters, local layers, or VLM-specific update targets.	TTT, TENT, MEMO, TPT, T3A, AdaContrast, TTT on nearest neighbors, CLIPTTA, ClipTTT, Ramen, GRPO-TTA, and Fair Context Learning Sun et al. (2020); Wang et al. (2021); Zhang et al. (2022); Shu et al. (2022); Iwasawa and Matsuo (2021); Chen et al. (2022); Hardt and Sun (2024); Lafon et al. (2025); Nath et al. (2026); Bao et al. (2026); Li et al. (2026a); Yun et al. (2026a).	From broad test-sample updates toward selective update targets, entropy, consistency, reward-guided, or evidence-balanced objectives, prompt/adaptor tuning, and sample-control strategies for mixed or shifted test distributions.
L2.B Activation and representation steering	Hidden states, activation directions, layer logits, latent trajectories, and behavior-associated representation subspaces.	PPLM, ActAdd, Function Vectors, Representation Engineering, ITI, CAA, DoLa, TruthX, LITO, lying-state diagnostics, and TTT layers Turner et al. (2023); Zou et al. (2023); Li et al. (2023); Todd et al. (2024); Chuang et al. (2024); Zhang et al. (2024b); Dathathri et al. (2020); Rimsky et al. (2024); Bayat et al. (2024); Azaria and Mitchell (2023); Sun et al. (2025).	From gradient-based activation perturbation toward reusable steering directions, representation-level monitoring, layer-contrast decoding, truthfulness or factuality steering, and designed hidden-state computation.
L2.C External memory and experience state	Writable stores, memory banks, databases, episodic traces, reflective records, tool histories, and skill libraries.	Memorizing Transformers, LongMem, MemoryBank, MemGPT, ChatDB, HippoRAG, Generative Agents, Reflexion, Voyager, ToolMem, LightMem, TokMem, xMemory, XSkill, CASCADE, Mnemis, and MeMo Zhong et al. (2024); Packer et al. (2023); Wang et al. (2023c); Wu et al. (2022); Wang et al. (2023a); Park et al. (2023); Xiao et al. (2025); Shinn et al. (2023); Gutiérrez et al. (2024); Hu et al. (2023); Fang et al. (2026b); Wu et al. (2026b); Hu et al. (2026); Jiang et al. (2026); Guo et al. (2026); Tang et al. (2026); Quek et al. (2026).	From longer context support toward explicit write, retrieve, index, compress, reflect, update, route, forget, modelize, and reuse operations across episodes.
L2.D Continual adaptation and state stabilization	State maintained under repeated updates, non-stationary streams, or long-horizon deployment.	CoTTA, EATA, SAR, RMT, EcoTTA, online TTA surveys, and video-stream TTT Wang et al. (2022); Niu et al. (2022, 2023); Döbler et al. (2023); Song et al. (2023); Wang et al. (2025b,f).	From one-shot or batch-wise adaptation toward anti-forgetting, reset/replay mechanisms, memory-efficient regularization, stream-aware update control, and stability under distribution drift.

on robust CLIP/VLM adaptation, CLIP-guided test-time training, and active sample selection for mixed or shifted test distributions (Lafon et al. 2025; Nath et al. 2026; Bao et al. 2026). GRPO-TTA further indicates that VLM test-time adaptation is moving from confidence or entropy objectives toward reward-guided update signals, reformulating class prediction as group-wise policy optimization without ground-truth labels (Li et al. 2026a). Fair Context Learning provides a complementary example in which test-time prompt/context adaptation is guided by evidence-balanced calibration rather than entropy minimization (Yun et al. 2026a). The resulting trend is not simply to update more parameters at test time, but to decide which variables should change, which test-time signals are reliable enough to guide the change, and how the update can remain tied to the current inference context.

3.2.2 Representation Steering

Beyond explicit test-time updates, another state-level route is to steer the internal representations through which generation or prediction unfolds. PPLM is an early example of this route: it uses gradients from attribute models during generation to perturb latent activations toward desired properties (Dathathri et al. 2020). Activation Addition makes the intervention lighter by contrasting the intermediate activations induced by prompt pairs, extracting a steering vector, and adding that vector during the forward pass to shift high-level output behavior (Turner et al. 2023). Function Vectors and Representation Engineering further treat internal representations as carriers of task or behavior information; Function Vectors extract reusable directions from in-context learning behavior, while RepE emphasizes population-level representations as objects that can be monitored and controlled (Zou et al. 2023; Todd et al. 2024). The movement is therefore from gradient-based activation perturbation toward reusable representation directions and representation-level control.

A later line of work applies this view to truthfulness, factuality, and behavior monitoring. ITI identifies directions associated with truthful answering in selected attention heads and shifts activations along those directions at inference time (Li et al. 2023). Contrastive Activation Addition extends the steering-vector idea through contrastive activation directions, while DoLa changes decoding by contrasting logits projected from different layers, aiming to improve factuality without retrieval or additional fine-tuning (Chuang et al. 2024; Rimsky et al. 2024). TruthX and LITO explore truthfulness-oriented latent-space intervention and uncertainty expression, and lying-state diagnostics examine whether internal states contain signals associated with deceptive or untruthful behavior (Zhang et al. 2024b; Bayat et al. 2024; Azaria and Mitchell 2023). Together, these methods show that state-level TTI can guide model behavior not only by updating parameters, but also by operating on the representations and layer-wise signals that already arise during inference.

3.2.3 External Memory and Experience State

State-level intervention is not limited to variables inside the model; it also includes external states that persist beyond a single inference episode. Memorizing Transformers and LongMem augment language models with memory mechanisms that retrieve from memory banks, allowing long past context or many-shot demonstrations to

influence later prediction (Wu et al. 2022; Wang et al. 2023a). MemoryBank makes long-term interaction memory explicit by storing, updating, and retrieving relevant memories over time (Zhong et al. 2024). MemGPT frames this problem as virtual context management: because the LLM context window is limited, the system manages information across different memory tiers and moves it into the active context when needed (Packer et al. 2023). ChatDB represents a more symbolic form of external state, using SQL databases as memory that the LLM can manipulate through generated SQL instructions (Hu et al. 2023). These methods move memory from passive context extension toward managed external state.

Agent systems extend this idea from stored information to accumulated experience. Generative Agents maintain a memory stream of observations, synthesize higher-level reflections, and retrieve relevant memories to support planning and behavior (Park et al. 2023). Reflexion stores verbal reflections in an episodic memory buffer so that feedback from prior attempts can guide later trials (Shinn et al. 2023). Voyager turns environment interaction into an expanding library of executable skills in Minecraft, using feedback, execution errors, and self-verification to improve and reuse temporally extended behaviors (Wang et al. 2023c). ToolMem similarly stores summaries of tool capabilities, including observed strengths and weaknesses, and retrieves them to guide future tool selection (Xiao et al. 2025). More recent systems such as LightMem, TokMem, xMemory, and XSkill further explore lightweight memory stages, procedural memory tokens, hierarchical retrieval, and skill-oriented experience reuse (Fang et al. 2026b; Wu et al. 2026b; Hu et al. 2026; Jiang et al. 2026). CASCADE extends this experience-state view to deployment-time learning, where an evolving episodic memory and contextual-bandit selection turn past cases into reusable decision support without modifying model parameters (Guo et al. 2026). Mnemis further shows that long-term memory retrieval may require both similarity-based access and deliberate global selection over hierarchical memory graphs, while MeMo represents a model-based variant in which new knowledge is encoded into a dedicated memory model with the base LLM kept frozen (Tang et al. 2026; Quek et al. 2026). Across these variants, memory becomes a state-level resource when information is not merely placed in a long context, but organized through operations such as writing, indexing, updating, compression, retrieval, forgetting, and reuse across episodes.

3.2.4 Continual Adaptation and Stabilization

When test-time updating is applied repeatedly to a non-stationary test stream, the problem shifts from one-time adaptation to preserving useful state over time. A state update that improves one batch may become unreliable when later samples are noisy, heterogeneous, or drawn from a gradually changing distribution. CoTTA places test-time adaptation in continual non-stationary streams, shifting attention from one-time adaptation to whether performance can be maintained as the target distribution changes (Wang et al. 2022). EATA reduces unnecessary or unreliable updates by emphasizing efficient adaptation and anti-forgetting constraints, while SAR focuses on stable adaptation in dynamic real-world conditions where test samples may be heterogeneous or unreliable (Niu et al. 2022, 2023). RMT further stabilizes continual adaptation through robust mean-teacher regularization, and EcoTTA addresses the

memory cost of continual adaptation through self-distilled regularization (Döbler et al. 2023; Song et al. 2023). These methods treat adaptation not as a single update event, but as an ongoing process in which the updated state must remain useful over time.

The shared mechanism is to regulate how state changes accumulate across a stream. A continually adapting system must decide which samples are reliable enough to update from, which parameters or statistics should change, when previous state should be restored or regularized, and how to prevent confident errors from becoming persistent state. Video-stream TTT extends this concern to temporally ordered visual streams, where adaptation must operate over sequential inputs rather than isolated test examples (Wang et al. 2025f). Online TTA surveys further show that deployment-like streams require explicit assumptions about ordering, distribution shift, update frequency, and evaluation protocol (Wang et al. 2025b). In this sense, continual adaptation extends test-time updating into a stabilization problem: useful state must remain plastic enough to adapt, but constrained enough to resist drift, forgetting, and error accumulation.

3.3 Process-Level Interventions

After input construction and state management, a third route to test-time improvement is to reorganize the procedure by which an answer, action, or decision is produced. Process-level methods make inference operate through intermediate traces, candidate spaces, evaluative loops, tool-mediated actions, or coordinated roles. Their common contribution is to turn generation from a single forward event into a controllable procedure in which steps can be decomposed, alternatives explored, outputs checked, external actions invoked, and responsibilities distributed. Table 8 summarizes the main process-level trajectories: reasoning decomposition, search and candidate exploration, feedback and verification, tool use and agentic action, and multi-agent coordination.

Across these trajectories, process-level TTI becomes effective when additional inference-time computation is organized into traces, candidate spaces, evaluative loops, action policies, or coordination protocols. The following subsections examine how these procedural forms develop and increasingly combine with input-level and state-level resources.

3.3.1 Reasoning Decomposition and Trace Design

Reasoning decomposition turns answer generation into a procedure with explicit intermediate steps. Chain-of-thought prompting made this mechanism visible by showing that exemplars containing intermediate reasoning steps can improve performance on arithmetic, commonsense, and symbolic reasoning tasks (Wei et al. 2022). Self-consistency changes the procedure further by sampling multiple reasoning paths and selecting the answer that is most consistent across them, rather than relying on a single decoded chain (Wang et al. 2023b). Least-to-most prompting introduces another decomposition pattern: a complex problem is first broken into simpler subproblems, and later steps build on the answers to earlier ones (Zhou et al. 2023). Together, these

Table 8 Mechanism trajectories in process-level TTI.

Process-level mechanism	Procedure being reorganized	Representative methods	Technical trajectory
L3.A Reasoning decomposition and trace design	Intermediate steps, subproblems, rationales, multimodal traces, reasoning boundaries, supervised reasoning paths, and compressed traces.	CoT, self-consistency, least-to-most prompting, compositional and visual CoT, Chain-of-Spot, RHO-1, LlamaV-o1, Mini-o3, adaptive CoT compression, Boundaries of Thought, LongRePS, and LaV-CoT Wei et al. (2022); Wang et al. (2023b); Zhou et al. (2023); Shao et al. (2024); Thawakar et al. (2025); Mitra et al. (2024); Liu et al. (2024a); Lin et al. (2024); Lai et al. (2025); Huang et al. (2025); Chen et al. (2024c); Zhu et al. (2025); Huang et al. (2026a).	From eliciting intermediate chains toward designing traces that are multi-path, multimodal, bounded, supervised, compressed, and compatible with search or verification.
L3.B Search and candidate exploration	Candidate states, trees, graphs, proof paths, retrieval paths, evidence graphs, visual search states, scorers, pruning rules, and inference budgets.	Tree of Thoughts, Graph of Thoughts, STATe-of-Thoughts, RouteGoT, AlphaMath Almost Zero, ODA, LiteSearch, Paths-over-Graph, ProgRAG, Relink, DeepRAG, DeepSeek-Prover-V2, compute-optimal TTS, PaT, Vision Value Model, and MMhops-R1 Yao et al. (2023b); Besta et al. (2024); Bamberger et al. (2026); Liu et al. (2026); Chen et al. (2024a); Liu et al. (2025a); Sun et al. (2024); Ren et al. (2025); Wang et al. (2025a); Tan et al. (2025a); Guan et al. (2026); Wang et al. (2025e); Zhang et al. (2026a); Park et al. (2026); Huang et al. (2026b); Yoon et al. (2026).	From fixed decoding toward explicit candidate spaces governed by scoring, pruning, task structure, evidence-path exploration, value models, adaptive routing, verification-triggered planning, and budget allocation.

Table 8 Mechanism trajectories in process-level TTI (continued).

Process-level mechanism	Procedure being reorganized	Representative methods	Technical trajectory
L3.C Feedback, verification, and refinement	Critique, revision, self-correction, reranking, verifiers, rewards, process labels, executable checks, and self-generated supervision.	Self-Refine, Reflexion, TTSR, Progressive Rectification Prompting, SLM Self-correct, Fine-Grained Refine, V-STaR, OPEN-RAG, RVR, MLLM reranking, KGR3, mR2AG, R1-style multimodal reasoning, Faithfulness-to-Correctness reward modeling, TTCS, OraPO, Vero, Self-Distillation Zero, and EVE Madaan et al. (2023); Shinn et al. (2023); He et al. (2026c); Hosseini et al. (2024); Heng et al. (2026); Islam et al. (2024); Qian et al. (2026); Huang et al. (2026d); Chen et al. (2024b); Li et al. (2025a); Zhang et al. (2024a); Pan et al. (2026); Tan et al. (2025b); Wang et al. (2025g); Yang et al. (2026); Chen et al. (2025a); Sarch et al. (2026); He et al. (2026b); Zhang et al. (2025b); Lai et al. (2026); Wu et al. (2024b); He et al. (2025); Chen et al. (2026); Ma et al. (2025).	Feedback develops as a set of functional roles: local critique, self-correction, candidate selection, verifier or reward signal, and supervision-like resource for later improvement.

Table 8 Mechanism trajectories in process-level TTI (continued).

Process-level mechanism	Procedure being reorganized	Representative methods	Technical trajectory
L3.D Tool use and agentic action	Tool calls, partial tool execution, code execution, browsing, visual programs, tool routing, multimodal search, environment observations, and action loops.	ReAct, Toolformer, Visual Programming, ViperGPT, GPT-4V web agents, HUSKY, PIVOT, Chain of Tools, AgentOccam, WebWalker, OctoTools, Search-R1, ToolRL, MMSearch-R1, Search-R2, PyVision-RL, VSearcher, Conveyor, Vision Search Assistant, Olympus, CodeDance, StepSearch, SpecEyes, MTA-Agent, and tool-use benchmarks/surveys Yao et al. (2023a); Schick et al. (2023); Surís et al. (2023); Zheng et al. (2024a); Lu et al. (2025b); Jin et al. (2025); Qian et al. (2025b); Kim et al. (2024); Gupta and Kembhavi (2023); Nasiriany et al. (2024); Shi et al. (2025b); Yang et al. (2025a); Wu et al. (2025d); Lu et al. (2025a); Wu et al. (2026a); He et al. (2026a); Zhao et al. (2026); Zhang et al. (2026c); Qu et al. (2025); Xu et al. (2024); Zhang et al. (2024c); Lin et al. (2025); Song et al. (2026); Zheng et al. (2025); Huang et al. (2026c); Peng et al. (2026b).	From isolated external calls toward routed, served, reinforced, multimodal, search-integrated, stateful, benchmarked, and failure-aware action loops.
L3.E Multi-agent and role-based inference	Role assignment, communication protocols, planning/solving/checking separation, specialist routing, sparse collaboration, self-organization, and collaborative verification.	AutoGen, MetaGPT, ChatDev, MAIN-RAG, SymAgent, RadAgents, ToG 3.0, MM-Zero, Mixture-of-Agents, AgentPrune, SMoA, ReSo, and Graph-of-Agents Chang et al. (2025); Liu et al. (2025b); Zhang et al. (2026b); Li et al. (2026c); Wu et al. (2025c, 2024a); Hong et al. (2024); Qian et al. (2024); Wang et al. (2025d); Zhang et al. (2025a); Li et al. (2025c); Zhou et al. (2025); Yun et al. (2026b).	From role prompting toward workflow architectures and communication topologies that coordinate retrieval, tools, memory, critique, routing, cost control, and self-evolution.

methods establish the reasoning trace as a procedural object that can be generated, decomposed, sampled, and aggregated during inference.

The trace-design problem becomes more complex in multimodal and interactive settings. Compositional CoT and Visual CoT extend intermediate reasoning from text-only tasks to visual evidence and multimodal rationales (Shao et al. 2024; Mitra et al. 2024). Chain-of-Spot, LlamaV-o1, and Mini-o3 further place visual grounding, perception steps, interaction turns, or visual search inside the reasoning trajectory (Thawakar et al. 2025; Liu et al. 2024a; Lai et al. 2025). In these settings, the contribution is not simply that additional visual information is provided to the model. The more important procedural change is that visual observations and intermediate operations are organized as parts of the inference trace itself.

Once reasoning traces become explicit, the next question is how to control them. RHO-1 and adaptive CoT compression examine whether reasoning tokens can be selected or compressed while preserving task performance (Lin et al. 2024; Huang et al. 2025). Boundaries of Thought, LongRePS, and LaV-CoT further examine how reasoning traces can be bounded, supervised, or optimized in multimodal settings (Chen et al. 2024c; Zhu et al. 2025; Huang et al. 2026a). The resulting trajectory moves from eliciting intermediate chains toward designing traces that are bounded, multimodal, supervised, compressed, and compatible with later search or verification.

3.3.2 Search and Test-Time Scaling

Once reasoning traces are explicit, search-based inference turns them into a candidate space. Tree of Thoughts generalizes chain-of-thought prompting by treating coherent intermediate units, or "thoughts," as states that can be explored with self-evaluation and backtracking when local choices affect global success (Yao et al. 2023b). Graph of Thoughts extends this procedural idea from trees to graph-structured generated information, where nodes and dependencies support aggregation, transformation, and feedback (Besta et al. 2024). The key mechanism is not simply producing more generations, but creating intermediate states that can be scored, compared, refined, or pruned during inference. Structured variants such as STATe-of-Thoughts replace unconstrained stochastic sampling with interpretable action templates, suggesting that search quality depends on how the candidate space is parameterized (Bamberger et al. 2026).

This candidate-space view becomes especially useful when the task has formal structure, multi-hop dependencies, or externally constrained solution paths. AlphaMath Almost Zero and DeepSeek-Prover-V2 illustrate how mathematical and theorem-proving settings can benefit from search, process supervision, subgoal decomposition, or proof-oriented exploration (Chen et al. 2024a; Ren et al. 2025). ODA, Paths-over-Graph, ProgRAG, and Relink show a related movement in knowledge-intensive reasoning, where the system explores observations, graph paths, progressive retrieval steps, or query-driven evidence graphs rather than relying on a single retrieved context (Sun et al. 2024; Tan et al. 2025a; Park et al. 2026; Huang et al. 2026b). LiteSearch and DeepRAG further connect lightweight tree search with stepwise retrieval decisions, making retrieval itself part of the search procedure (Wang et al. 2025a; Guan et al. 2026).

