Fast or Slow? Human-Inspired Self-Evolving Framework for Resilient AI Systems

Haoran Qiu[1], Phuong Cao[2,3], Shengkun Cui[2], Archit Patke[2], and Ravishankar K. Iyer[2]
[1]Microsoft Azure Research, [2]UIUC, [3]National Center for Supercomputing Applications

***Abstract*—This paper proposes a disruptive shift toward human-like self-evolving loops as a foundation for resilient AI systems. At the core of our proposal is the PURER loop (Perceive, Update, Reason, Execute, Reflection), a cognitive-inspired framework that enables intelligent AI agents to treat experiences (including failures) as fundamental learning triggers. PURER is centered on five tightly coupled stages: (1) Perceive by gathering raw and engineered states, (2) Update by revising memory and internal world models, (3) Reason by planning and prioritizing actions based on reward models, (4) Execute by enacting behaviors via tool calls, and (5) Reflect by learning from outcomes to adjust future behavior. Unlike current static models that rely on periodic updates via retraining, ad-hoc recovery pipelines, or vulnerable fine-tuning practices, PURER enables continuous exploration, functionality transfer, and adaptation without manual intervention.**

***Index Terms*—Resilience, Verification, Dependability, Agents.**

I. Introduction

Despite their impressive capabilities, most deployed AI systems rely on reactive dependability mechanisms where failures are detected via fixed guardrails [36], mitigated through predefined rules [32], and resolved only through manual, offline intervention [25], [26], [44]. This paradigm does not scale; as systems grow more complex, failures become harder to enumerate, and human-in-the-loop remediation becomes prohibitively expensive. Furthermore, while advances in agentic frameworks such as ReAct [47] and search-based reasoning [39] enable planning, they lack a principled, system-level approach to learning from failures at runtime. In contrast, human decision-making, though often slower, is remarkably robust because it treats failure as a primary learning mechanism [4]. By continuously perceiving signals, revising internal models, and reflecting on outcomes [12], [16], [35], humans internalize errors to adapt functionality to new situations [15], [24], a self-evolving loop currently missing in modern AI.

We propose a disruptive shift toward human-like, self-evolving loops for resilient AI systems, embodied in the **PURER** loop: Perceive, Update, Reason, Execute, and Reflection. Unlike reactive dependability mechanisms—where failures are detected, masked, or rolled back using predefined rules—PURER treats operational experience, including faults, degradations, and near-misses, as a first-class signal for improving system behavior over time. Instead of relying on manual retraining, static recovery policies, or one-off mitigations, a PURER-driven system continuously updates internal state and memory, reasons about alternative recovery actions via a reward-based system model, executes the optimal recovery behavior through tools, and reflects on outcomes to reduce the likelihood, impact, and recurrence of future failures via optimal, in-flight decision making. Figure 1 provides an overview of the PURER framework. We are integrating PURER into a Top 500 supercomputer, powered by NVIDIA H200 and Blackwell GPUs, to validate the concept using realistic AI training, inference, and HPC workloads.

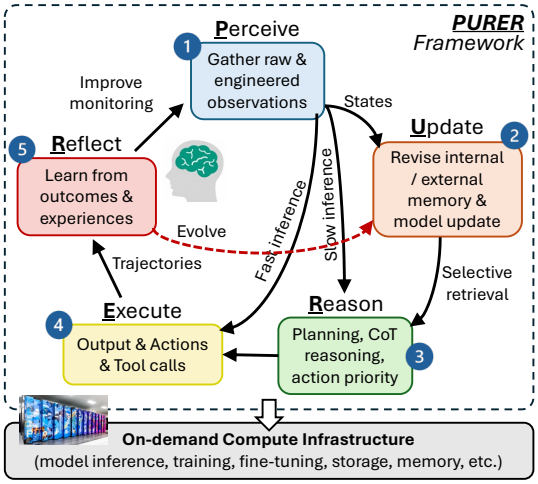

Fig. 1. A human-inspired self-evolving framework for LLM-powered AI systems: (1) *Perceive* gathers raw sensor signals and observations; (2) *Update* evolves system state through selective internal and external memory and model updates; (3) *Reason* performs planning, chain-of-thought reasoning, and action prioritization; (4) *Execute* issues actions and tool calls; and (5) *Reflect* learns from outcomes to improve monitoring and guide future evolution.