Search also turns process-level TTI into a resource-allocation problem. Vision Value Model and MMhops-R1 indicate that test-time search is expanding beyond text and formal reasoning into visual comprehension and multimodal multi-hop reasoning (Wang et al. 2025e; Zhang et al. 2026a). Compute-optimal TTS makes the cost side explicit by analyzing how inference-time computation should be allocated across policy models, process reward models, and problem difficulty (Liu et al. 2025a). RouteGoT makes this control problem explicit by routing stronger models to globally coupled planning or synthesis nodes while assigning lighter strategies to localized subtasks under a token budget (Liu et al. 2026). PaT further refines this allocation view in code generation by invoking planning only after verification failure, showing that planning itself can be treated as an adaptive test-time resource rather than a fixed pre-generation step (Yoon et al. 2026). In process-level use, a graph or retrieval structure matters not only as stored information, but as a space over which inference actively explores, scores, prunes, or updates candidate paths.

3.3.3 Feedback, Verification, and Refinement

Once inference produces a draft, candidate, or intermediate result, feedback mechanisms add a procedure for judging and revising it. Self-Refine illustrates the basic pattern: an initial output is generated, feedback is produced by the model itself, and the output is iteratively refined without requiring additional supervised training data or reinforcement learning (Madaan et al. 2023). Reflexion adapts the same procedural idea to language agents by turning feedback into verbal reflections that are stored in episodic memory and reused in later attempts (Shinn et al. 2023). TTSR offers an early example of feedback-to-update migration, where failed reasoning trajectories are converted into teacher-generated variant questions for test-time self-improvement (He et al. 2026c). Progressive Rectification Prompting, SLM Self-correct, and Fine-Grained Refine further show that feedback can be organized as staged rectification, model self-correction, or fine-grained revision signals (Wu et al. 2024b; He et al. 2025; Chen et al. 2026). In these methods, the final output is produced through generation, critique, and revision rather than through a single pass.

Feedback also becomes a selection mechanism when search, retrieval, or multimodal generation produces multiple candidates. V-STaR trains a verifier from both correct and incorrect self-generated solutions and uses it at inference time to select among candidate answers (Hosseini et al. 2024). OPEN-RAG, MLLM reranking, KGR3, and mR2AG apply related ideas to retrieval-augmented or multimodal settings, where feedback is converted into ranking, filtering, reflection, or evidence-quality assessment (Islam et al. 2024; Chen et al. 2024b; Li et al. 2025a; Zhang et al. 2024a). This role is especially important for process-level TTI: once a system generates multiple reasoning paths, retrieved contexts, or candidate outputs, it needs an evaluative procedure for deciding which ones should influence the final answer.

A further extension is to reuse feedback-derived signals beyond the current revision step. R1-VL, Vision-R1, MedVLM-R1, Med-R1, Reason-RFT, and unified multimodal reward models explore reward-shaped reasoning, reinforcement fine-tuning, or reward-model guidance for multimodal and medical reasoning (Huang et al. 2026d; Pan et al. 2026; Tan et al. 2025b; Wang et al. 2025g; Zhang et al. 2025b; Lai et al. 2026). TTCS,

OraPO, Vero, Self-Distillation Zero, and EVE extend this direction through test-time curricula, oracle-educated rewards, scalable reinforcement-learning recipes, dense self-supervision, or executable visual transformations (Heng et al. 2026; Yang et al. 2026; Chen et al. 2025a; Sarch et al. 2026; He et al. 2026b). Faithfulness-to-Correctness highlights a key requirement for this family: critics and reward models must be aligned with correctness rather than merely with plausible surface form (Ma et al. 2025). The mechanism-level point is that feedback can function not only as local critique, but also as candidate selection, verification, or a reusable signal for later improvement.

3.3.4 Tool Use and Agentic Action

Tool-use methods turn inference into a loop in which the model reasons, invokes an external action, observes the result, and updates the next step. ReAct makes this loop explicit by interleaving reasoning traces with task-specific actions and observations from knowledge bases or environments (Yao et al. 2023a). Toolformer follows a complementary route: it trains a model to decide when to call APIs, what arguments to pass, and how to incorporate returned results into subsequent prediction (Schick et al. 2023). Visual Programming and ViperGPT move this procedural idea into visual reasoning by generating executable programs over vision-language modules or Python code, while GPT-4V web-agent work extends action loops to interactive web environments (Surís et al. 2023; Zheng et al. 2024a; Gupta and Kembhavi 2023). In these methods, inference is no longer only generation; it includes acting, observing, and revising the next step.

The mechanism becomes broader as tool use develops from isolated calls into routed, learned, or search-integrated action policies. PIVOT, Chain of Tools, AgentOccam, WebWalker, and OctoTools illustrate tool routing, chained tool use, web traversal, and extensible tool workflows (Nasiriany et al. 2024; Shi et al. 2025b; Yang et al. 2025a; Wu et al. 2025d; Lu et al. 2025a). Search-R1, ToolRL, and Search-R2 connect tool use with reinforcement learning, search engines, and actor-refiner collaboration, while MMSearch-R1, PyVision-RL, and VSearcher extend the same direction to multimodal search, visual tool use, and long-horizon multimodal interaction (Jin et al. 2025; Qian et al. 2025b; Wu et al. 2026a; He et al. 2026a; Zhao et al. 2026; Zhang et al. 2026c). Across these systems, the central question shifts from whether a model can call a tool to how the action policy governing tool choice, argument construction, observation use, and recovery from failure is learned or controlled.

Tool-augmented inference also exposes infrastructure and evaluation problems. Conveyor and Olympus highlight serving, routing, and task-router issues in tool-aware or computer-vision systems, while Vision Search Assistant, CodeDance, StepSearch, SpecEyes, and MTA-Agent extend the trajectory toward multimodal search engines, executable visual reasoning, step-wise search optimization, speculative perception and planning, and deep search agents (Xu et al. 2024; Zhang et al. 2024c; Lin et al. 2025; Song et al. 2026; Zheng et al. 2025; Huang et al. 2026c; Peng et al. 2026b). ToolSandbox and broader tool-learning surveys further show that tool availability, tool choice, argument construction, stateful interaction, and failure recovery require dedicated evaluation (Lu et al. 2025b; Qu et al. 2025). When action traces are stored, summarized, or reused, tool use also becomes a bridge to state-level experience memory;

when actions are distributed across roles, it connects naturally to the coordination mechanisms discussed next.

3.3.5 Multi-Agent and Role-Based Inference

After tool-use methods expand inference through external actions, multi-agent methods further organize inference through communication, division of labor, and role-based workflows. Multi-agent TTI makes coordination itself part of the inference procedure: different roles may plan, retrieve, solve, critique, verify, or integrate partial results before a final output is produced. General frameworks such as AutoGen, MetaGPT, and ChatDev illustrate this shift by organizing conversation, role assignment, communication, and collaborative task execution across interacting agents (Wu et al. 2024a; Hong et al. 2024; Qian et al. 2024). Their process-level contribution is not merely the use of multiple prompts or models, but the workflow that specifies how intermediate decisions and messages move among roles.

This coordination logic becomes more task-specific when multi-agent systems are coupled with retrieval, symbolic reasoning, medical workflows, or multimodal self-improvement. MAIN-RAG uses multi-agent filtering to support retrieval-augmented generation, while SymAgent and ToG 3.0 connect agent coordination with symbolic or graph-based reasoning (Chang et al. 2025; Liu et al. 2025b; Wu et al. 2025c). RadAgents adapts role-based workflows to chest X-ray interpretation, and MM-Zero uses role-like components in a self-evolving multimodal setting (Zhang et al. 2026b; Li et al. 2026c). These systems often combine input-level resources, state-like memories, feedback loops, and tool actions, but their distinctive process-level mechanism is the protocol that determines how subtasks, critiques, and decisions are distributed across agents.

Recent work further shifts attention from adding more agents to controlling the structure of collaboration. Mixture-of-Agents and SMoA explore mixture or sparse collaboration, AgentPrune reduces unnecessary communication, ReSo studies reward-driven self-organization, and Graph-of-Agents represents communication through graph-structured agent interaction (Wang et al. 2025d; Zhang et al. 2025a; Li et al. 2025c; Zhou et al. 2025; Yun et al. 2026b). The resulting trajectory moves from role prompting toward workflow architectures and communication topologies that coordinate retrieval, tools, memory, critique, routing, and verification. In this sense, multi-agent inference serves as an integration point for earlier process mechanisms while introducing coordination as a mechanism in its own right.

3.4 Cross-Locus Integration and Hybrid Systems

The preceding subsections describe input-, state-, and process-level mechanisms separately, but recent TTI systems increasingly connect these loci through reusable inference artifacts. Retrieved evidence can become persistent memory; demonstrations can be selected, compressed, and reused; tool-use episodes can leave experience traces; reasoning paths can be expanded into search spaces; feedback can become a verifier or reward signal; and local adaptation can develop into sustained state management.

Figure 4 places these representative migration trajectories along a temporal axis and highlights the reusable resources formed at the end of each chain.

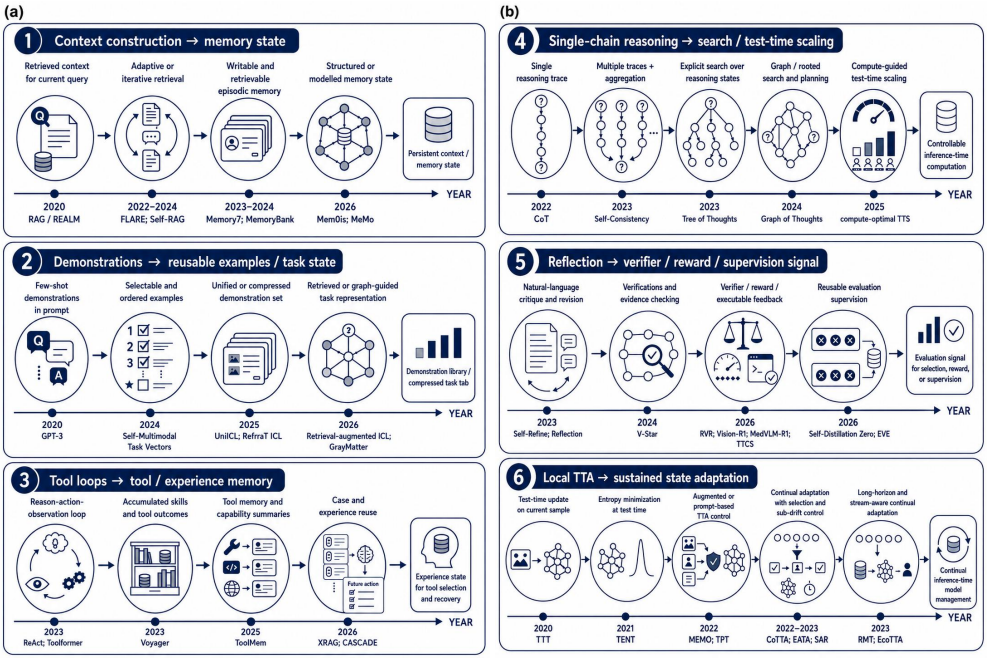


Fig. 4 Representative cross-locus migration trajectories in TTI mechanism development. The figure illustrates how local inference artifacts and operations, including retrieved context, demonstrations, tool-use traces, reasoning paths, feedback signals, and test-time updates, are progressively organized into reusable system resources. Method names are abbreviated labels; full titles are provided in the bibliography.

These trajectories provide a concrete view of the mechanism shifts summarized in Table 9. Across the six chains, the recurring pattern is that an artifact or operation initially tied to a local inference setting becomes available for later selection, storage, retrieval, verification, routing, or stabilization. The table abstracts these cases into migration patterns, the corresponding mechanism shifts, and the system resources formed by the migration.

The migration trajectories in Figure 4 and the patterns summarized in Table 9 show that TTI mechanisms increasingly operate as coupled inference resources. A local mechanism may first appear as a prompting, retrieval, reasoning, feedback, tool-use, or adaptation technique, but its output can later be reused as memory, examples, search candidates, evaluative signals, experience traces, or sustained state. This shift from local intervention to reusable resource provides the basis for the next Section, which examines what bottlenecks these resources relieve and what control requirements they introduce.

Table 9 Representative cross-locus migration patterns in TTI mechanism development.

Migration pattern	Mechanism shift	Resulting system resource	Representative methods
Context construction → memory state	Information initially assembled for the current context becomes writable, retrievable, compressible, updateable, and reusable across episodes.	Persistent context or knowledge state.	RAG, REALM, FLARE, Self-RAG, LongMem, MemoryBank, MemGPT, HippoRAG, LightMem, TokMem, xMemory, Doc-to-LoRA, Latent Context Compilation, Mnemis, and MeMo Lewis et al. (2020); Guu et al. (2020); Jiang et al. (2023a); Asai et al. (2024); Wang et al. (2023a); Zhong et al. (2024); Packer et al. (2023); Gutiérrez et al. (2024); Fang et al. (2026b); Wu et al. (2026b); Hu et al. (2026); Charakorn et al. (2026); Li et al. (2026b); Tang et al. (2026); Quek et al. (2026).
Demonstrations → reusable examples or task state	Examples shift from one-off prompt content to selected, ordered, compressed, or retrieved task-conditioning resources.	Demonstration library, compressed task state, or retrieval target.	GPT-3, visual demonstration selection, Se2, Multimodal Task Vectors, UniICL, Refract ICL, retrieval-augmented ICL, and GraphWalker Brown et al. (2020); Lee et al. (2026); Liu et al. (2024c); Huang et al. (2024b); Gao et al. (2025); Akula et al. (2025); Zhan et al. (2026); Fang et al. (2026a).
Tool loops → tool and experience memory	One-episode actions generate success and failure traces that can be summarized, stored, and routed in later tasks.	Experience state for tool selection, execution planning, and task recovery.	ReAct, Toolformer, Voyager, ToolMem, XSkill, and CASCADE Yao et al. (2023a); Schick et al. (2023); Wang et al. (2023c); Xiao et al. (2025); Jiang et al. (2026); Guo et al. (2026).

Table 9 Representative cross-locus migration patterns in TTI mechanism development (continued).

Migration pattern	Mechanism shift	Resulting system resource	Representative methods
Single-chain reasoning → search and test-time scaling	One reasoning trace becomes a candidate space explored through sampling, branching, scoring, backtracking, and pruning.	Inference-time computation as a controllable resource.	CoT, self-consistency, Tree of Thoughts, Graph of Thoughts, compute-optimal TTS, STATe-of-Thoughts, RouteGoT, and PaT Wei et al. (2022); Wang et al. (2023b); Yao et al. (2023b); Besta et al. (2024); Liu et al. (2025a); Bamberger et al. (2026); Liu et al. (2026); Yoon et al. (2026).
Reflection → verifier, reward, and supervision signal	Natural-language critique becomes a score, verifier, reward, preference signal, executable check, or self-generated supervision signal.	Evaluative signal for selection and future improvement.	Self-Refine, Reflexion, TTSR, RVR, V-STaR, Vision-R1, MedVLM-R1, TTCS, OraPO, Vero, Self-Distillation Zero, and EVE Madaan et al. (2023); Shinn et al. (2023); He et al. (2026c); Qian et al. (2026); Hosseini et al. (2024); Huang et al. (2026d); Pan et al. (2026); Yang et al. (2026); Chen et al. (2025a); Sarch et al. (2026); He et al. (2026b); Heng et al. (2026).
Local TTA → sustained state adaptation	Single-sample or batch-wise updates become stream-aware adaptation with mechanisms for preserving useful state.	Continual inference-time state management.	TTT, TENT, MEMO, TPT, CoTTA, EATA, SAR, RMT, EcoTTA, online TTA surveys, video-stream TTT, GRPO-TTA, and Fair Context Learning Sun et al. (2020); Wang et al. (2021); Zhang et al. (2022); Shu et al. (2022); Wang et al. (2022); Niu et al. (2022, 2023); Döbler et al. (2023); Song et al. (2023); Liang et al. (2025); Wang et al. (2025b,f); Li et al. (2026a); Yun et al. (2026a).

4 Bottleneck Conversion and Resource Control

Section 3 described how test-time mechanisms develop within input, state, and process loci, and how local interventions increasingly migrate into coupled TTI systems. This Section shifts from mechanism development to improvement logic. The central question is no longer only where a test-time method intervenes, but why a migrated mechanism can improve model behavior, what bottleneck it relieves, and what new cost or control requirement it introduces.

Across the systems reviewed above, improvement often comes from making otherwise constrained inference resources more actionable at test time. Some methods use additional computation to explore and select among alternative solutions; others preserve information, interaction history, or tool experience as reusable state; still others turn feedback into signals for revision, ranking, or supervision. The following sections examine these resource shifts through three recurring bottlenecks—capacity, context, and feedback—and analyze how each shift creates both an improvement pathway and a corresponding control problem.

4.1 Capacity and Inference-Time Computation

The capacity bottleneck becomes visible when a trained model is evaluated through a narrow inference procedure. A model may contain latent knowledge or reasoning potential, yet greedy decoding or a single reasoning trace exposes only one limited route through that potential. Recent test-time scaling results make this point concrete: under compute-optimal test-time scaling, a smaller policy model can outperform a much larger model using greedy decoding on selected mathematical reasoning benchmarks (Liu et al. 2025a). This finding does not diminish the role of model scale; rather, it shows that the effective capacity available at test time depends not only on parameters, but also on how the inference procedure generates, explores, and selects candidate solutions.

The first way to relieve this bottleneck is to broaden candidate exposure. Self-consistency does this by sampling multiple reasoning paths and selecting the answer that is most consistent across them, thereby turning single-path instability into aggregation over candidate traces (Wang et al. 2023b). Large Language Monkeys provides broader evidence for the same principle: repeated sampling can increase problem coverage, and in coding or formal-proof settings where outputs can be automatically verified, this increased coverage can translate into better task performance (Brown et al. 2024). At the same time, the same study shows that candidate exposure alone is not sufficient; when reliable verification is unavailable, majority voting or reward-model selection can plateau as the sample budget grows (Brown et al. 2024).

A second route is to organize candidate exploration. Tree of Thoughts and Graph of Thoughts turn implicit reasoning alternatives into explicit candidate spaces that can be explored, scored, aggregated, or pruned (Yao et al. 2023b; Besta et al. 2024). Their relevance here is not the specific tree or graph representation, but the way they convert latent reasoning uncertainty into a search space over which inference-time computation can be allocated. Test-time compute scaling extends this logic from candidate construction to resource allocation: computation must be assigned according to

where it is most likely to change the outcome, rather than expanded uniformly. Scaling LLM Test-Time Compute Optimally shows that the benefit of additional inference compute depends on problem difficulty and allocation strategy, and that compute-optimal allocation can outperform uniform best-of-N sampling (Snell et al. 2025). Recent search-control methods make this point more concrete: STATe-of-Thoughts controls candidate formation through structured action templates, RouteGoT controls graph reasoning cost through node-adaptive routing, and PaT controls planning cost by delaying planning until verification failure (Bamberger et al. 2026; Liu et al. 2026; Yoon et al. 2026). Beyond Chinchilla-Optimal gives the broader resource background for this claim by showing that inference demand can change the optimal trade-off among model size, training data, and cost (Sardana et al. 2024).

Taken together, these studies specify when inference-time computation becomes an improvement resource. Sampling helps when it increases the chance that a correct or useful candidate is produced; search helps when it organizes alternatives into a space that can be evaluated; and test-time scaling helps when budget is allocated according to task difficulty and selector reliability. The corresponding failure mode is equally clear: additional computation becomes an open-ended cost when it produces more samples, tokens, or branches without improving the system's ability to identify better candidates. Capacity bottlenecks are therefore relieved not by computation in general, but by computation that is coupled to candidate diversity, search-space control, budget allocation, and reliable evaluation.

4.2 Context and Reusable State

The context bottleneck appears when information needed for reliable inference is absent from the current prompt, difficult to locate within a long prompt, or useful beyond the current episode. Retrieval-augmented methods such as RAG, REALM, Atlas, Self-RAG, and FLARE address missing or stale information by retrieving, ranking, or actively requesting information for the current query (Lewis et al. 2020; Guu et al. 2020; Asai et al. 2024; Jiang et al. 2023a; Izacard et al. 2023). Long-context and compression methods respond to a related pressure: even when more text can be placed in the prompt, models may not use it uniformly, and the cost of processing long inputs can become substantial. Lost in the Middle shows that language models often use long contexts unevenly, with performance degrading when relevant information appears in the middle of the input (Liu et al. 2024b). LLMLingua and LongLLMLingua show that prompt compression can reduce inference cost, latency, and information overload by retaining more salient content in the current context (Jiang et al. 2023b, 2024). Latent Context Compilation similarly treats compressed context as a portable memory artifact, separating reusable context state from the base model parameters (Li et al. 2026b).

These methods improve the construction of the current inference context, but they do not fully address the persistence problem. They do not determine what should be retained beyond the current episode, when stored information should be revised, or how outdated or conflicting records should be handled. The focus therefore shifts from constructing a better current context to maintaining state that can support later inference.

Memory methods address this persistence problem. LongMem, MemoryBank, and MemGPT move beyond one-episode context by retrieving long past context, storing and updating long-term interaction memories, or managing information across memory tiers (Zhong et al. 2024; Packer et al. 2023; Wang et al. 2023a). Generative Agents, Voyager, and ToolMem extend this idea from stored information to accumulated experience: observations, reflections, executable skills, and tool-use outcomes can influence later planning or tool selection (Wang et al. 2023c; Park et al. 2023; Xiao et al. 2025). MemoRAG illustrates an intermediate form of this transition by using global memory-enhanced retrieval for long-context processing, turning information from an extended context into retrieval clues for later generation (Qian et al. 2025a). Doc-to-LoRA pushes context-to-state migration further by converting a document context into a generated adapter, reducing the need to repeatedly consume the original context during later inference (Charakorn et al. 2026). MeMo represents a model-based variant of reusable state: instead of only retrieving passages, it encodes new knowledge into a dedicated memory model while keeping the base LLM frozen (Quek et al. 2026). The shared movement is from making information available in the current prompt toward maintaining reusable state that can be written, indexed, retrieved, updated, and governed across episodes.

Recent memory systems such as Mnemis make the retrieval policy itself a control surface, not merely the memory store (Tang et al. 2026). Long-term memory evaluation makes the control requirement of this shift explicit. LongMemEval evaluates chat assistants on information extraction, multi-session reasoning, temporal reasoning, knowledge updates, and abstention, showing that sustained interaction memory remains challenging even for commercial assistants and long-context LLMs (Wu et al. 2025b). Surveys of LLM-agent memory similarly emphasize that long-term agent-environment interaction depends on how information is stored, retrieved, and reused (Zhang et al. 2025d). The gain of reusable state is continuity: systems can preserve user preferences, domain facts, tool outcomes, procedural knowledge, and prior mistakes instead of reconstructing them in every prompt. The corresponding risk is that persistence can turn local errors into durable state, making write policies, temporal validation, forgetting mechanisms, source trust, and privacy control central to context-to-state TTI.

4.3 Feedback and Reliable Evaluation

The feedback bottleneck becomes visible when a test-time system can generate outputs, critiques, reasoning paths, retrieved contexts, or tool results, but cannot reliably determine which of them should guide the final answer or future behavior. A concrete form of this bottleneck appears in intrinsic self-correction. Large Language Models Cannot Self-Correct Reasoning Yet shows that asking a model to revise its own reasoning without external feedback often fails to improve performance and can even degrade it (Huang et al. 2024a). The issue is therefore not simply whether the system can produce feedback-like text. The deeper constraint is whether feedback can function as a dependable evaluative signal: it must identify errors, distinguish better candidates from worse ones, or provide targets that remain useful when reused.

The first way to relieve this bottleneck is to make a generated output checkable and revisable. Self-Refine follows this pattern by using model-generated feedback to revise an initial answer, while Reflexion stores verbal reflections from previous attempts so that later trials can avoid repeated failures (Madaan et al. 2023; Shinn et al. 2023). These methods convert an answer from a fixed endpoint into an object that can be examined and improved. Their value, however, depends on whether the critique is accurate enough and specific enough to be incorporated. Feedback helps not because it adds another round of text, but because it supplies a usable judgment about what should change in the current output.

A second route is to make candidate spaces selectable. Sampling, search, retrieval, and multimodal generation can expose many possible answers, reasoning traces, contexts, or tool results, but additional candidates improve inference only when the system can identify which candidates are stronger. V-STaR addresses this problem by training a verifier from both correct and incorrect self-generated answers and using it to select among candidate solutions (Hosseini et al. 2024). Retrieval-augmented and multimodal systems extend the same principle by converting reranking scores, reflection signals, or context-quality assessments into procedures for ranking, filtering, and evidence evaluation (Islam et al. 2024; Chen et al. 2024b; Li et al. 2025a; Zhang et al. 2024a). This connects feedback limits directly to the compute gains discussed in Section 4.1: more candidates are useful only when evaluation can turn candidate diversity into reliable selection. RewardBench and LLM-as-a-Judge bias studies further show that evaluators must themselves be assessed, because reward models and model-based judges can fail under difficult comparisons or carry systematic preferences that distort selection (Lambert et al. 2025; Shi et al. 2025a).

The most consequential conversion occurs when evaluative signals are reused beyond the current decision. In this setting, feedback no longer only revises an answer or selects a candidate; it becomes a supervision-like resource that can shape later reasoning, adaptation, or self-improvement. Reward-shaped multimodal and medical reasoning methods use verifier or reward signals to guide reasoning behavior, while test-time curriculum, oracle-educated reward, self-distillation, and executable-check systems further turn feedback-derived signals into reusable targets for future improvement (Heng et al. 2026; Huang et al. 2026d; Pan et al. 2026; Tan et al. 2025b; Wang et al. 2025g; Yang et al. 2026; Chen et al. 2025a; Sarch et al. 2026; He et al. 2026b; Zhang et al. 2025b; Lai et al. 2026). The gain is that feedback can create explicit selection pressure over answers, trajectories, retrieved contexts, tool results, or later updates. The corresponding risk is that evaluator errors become system-level errors: imperfect evaluators may be optimized, biased judges may reward surface plausibility, and proxy rewards may rise while the intended target declines, as shown in reward-hacking analyses of iterative refinement and inference-time optimization (Pan et al. 2024; Khalaf et al. 2025). Feedback limits are therefore relieved not by adding more feedback, but by turning feedback into evaluative supervision that is calibrated, task-aligned, and constrained against evaluator bias, reward hacking, pseudo-label degradation, and self-confirmation loops.

4.4 Resource-Aware Evaluation Principles

The three bottleneck-relief pathways show that TTI does not eliminate capacity, context, or feedback limitations. Instead, it converts these limitations into resource-control problems over inference-time computation, reusable state, and reliable evaluation. This conversion is valuable only when the introduced resource is controllable. Additional computation is useful when candidate exposure, search spaces, budgets, and stopping rules are controlled. Reusable state is useful when writing, retrieval, updating, forgetting, auditing, and privacy policies are governed. Reliable evaluation is useful when evaluators are calibrated, task-aligned, and resistant to bias or optimization artifacts. The same mechanism that creates a gain can therefore become the next bottleneck when its control surface is weak.

Table 10 summarizes these pathways by relating each bottleneck to its associated improvement resource, control requirement, and dominant failure mode. Rather than ranking the pathways, the table specifies the resource introduced by each pathway, the control problem associated with that resource, and the failure mode that may arise when the control problem is left unaddressed. This framing connects performance reporting with the inference-time resource and control condition under which a TTI gain is obtained.

Table 10: Bottleneck-relief pathways, gains, and control requirements.

Bottleneck-relief pathway	**Primary gain**	**Control requirement**	**Dominant failure mode**
Capacity bottlenecks -> inference-time computation	Improved performance when candidate exposure, search organization, or compute allocation increases the chance of finding and selecting a better solution.	Explicit budget allocation, diverse candidate generation, search-space design, stopping rules, and reliable scoring.	Latency, token cost, search blow-up, verifier dependence, and diminishing returns.
Context bottlenecks -> reusable state	Continuity, personalization, cross-session memory, experience reuse, and lower repeated context-construction cost.	Write, retrieve, compress, update, forget, audit, privacy, and source-trust policies, together with memory-specific evaluation.	Memory pollution, stale state, privacy leakage, persistent retrieval errors, and unexamined personalization drift.

Bottleneck-relief pathway	Primary gain	Control requirement	Dominant failure mode
Feedback bottlenecks -> reliable evaluation	More reliable revision, candidate selection, evaluator-guided reasoning, and feedback-derived supervision for later improvement.	Evaluator calibration, task alignment, process-level checks, reward validation, separation of generation and judging when possible, and task-appropriate verification targets.	Reward hacking, verifier or judge bias, pseudo-label degeneration, self-confirmation loops, and evaluator-task mismatch.

These control requirements suggest several evaluation principles for future TTI systems. First, performance should be reported against strong baselines at explicit inference-time cost, including token budget, number of samples, search depth, update steps, tool calls, latency, or memory overhead when applicable. Second, resource efficiency should be distinguished from simply spending more test-time resources. Third, the introduced resource should be evaluated under disturbances that match its risk surface, such as noisy retrieval, tool errors, heterogeneous streams, distribution shift, repeated updates, or evaluator bias. Fourth, persistent state should be evaluated not only for short-term usefulness, but also for staleness, harmful retention, privacy sensitivity, update reliability, and safety. Fifth, when feedback is used, the verifier, reward model, model-based judge, or execution check should be evaluated as part of the system rather than treated as an oracle.

Viewed in this way, TTI methods can be compared by the bottleneck they relieve, the inference-time resource that supports the gain, and the control mechanism that prevents the resource from becoming a new failure mode. This perspective connects the Locus $\times$ Source map with the mechanism migrations discussed earlier: TTI is not a collection of unrelated inference techniques, but a design space for converting capacity, context, and feedback limitations into controlled inference-time resources. The remaining issue is that these conversions are not always stable. The next subsection summarizes the open contradictions that arise when the expected exchange between resource cost and inference gain becomes difficult to maintain.

4.5 Open Tensions in Resource Conversion

The resource-control view developed above treats TTI gains as exchanges between added test-time resources and improved inference behavior. Such exchanges are useful only when the added resource remains controllable and its cost remains acceptable. Table 11 summarizes four open contradictions that arise when this exchange becomes unstable.

The first two contradictions concern resources that operate within the current inference procedure. Additional computation and evaluative feedback can both improve inference, but only when their control mechanisms remain reliable. Extra

Table 11 Open contradictions in bottleneck-to-resource conversion.

Bottleneck-to-resource conversion	Intended exchange	Open contradiction	Case anchors
Capacity bottleneck → inference-time computation	Spend additional decoding, sampling, search, or verification cost to expose and select better candidates.	The gain from extra computation is conditional on candidate quality, selector reliability, budget allocation, and stopping rules; otherwise, more computation can expand cost or error space rather than improve selection.	Compute-optimal TTS, repeated sampling, and test-time compute allocation studies (Liu et al. 2025a; Brown et al. 2024; Snell et al. 2025).
Feedback bottleneck → evaluative supervision	Use feedback, verifiers, rewards, judges, or executable checks to select, revise, or supervise outputs.	The evaluator that enables improvement can become a proxy optimization target; gains may reflect evaluator exploitation rather than better task behavior.	Self-correction failure, verifier training, judge-bias analysis, and reward-hacking studies (Hosseini et al. 2024; Huang et al. 2024a; Lambert et al. 2025; Shi et al. 2025a; Pan et al. 2024; Khalaf et al. 2025).
Context bottleneck → reusable state	Preserve memory, interaction history, tool outcomes, or failure cases to support later inference.	The persistence that enables reuse can also preserve stale, unsafe, private, or erroneous information unless state governance is explicit.	Long-memory evaluation, memory/experience systems, and memory retrieval-control studies (Zhong et al. 2024; Packer et al. 2023; Wang et al. 2023c; Park et al. 2023; Xiao et al. 2025; Wu et al. 2025b; Tang et al. 2026; Quek et al. 2026).
Test-time resource control → reported improvement	Adapt the inference procedure to task difficulty, stream conditions, evaluator signals, or resource budgets.	Reported gains may depend on protocol choices such as model, prompt, budget, stream order, update steps, retrieval index, verifier, or tool setting, making conclusions difficult to compare across studies.	Online TTA protocol concerns and protocol-sensitive compute/evaluator settings (Wang et al. 2025b; Liu et al. 2025a; Snell et al. 2025; Lambert et al. 2025; Shi et al. 2025a).

Note. The table summarizes contradictions internal to the resource-control analysis in Section 4; case anchors refer to representative works discussed in the preceding subsections.

computation must be connected to useful candidate generation, selection, and stopping, rather than to uncontrolled expansion of samples or search paths (Liu et al. 2025a; Brown et al. 2024; Snell et al. 2025). Similarly, verifier, reward, judge, or execution signals must remain aligned with the intended task, rather than becoming proxy targets for optimization (Hosseini et al. 2024; Huang et al. 2024a; Lambert et al. 2025; Shi et al. 2025a; Pan et al. 2024; Khalaf et al. 2025).

The other two contradictions concern stability across time and experimental settings. Reusable state can support long-horizon interaction, but persistence also makes errors, outdated information, unsafe preferences, or private content more durable (Zhong et al. 2024; Packer et al. 2023; Wang et al. 2023c; Park et al. 2023; Xiao et al. 2025; Wu et al. 2025b; Tang et al. 2026; Quek et al. 2026). Protocol-dependent improvement raises a parallel concern: because TTI changes the inference procedure itself, results may vary with the model, prompt, retrieval index, budget, stream order, update schedule, verifier, or tool environment (Wang et al. 2025b; Liu et al. 2025a; Snell et al. 2025; Lambert et al. 2025; Shi et al. 2025a). These contradictions show that bottleneck-to-resource conversion should be interpreted not only by final task performance, but also by whether the added resource remains cost-aware, evaluator-aligned, memory-governed, and reproducible under its use conditions.

This completes the resource-control analysis and provides the critical basis for the landscape-level analysis that follows.

5 Research Landscape and Opportunities

5.1 From Methods Analysis to the Research Landscape

The preceding Sections have established the analytical basis for viewing test-time improvement (TTI) as an inference-stage research space. The Locus × Source map provides the structural coordinates for positioning test-time interventions, while the mechanism and bottleneck analyses show how local interventions migrate across loci and become associated with new forms of resource control. This progression moves the discussion from individual mechanisms toward the broader organization of the field.

Figure 5 summarizes how the analytical framework developed in Sections 2–4 supports the landscape analysis in this section.

Building on this progression, the present section moves from method-level analysis to landscape-level analysis. It considers how the field is organized when the Locus × Source map, cross-locus mechanism migration, and resource-control perspective are considered jointly. This analysis reveals several research patterns, including configurations that have developed into recognizable research areas, couplings that are becoming increasingly important as mechanisms migrate across loci, and intersections that remain less systematically organized but may support future research when they connect to clear mechanisms, demands for resource control, and evaluable settings. In this way, the Section provides a bridge between the analytical Sections above and the open problems discussed in Section 6. The landscape analysis follows the same positioning logic developed earlier, where density indicates the relative consolidation of recurring mechanisms within the Locus × Source map, while opportunity regions

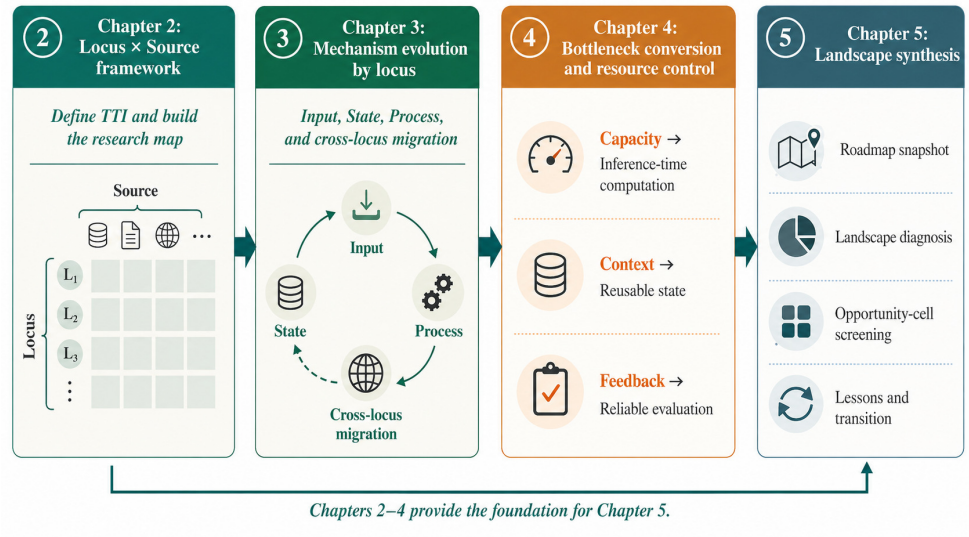


Fig. 5 From locus-based mapping and bottleneck conversion to landscape analysis.

are discussed in relation to mechanism trajectories, resource-control questions, and evaluable settings.

5.2 Temporal Research Trends

A landscape-level analysis of TTI requires a temporal view of how research activity has accumulated across mechanism branches. Figure 6 presents this temporal distribution using the literature collected and reviewed in this survey. Panel (a) presents the relative paper density of representative works across major TTI mechanism branches from 2020 to 2026. Panel (b) further aggregates these branch-level patterns into locus-level relative density trends and summarizes several migration patterns identified from the preceding mechanism analysis. Relative paper density denotes the concentration of representative works reviewed in this survey across mechanism branches and time periods. In this survey, relative paper density refers to ordinal concentration within the screened core-method corpus described in Appendix A, rather than complete publication volume in the field.

Several broad patterns emerge from the branch-level heatmap. In the earlier period, reviewed works are concentrated around prompt-based intervention, retrieval-augmented context construction, test-time adaptation, and explicit reasoning. These branches correspond to early ways of improving model behavior by modifying the input surface, adapting inference-relevant variables, or reorganizing the immediate reasoning procedure. From 2023 to 2024, the distribution becomes more uneven and more specialized. RAG-related work becomes particularly concentrated, while verification, tool use, memory, and search-related process mechanisms become more prominent. This pattern indicates that research activity begins to extend beyond a small set of entry-point techniques toward branches that support retrieval control, candidate evaluation, external action, and reusable state.

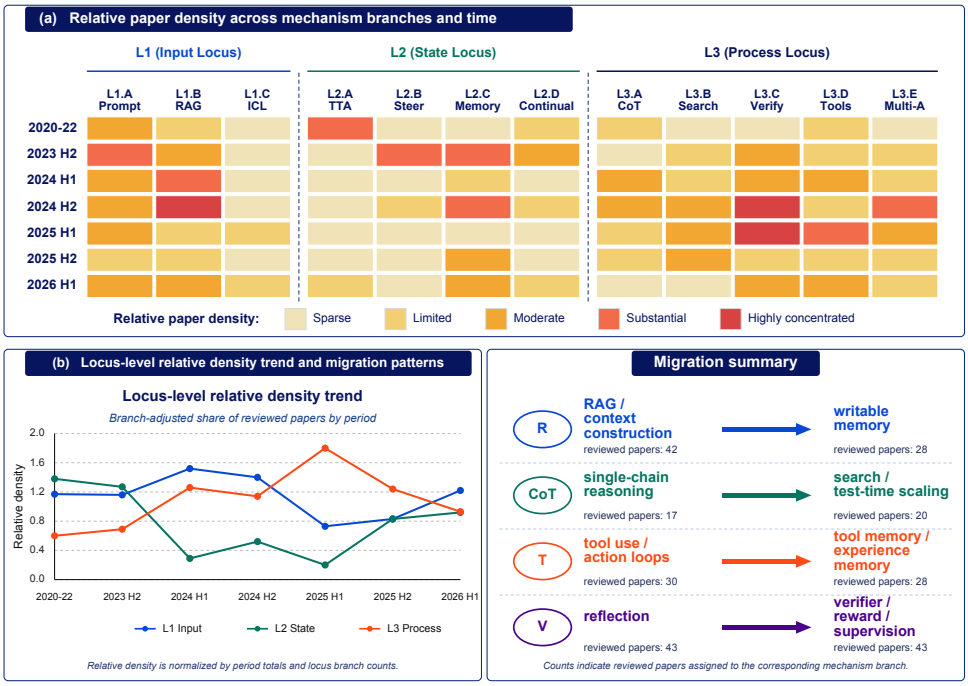


Fig. 6 Temporal shifts in relative paper density and migration patterns across TTI mechanisms, 2020–2026. Density levels are ordinal indicators derived from the screened core-method corpus described in Appendix A.