Rather than presenting a complete architecture, this paper articulates a conceptual framework and research agenda for *self-evolving dependability* in AI systems. We examine how recent advances in LLMs, search-based reasoning, and self-reflective agents can serve as building blocks for resilient operation, while highlighting open system-level challenges central to the DSN community: efficiency overheads, stability under feedback loops, failure amplification, context drift, and safety under continuous adaptation. We argue that dependability for generative AI (e.g., LLM)-powered systems must move beyond static correctness and predefined recovery paths, toward a learning-driven, failure-informed process that improves reliability, availability, and safety in deployments. This perspective opens a new research frontier for DSN, bridging classical fault tolerance with adaptive evolution.

II. Limitations of Today's Approaches

We examine why dominant reactive dependability mechanisms for LLM systems fail to provide sustained robustness under evolving, adversarial, or long-horizon conditions.

Static Guardrails and Rules. Rule-based guardrails and safety filters are widely deployed in LLM systems [2], [32], [36], relying on static heuristics, keyword matching, or classifiers to block undesired behaviors, offering limited robustness.

Guardrails require failure modes to be enumerated *a priori*, despite acceptable behavior being highly context-dependent across tasks and domains, leading to brittle policies and frequent false positives [42]. False negatives are equally common: adversarial prompting, indirect instruction following, and multi-step jailbreaks routinely bypass filters [44], [49].

Critically, guardrails are static. Once deployed, they remain unchanged until manually updated, allowing adversaries to iteratively exploit uncovered weaknesses. This rigidity has made prompt injection a pervasive real-world vulnerability [43], yielding fragile and short-lived guarantees.

Offline Post-Training Recovery. Failures that evade runtime defenses are typically addressed offline via fine-tuning, retraining, or prompt updates [25], [26]. While sometimes effective, these approaches are slow, costly, and reactive.

Large-model updates require significant compute, labeled data, and long turnaround times. Sequential fine-tuning can also cause *catastrophic forgetting*, degrading previously learned behaviors [15], [24], and may fail to improve robustness under distribution shift or even worsen hallucinations [26]. More fundamentally, offline recovery treats failures as isolated events rather than structured learning signals, resulting in repeated human intervention and limited scalability.

Contextual and Memory-Based Mitigations. To address statelessness, recent systems employ long context windows, external memory, or experience summarization [11], [39], [47]. These mechanisms provide limited temporal continuity but do not enable true adaptation. Context-based memory is bounded by finite windows and attention degradation, leading to context drift or silent loss of critical state [5], [46]. External memory can be incomplete, stale, or poorly aligned with current reasoning needs [14], while summarization irreversibly discards potentially relevant details. Most importantly, these mechanisms do not modify the model's internal decision process: past failures may be retrieved, but they do not influence future action selection, allowing similar errors to recur.

Need for a Paradigm Shift. Overall, today's dependability pipelines are fundamentally reactive and rely on fixed detection and recovery mechanisms that neither anticipate novel failures nor learn persistently from experience. As LLM systems face adversarial situations and long-horizon interactions, these limitations motivate new paradigms that treat adaptation and learning as first-class runtime concerns.

III. The PURER Framework

Figure 1 illustrates the core design of PURER, a self-evolving decision framework inspired by human resilience under uncertainty and failure. PURER departs from conventional reactive or statically optimized AI systems by treating failure, environmental change, and partial observability as first-class signals for adaptation. Rather than optimizing a fixed policy or model in isolation, PURER continuously co-evolves its internal components—models, memory, context, and tools—through a closed-loop interaction with the environment.

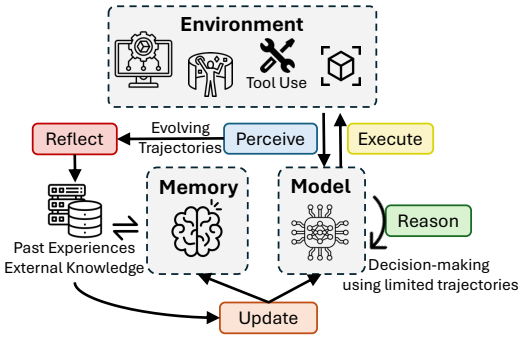

Fig. 2. Deep dive into system-state evolution: from perception to execution. The key components are: an adaptive memory integrated with a generative model interacting with the environment.

Human decision-making under stress is often slower and less optimized for immediate performance, yet remarkably robust over time. This resilience emerges from several key properties that motivate the PURER design.