The later period further changes the temporal pattern. In 2025 and 2026, the heatmap shows stronger concentrations around verification, tools, memory, and multi-agent or agentic process structures. The locus-level trend in Panel (b) is consistent with this branch-level pattern, as state-oriented work rises sharply in the later period, while process-level work remains highly visible across the whole timeline. Input-level work remains present, but its later role is increasingly associated with systems that combine context construction with memory, verification, tool use, or process control. The trend lines therefore summarize how paper density shifts from early input-side and reasoning-oriented foundations toward state- and process-centered configurations.

The migration summary in Panel (b) condenses several movements across loci identified in the preceding sections. RAG and context construction are shown as moving toward writable or reusable memory; single-chain reasoning is linked to search and test-time scaling; tool use and action loops are linked to tool memory or experience memory; and reflection is linked to verifier, reward, or supervision mechanisms. Placed alongside the paper-density trends, these movements summarize how several mechanism branches become connected over time.

5.3 Locus × Source Landscape Patterns

The temporal view in Section 5.2 shows how relative paper density and several mechanism-migration paths have changed across TTI research from 2020 to 2026. When analyzed in relation to the Locus × Source map established in Section 2, these temporal patterns reveal how research activity is organized across the current TTI landscape. Some locus–source combinations have developed into relatively stable research configurations; some gain prominence through cross-locus coupling; and others remain less systematically organized. This section uses the map as an analytical device for landscape analysis to summarize these regional patterns before Section 5.4 narrows the discussion to opportunity cells.

Table 12 summarizes the major landscape patterns on the Locus × Source map.

Table 12: Landscape analysis of TTI research patterns on the Locus × Source map.

Landscape pattern	**Diagnostic basis**	**Representative regions**	**Interpretation**
Established configurations	Visible paper-density accumulation; recurring mechanism forms; clearer evaluation concerns	L1×S1; L2×S3; L3×S0/S4; L3×S2	Regions where intervention locus and information source have formed relatively stable research configurations
Emerging couplings across loci	Rising visibility; connection to cross-locus migration; emerging resource-control problems	L2×S2; L2×S4; multimodal L2/L3×S2/S4	Regions becoming more important as local artifacts become reusable state, feedback signals, or tool-mediated process resources
Intersections with limited systematic organization	Conceptual position is identifiable; mechanism forms, system roles, or evaluation settings remain less settled	L1×S3; L1×S4; L3×S3; L3.E×S3	Regions that can be located on the map but still require clearer mechanism design and evaluation settings
Candidate opportunity cells	Selected from less systematically organized intersections through additional screening	Discussed in Section 5.4	Regions whose research value depends on mechanism linkage, resource-control demand, and evaluability

Note. L1, L2, and L3 refer to the input, state, and process loci; S0–S4 follow the Source axis defined in Section 2. The table is intended as a qualitative analysis grounded in the preceding map, mechanism review, and resource-control analysis.

Established configurations are regions where the relation between intervention locus and information source has become comparatively stable. Representative examples include input-level methods grounded in static knowledge sources, state-level methods driven by test-stream information, and process-level methods organized around internal reasoning, live tools, or verifier-like feedback. These regions are recognizable because they have developed recurring mechanism forms, representative tasks, and evaluation concerns. They function as reference points in the current landscape. Although the methods within each region may differ substantially, the relation between where the intervention acts and where its useful information comes from has become clear enough to support sustained research activity.

Emerging couplings across loci describe regions whose importance increases when migration across loci is considered. They often appear when an artifact produced in one part of inference becomes a reusable or controllable resource elsewhere. Context construction may develop into writable or retrievable memory; tool-use traces may become experience memory; feedback may become a verifier, reward, or supervision-like signal; and multimodal reasoning may connect input formalization with tool use and process control. These couplings show that TTI research is not only organized within individual cells of the map, but also along the connections between loci and sources, where local mechanisms begin to support broader inference architectures.

Intersections with limited systematic organization refer to regions that can be located on the map but whose mechanism forms, system roles, or evaluation settings remain less settled. Examples include test-stream-conditioned input construction, feedback-conditioned input organization, test-stream-aware process control, and multi-agent inference under changing test distributions. Recent methods such as RVR and Fair Context Learning illustrate how these less consolidated regions can begin to acquire mechanism-level form through verifier-conditioned retrieval or evidence-balanced test-time context adaptation (Qian et al. 2026; Yun et al. 2026a). Analytically, these regions indicate where the current map is not yet fully organized by stable mechanisms and evaluation practices. Whether such intersections should be treated as opportunity cells depends on a further screening step, discussed in the next section.

5.4 Evaluable Opportunity Areas

Section 5.3 identified several regional patterns in the Locus $\times$ Source landscape, including established configurations, emerging couplings across loci, and less systematically organized intersections. The analysis then determines which regions can be treated as opportunity cells rather than merely as descriptive positions on the map. In this survey, candidate cells are suggested by two types of landscape signals. The first is an activity signal, where the paper-density pattern indicates that a mechanism branch or related landscape region is gaining prominence. The second is a structural gap, where paper density remains comparatively weak but the region is structurally plausible given the mechanism migrations and resource-control problems identified in the preceding Sections. These two signals provide different reasons for considering a region. One reflects emerging research activity, while the other reflects an under-organized but theoretically coherent position in the landscape.

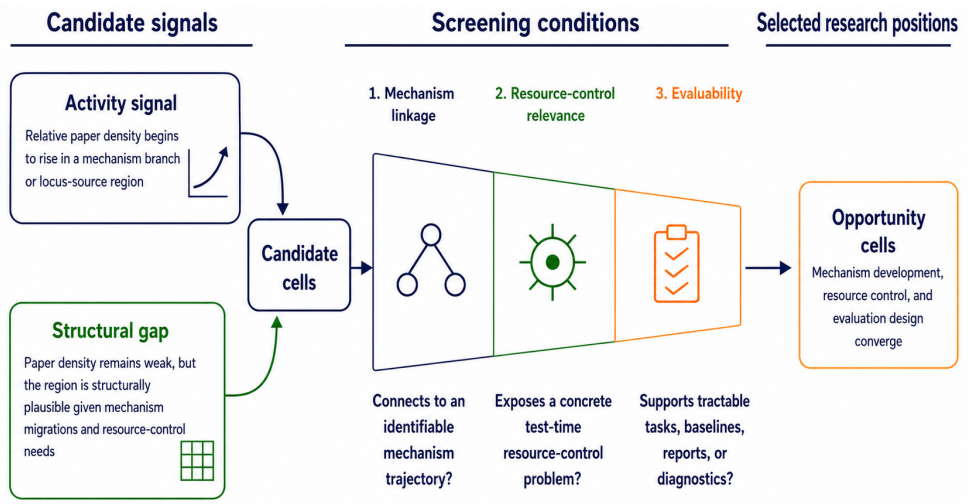


Fig. 7 From landscape signals to opportunity cells. Candidate-cell screening is supported by the mechanism-linkage, resource-control, and evaluability protocol described in Appendix A.

As summarized in Figure 7, candidate cells are treated as meaningful opportunity cells only when they satisfy three screening conditions. First, mechanism linkage assesses whether the cell can be connected to an identifiable mechanism trajectory, such as the migration from context construction to memory state or from reflection to verifier and reward signals. Second, resource-control relevance evaluates whether the cell exposes a concrete test-time control problem, including storage, retrieval, updating, selection, verification, or budget allocation. Third, evaluability assesses whether the cell can be studied through tractable tasks, baselines, cost reports, stability analyses, stream protocols, or evaluator diagnostics. Together, these conditions identify opportunity cells as research positions where mechanism development, resource control, and evaluation design converge. The screening procedure and its relation to the coded literature corpus are detailed in Appendix A.

Table 13 provides representative examples of this screening logic.

These examples show that opportunity cells should be understood as analytically selected regions rather than simple extensions of less-developed parts of the map. An activity signal may indicate that a region is beginning to attract research attention, while a structural gap may indicate that a plausible locus–source combination has not yet been systematically organized. In both cases, the region becomes more convincing when it can be connected to a mechanism trajectory, framed as a resource-control problem, and evaluated through concrete tasks or protocols.

Table 13 Representative opportunity-cell screening examples.

Candidate signal	**Candidate cell (Locus × Source)**	**Mechanism linkage (example trajectory)**	**Resource-control relevance (key control problem)**	**Evaluation handle (examples)**
Emerging activity signal	L2 × S2 Tool / experience memory	Tool-use traces, deployment cases, and interaction experience become reusable state, as in CASCADE (Guo et al. 2026)	What tool or deployment experience should be stored, updated, retrieved, or forgotten?	Long-horizon tool-use tasks; case-memory ablations; tool-selection and experience-reuse accuracy
Emerging activity signal	L2 × S4 Feedback-driven state update	Feedback, verifier, reward, or reflection signals guide adaptive state, as in GRPO-TTA (Li et al. 2026a) and TTSR (He et al. 2026c)	How to control evaluator error, pseudo-label drift, reward hacking, and update reliability?	Verifier diagnostics; held-out evaluation; reward and stream stability tests
Structural gap	L3 × S3 Test-stream-aware process control	Test-stream or reliability signals guide search, selection, retrieval, or interaction policies	How to allocate budget, reset process state, verify intermediate results, and assess sample reliability under changing streams?	Online or continual benchmarks; cost-aware reporting; stream protocols

Note. Candidate cells are representative examples selected to illustrate the screening logic.

5.5 Landscape Lessons and Research Implications

The preceding sections have used density shifts in the reviewed corpus, the Locus × Source landscape, and opportunity-cell screening to analyze how TTI research is currently organized. Table 14 summarizes the main lessons from this analysis and links each lesson to its implication for future TTI research.

Table 14: Lessons from the landscape analysis of TTI research

Lesson	**Core message**	**Implication for TTI research**
Growth is clustered and coupled	Research activity gathers around recurring configurations and migration paths rather than expanding uniformly.	Analyze TTI through mechanism combinations and system couplings, not only isolated method families.
Input remains foundational	Input construction remains the main surface through which tasks, evidence, examples, and multimodal information enter inference.	Study input-level methods together with retrieval control, feedback, memory, and process organization.

Lesson	Core message	Implication for TTI research
State is the long-horizon frontier	Memory, tool experience, adaptation, and continual stabilization extend TTI beyond a single inference episode.	Develop policies for writing, updating, retrieving, trusting, and forgetting persistent resources.
Process needs control	Reasoning, search, verification, tools, and agents improve inference only when computation is organized and controlled.	Report budget, selection, verification, coordination, and stopping conditions.
Opportunity requires evaluability	Candidate cells become useful research opportunities when they support clear tasks, baselines, cost reports, and stability or reliability diagnostics.	Prioritize methodologically tractable directions rather than map gaps alone.

Note. The lessons summarize the landscape analysis developed in Sections 5.2–5.4.

Collectively, these lessons suggest that TTI research is moving from isolated test-time techniques toward coupled inference systems. Input construction remains a central entry point, but it increasingly interacts with retrieval control, memory, feedback, and process organization. State-level mechanisms extend improvement beyond a single inference episode, making persistence and update control central to long-horizon TTI. Process-level methods organize additional computation through reasoning, search, verification, tool use, and coordination, but their value depends on whether the added computation can be controlled. This analysis also explains why the open problems discussed next arise naturally from the landscape itself. As TTI methods become more persistent, evaluative, tool-mediated, and process-oriented, their central challenges increasingly concern how inference-time resources can be controlled, evaluated, and stabilized.

6 Discussion and Conclusion

6.1 Limitations

This survey provides a research map and mechanism-level analysis of test-time improvement rather than a unified empirical evaluation of all TTI methods. Its conclusions should therefore be interpreted as framework-level findings about how inference-stage interventions are positioned, connected, and controlled. This orientation helps organize a rapidly expanding field, but it also defines the scope within which the proposed map, mechanism analysis, bottleneck conversion, and landscape analysis should be interpreted.

Based on this orientation, the limitations of this survey fall into the following categories.

Literature coverage and interpretation of density indicators. The reviewed literature is selected to support an analysis of recurring TTI mechanisms rather than a comprehensive bibliometric analysis. As a result, the paper-density shifts in Section 5 should be interpreted as qualitative signals of research activity and mechanism migration, not as complete publication statistics for the entire TTI field. Appendix A makes the search, screening, coding, and density-construction protocol explicit; nevertheless, the resulting heatmaps and density trends remain qualitative analyses shaped by the protocol rather than comprehensive bibliometric measurements.

Granularity of the analytical framework. The Locus × Source map and the Temporal / Cost descriptors operate at a deliberately abstract level. This abstraction makes it possible to compare prompting, retrieval, adaptation, memory, verification, tool use, and agentic systems within a common framework, but it cannot fully capture implementation-level differences such as latency, energy use, API cost, storage overhead, privacy risk, human supervision, or long-term system maintenance.

Lack of unified empirical comparison. This survey compares TTI methods primarily by intervention locus, information source, mechanism trajectory, and resource-control requirement. It does not reimplement methods under a unified experimental protocol, nor does it provide a performance ranking across tasks, datasets, or model families. Such benchmark-level comparison remains necessary for assessing the empirical strength of specific methods.

6.2 Future Directions

This survey has argued that test-time improvement is best understood as a research space centered on inference-stage resource control. Across prompting, retrieval, adaptation, reasoning, memory, verification, tool use, and agentic action, the central question is how additional information, computation, feedback, state, and external actions can be introduced at inference time in ways that remain useful, controllable, and evaluable. From this perspective, TTI is no longer a collection of isolated test-time techniques, but an emerging design space for organizing how trained models use resources available during inference.

Looking forward, several directions deserve further systematic study.

Stable gains from additional test-time resources. Extra sampling, search, tool calls, state updates, and memory retrieval can improve inference, but they can also increase cost without changing the final outcome. **Open problem:** Future work should clarify when additional inference-time resources are likely to produce genuine task improvement, when they should be stopped, and how they should be reallocated across samples, candidates, tools, updates, or memory accesses. **Evaluation handle:** A useful evaluation protocol would compare fixed-budget and adaptive-budget systems under matched task settings, while reporting tokens, samples, search depth, tool calls, update steps, latency, memory access, and marginal performance gain. Strong baselines should include uniform best-of-N sampling, fixed-depth search, fixed retrieval, and fixed update schedules, so that improvements can be attributed to resource allocation rather than greater resource spending alone.

Reliable feedback in the absence of ground-truth supervision. Verifiers, reward models, model-based judges, execution checks, and self-generated critiques

are increasingly used to select, revise, or supervise test-time behavior. **Open problem:** Future work should develop calibration and robustness metrics for evaluators whose feedback is reused across iterative refinement, candidate selection, adaptation, or self-improvement, especially when the evaluator itself can become a proxy optimization target. **Evaluation handle:** Evaluation should track verifier–task alignment over multiple refinement rounds on held-out benchmarks, compare evaluator scores with task correctness or executable outcomes, and stress-test systems under biased, noisy, or underspecified feedback. Such protocols would help distinguish genuine task improvement from improvement only with respect to the evaluator.

Governance of persistent state and experience. Memory, tool experience, and continual adaptation extend TTI beyond a single inference episode, but they also make local information durable across later interactions or tasks. **Open problem:** Future work should design write, update, retrieve, forget, and audit policies that preserve useful memory or experience while preventing stale, private, unsafe, or erroneous information from becoming persistent system state. **Evaluation handle:** A practical benchmark design should include long-horizon memory and agent tasks with controlled knowledge updates, conflicting facts, deletion requests, privacy-sensitive records, and tool-failure traces. Evaluation should report not only downstream task performance, but also memory usefulness, staleness, harmful retention, forgetting reliability, privacy sensitivity, and downstream error propagation after memory retrieval or state reuse.

Control of dynamic inference processes. Tool-augmented, multi-agent, and stream-aware systems organize inference through search, tool use, verification loops, communication, reset strategies, and stopping rules. **Open problem:** Future work should determine how these process-control mechanisms should change when tasks, environments, tools, or test streams are non-stationary, and how intermediate errors can be prevented from propagating through later tool calls, agent messages, search branches, or repeated updates. **Evaluation handle:** Online or continual evaluation settings should vary stream order, tool availability, API reliability, task distributions, and resource budgets. In addition to task performance, evaluation should report failure recovery, communication overhead, tool-call efficiency, verifier intervention rate, reset behavior, and stopping-rule reliability.

Multimodal and embodied extensions of TTI. As TTI extends to visual, interactive, and embodied settings, inference-time resources become more heterogeneous and harder to control. **Open problem:** Future work should align visual evidence, language reasoning, environment feedback, executable verification, and cross-modal reward signals within the same inference procedure, while distinguishing failures caused by perception, grounding, action execution, environment interaction, or evaluator mismatch. **Evaluation handle:** Multimodal and embodied benchmarks should jointly report answer accuracy, grounding consistency, execution success, environment-interaction cost, reward–task alignment, and robustness under visual, textual, or environmental perturbations. Such evaluation should clarify whether a test-time intervention improves multimodal reasoning itself, tool-mediated perception, action reliability, or only the final textual response.

The next stage of TTI research will depend less on increasing the number of prompts, retrieved documents, samples, tools, memories, or agents than on whether

these inference-time resources can be organized through explicit control policies, evaluated through concrete protocols, and stabilized across tasks, time, modalities, and environments. Accordingly, TTI provides a lens for studying how trained models can continue to improve after training through principled and controlled use of inference-time resources.

Declarations

- **Funding:** This research was supported by the National Natural Science Foundation of China (Nos. 62576046, 62301066, and 62406028), the Beijing Academy of Artificial Intelligence (No. Z251100008125041), the Key Project of Philosophy and Social Sciences Research, Ministry of Education, China (No. 24JZD040), the Graduate Education and Teaching Reform Research Project of Beijing University of Posts and Telecommunications (No. 2025YZ010), and the Beijing Natural Science Foundation (No. L259043).

- **Conflict of interest:** The authors declare that they have no conflict of interest.
- **Data availability:** The coded literature screening workbook, along with the count matrices used to generate the heatmap and temporal-density analyses, can be provided upon reasonable request from the corresponding author. No new experimental datasets were generated in this study.
- **Author contribution:** Yiming Tang, Ziyu Liu, and Zihan Chen drafted the main manuscript text. Jinghan Yang contributed to the discussion and revision of the manuscript. Liuyu Xiang, Zhenbo Xu, and Zhaofeng He supervised the study and contributed to the discussion and revision of the manuscript. All authors reviewed and approved the final manuscript.

References

Akula, A.R., Hashimoto, K., Srinivasan, K., Chaudhary, A., Raman, K., Bendersky, M.: Refract ICL: Rethinking example selection in the era of million-token models. arXiv preprint arXiv:2506.12346 (2025) https://doi.org/10.48550/arXiv.2506.12346

Alomrani, M.A., Zhang, Y., Li, D., Sun, Q., Pal, S., Zhang, Z., Hu, Y., Ajwani, R.D., Valkanas, A., Karimi, R., Cheng, P., Wang, Y., Liao, P., Huang, H., Wang, B., Hao, J., Coates, M.: Reasoning on a budget: A survey of adaptive and controllable test-time compute in LLMs. arXiv preprint arXiv:2507.02076 (2025) https://doi.org/10.48550/arXiv.2507.02076

Asai, A., Wu, Z., Wang, Y., Sil, A., Hajishirzi, H.: Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In: International Conference on Learning Representations (2024)

Azaria, A., Mitchell, T.: The internal state of an LLM knows when it's lying. In: Findings of the Association for Computational Linguistics: EMNLP 2023, pp. 967–976 (2023)

Baldassini, F.B., Shukor, M., Cord, M., Soulier, L., Piwowarski, B.: What makes multimodal in-context learning work? In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 1539–1550 (2024)

Bamberger, Z., Saenger, T.R., Morad, G., Amir, O., Stewart, B.M., Feder, A.: STATe-of-Thoughts: Structured action templates for tree-of-thoughts. arXiv preprint arXiv:2602.14265 (2026) https://doi.org/10.48550/arXiv.2602.14265

Bao, W., Zhao, Y., Yang, X., He, J.: Ramen: Robust test-time adaptation of vision-language models with active sample selection. arXiv preprint arXiv:2604.21728 (2026) https://doi.org/10.48550/arXiv.2604.21728

Bayat, F.F., Liu, X., Jagadish, H., Wang, L.: Enhanced language model truthfulness with learnable intervention and uncertainty expression. In: Findings of the Association for Computational Linguistics: ACL 2024, pp. 12388–12400 (2024)

Besta, M., Blach, N., Kubicek, A., Gerstenberger, R., Podstawski, M., Gianinazzi, L., Gajda, J., Lehmann, T., Niewiadomski, H., Nyczyk, P.: Graph of thoughts: Solving elaborate problems with large language models. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 38, pp. 17682–17690 (2024)

Brown, B., Juravsky, J., Ehrlich, R., Clark, R., Le, Q.V., Ré, C., Mirhoseini, A.: Large language monkeys: Scaling inference compute with repeated sampling. arXiv preprint arXiv:2407.21787 (2024) https://doi.org/10.48550/arXiv.2407.21787

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A.: Language models are few-shot learners. Advances in Neural Information Processing Systems **33**, 1877–1901 (2020)

Bsharat, S.M., Myrzakhan, A., Shen, Z.: Principled instructions are all you need for questioning LLaMA-1/2, GPT-3.5/4. arXiv preprint arXiv:2312.16171 (2023) https://doi.org/10.48550/arXiv.2312.16171

Chang, C.-Y., Jiang, Z., Rakesh, V., Pan, M., Yeh, C.-C.M., Wang, G., Hu, M., Xu, Z., Zheng, Y., Das, M.: Main-RAG: Multi-agent filtering retrieval-augmented generation. In: Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 2607–2622 (2025)

Charakorn, R., Cetin, E., Uesaka, S., Lange, R.T.: Doc-to-LoRA: Learning to instantly internalize contexts. arXiv preprint arXiv:2602.15902 (2026) https://doi.org/10.48550/arXiv.2602.15902

Chen, D., Wang, D., Darrell, T., Ebrahimi, S.: Contrastive test-time adaptation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 295–305 (2022)

Chen, G., Liao, M., Li, C., Fan, K.: Alphamath almost zero: process supervision

without process. Advances in Neural Information Processing Systems **37**, 27689–27724 (2024)

Chen, J., Wu, H., Pang, J., Wang, Y., Zhang, D., Sun, C.: Tool learning with language models: A comprehensive survey of methods, pipelines, and benchmarks. Vicinagearth **2**(16) (2025) https://doi.org/10.1007/s44336-025-00024-x

Chen, Q., Qin, L., Wang, J., Zhou, J., Che, W.: Unlocking the capabilities of thought: A reasoning boundary framework to quantify and optimize chain-of-thought. Advances in Neural Information Processing Systems **37**, 54872–54904 (2024)

Chen, X., Gan, Y., Purver, M.: Fine-Refine: Iterative fine-grained refinement for mitigating dialogue hallucination. arXiv preprint arXiv:2602.15509 (2026) https://doi.org/10.48550/arXiv.2602.15509

Chen, Z., Xu, C., Qi, Y., Guo, J.: MLLM is a strong reranker: Advancing multimodal retrieval-augmented generation via knowledge-enhanced reranking and noise-injected training. arXiv preprint arXiv:2407.21439 (2024) https://doi.org/10.48550/arXiv.2407.21439

Chen, Z., Yu, H., Xu, Y., Luo, Y., Duong, L., Li, Y.-F.: OraPO: Oracle-educated reinforcement learning for data-efficient and factual radiology report generation. arXiv preprint arXiv:2509.18600 (2025) https://doi.org/10.48550/arXiv.2509.18600

Chuang, Y.-S., Xie, Y., Luo, H., Kim, Y., Glass, J.R., He, P.: Dola: Decoding by contrasting layers improves factuality in large language models. In: International Conference on Learning Representations (2024)

Dathathri, S., Madotto, A., Lan, J., Hung, J., Frank, E., Molino, P., Yosinski, J., Liu, R.: Plug and play language models: A simple approach to controlled text generation. In: International Conference on Learning Representations (2020). https://openreview.net/forum?id=H1edEyBKDS

Döbler, M., Marsden, R.A., Yang, B.: Robust mean teacher for continual and gradual test-time adaptation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7704–7714 (2023)

Fan, W., Ding, Y., Ning, L., Wang, S., Li, H., Yin, D., Chua, T.-S., Li, Q.: A survey on RAG meeting LLMs: Towards retrieval-augmented large language models. In: Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 6491–6501. Association for Computing Machinery, Barcelona, Spain (2024). https://doi.org/10.1145/3637528.3671470

Fang, J., Deng, X., Xu, H., Jiang, Z., Tang, Y., Xu, Z., Deng, S., Yao, Y., Wang,

M., Qiao, S., Chen, H., Zhang, N.: LightMem: Lightweight and efficient memory-augmented generation. In: The Fourteenth International Conference on Learning Representations (2026). https://openreview.net/forum?id=dyJ0GWpjJB

Fang, L., Wang, Y., Gatmiry, K., Fang, L., Wang, Y.: Rethinking invariance in in-context learning. In: The Thirteenth International Conference on Learning Representations (2025). https://openreview.net/forum?id=q1UyoY3MgJ

Fang, Y., Liao, W., Guo, Y., Gao, J., Ding, H., Zhang, J., Jiang, X., Yang, Z., Zhao, J., Wang, Y., Ma, L.: GraphWalker: Graph-guided in-context learning for clinical reasoning on electronic health records. arXiv preprint arXiv:2604.06684 (2026) https://doi.org/10.48550/arXiv.2604.06684

Gao, J., Lv, Q., Wang, Z., Wu, T., Cao, Z., Li, W.: UniICL: An efficient unified framework unifying compression, selection, and generation. In: Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (2025). https://doi.org/10.18653/v1/2025.acl-long.24

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., Wang, H.: Retrieval-augmented generation for large language models: A survey. arXiv preprint arXiv:2312.10997 (2024) https://doi.org/10.48550/arXiv.2312.10997

Guan, X., Zeng, J., Meng, F., Xin, C., Lu, Y., Lin, H., Han, X., Sun, L., Zhou, J.: DeepRAG: Thinking to retrieve step by step for large language models. In: The Fourteenth International Conference on Learning Representations (2026). https://openreview.net/forum?id=VI2YaggHIF

Guo, S., Du, Y., Chen, H., Chang, Y., Wang, J.: CASCADE: Case-based continual adaptation for large language models during deployment. arXiv preprint arXiv:2605.06702 (2026) https://doi.org/10.48550/arXiv.2605.06702

Gupta, T., Kembhavi, A.: Visual programming: Compositional visual reasoning without training. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 14953–14962 (2023)

Gutiérrez, B.J., Shu, Y., Gu, Y., Yasunaga, M., Su, Y.: Hipporag: Neurobiologically inspired long-term memory for large language models. Advances in Neural Information Processing Systems **37**, 59532–59569 (2024)

Guu, K., Lee, K., Tung, Z., Pasupat, P., Chang, M.: Retrieval augmented language model pre-training. In: Proceedings of the 37th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 119, pp. 3929–3938. Virtual Event (2020)

Hardt, M., Sun, Y.: Test-time training on nearest neighbors for large language models. In: International Conference on Learning Representations (2024)

He, B., Hu, M., Xu, Z., Wang, H., Zong, L., Chen, Y., Ma, C., Liu, X., Zhou, P., King, I.: Search-R2: Enhancing search-integrated reasoning via actor-refiner collaboration. arXiv preprint arXiv:2602.03647 (2026) https://doi.org/10.48550/arXiv.2602.03647

He, H., Rong, Z., Zhao, L., Zhao, Y., Yang, L., Zhang, H.: TTSR: Test-time self-reflection for continual reasoning improvement. arXiv preprint arXiv:2603.03297 (2026) https://doi.org/10.48550/arXiv.2603.03297

He, J., Lin, H., Wang, Q., Fung, Y.R., Ji, H.: Self-correction is more than refinement: A learning framework for visual and language reasoning tasks. In: Findings of the Association for Computational Linguistics: ACL 2025, pp. 6405–6421 (2025)

He, Y., Kaur, S., Bhaskar, A., Yang, Y., Liu, J., Ri, N., Fowl, L., Panigrahi, A., Chen, D., Arora, S.: Self-distillation zero: Self-revision turns binary rewards into dense supervision. arXiv preprint arXiv:2604.12002 (2026) https://doi.org/10.48550/arXiv.2604.12002

Heng, Y., Jiang, C., Yang, H., Zhang, S., Ye, W.: EVE: Verifiable self-evolution of MLLMs via executable visual transformations. arXiv preprint arXiv:2604.18320 (2026) https://doi.org/10.48550/arXiv.2604.18320

Hong, S., Zhuge, M., Chen, J., Zheng, X., Cheng, Y., Wang, J., Zhang, C., Yau, S., Lin, Z., Zhou, L.: MetaGPT: Meta programming for a multi-agent collaborative framework. In: International Conference on Learning Representations (2024)

Hosseini, A., Yuan, X., Malkin, N., Courville, A., Sordoni, A., Agarwal, R.: V-STaR: Training verifiers for self-taught reasoners. In: First Conference on Language Modeling (2024). https://openreview.net/forum?id=stmqBSW2dV

Hu, C., Fu, J., Du, C., Luo, S., Zhao, J., Zhao, H.: ChatDB: Augmenting LLMs with databases as their symbolic memory. arXiv preprint arXiv:2306.03901 (2023) https://doi.org/10.48550/arXiv.2306.03901

Hu, Z., Zhu, Q., Yan, H., He, Y., Gui, L.: Beyond RAG for agent memory: Retrieval by decoupling and aggregation. arXiv preprint arXiv:2602.02007 (2026) https://doi.org/10.48550/arXiv.2602.02007

Huang, B., Mitra, C., Arbelle, A., Karlinsky, L., Darrell, T., Herzig, R.: Multimodal task vectors enable many-shot multimodal in-context learning. Advances in Neural Information Processing Systems **37**, 22124–22153 (2024)

Huang, H., Huang, J., Wan, Z., Zheng, X., Ji, R., Luo, J.: SpecEyes: Accelerating agentic multimodal LLMs via speculative perception and planning. arXiv preprint arXiv:2603.23483 (2026) https://doi.org/10.48550/arXiv.2603.23483

Huang, J., Chen, X., Mishra, S., Zheng, H.S., Yu, A., Song, X., Zhou, D.: Large

language models cannot self-correct reasoning yet. In: International Conference on Learning Representations (2024)

Huang, J., Tan, Z., Gong, S., Zeng, F., Zhou, J.T., Miao, C., Tan, H., Yao, W., Li, J.: LaV-CoT: Language-aware visual CoT with multi-aspect reward optimization for multilingual text-centric VQA. In: Proceedings of the ACM Web Conference 2026, pp. 8262–8273 (2026). https://doi.org/10.1145/3774904.3792793

Huang, K., Liu, S., Hu, X., Xu, T., Bao, L., Xia, X.: Reasoning efficiently through adaptive chain-of-thought compression: A self-optimizing framework. arXiv preprint arXiv:2509.14093 (2025) https://doi.org/10.48550/arXiv.2509.14093

Huang, M., Bu, C., He, Y., Zhuo, X., Wu, X.: Relink: Constructing query-driven evidence graph on-the-fly for GraphRAG. Proceedings of the AAAI Conference on Artificial Intelligence **40**(37), 31202–31210 (2026) https://doi.org/10.1609/aaai.v40i37.40382

Huang, W., Jia, B., Zhai, Z., Cao, S., Ye, Z., Zhao, F., Yao, H., Lin, S.: Vision-R1: Incentivizing reasoning capability in multimodal large language models. In: The Fourteenth International Conference on Learning Representations (2026). https://openreview.net/forum?id=UZIjskfbfU

Huang, X., Liu, W., Chen, X., Wang, X., Wang, H., Lian, D., Wang, Y., Tang, R., Chen, E.: Understanding the planning of LLM agents: A survey. arXiv preprint arXiv:2402.02716 (2024)

Islam, S.B., Rahman, M.A., Hossain, K.T., Hoque, E., Joty, S., Parvez, M.R.: Open-RAG: Enhanced retrieval augmented reasoning with open-source large language models. In: Findings of the Association for Computational Linguistics: EMNLP 2024, pp. 14231–14244 (2024)

Iwasawa, Y., Matsuo, Y.: Test-time classifier adjustment module for model-agnostic domain generalization. Advances in Neural Information Processing Systems **34**, 2427–2440 (2021)

Izacard, G., Lewis, P., Lomeli, M., Hosseini, L., Petroni, F., Schick, T., Dwivedi-Yu, J., Joulin, A., Riedel, S., Grave, E.: Atlas: Few-shot learning with retrieval augmented language models. Journal of Machine Learning Research **24**(251), 1–43 (2023)

Jiang, G., Su, Z., Qu, X., Fung, Y.R.: XSkill: Continual learning from experience and skills in multimodal agents. arXiv preprint arXiv:2603.12056 (2026) https://doi.org/10.48550/arXiv.2603.12056

Jiang, H., Wu, Q., Lin, C.-Y., Yang, Y., Qiu, L.: Llmlingua: Compressing prompts for accelerated inference of large language models. In: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pp. 13358–13376 (2023)

Jiang, H., Wu, Q., Luo, X., Li, D., Lin, C.-Y., Yang, Y., Qiu, L.: LongLLMLingua: Accelerating and enhancing LLMs in long context scenarios via prompt compression. In: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 1658–1677 (2024)

Jiang, Z., Xu, F.F., Gao, L., Sun, Z., Liu, Q., Dwivedi-Yu, J., Yang, Y., Callan, J., Neubig, G.: Active retrieval augmented generation. In: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pp. 7969–7992 (2023)

Jin, B., Zeng, H., Yue, Z., Yoon, J., Arik, S.O., Wang, D., Zamani, H., Han, J.: Search-R1: Training LLMs to reason and leverage search engines with reinforcement learning. In: Second Conference on Language Modeling (2025). https://openreview.net/forum?id=Rwhi91ideu

Khalaf, H., Mayrink Verdun, C., Oesterling, A., Lakkaraju, H., Calmon, F.d.P.: Inference-time reward hacking in large language models. Advances in Neural Information Processing Systems **38**, 61720–61760 (2025)

Kim, J., Paranjape, B., Khot, T., Hajishirzi, H.: Husky: A unified, open-source language agent for multi-step reasoning. arXiv preprint arXiv:2406.06469 (2024) https://doi.org/10.48550/arXiv.2406.06469

Lafon, M., Vargas Hakim, G.A., Rambour, C., Desrosiers, C., Thome, N.: CLIPTTA: Robust contrastive vision-language test-time adaptation. Advances in Neural Information Processing Systems **38**, 71849–71883 (2025)

Lai, X., Li, J., Li, W., Liu, T., Li, T., Zhao, H.: Mini-o3: Scaling up reasoning patterns and interaction turns for visual search. arXiv preprint arXiv:2509.07969 (2025) https://doi.org/10.48550/arXiv.2509.07969

Lai, Y., Zhong, J., Li, M., Zhao, S., Li, Y., Psounis, K., Yang, X.: Med-R1: Reinforcement learning for generalizable medical reasoning in vision-language models. IEEE Transactions on Medical Imaging (2026)

Lambert, N., Pyatkin, V., Morrison, J., Miranda, L., Lin, B.Y., Chandu, K., Dziri, N., Kumar, S., Zick, T., Choi, Y., Smith, N.A., Hajishirzi, H.: RewardBench: Evaluating reward models for language modeling. In: Findings of the Association for Computational Linguistics: NAACL 2025, pp. 1755–1797. Association for Computational Linguistics, Albuquerque, New Mexico (2025). https://doi.org/10.18653/v1/2025.findings-naacl.96

Lee, E., Lin, Y.-C., Diao, J.: Learning to select visual in-context demonstrations. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2026). Findings Track. https://arxiv.org/abs/2603.26775

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T.: Retrieval-augmented generation for knowledge-intensive NLP tasks. Advances in Neural Information Processing Systems **33**, 9459–9474 (2020)

Li, D., Tan, Z., Qian, P., Li, Y., Chaudhary, K., Hu, L., Shen, J.: SMoA: Improving multi-agent large language models with sparse mixture-of-agents. In: Advances in Knowledge Discovery and Data Mining. Lecture Notes in Computer Science, vol. 15872, pp. 54–65 (2025). https://doi.org/10.1007/978-981-96-8180-8_5

Li, K., Patel, O., Viégas, F., Pfister, H., Wattenberg, M.: Inference-time intervention: Eliciting truthful answers from a language model. Advances in Neural Information Processing Systems **36**, 41451–41530 (2023)

Li, M., Miao, S., Li, P.: Simple is effective: The roles of graphs and large language models in knowledge-graph-based retrieval-augmented generation. In: International Conference on Learning Representations (2025)

Li, M., Yang, C., Xu, C., Jiang, X., Qi, Y., Guo, J., Leung, H.-f., King, I.: Retrieval, reasoning, re-ranking: A context-enriched framework for knowledge graph completion. In: Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pp. 4349–4363 (2025)

Li, X.: A review of prominent paradigms for LLM-based agents: Tool use (Including RAG), planning, and feedback learning. In: Proceedings of the 31st International Conference on Computational Linguistics, pp. 9760–9779. Association for Computational Linguistics, Abu Dhabi, United Arab Emirates (2025)

Li, Y., Zhang, H., Yuan, Y.: GRPO-TTA: Test-time visual tuning for vision-language models via GRPO-driven reinforcement learning. arXiv preprint arXiv:2605.03403 (2026) https://doi.org/10.48550/arXiv.2605.03403

Li, Z., Zhou, Y., Xu, Q.: Latent context compilation: Distilling long context into compact portable memory. arXiv preprint arXiv:2602.21221 (2026) https://doi.org/10.48550/arXiv.2602.21221

Li, Z., Du, H., Huang, C., Wu, X., Yu, L., He, Y., Xie, J., Wu, X., Liu, Z., Zhang, J., Liu, F.: MM-Zero: Self-evolving multi-model vision language models from zero data. arXiv preprint arXiv:2603.09206 (2026) https://doi.org/10.48550/arXiv.2603.09206

Liang, J., He, R., Tan, T.: A comprehensive survey on test-time adaptation under distribution shifts. International Journal of Computer Vision **133**(1), 31–64 (2025) https://doi.org/10.1007/s11263-024-02181-w

Lin, Y., Li, Y., Chen, D., Xu, W., Clark, R., Torr, P.: Olympus: A universal task

router for computer vision tasks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 14235–14246 (2025)

Lin, Z., Gou, Z., Gong, Y., Liu, X., Shen, Y., Xu, R., Lin, C., Yang, Y., Jiao, J., Duan, N., Chen, W.: Not all tokens are what you need for pretraining. In: Advances in Neural Information Processing Systems (2024). https://openreview.net/forum?id=0NMzBwqaAJ