First, humans engage in *slow but robust* decision-making when operating under uncertainty, prioritizing adaptability over short-term optimality. Second, they continuously integrate new observations with prior experience, transferring lessons learned across tasks, contexts, and time horizons. Third, and most critically, failure is not treated as an exception to be masked or merely reacted to, but as a valuable signal that informs future behavior.

PURER operationalizes these principles by embedding learning, reflection, and structural adaptation directly into the decision loop, enabling systems to remain effective even under distribution shift, incomplete information, and evolving operational constraints.

A. Core Components

Figure 2 presents the essential components of PURER using the same five overarching phases:

- **Model:** The core decision-making and reasoning substrate, which may include Dense LLMs or MoEs.
- **Memory:** Both *internal memory* (e.g., latent states, short-term context) and *external memory* (e.g., RAG database and knowledge base) that the Model can refer to.
- **Environment:** The external world with which PURER interacts, including tools, APIs, and simulators.
- **Trajectory / Experience Database:** A structured repository of past interactions, outcomes, and rewards used to support evaluation, reflection, and continuous evolving.

B. The PURER Loop: A Self-Evolving Decision Framework

At its core, PURER consists of five phases: Perceive, Update, Reason, Execute, and Reflection. These phases are orchestrated to interact with a dynamic environment and continuously adapt system behavior over time.

The Perceive phase ingests raw and engineered observations from both the system and its environment. This includes sensor

inputs, system logs, tool outputs, intermediate model signals, and contextual metadata. Perception is intentionally broad, allowing PURER to operate under partial observability and heterogeneous data sources.

In the Update phase, PURER revises its internal state based on new observations. This includes updating internal memory representations as well as external memory stores such as trajectory or experience databases. Updates may refine world models, summarize recent interactions, or compress long-term experience for efficient reuse.

The Reason phase performs planning, prioritization, and decision-making conditioned on current perceptions, goals, constraints, and learned reward models. Rather than executing a fixed policy, reasoning explicitly weighs alternative actions, uncertainty, and long-term utility, enabling deliberate trade-offs between exploration and exploitation.

During Execute, PURER enacts decisions through concrete actions, including tool invocations, control commands, environment interactions, or behavior generation. Execution may involve external systems, APIs, simulators, or real-world actuators, closing the loop between cognition and action.

Finally, the Reflect phase evaluates outcomes relative to expectations, goals, and rewards. Reflection transforms successes and failures into learning signals, shaping future perception, reasoning, and adaptation. Unlike traditional feedback mechanisms, reflection in PURER explicitly informs *what should evolve next* within the system.

C. Self-Evolution via Reinforcement Learning

A key innovation of PURER is that reinforcement learning is not solely used to optimize action selection, but to decide (1) *which component of the system should evolve*, and (2) *how to evolve*. Based on observed states, failure patterns, and long-term rewards, PURER adaptively chooses to: (1) update or reorganize external memory, (2) modify context construction or perception strategies, (3) fine-tune or replace (partial) model parameters, (4) or evolve the environment itself, such as introducing new tools, abstractions, or execution pathways.

This meta-level learning enables PURER to co-evolve its cognition, memory, and operational substrate, rather than assuming a static system boundary. As a result, PURER supports sustained robustness in long-running, safety-critical, and rapidly changing environments.

IV. Validating PURER on AI Live Traffic Testbed

We plan to validate PURER on a live traffic testbed on a supercomputer ranked in the Top 500 that includes realistic AI training, inference, and HPC traffic at a national scale. The testbed encompasses data collection, analytics, detection, and probationary environments for experimentation with different resilience scenarios in a 1,200 Gbps network.

A. Perceiving AI Traffic for Updating Belief

In the Perceive and Update step of PURER, to evaluate the resilience of full-scale AI systems, we would need to replicate almost the entire AI/scientific workflow; instead, we will develop a fractional system that maintains the essential complexities of the actual AI cyberinfrastructure. For instance, disruptions to AI interconnects and traffic are detected based on mirrored optical network traffic. Specifically, duplicate Ethernet frames on a source interface will be sent to a network observability cluster, e.g., Zeek, in real time. This optical tap (e.g., Arista) will be used on a `/16` CIDR subnet, mirroring both operational and disrupted inference traffic to the testbed.