Liu, B., Zhang, J., Lin, F., Yang, C., Peng, M., Yin, W.: SymAgent: A neural-symbolic self-learning agent framework for complex reasoning over knowledge graphs. In: Proceedings of the ACM Web Conference 2025, Sydney NSW Australia, pp. 98–108 (2025). https://doi.org/10.1145/3696410.3714768

Liu, H., Liu, J., Huang, S., Zhan, Y., Sun, H., Deng, W., Wei, F., Zhang, Q.: Se^2: Sequential example selection for in-context learning. In: Findings of the Association for Computational Linguistics: ACL 2024, pp. 5262–5284. Association for Computational Linguistics, Bangkok, Thailand (2024). https://doi.org/10.18653/v1/2024.findings-acl.312

Liu, N.F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., Liang, P.: Lost in the middle: How language models use long contexts. Transactions of the Association for Computational Linguistics **12**, 157–173 (2024)

Liu, R., Gao, J., Zhao, J., Zhang, K., Li, X., Qi, B., Ouyang, W., Zhou, B.: Can 1B LLM surpass 405B LLM? rethinking compute-optimal test-time scaling. In: Reasoning and Planning for LLMs Workshop at the International Conference on Learning Representations (2025). https://openreview.net/forum?id=CvjX9Lhpze

Liu, Y., Wang, R., Chu, Y., Hao, B., Lin, Y., Liu, S., Shao, M.: RouteGoT: Node-adaptive routing for cost-efficient graph of thoughts reasoning. arXiv preprint arXiv:2603.05818 (2026) https://doi.org/10.48550/arXiv.2603.05818

Liu, Z., Dong, Y., Rao, Y., Zhou, J., Lu, J.: Chain-of-Spot: Interactive reasoning improves large vision-language models. arXiv preprint arXiv:2403.12966 (2024) https://doi.org/10.48550/arXiv.2403.12966

Lu, J., Holleis, T., Zhang, Y., Aumayer, B., Nan, F., Bai, H., Ma, S., Ma, S., Li, M., Yin, G.: ToolSandbox: A stateful, conversational, interactive evaluation benchmark for LLM tool use capabilities. In: Findings of the Association for Computational Linguistics: NAACL 2025, pp. 1160–1183 (2025)

Lu, P., Chen, B., Liu, S., Thapa, R., Boen, J., Zou, J.: OctoTools: An agentic framework with extensible tools for complex reasoning. In: Reasoning and Planning for LLMs Workshop at the International Conference on Learning Representations (2025). https://openreview.net/forum?id=YzKOW0zD1d

Ma, Q., Shi, Y., Tian, H., Wang, C., Chang, W., Yao, T.: From faithfulness

to correctness: Generative reward models that think critically. arXiv preprint arXiv:2509.25409 (2025) https://doi.org/10.48550/arXiv.2509.25409

Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y.: Self-refine: Iterative refinement with self-feedback. Advances in Neural Information Processing Systems **36**, 46534–46594 (2023)

Mavromatis, C., Karypis, G.: GNN-RAG: Graph neural retrieval for efficient large language model reasoning. In: Findings of the Association for Computational Linguistics: ACL 2025, pp. 16682–16699 (2025). https://doi.org/10.18653/v1/2025.findings-acl.856

Mialon, G., Dessì, R., Lomeli, M., Nalmpantis, C., Pasunuru, R., Raileanu, R., Rozière, B., Schick, T., Dwivedi-Yu, J., Celikyilmaz, A., Grave, E., LeCun, Y., Scialom, T.: Augmented language models: A survey. Transactions on Machine Learning Research (2023)

Mitra, C., Huang, B., Darrell, T., Herzig, R.: Compositional chain-of-thought prompting for large multimodal models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 14420–14431 (2024)

Nasiriany, S., Xia, F., Yu, W., Xiao, T., Liang, J., Dasgupta, I., Xie, A., Driess, D., Wahid, A., Xu, Z., Vuong, Q., Zhang, T., Lee, T.-W.E., Lee, K.-H., Xu, P., Kirmani, S., Zhu, Y., Zeng, A., Hausman, K., Heess, N., Finn, C., Levine, S., Ichter, B.: PIVOT: Iterative visual prompting elicits actionable knowledge for VLMs. In: Proceedings of the 41st International Conference on Machine Learning (2024)

Nath, M., Das, A., Xie, J., Schiele, B.: ClipTTT: CLIP-guided test-time training helps LVLMs see better. arXiv preprint arXiv:2603.26486 (2026) https://doi.org/10.48550/arXiv.2603.26486

Ni, B., Liu, Z., Wang, L., Lei, Y., Zhao, Y., Cheng, X., Zeng, Q., Dong, L., Xia, Y., Kenthapadi, K., Rossi, R., Dernoncourt, F., Tanjim, M.M., Ahmed, N., Liu, X., Fan, W., Blasch, E., Wang, Y., Jiang, M., Derr, T.: Towards trustworthy retrieval-augmented generation for large language models: A survey. arXiv preprint arXiv:2502.06872 (2025) https://doi.org/10.48550/arXiv.2502.06872

Niu, S., Wu, J., Zhang, Y., Chen, Y., Zheng, S., Zhao, P., Tan, M.: Efficient test-time model adaptation without forgetting. In: Proceedings of the 39th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 162, pp. 16888–16905. Baltimore, MD, USA (2022)

Niu, S., Wu, J., Zhang, Y., Wen, Z., Chen, Y., Zhao, P., Tan, M.: Towards stable test-time adaptation in dynamic wild world. In: The Eleventh International Conference on Learning Representations (2023). https://openreview.net/forum?id=g2YraF75Tj

Nori, H., Lee, Y.T., Zhang, S., Carignan, D., Edgar, R., Fusi, N., King, N., Larson, J., Li, Y., Liu, W., Luo, R., McKinney, S.M., Ness, R.O., Poon, H., Qin, T., Usuyama, N., White, C., Horvitz, E.: Can generalist foundation models outcompete special-purpose tuning? case study in medicine. arXiv preprint arXiv:2311.16452 (2023) https://doi.org/10.48550/arXiv.2311.16452

Packer, C., Wooders, S., Lin, K., Fang, V., Patil, S.G., Stoica, I., Gonzalez, J.E.: MemGPT: Towards LLMs as operating systems. arXiv preprint arXiv:2310.08560 (2023) https://doi.org/10.48550/arXiv.2310.08560

Pan, J., He, H., Bowman, S.R., Feng, S.: Spontaneous reward hacking in iterative self-refinement. arXiv preprint arXiv:2407.04549 (2024) https://doi.org/10.48550/arXiv.2407.04549

Pan, J., Liu, C., Wu, J., Liu, F., Zhu, J., Li, H.B., Chen, C., Ouyang, C., Rueckert, D.: MedVLM-R1: Incentivizing medical reasoning capability of vision-language models (VLMs) via reinforcement learning. In: Medical Image Computing and Computer Assisted Intervention – MICCAI 2025. Lecture Notes in Computer Science, vol. 15966, pp. 337–347 (2026). https://doi.org/10.1007/978-3-032-04981-0_32

Park, J.S., O'Brien, J., Cai, C.J., Morris, M.R., Liang, P., Bernstein, M.S.: Generative agents: Interactive simulacra of human behavior. In: Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology, San Francisco CA USA, pp. 1–22 (2023). https://doi.org/10.1145/3586183.3606763

Park, M., Yang, H., Kim, J., Park, K., Kim, H.: ProgRAG: Hallucination-resistant progressive retrieval and reasoning over knowledge graphs. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 40, pp. 32674–32682 (2026)

Peng, B., Zhu, Y., Liu, Y., Bo, X., Shi, H., Hong, C., Zhang, Y., Tang, S.: Graph retrieval-augmented generation: A survey. ACM Transactions on Information Systems **44**(2), 1–52 (2026) https://doi.org/10.1145/3777378

Peng, X., Qin, C., Yan, A., Yang, X., Chen, Z., Xu, R., Wu, C.-S.: MTA-Agent: An open recipe for multimodal deep search agents. arXiv preprint arXiv:2604.06376 (2026) https://doi.org/10.48550/arXiv.2604.06376

Qian, C., Liu, W., Liu, H., Chen, N., Dang, Y., Li, J., Yang, C., Chen, W., Su, Y., Cong, X.: Chatdev: Communicative agents for software development. In: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 15174–15186 (2024)

Qian, C., Acikgoz, E.C., He, Q., Wang, H., Chen, X., Hakkani-Tur, D., Tur, G., Ji, H.: ToolRL: Reward is all tool learning needs. Advances in Neural Information Processing Systems **38**, 105523–105553 (2025)

Qian, D., Chen, H.-T., Choi, E.: RVR: Retrieve-verify-retrieve for comprehensive question answering. arXiv preprint arXiv:2602.18425 (2026) https://doi.org/10.48550/arXiv.2602.18425

Qian, H., Liu, Z., Zhang, P., Mao, K., Lian, D., Dou, Z., Huang, T.: MemoRAG: Boosting long context processing with global memory-enhanced retrieval augmentation. In: Proceedings of the ACM Web Conference 2025 (2025). https://doi.org/10.1145/3696410.3714805

Qin, L., Chen, Q., Fei, H., Chen, Z., Li, M., Che, W.: What factors affect multimodal in-context learning? an in-depth exploration. Advances in Neural Information Processing Systems **37**, 123207–123236 (2024)

Qu, C., Dai, S., Wei, X., Cai, H., Wang, S., Yin, D., Xu, J., Wen, J.-r.: Tool learning with large language models: a survey. Frontiers of Computer Science **19**(8), 198343 (2025) https://doi.org/10.1007/s11704-024-40678-2

Quek, R.W.H., Lee, S., Leong, A.W.L., Verma, A., Prakash, A., Chen, N.F., Low, B.K.H., Rus, D., Solar-Lezama, A.: MeMo: Memory as a model. arXiv preprint arXiv:2605.15156 (2026) https://doi.org/10.48550/arXiv.2605.15156

Ren, Z.Z., Shao, Z., Song, J., Xin, H., Wang, H., Zhao, W., Zhang, L., Fu, Z., Zhu, Q., Yang, D., Wu, Z.F., Gou, Z., Ma, S., Tang, H., Liu, Y., Gao, W., Guo, D., Ruan, C.: DeepSeek-Prover-V2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition. arXiv preprint arXiv:2504.21801 (2025) https://doi.org/10.48550/arXiv.2504.21801

Rimsky, N., Gabrieli, N., Schulz, J., Tong, M., Hubinger, E., Turner, A.M.: Steering Llama 2 via contrastive activation addition. In: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 15504–15522 (2024). https://doi.org/10.18653/v1/2024.acl-long.828

Sarch, G., Cai, L., Wang, Q., Wu, H., Chen, D., Liu, Z.: Vero: An open RL recipe for general visual reasoning. arXiv preprint arXiv:2604.04917 (2026) https://doi.org/10.48550/arXiv.2604.04917

Sardana, N., Portes, J., Doubov, S., Frankle, J.: Beyond Chinchilla-Optimal: Accounting for inference in language model scaling laws. In: Proceedings of the 41st International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 235, pp. 43445–43460 (2024)

Schick, T., Schütze, H.: Exploiting cloze-questions for few-shot text classification and natural language inference. In: Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, pp. 255–269 (2021)

Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Hambro, E., Zettlemoyer, L., Cancedda, N., Scialom, T.: Toolformer: Language models can teach themselves to use tools. Advances in Neural Information Processing Systems **36**, 68539–68551 (2023)

Shao, H., Qian, S., Xiao, H., Song, G., Zong, Z., Wang, L., Liu, Y., Li, H.: Visual CoT: Advancing multi-modal language models with a comprehensive dataset and benchmark for chain-of-thought reasoning. Advances in Neural Information Processing Systems **37**, 8612–8642 (2024)

Shi, L., Ma, C., Liang, W., Diao, X., Ma, W., Vosoughi, S.: Judging the judges: A systematic study of position bias in LLM-as-a-judge. In: Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, pp. 292–314. The Asian Federation of Natural Language Processing and The Association for Computational Linguistics, Mumbai, India (2025). https://doi.org/10.18653/v1/2025.ijcnlp-long.18

Shi, Z., Gao, S., Yan, L., Feng, Y., Chen, X., Chen, Z., Yin, D., Verberne, S., Ren, Z.: Tool learning in the wild: Empowering language models as automatic tool agents. In: Proceedings of the ACM Web Conference 2025, Sydney NSW Australia, pp. 2222–2237 (2025). https://doi.org/10.1145/3696410.3714825

Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., Yao, S.: Reflexion: Language agents with verbal reinforcement learning. Advances in Neural Information Processing Systems **36**, 8634–8652 (2023)

Shu, M., Nie, W., Huang, D.-A., Yu, Z., Goldstein, T., Anandkumar, A., Xiao, C.: Test-time prompt tuning for zero-shot generalization in vision-language models. Advances in Neural Information Processing Systems **35**, 14274–14289 (2022)

Snell, C., Lee, J., Xu, K., Kumar, A.: Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In: The Thirteenth International Conference on Learning Representations (2025). https://openreview.net/forum?id=4FWAwZtd2n

Song, J., Lee, J., Kweon, I.S., Choi, S.: Ecotta: Memory-efficient continual test-time adaptation via self-distilled regularization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 11920–11929 (2023)

Song, Q., Li, H., Yu, Y., Zhou, H., Yang, L., Bai, S., She, Q., Huang, Z., Zhao, Y.: CodeDance: A dynamic tool-integrated MLLM for executable visual reasoning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2026). https://arxiv.org/abs/2512.17312

Sun, L., Tao, Z., Li, Y., Arakawa, H.: Oda: Observation-driven agent for integrating LLMs and knowledge graphs. In: Findings of the Association for Computational

Linguistics: ACL 2024, pp. 7417–7431 (2024)

Sun, Y., Wang, X., Liu, Z., Miller, J., Efros, A., Hardt, M.: Test-time training with self-supervision for generalization under distribution shifts. In: Proceedings of the 37th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 119, pp. 9229–9248. Virtual Event (2020)

Sun, Y., Li, X., Dalal, K., Xu, J., Vikram, A., Zhang, G., Dubois, Y., Chen, X., Wang, X., Koyejo, S., Hashimoto, T., Guestrin, C.: Learning to (learn at test time): RNNs with expressive hidden states. In: Proceedings of the 42nd International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 267, pp. 57503–57522 (2025)

Surís, D., Menon, S., Vondrick, C.: Vipergpt: Visual inference via python execution for reasoning. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 11888–11898 (2023)

Tan, H., Ji, Y., Hao, X., Lin, M., Wang, P., Wang, Z., Zhang, S.: Reason-RFT: Reinforcement fine-tuning for visual reasoning of vision-language models. Advances in Neural Information Processing Systems **38**, 5772–5822 (2025)

Tan, X., Wang, X., Liu, Q., Xu, X., Yuan, X., Zhang, W.: Paths-over-Graph: Knowledge graph empowered large language model reasoning. In: Proceedings of the ACM Web Conference 2025, Sydney NSW Australia, pp. 3505–3522 (2025). https://doi.org/10.1145/3696410.3714892

Tang, Y., Yang, Y.: MultiHop-RAG: Benchmarking retrieval-augmented generation for multi-hop queries. In: First Conference on Language Modeling (2024). https://openreview.net/forum?id=t4eB3zYWBK

Tang, Z., Yu, X., Xiao, Z., Wen, Z., Li, Z., Zhou, J., Wang, H., Wang, H., Huang, H., Deng, W., Sun, F., Zhang, Q.: Mnemis: Dual-route retrieval on hierarchical graphs for long-term LLM memory. In: Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (2026). To appear. https://arxiv.org/abs/2602.15313

Thawakar, O., Dissanayake, D., More, K.P., Thawkar, R., Heakl, A., Ahsan, N., Li, Y., Zumri, I.Z.M., Lahoud, J., Anwer, R.M., Cholakkal, H., Laptev, I., Shah, M., Khan, F.S., Khan, S.: LlamaV-o1: Rethinking step-by-step visual reasoning in LLMs. In: Findings of the Association for Computational Linguistics: ACL 2025, pp. 24290–24315 (2025). https://doi.org/10.18653/v1/2025.findings-acl.1247

Tian, K., Peng, Z.: FewRelEx: Exploring multimodal few-shot relation extraction with enhanced visual–textual mapping. Expert Systems with Applications **277**, 127104 (2025)

Todd, E., Li, M., Sen Sharma, A., Mueller, A., Wallace, B., Bau, D.: Function vectors

in large language models. In: International Conference on Learning Representations (2024)

Trivedi, H., Balasubramanian, N., Khot, T., Sabharwal, A.: Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 10014–10037 (2023)

Turner, A.M., Thiergart, L., Leech, G., Udell, D., Vazquez, J.J., Mini, U., MacDiarmid, M.: Activation addition: Steering language models without optimization. arXiv preprint arXiv:2308.10248 (2023) https://doi.org/10.48550/arXiv.2308.10248

Venktesh, V., Rathee, M., Anand, A.: Trust but verify! a survey on verification design for test-time scaling. arXiv preprint arXiv:2508.16665 (2025) https://doi.org/10.48550/arXiv.2508.16665

Wang, A., Song, L., Tian, Y., Peng, B., Yu, D., Mi, H., Su, J., Yu, D.: Efficient tree search with dynamic exploration budget for LLM reasoning. In: Proceedings of the AAAI Conference on Artificial Intelligence (2025)

Wang, D., Shelhamer, E., Liu, S., Olshausen, B., Darrell, T.: Tent: Fully test-time adaptation by entropy minimization. In: International Conference on Learning Representations (2021). https://openreview.net/forum?id=uXl3bZLkr3c

Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L., Anandkumar, A.: Voyager: An open-ended embodied agent with large language models. arXiv preprint arXiv:2305.16291 (2023) https://doi.org/10.48550/arXiv.2305.16291

Wang, J., Wang, J., Athiwaratkun, B., Zhang, C., Zou, J.Y.: Mixture-of-agents enhances large language model capabilities. In: International Conference on Learning Representations (2025)

Wang, L., Ma, C., Feng, X., Zhang, Z., Yang, H., Zhang, J., Chen, Z., Tang, J., Chen, X., Lin, Y., Zhao, W.X., Wei, Z., Wen, J.-R.: A survey on large language model based autonomous agents. Frontiers of Computer Science **18**(6), 186345 (2024) https://doi.org/10.1007/s11704-024-40231-1

Wang, Q., Fink, O., Van Gool, L., Dai, D.: Continual test-time domain adaptation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7201–7211 (2022)

Wang, R., Sun, Y., Tandon, A., Gandelsman, Y., Chen, X., Efros, A.A., Wang, X.: Test-time training on video streams. Journal of Machine Learning Research **26**(9), 1–29 (2025)

Wang, W., Dong, L., Cheng, H., Liu, X., Yan, X., Gao, J., Wei, F.: Augmenting language models with long-term memory. Advances in Neural Information Processing Systems **36**, 74530–74543 (2023)

Wang, X., Wang, Z., Gao, X., Zhang, F., Wu, Y., Xu, Z., Shi, T., Wang, Z., Li, S., Qian, Q.: Searching for best practices in retrieval-augmented generation. In: Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pp. 17716–17736 (2024)

Wang, X., Yang, Z., Li, L., Lu, H., Xu, Y., Lin, C.-C., Lin, K., Huang, F., Wang, L.: Scaling inference-time search with vision value model for improved visual comprehension. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 1173–1184 (2025)

Wang, X., Wei, J., Schuurmans, D., Le, Q.V., Chi, E.H., Narang, S., Chowdhery, A., Zhou, D.: Self-consistency improves chain of thought reasoning in language models. In: The Eleventh International Conference on Learning Representations (2023). https://openreview.net/forum?id=1PL1NIMMrw