B. Replaying System Disruptions

In the Reason and Execute step of PURER, we will support the detection and replay of AI-system disruptions, including accidental failures and cyberattacks, in live traffic. The idea of replaying an attack is to enable measurement and quantification of various aspects. While several AI Testbeds exist at national labs [21] have gained attention, no such *resilience* testbed exists in the public domain (academia or industry) in the AI-for-Science domain. In that case, the target of interest (e.g., a faulty GPU or a model with a backdoor) must be integrated into our testbed to collect GPU driver logs.

C. Optimizing Outcomes from Past Recoveries

Finally, in the Reflect step, we optimize the timing and selection of which monitor to activate to maximize information entropy about the user, while minimizing disruption risks (system misuse and data exfiltration). We define the objective function $f(x,t)$ as the balance between the information entropy $H(U|M_x)$ and the risk function $R(M_x)$, where M_x is the subset of active monitors. Near-miss and latent failure signals—such as uncertainty spikes and guardrail activations—are inferred from internal traces and incorporated into $H(U|M_x)$ as proxy indicators of potential failure. If we view this as a game against an adversary, we can formalize it as: $\min_R \max_{x,t} [H(U|M_x,t) - \lambda R(M_x,t)]$, where: $H(U|M_x,t)$ is the Information Entropy gained at time t using the selected monitors; $R(M_x,t)$ is the Risk Function, quantifying the potential for disruptions, whereas λ is a regularization parameter to balance resilient and data collection.

D. Risk Mitigation

To reduce the risk that the PURER loop may introduce overhead for AI systems, the reasoning framework will be executed in massively parallel threads on a virtual GPU workstation (VGM). We provide an immutable VGM image for launch on the testbed. Each instance of the VGM is short-lived to reduce the risk of permanent compromise and can be quickly provisioned after successful collection of AI traces. This setup also enables auto-scaling of a network of VGM instances, e.g., simulating a distributed federation of databases, allowing us to capture realistic fault propagation, derive valuable insights into the origin of the fault, and assess the potential impacts of the traces obtained.

V. Going Forward

In this section, we examine how existing techniques partially support individual stages of the PURER loop.

A. Enabling Technologies and Where They Fall Short

We organize prior work around the stages of the PURER loop, with emphasis on Update, Reason, and Reflect, as these phases are central to sustained adaptation and dependability.

LLMs as Abstraction, Reasoning, and Synthesis Engines. LLMs already serve as powerful substrates for abstraction, semantic compression, and cross-modal synthesis. Within PURER, they naturally support the Reason phase by providing a natural language interface and enabling goal-conditioned planning, decomposition of complex tasks, and synthesis of action sequences from unstructured observations. They also partially support Update by summarizing experience, distilling trajectories, and maintaining short-term contextual coherence.

However, these capabilities remain fundamentally *context-bound*. As interaction histories grow, LLM performance degrades due to context truncation, attention dilution, and *lost-in-the-middle* effects [17], [28], [33]. More importantly, LLMs lack intrinsic mechanisms for deciding *what should be updated*, *where learning should reside*, or *when adaptation is warranted*. Model weights, context windows, and external memory are treated as static substrates rather than decision variables. As a result, updates are triggered via ad hoc heuristics, leaving systems brittle under long-horizon deployments.

Agent Frameworks. Existing agent frameworks primarily support ReAct-style agents, tool-augmented planners, and self-refining execution loops [6], [22], [23], [29], [31], [37], [47]. These frameworks typically assume a fixed architecture: a static model, an attached context window, a predefined toolset, and reactive control flow. While effective for short-horizon tasks, these assumptions conflict with PURER's objective of long-term resilience. In particular, current frameworks lack: (1) explicit representations of experience trajectories as first-class learning objects, (2) mechanisms to arbitrate between context adaptation, memory updates, and model evolution, and (3) principled reward or feedback signals that couple task success with system-level robustness. Consequently, adaptation remains reactive rather than strategic.

Insight 1: *Adaptation as a Control Decision. PURER uses reward-driven control to decide* ***when to reason, reflect, or update****, and* ***where learning should reside****—context, memory, or model weights—to achieve long-horizon robustness.*

Structured Reasoning and Self-Reflective Agents. Recent work on dependency-structured reasoning [7], [9], [19], [40], [41] has significantly improved long-horizon planning by explicitly modeling task structure and context dependencies. For example, Cui et al. [9] impose hierarchical graphs for cloud incident analysis, outperforming ReAct while reducing token consumption. These approaches strengthen the Reason phase by enabling explicit state transitions, dependency tracking, and backtracking for correction. Self-reflective agents introduce nascent mechanisms for critique, outcome evaluation, and iterative refinement [1], [3], [8], [10], [13], [18], [20], [27], [34], [45], [48]. These systems improve task-level performance while exposing a rich space of research problems.