Wang, Y., Li, Z., Zang, Y., Wang, C., Lu, Q., Jin, C., Wang, J.: Unified multimodal chain-of-thought reward model through reinforcement fine-tuning. Advances in Neural Information Processing Systems **38**, 159130–159157 (2025)

Wang, Y., Zhang, H., Pang, L., Guo, B., Zheng, H., Zheng, Z.: Maferw: Query rewriting with multi-aspect feedbacks for retrieval-augmented large language models. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 39, pp. 25434–25442 (2025)

Wang, Z., Wang, J., Xu, H., Yan, M., Huang, F., Yang, X., Wei, X.-S., Mi, S., Zhang, Y.: Efficient and effective in-context demonstration selection with coreset. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 40, pp. 10458–10466 (2026)

Wang, Z., Luo, Y., Zheng, L., Chen, Z., Wang, S., Huang, Z.: In search of lost online test-time adaptation: A survey. International Journal of Computer Vision **133**(3), 1106–1139 (2025) https://doi.org/10.1007/s11263-024-02213-5

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q.V., Zhou, D.: Chain-of-thought prompting elicits reasoning in large language models. Advances in Neural Information Processing Systems **35**, 24824–24837 (2022)

Wu, D., Wang, H., Yu, W., Zhang, Y., Chang, K.-W., Yu, D.: LongMemEval: Benchmarking chat assistants on long-term interactive memory. In: The Thirteenth International Conference on Learning Representations (2025). https://openreview.net/forum?id=pZiyCaVuti

Wu, J., Yin, W., Jiang, Y., Wang, Z., Xi, Z., Fang, R., Zhang, L., He, Y., Zhou, D.,

Xie, P.: WebWalker: Benchmarking LLMs in web traversal. In: Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 10290–10305 (2025)

Wu, J., Deng, Z., Li, W., Liu, Y., You, B., Li, B., Ma, Z., Liu, Z.: MMSearch-R1: Incentivizing LMMs to search. In: Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (2026). To appear. https://arxiv.org/abs/2506.20670

Wu, J., Zhu, J., Qi, Y., Chen, J., Xu, M., Menolascina, F., Grau, V.: Medical GraphRAG: Towards safe medical large language model via graph retrieval-augmented generation. arXiv preprint arXiv:2408.04187 (2024) https://doi.org/10.48550/arXiv.2408.04187

Wu, Q., Bansal, G., Zhang, J., Wu, Y., Li, B., Zhu, E., Jiang, L., Zhang, X., Zhang, S., Liu, J.: AutoGen: Enabling next-gen LLM applications via multi-agent conversations. In: First Conference on Language Modeling (2024)

Wu, X., Yang, C., Lin, X., Xu, C., Jiang, X., Sun, Y., Xiong, H., Li, J., Guo, J.: Think-on-Graph 3.0: Efficient and adaptive LLM reasoning on heterogeneous graphs via multi-agent dual-evolving context retrieval. arXiv preprint arXiv:2509.21710 (2025) https://doi.org/10.48550/arXiv.2509.21710

Wu, Y., Liang, S., Zhang, C., Wang, Y., Zhang, Y., Guo, H., Tang, R., Liu, Y.: From human memory to AI memory: A survey on memory mechanisms in the era of LLMs. arXiv preprint arXiv:2504.15965 (2025) https://doi.org/10.48550/arXiv.2504.15965

Wu, Y., Rabe, M.N., Hutchins, D., Szegedy, C.: Memorizing transformers. In: International Conference on Learning Representations (2022). https://openreview.net/forum?id=TrjbxzRcnf-

Wu, Z., Jiang, M., Shen, C.: Get an a in math: Progressive rectification prompting. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 38, pp. 19288–19296 (2024)

Wu, Z., Hao, Y., Mou, L.: TokMem: One-token procedural memory for large language models. In: The Fourteenth International Conference on Learning Representations (2026). https://openreview.net/forum?id=RWjEf9PdiJ

Xiao, Y., Li, Y., Wang, H., Tang, Y., Wang, Z.Z.: ToolMem: Enhancing multimodal agents with learnable tool capability memory. arXiv preprint arXiv:2510.06664 (2025) https://doi.org/10.48550/arXiv.2510.06664

Xiao, Z., Snoek, C.G.M.: Beyond model adaptation at test time: A survey. arXiv preprint arXiv:2411.03687 (2024) https://doi.org/10.48550/arXiv.2411.03687

Xu, Y., Kong, X., Chen, T., Zhuo, D.: Conveyor: Efficient tool-aware LLM serving with tool partial execution. arXiv preprint arXiv:2406.00059 (2024) https://doi.org/10.48550/arXiv.2406.00059

Yang, C., Xiang, Z., Tang, Y., Teng, Z., Huang, C., Long, F., Liu, Y., Su, J.: TTCS: Test-time curriculum synthesis for self-evolving. In: LLA Workshop at the International Conference on Learning Representations (2026). https://openreview.net/forum?id=ZkAJQNXQkr

Yang, K., Liu, Y., Chaudhary, S., Fakoor, R., Chaudhari, P.A., Karypis, G., Rangwala, H.: AgentOccam: A simple yet strong baseline for LLM-based web agents. In: International Conference on Learning Representations (2025)

Yang, X., Sun, K., Xin, H., Sun, Y., Bhalla, N., Chen, X., Choudhary, S., Gui, R.D., Jiang, Z.W., Jiang, Z.: Crag-comprehensive RAG benchmark. Advances in Neural Information Processing Systems **37**, 10470–10490 (2024)

Yang, Y., He, X., Pan, H., Jiang, X., Deng, Y., Yang, X., Lu, H., Yin, D., Rao, F., Zhu, M., Zhang, B., Chen, W.: R1-Onevision: Advancing generalized multimodal reasoning through cross-modal formalization. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pp. 2376–2385 (2025)

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T., Cao, Y., Narasimhan, K.: Tree of thoughts: Deliberate problem solving with large language models. Advances in Neural Information Processing Systems **36**, 11809–11822 (2023)

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K.R., Cao, Y.: ReAct: Synergizing reasoning and acting in language models. In: The Eleventh International Conference on Learning Representations (2023). https://openreview.net/forum?id=WE_vluYUL-X

Yoon, Y., Lee, S., Song, S., Wang, S., Chen, W., Ok, J.: PaT: Planning-after-trial for efficient test-time code generation. In: Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (2026). To appear. https://arxiv.org/abs/2605.07248

Yu, Y., Ping, W., Liu, Z., Wang, B., You, J., Zhang, C., Shoeybi, M., Catanzaro, B.: RankRAG: Unifying context ranking with retrieval-augmented generation in LLMs. Advances in Neural Information Processing Systems **37**, 121156–121184 (2024)

Yun, S., Masukawa, R., Jeong, S., Huang, W., Chen, H., Imani, M.: Fair context learning for evidence-balanced test-time adaptation in vision-language models. arXiv preprint arXiv:2602.07027 (2026) https://doi.org/10.48550/arXiv.2602.07027

Yun, S., Peng, J., Li, P., Fan, W., Chen, J., Zou, J., Li, G., Chen, T.: Graph-of-Agents: A graph-based framework for multi-agent LLM collaboration. In: The Fourteenth International Conference on Learning Representations (2026). https://openreview

.net/forum?id=34cANdsHKV

Zhan, Z., Zhou, S., Zhou, X., Xiao, Y., Wang, J., Deng, J., Zhu, H., Hou, Y., Song, Y., Lin, M.: Retrieval-augmented in-context learning for multimodal large language models in disease classification. Journal of Biomedical Informatics, 105017 (2026)

Zhang, G., Yue, Y., Li, Z., Yun, S., Wan, G., Wang, K., Cheng, D., Yu, J., Chen, T.: Cut the crap: An economical communication pipeline for LLM-based multi-agent systems. In: International Conference on Learning Representations (2025)

Zhang, J., Huang, J., Yao, H., Liu, S., Zhang, X., Lu, S., Tao, D.: R1-VL: Learning to reason with multimodal large language models via step-wise group relative policy optimization. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (2025). https://openaccess.thecvf.com/content/ICCV2025/papers/Zhang_R1-VL_Learning_to_Reason_with_Multimodal_Large_Language_Models_via_ICCV_2025_paper.pdf

Zhang, K., Barrett, C.D., Kim, J., Sun, L., Taghavi, T., Kenthapadi, K.: RadAgents: Multimodal agentic reasoning for chest X-ray interpretation with radiologist-like workflows. In: Proceedings of The 9th International Conference on Medical Imaging with Deep Learning. Proceedings of Machine Learning Research, vol. 315, pp. 3496–3519 (2026). https://proceedings.mlr.press/v315/zhang26b.html

Zhang, M., Levine, S., Finn, C.: Memo: Test time robustness via adaptation and augmentation. Advances in Neural Information Processing Systems **35**, 38629–38642 (2022)

Zhang, Q., Lyu, F., Sun, Z., Wang, L., Zhang, W., Hua, W., Wu, H., Guo, Z., Wang, Y., Muennighoff, N., King, I., Liu, X., Ma, C.: A survey on test-time scaling in large language models: What, how, where, and how well? arXiv preprint arXiv:2503.24235 (2025) https://doi.org/10.48550/arXiv.2503.24235

Zhang, R., Sun, Q., Song, C., Qi, Y., Zheng, Z.: VSearcher: Long-horizon multimodal search agent via reinforcement learning. arXiv preprint arXiv:2603.02795 (2026) https://doi.org/10.48550/arXiv.2603.02795

Zhang, S., Yu, T., Feng, Y.: Truthx: Alleviating hallucinations by editing large language models in truthful space. In: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 8908–8949 (2024)

Zhang, T., Zhang, Z., Ma, Z., Chen, Y., Qi, Z., Yuan, C., Li, B., Pu, J., Zhao, Y., Xie, Z., Ma, J., Shan, Y., Hu, W.: mR^2AG: Multimodal retrieval-reflection-augmented generation for knowledge-based VQA. arXiv preprint arXiv:2411.15041 (2024) https://doi.org/10.48550/arXiv.2411.15041

Zhang, T., Zhang, Z., Ma, Z., Chen, Y., Li, B., Yuan, C., Wang, G., Rao, F., Shan,

Y., Hu, W.: MMHops-R1: Multimodal multi-hop reasoning. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 40, pp. 28391–28399 (2026)

Zhang, Z., Dai, Q., Bo, X., Ma, C., Li, R., Chen, X., Zhu, J., Dong, Z., Wen, J.-R.: A survey on the memory mechanism of large language model-based agents. ACM Transactions on Information Systems **43**(6), 1–47 (2025) https://doi.org/10.1145/3748302

Zhang, Z., Zhang, Y., Ding, X., Yue, X.: Vision search assistant: Empower vision-language models as multimodal search engines. arXiv preprint arXiv:2410.21220 (2024) https://doi.org/10.48550/arXiv.2410.21220

Zhao, S., Lin, S., Li, M., Zhang, H., Peng, W., Zhang, K., Wei, C.: PyVision-RL: Forging open agentic vision models via RL. arXiv preprint arXiv:2602.20739 (2026) https://doi.org/10.48550/arXiv.2602.20739

Zheng, B., Gou, B., Kil, J., Sun, H., Su, Y.: GPT-4V(ision) is a generalist web agent, if grounded. In: Proceedings of the 41st International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 235, pp. 61349–61385 (2024). https://proceedings.mlr.press/v235/zheng24e.html

Zheng, M., Pei, J., Logeswaran, L., Lee, M., Jurgens, D.: When "a helpful assistant" is not really helpful: Personas in system prompts do not improve performances of large language models. In: Findings of the Association for Computational Linguistics: EMNLP 2024, pp. 15126–15154 (2024)

Zheng, X., An, K., Wang, Z., Wang, Y., Wu, Y.: StepSearch: Igniting LLMs search ability via step-wise proximal policy optimization. In: Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pp. 21805–21830 (2025). https://doi.org/10.18653/v1/2025.emnlp-main.1106

Zhong, W., Guo, L., Gao, Q., Ye, H., Wang, Y.: Memorybank: Enhancing large language models with long-term memory. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 38, pp. 19724–19731 (2024)

Zhou, D., Schärli, N., Hou, L., Wei, J., Scales, N., Wang, X., Schuurmans, D., Cui, C., Bousquet, O., Le, Q.V., Chi, E.H.: Least-to-most prompting enables complex reasoning in large language models. In: The Eleventh International Conference on Learning Representations (2023). https://openreview.net/forum?id=WZH7099tgfM

Zhou, H., Geng, H., Xue, X., Kang, L., Qin, Y., Wang, Z., Yin, Z., Bai, L.: Reso: A reward-driven self-organizing LLM-based multi-agent system for reasoning tasks. In: Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pp. 15990–16009 (2025)

Zhou, Y., Liu, Y., Li, X., Jin, J., Qian, H., Liu, Z., Li, C., Dou, Z., Ho, T.-Y., Yu,

P.S.: Trustworthiness in retrieval-augmented generation systems: A survey. arXiv preprint arXiv:2409.10102 (2024) https://doi.org/10.48550/arXiv.2409.10102

Zhu, D., Wei, X., Zhao, G., Wu, W., Zou, H., Ran, J., Wang, X., Sun, L., Zhang, X., Li, S.: Chain-of-thought matters: Improving long-context language models with reasoning path supervision. In: Findings of the Association for Computational Linguistics: EMNLP 2025, pp. 3197–3211 (2025). https://doi.org/10.18653/v1/2025.findings-emnlp.170

Zou, A., Phan, L., Chen, S., Campbell, J., Guo, P., Ren, R., Pan, A., Yin, X., Mazeika, M., Dombrowski, A.-K., Goel, S., Li, N., Byun, M.J., Wang, Z., Mallen, A., Basart, S., Koyejo, S., Song, D., Fredrikson, M., Kolter, J.Z., Hendrycks, D.: Representation engineering: A top-down approach to AI transparency. arXiv preprint arXiv:2310.01405 (2023) https://doi.org/10.48550/arXiv.2310.01405

Appendix A Review Protocol

This appendix describes the literature search, screening, coding, and density-construction protocol used to support the Locus × Source heatmap, the temporal-density visualization, and the opportunity-cell analysis in this survey. The purpose of this appendix is not to present an exhaustive bibliometric census of all publications related to test-time improvement, but to make explicit how the reviewed corpus was constructed, how papers were assigned to the analytical framework, and how the qualitative density claims in the main text were derived from a screened and coded literature corpus.

A.1 Literature Search

The reviewed literature database was constructed through structured searches over scholarly databases, major publication venues, and citation links. The search covered work published or publicly released between 2020 and 2026, with particular attention to large models, multimodal models, retrieval-augmented generation, test-time adaptation, reasoning-time computation, verifier- or reward-guided inference, memory mechanisms, tool use, and agentic systems. A small number of necessary 2019 anchor works that initiated later TTI branches were retained and grouped into the 2020–2022 foundation period.

The search sources included Google Scholar, Semantic Scholar, arXiv, ACL Anthology, OpenReview, DBLP, ACM Digital Library, IEEE Xplore, SpringerLink, and major conference or journal proceedings relevant to machine learning, natural language processing, computer vision, artificial intelligence, and large-model systems. The database was also expanded through backward citation chasing from representative papers and surveys, and through forward citation chasing for anchor works that initiated major intervention families.

The search was organized around mechanism-oriented keyword groups. Instead of using only the term "test-time improvement," which is not yet a standardized field label, we searched across adjacent intervention families that satisfy the TTI boundary defined in Section 2.1. The keyword groups included prompting and task expression, retrieval-augmented generation, test-time adaptation, reasoning and search, test-time scaling, verification and reward mechanisms, memory, tool use, and agentic systems. Table A1 summarizes the main search dimensions.

Table A1: Literature search protocol.

Item	Specification
Search sources	Google Scholar; Semantic Scholar; arXiv; ACL Anthology; OpenReview; DBLP; ACM Digital Library; IEEE Xplore; SpringerLink; major AI, ML, NLP, CV, and large-model venues.
Time span	2020–2026. The corpus was last updated in May 2026.

Table A1: Literature search protocol (continued).

Item	Specification
Search expansion	Structured keyword search; backward citation chasing; forward citation chasing; checking references from adjacent surveys on test-time scaling, test-time adaptation, RAG, tool learning, memory, verification, and agents.
Initial literature database	A reviewed literature database covering TTI-related and adjacent inference-stage mechanisms. The database included recent papers collected from the project archive and necessary anchor works from earlier years.
Review type	Protocol-guided scoping synthesis for mechanism-level review, not an exhaustive bibliometric census.
Density corpus	Only screened core-method records were used for heatmap and temporal-density construction. Survey, benchmark, dataset, and background papers were tracked separately.

After specifying the search sources and time span in Table A1, we further organized the search terms into mechanism-oriented keyword groups. These groups were designed to capture the major TTI intervention families reviewed in the paper. Table A2 lists representative examples.

Table A2: Representative keyword groups.

Direction	Example search keywords
Prompting and task expression	prompt engineering; prompt optimization; instruction prompting; task reformulation; query rewriting; input reconstruction; multimodal prompting
Retrieval and context construction	retrieval-augmented generation; RAG; active retrieval; iterative retrieval; GraphRAG; KG-RAG; trustworthy RAG; context ranking; retrieval evaluation
Test-time adaptation	test-time adaptation; test-time training; online test-time adaptation; continual test-time adaptation; entropy minimization; prompt tuning at test time; black-box TTA
Reasoning and search	chain-of-thought; self-consistency; least-to-most prompting; tree of thoughts; graph of thoughts; test-time search; candidate exploration; inference-time search
Test-time scaling and compute	test-time scaling; test-time compute; inference-time scaling; compute-optimal inference; adaptive test-time compute; budgeted reasoning
Feedback, verifier, and reward	self-refine; Reflexion; verifier-guided decoding; reward model; LLM-as-a-judge; self-correction; reward hacking; process supervision
Memory and reusable state	long-term memory; agent memory; memory-augmented LLM; experience memory; tool memory; skill memory; persistent state

Table A2: Representative keyword groups (continued).

Direction	Example search keywords
Tool use and agents	tool learning; tool use; code execution; web agent; multimodal agent; agentic search; multi-agent collaboration; agent workflow

In practice, keywords within each group were combined with large-model-related terms such as "large language model," "LLM," "vision-language model," "multimodal model," "test-time," and "inference-time," and were adapted to the syntax of each search source.

The literature database was then screened and transformed into a coded reviewed corpus. In the current coding workbook, the three main mechanism sheets contain 318 records before density exclusion. After removing records used only as surveys, benchmarks, analysis papers, or background/exclusion cases, 272 core-method records are retained for the density construction reported in this appendix. These counts make the current heatmap and temporal-density analysis auditable and correspond to the screened coding workbook used for this version of the manuscript.

Table A3 summarizes the screening and filtering steps used to construct the density corpus.

Table A3: Screening flow used for corpus construction.

Step	Operation	Decision rule	Output in current coding workbook
1	Structured literature search and corpus assembly	Retrieve papers from databases, venues, and citation links using mechanism-oriented keyword groups.	Candidate literature database.
2	Version merging and deduplication	Merge arXiv, conference, and journal versions of the same work when identifiable; keep the formal version where available.	Deduplicated candidate records.
3	Topic screening	Retain papers related to large models, multimodal models, or inference-stage improvement mechanisms.	Topic-relevant records.
4	TTI-boundary screening	Retain records whose method or system directly affects testing or inference behavior.	TTI-eligible records.
5	Role classification	Classify records as core-method, system, anchor, representative, emerging, extension, survey, benchmark, analysis, or background/exclusion.	318 records in the three main mechanism sheets before density exclusion.

Table A3: Screening flow used for corpus construction (continued).

Step	Operation	Decision rule	Output in current coding workbook
6	Density-corpus filtering	Exclude survey, benchmark, analysis, and background-only records from heatmap and temporal-density counts.	272 core-method records used for density construction.
7	Mechanism coding and aggregation	Code each core-method record by branch, locus, source, temporal descriptor, cost descriptor, and secondary links.	Locus × Source counts and branch-year density matrices.

The coded corpus is larger than the set of works directly cited in the main text. The bibliography includes papers explicitly discussed or cited in the manuscript, whereas the density matrices are computed from the screened coding workbook. The coding workbook may be provided as supplementary material if required during review. The next subsection specifies the criteria used to determine which records entered the core-method corpus and which records were retained only as background, survey, benchmark, or analysis support.

A.2 Screening Criteria

The inclusion and exclusion criteria follow the bounded definition of TTI developed in Section 2.1. A paper is included in the core-method corpus only when it describes a method, system, or mechanism that directly changes how a trained model behaves during testing or inference. The intervention may operate by changing the input, updating or steering inference-relevant states, reorganizing the reasoning or action process, introducing external information, using feedback or verification, or maintaining memory or experience across episodes.

A paper is not included in the core-method density count solely because it improves a model. Conventional pretraining, standalone supervised fine-tuning, offline domain retraining, dataset construction, benchmark design, or general evaluation papers are excluded from density construction unless they contain an explicit inference-stage mechanism that directly affects the current solving behavior of the model. Such papers may still be cited in the main text as background, evaluation context, adjacent surveys, or supporting evidence for resource-control discussions.

Table A4 summarizes the inclusion and exclusion rules applied during screening.

Table A4: Inclusion and exclusion criteria.

Paper type	**Included in core-method density corpus?**	**Rule**
Prompting and task-expression methods	Yes	Included if the method changes task formulation, instruction, query, output format, or input representation at inference time.
Retrieval and context-construction methods	Yes	Included if retrieval, ranking, or context assembly supplies information for the current inference episode.
Demonstration-based in-context learning methods	Yes	Included if examples are selected, ordered, compressed, or retrieved to condition current inference behavior.
Test-time adaptation methods	Yes	Included if the method updates parameters, statistics, prompt parameters, adapters, classifiers, or other inference-relevant state using test samples, batches, or streams.
Activation or representation steering methods	Yes	Included if the method steers hidden states, activation directions, logits, or internal representation trajectories during inference.
Memory and experience-state methods	Yes	Included if the method writes, retrieves, updates, compresses, routes, forgets, or reuses state across inference episodes.
Reasoning, search, and test-time scaling methods	Yes	Included if the method reorganizes inference through traces, candidates, search, sampling, scoring, pruning, or budget allocation.
Feedback, verifier, and reward methods	Yes	Included if feedback, critique, reward, verifier, execution check, or judge signal is used to revise, select, score, or supervise inference behavior.
Tool-use and agentic systems	Yes	Included if tools, APIs, code execution, web interaction, environments, or agent roles are used during inference.
Pure pretraining papers	No, unless inference-coupled	Excluded if the contribution is only pretraining without an inference-stage mechanism.
Offline fine-tuning only	No, unless inference-coupled	Excluded if the method improves a model only through offline training and does not directly change testing or inference behavior.
Dataset or benchmark papers	No for density	Used as evaluation context, but not counted as core TTI mechanism papers.
Survey papers	No for density	Used to position adjacent fields and compare scope, but not counted in heatmap or temporal-density statistics.

Table A4: Inclusion and exclusion criteria (continued).

Paper type	Included in core-method density corpus?	Rule
Analysis papers without a new inference-stage mechanism	Usually no for density	Cited when useful for bottleneck or failure-mode discussion, but not counted as core-method records unless they define a mechanism used in inference.

This distinction is important for interpreting the density visualizations. The heatmap and temporal-density plot are based on core-method records rather than all references in the bibliography. Therefore, the count matrices below should be interpreted as reviewed-paper concentration within the screened mechanism corpus, not as total publication volume in the broader field.

A.3 Coding Procedure

Each core-method record was coded using the framework introduced in Section 2. The primary coding fields record the publication year, mechanism branch, intervention locus, information source, temporal persistence, inference-time cost, and secondary mechanism links. These fields support three uses in the main text: the Locus × Source heatmap, the temporal-density visualization, and the opportunity-cell screening.

The coding process followed three principles. First, each core-method paper was assigned to one primary mechanism branch. The branch describes the main functional role of the method in the TTI system, such as prompt and task expression, context construction, test-time adaptation, activation steering, memory, reasoning decomposition, search, verification, tool use, or multi-agent coordination. Second, each core-method paper was assigned to one primary Locus × Source cell for count construction. This primary cell reflects the main intervention object and the main source of useful information. When a paper combined multiple mechanisms, the primary cell was chosen according to the component most central to the paper's claimed test-time improvement. Third, secondary links were recorded separately for hybrid methods. A method may combine retrieval with feedback, tool use with memory, or reasoning search with verifier guidance. These secondary links are used in the migration analysis, but they are not double-counted in the heatmap.

Table A5 summarizes the fields recorded for each core-method record.

Table A5: Coding fields for the reviewed corpus.

Coding field	Purpose in this survey
Title	Identifies the paper or method record.
Publication year	Used for temporal-density visualization.

Table A5: Coding fields for the reviewed corpus (continued).

Coding field	Purpose in this survey
Venue or source	Records whether the work appeared in a conference, journal, arXiv, benchmark, survey, or other source.
Paper role	Distinguishes core-method, anchor, representative, emerging, extension, system, survey, benchmark, analysis, and background/exclusion records.
Mechanism branch	Organizes papers into L1.A–L3.E intervention families.
Main intervention locus	Determines the Locus axis in the heatmap: L1 Input, L2 State, or L3 Process.
Main information source	Determines the Source axis in the heatmap: S0–S4.
Secondary mechanism links	Records hybrid connections used for migration analysis, such as context-to-memory, reflection-to-verifier, or tool-loop-to-experience migration.
Temporal descriptor	Records whether the intervention operates within a single episode, across multiple rounds, across test streams, or across persistent sessions.
Cost descriptor	Records the main inference-time cost, such as single forward pass, multi-decoding, online gradient update, search, or tool/agent interaction.
Counted in heatmap?	Indicates whether the record enters the Locus × Source count matrix.
Counted in temporal density?	Indicates whether the record enters branch-year temporal-density aggregation.
Exclusion/background note	Explains why a survey, benchmark, analysis paper, or non-TTI paper is not counted as a core mechanism record.

The temporal and cost descriptors are not used as primary axes in the heatmap, but they provide auxiliary information for the resource-control discussion in Section 4. For example, two papers may both be assigned to L3 × S4, but one may require only a local self-refinement loop while another may require verifier-guided search, reward modeling, or online reinforcement learning. The auxiliary descriptors therefore help interpret the type of inference-time resource introduced by each mechanism.

Algorithm A1 summarizes the literature search, screening, and coding workflow.

A.4 Assignment Rules

The Locus × Source map is used to position TTI methods according to where the intervention acts and where the useful inference-stage information comes from. Locus records the main intervention object, while Source records the origin of the information or signal that enables the intervention.

For count construction, each core-method record is assigned to one primary Locus × Source cell. Hybrid methods are not double-counted. Instead, secondary mechanism

Algorithm A1 Literature search, screening, and coding protocol.

Input: Search sources, major venues, keyword groups, citation links, and publication years 2020–2026.
Output: Screened core-method corpus; Locus × Source count matrix; mechanism-branch temporal-density matrix; opportunity-cell examples.
Search scholarly databases, major venues, and preprint repositories using mechanism-oriented keyword groups.
Expand candidate records through backward and forward citation chasing.
Merge duplicate records and multiple versions of the same work when identifiable.
Screen records for topical relevance to large models, multimodal models, and inference-stage mechanisms.
Apply the TTI boundary conditions defined in Section 2.1.
Classify retained records by role: core-method, system, anchor, representative, emerging, extension, survey, benchmark, analysis, or background/exclusion.
Exclude survey, benchmark, analysis, and background-only records from density construction.
For each core-method record, code publication year, mechanism branch, primary locus, primary source, temporal descriptor, cost descriptor, and secondary mechanism links.
Assign each core-method record to one primary Locus × Source cell for count construction.
Aggregate primary cell counts for the Locus × Source heatmap.
Aggregate branch-year counts for temporal-density visualization.
Screen candidate opportunity cells using mechanism linkage, resource-control relevance, and evaluability.

links are recorded separately and used for the migration analysis in Section 3.4 and the opportunity-cell discussion in Section 5.4.

Tables A6 and A7 summarize the assignment rules for the Locus and Source axes.

Table A6: Locus assignment rules.

Locus	**Assignment rule**	**Representative mechanism examples**
L1 Input	The primary intervention changes the information presented to the model before or during the current inference episode.	Prompt rewriting, instruction design, query rewriting, RAG context assembly, demonstration selection, input or view reconstruction.
L2 State	The primary intervention updates, steers, stores, retrieves, or preserves an inference-relevant state.	Test-time parameter/statistic adaptation, prompt-parameter updates, activation steering, external memory, experience state, continual adaptation.

Table A6: Locus assignment rules (continued).

Locus	Assignment rule	Representative mechanism examples
L3 Process	The primary intervention reorganizes the procedure through which outputs, actions, or decisions are produced.	CoT, self-consistency, search, verifier-guided decoding, self-refinement, tool use, web agents, code execution, multi-agent coordination.

Table A7: Source assignment rules.

Source	Assignment rule	Representative examples
S0 Internal-only	The useful signal comes from internal model knowledge, generated samples, reasoning traces, activations, logits, or latent states, without adding new external information during the inference episode.	CoT, self-consistency, activation steering, latent reasoning, prompt-only reasoning.
S1 Static-KB	The useful signal comes from a static corpus, knowledge base, retriever index, knowledge graph, document collection, or prebuilt external resource.	RAG, REALM, Atlas, GraphRAG, KG-RAG, medical RAG, retrieval-based memory.
S2 Live-tool/env	The useful signal comes from live tools, APIs, web pages, code execution, interactive environments, visual tools, or environment observations.	ReAct, Toolformer, web agents, code agents, visual programming, tool-use agents.
S3 Test-stream TTA	The useful signal comes from the current test sample, test batch, online test stream, or non-stationary deployment stream.	TTT, TENT, CoTTA, MEMO, TPT, black-box TTA, continual TTA.
S4 Self/Verifier	The useful signal comes from critique, reflection, verifier scores, reward models, LLM judges, execution checks, self-generated supervision, or evaluative feedback.	Self-Refine, Reflexion, verifier-guided decoding, reward-guided reasoning, self-distillation, executable verification.

For hybrid systems, the primary plotted cell is selected according to the component most central to the method's claimed improvement. Secondary links are recorded to preserve the hybrid nature of the method. Table A8 illustrates how this rule is applied.

Table A8: Examples of primary-cell assignment for hybrid methods.

Method type	Primary assignment logic	Example primary cell	Secondary links
Retrieval with self-critique	If the paper's main contribution is context construction, assign to L1 × S1; if the main contribution is critique or feedback control, assign to L3 × S4.	Self-RAG may be coded according to its dominant role in the analysis.	Retrieval ↔ critique; context construction ↔ feedback.
Agent reflection with memory	If the main mechanism is feedback-driven revision during task execution, assign to L3 × S4; if the main mechanism is persistent storage, assign to L2 × S4.	Reflexion is typically treated as L3 × S4 when analyzed as feedback/refinement.	Feedback → episodic memory.
Tool-use agent with reusable skill memory	If the main contribution is tool/environment action, assign to L3 × S2; if the main contribution is reusable tool or skill state, assign to L2 × S2.	Voyager-like systems may be discussed through both links but counted once.	Tool loop → experience state.
Classical online test-time adaptation	If the model updates inference-relevant parameters or statistics from test samples or streams, assign to L2 × S3.	TTT, TENT, CoTTA.	Local update → continual state stabilization.
Verifier-guided search	If the central mechanism is candidate-space exploration, assign to L3 × S4 when verifier/reward signal drives selection or pruning.	Verifier-guided decoding, reward-guided search, compute-optimal TTS.	Search ↔ evaluation; compute budget ↔ verifier reliability.

These rules reduce ambiguity in the heatmap and prevent the same paper from being counted multiple times. At the same time, secondary links allow the survey to discuss mechanism migration without forcing hybrid systems into overly rigid single-label categories.

A.5 Density and Opportunity Screening

A.5.1 Locus × Source Heatmap Construction

The Locus × Source heatmap is constructed from the primary cell assignments of the screened core-method corpus. Each core-method record contributes one count to one primary Locus × Source cell. Survey papers, benchmark papers, general analysis papers, and background-only records are not included in the heatmap count, although they may be cited elsewhere in the paper.

Table A9 reports the current per-cell count matrix computed from the screened coding workbook. The total number of core-method records used for this matrix is 272.

Table A9 Locus × Source count matrix for the screened core-method corpus.

Locus × Source	S0 Internal	S1 Static	S2 Tool/env	S3 Stream	S4 Verifier	Row total
L1 Input	28	41	4	1	8	82
L2 State	11	13	9	18	6	57
L3 Process	27	24	34	0	48	133
Column total	66	78	47	19	62	272

The heatmap color intensity in Figure 3 is derived from these counts but is interpreted ordinally rather than as an exact bibliometric measurement. High-count cells indicate that the screened reviewed corpus contains a relatively larger concentration of core-method records and that the corresponding mechanism family is more consolidated. Low-count cells indicate sparser or less systematized regions within the screened corpus. The heatmap should therefore be read as a positioning device for mechanism-level synthesis, not as a performance ranking or complete field-wide publication statistic.

Table A10 defines the ordinal density levels used for heatmap interpretation.

Table A10 Ordinal density levels used for heatmap interpretation.

Count range in a Locus × Source cell	Density level	Interpretation
0	Empty or not observed in the screened corpus	No core-method record assigned to the cell in the current corpus.
1–4	Sparse	Isolated or boundary examples; mechanism form not yet stable.
5–10	Emerging	Repeated examples appear, but mechanism and evaluation practices remain less consolidated.
11–24	Active	A visible research region with recurring method forms and evaluation concerns.
25–40	Dense	A strongly represented region with multiple mechanism variants and system extensions.
> 40	Highly concentrated	A central region of the current reviewed corpus with extensive accumulated work.

This ordinal scale is used to avoid overclaiming precision. It supports statements such as "relative concentration," "visible accumulation," or "emerging activity," but not claims about complete publication volume in the entire field.

A.5.2 Temporal Density Construction

Temporal density was constructed by aggregating the screened core-method records by mechanism branch and publication year or period. The resulting matrix supports the temporal visualization in the main text. Because the corpus is designed for mechanism-level synthesis rather than exhaustive bibliometric measurement, the temporal density levels indicate relative concentration within the reviewed corpus. If 2026 papers are retained in the final manuscript, 2026 should be interpreted as a partial-year observation and the final corpus update date should be reported.

Because the corpus was last updated in May 2026, the 2026 counts should be interpreted as partial-year observations rather than full-year publication counts.

Table A11: Mechanism-branch temporal counts in the screened core-method corpus.

Mechanism branch	2020–2022	2023	2024	2025	2026	Total
L1.A Prompt and task expression	4	9	12	6	4	35
L1.B Context assembly and RAG	2	4	26	5	5	42
L1.C Demonstration-based ICL	1	0	0	2	2	5
L2.A Test-time parameter/statistic adaptation	6	1	1	2	2	12
L2.B Activation and representation steering	1	6	3	1	1	12
L2.C External memory and experience state	1	7	7	6	5	26
L2.D Continual adaptation and state stabilization	2	3	1	1	0	7
L3.A Reasoning decomposition	3	1	8	5	0	17
L3.B Search and candidate exploration	0	2	7	10	1	20
L3.C Feedback, verification, and refinement	1	5	17	15	5	43
L3.D Tool use and agentic action	2	3	9	11	5	30
L3.E Multi-agent and role-based inference	0	2	11	7	3	23
Total	23	43	102	71	33	272

Table A11 supports three observations used in the main text. First, early works concentrate around prompting, RAG, TTA, and explicit reasoning foundations. Second, 2024 shows a strong accumulation in RAG, feedback, tools, and multi-agent mechanisms. Third, the later period shows stronger activity around verifier/reward

mechanisms, tool and agentic action, memory, and search-based inference. These patterns are interpreted as relative trends within the screened corpus rather than as complete annual publication statistics.

Table A12 reports the corresponding locus-level temporal counts.

Table A12 Locus-level temporal counts in the screened core-method corpus. The 2026 entries reflect partial-year data as of May 2026.

Locus	2020–2022	2023	2024	2025	2026	Total
L1 Input	7	13	38	13	11	82
L2 State	10	17	12	10	8	57
L3 Process	6	13	52	48	14	133
Total	23	43	102	71	33	272

These locus-level counts support the landscape reading that process-level TTI remains highly visible across the timeline, while input-level work peaks around the RAG and context-construction wave and state-level work becomes more important as memory, experience state, continual adaptation, and self-evolving mechanisms develop.

A.5.3 Opportunity Cell Screening

Opportunity cells are not selected solely because they appear visually sparse in the heatmap. A sparse cell is not automatically a research opportunity, and a dense cell may still contain unresolved problems. In this survey, candidate opportunity cells are screened using three criteria.

First, mechanism linkage assesses whether the cell can be connected to an identifiable mechanism trajectory discussed in the mechanism review. Examples include context-to-memory migration, tool-loop-to-experience migration, reflection-to-verifier migration, and local TTA-to-continual-state migration. Second, resource-control relevance evaluates whether the cell exposes a concrete inference-time control problem. Examples include budget allocation, stopping rules, memory writing and forgetting, tool-selection reliability, verifier calibration, reward hacking, pseudo-label drift, or stream-aware reset policies. Third, evaluability assesses whether the cell can be studied through tractable tasks, baselines, cost reports, stability analyses, stream protocols, ablation studies, verifier diagnostics, or long-horizon evaluation.

A candidate cell is retained as an opportunity example only if it satisfies all three criteria. Table A13 reports the representative opportunity-cell screening examples used in the main text.

This screening procedure makes the selection of opportunity cells more explicit by requiring each candidate region to be linked to a mechanism trajectory, a resource-control problem, and feasible evaluation settings. In this way, opportunity cells are

Table A13 Representative opportunity-cell screening examples.

Candidate signal	Candidate cell	Mechanism linkage	Resource-control relevance	Evaluation handle
Emerging activity signal	L2 × S2: tool and experience memory	Tool-use traces, deployment cases, and interaction histories become reusable state.	What tool experience should be stored, updated, retrieved, routed, or forgotten?	Long-horizon tool-use tasks; tool-selection accuracy; experience-reuse ablations; failure-recovery evaluation.
Emerging activity signal	L2 × S4: feedback-driven state update	Feedback, verifier, reward, or reflection signals guide adaptive state or memory.	How to control evaluator error, pseudo-label drift, reward hacking, and update reliability?	Verifier diagnostics; held-out evaluation; reward stability tests; stream stability tests.
Structural gap	L3 × S3: test-stream-aware process control	Test-stream or reliability signals guide search, selection, retrieval, interaction, or stopping policies.	How should search budget, reset policy, tool invocation, or intermediate verification change under non-stationary streams?	Online or continual benchmarks; cost-aware protocols; stream-level ablations; reliability-aware stopping tests.

treated as analytically selected research positions rather than as arbitrary gaps in a map.

A.5.4 Interpretation Limits

The density analyses in this appendix are protocol-dependent. They depend on the search sources, keyword groups, corpus update date, inclusion/exclusion criteria, and coding decisions described above. They are therefore intended to support qualitative landscape analysis rather than exact bibliometric measurement. The reported counts make the construction of the heatmap and temporal-density visualization auditable, but they should not be interpreted as complete publication statistics for the entire TTI field.

For this reason, the main text describes the figures using terms such as "relative concentration," "reviewed-paper density," "emerging activity," and "mechanism consolidation," rather than "complete publication volume" or "field-wide bibliometric distribution."