Despite recent progress, current reflection mechanisms remain shallow and mechanistic, bearing little resemblance to how learning and recovery occur in the human brain. In humans, when a region deteriorates or fails—as in neurodegenerative conditions such as Alzheimer's—other regions can gradually assume lost functionality through sustained learning, adaptation, and repeated exercise [30], [38]. By contrast, LLM agents often stall at local optima, lacking mechanisms to redistribute or reconstitute capabilities by transferring experience from prior agents or progressively training new ones to recover lost functionality or further evolve.

Insight 2: *Experience Transfer Across Agents. Enable new agents or models to progressively recover lost or failed capabilities by transferring knowledge, failure signals, and learned behaviors from prior agents, breaking local optima and supporting* ***long-term*** *functional evolution.*

B. System-Level Challenges

Beyond algorithmic limitations, PURER exposes a set of system-level challenges that must be addressed to enable continuous adaptation at scale.

Memory, Context, and Model Updates as a Unified Decision Problem. A central challenge is deciding *where learning should reside*. Updating external memory is cheap but may suffer from retrieval inefficiencies; reshaping context improves short-term reasoning but is ephemeral; updating model weights provides durable learning but incurs high computational and safety costs. PURER reframes this trade-off as a runtime decision problem across timescales, requiring new abstractions to reason about learning cost, benefit, and risk.

Stability, Safety, and Preventing Pathological Adaptation. Self-evolving systems risk destabilizing feedback loops, reward hacking, or unintended behavioral drift. Ensuring that adaptation remains bounded, interpretable, and aligned with system objectives requires safeguards that constrain evolution without eliminating flexibility, such as meta-policies, invariants, or human-in-the-loop oversight.

Observability and Correctness Under Continuous Evolution. As systems evolve internally, maintaining observability and correctness becomes increasingly difficult. Debugging and auditing require tracking not only actions, but the evolution of memory, models, and environment interactions over time. This calls for new tooling to introspect adaptive systems and reason about correctness under non-stationarity.

Insight 3: *Novel System-Level Abstractions. Design runtime* ***mechanisms*** *and* ***abstractions*** *that jointly manage memory, context, model updates, and resources, while ensuring stability, observability, and correctness, enabling self-evolving agents to adapt safely and efficiently at scale.*

Taken together, these challenges suggest that achieving human-like resilience requires rethinking AI systems as *continuously evolving entities* rather than static pipelines. PURER provides a conceptual and architectural foundation for this shift, while exposing a rich space of open research problems at the intersection of learning, reasoning, and systems design.

REFERENCES

[1] Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, et al. Gepa: Reflective prompt evolution can outperform reinforcement learning. arXiv preprint arXiv:2507.19457, 2025.

[2] Suriya Ganesh Ayyamperumal and Limin Ge. Current state of llm risks and ai guardrails. arXiv preprint arXiv:2406.12934, 2024.

[3] Ali Behrouz, Meisam Razaviyayn, Peilin Zhong, and Vahab Mirrokni. Nested learning: The illusion of deep learning architectures. In Advances in Neural Information Processing Systems (NeurIPS), 2025.

[4] Kaileigh A Byrne, Astin C Cornwall, and Darrell A Worthy. Acute stress improves long-term reward maximization in decision-making under uncertainty. Brain and cognition, 133:84–93, 2019.

[5] Bowen Cao, Deng Cai, and Wai Lam. InfiniteICL: Breaking the limit of context window size via long short-term memory transformation. arXiv preprint arXiv:2504.01707, 2025.

[6] Gohar Irfan Chaudhry, Esha Choukse, Haoran Qiu, Íñigo Goiri, Rodrigo Fonseca, Adam Belay, and Ricardo Bianchini. Murakkab: Resource-efficient agentic workflow orchestration in cloud platforms. arXiv preprint arXiv:2508.18298, 2025.

[7] Liyi Chen, Panrong Tong, Zhongming Jin, Ying Sun, Jieping Ye, and Hui Xiong. Plan-on-graph: Self-correcting adaptive planning of large language model on knowledge graphs. In Advances in Neural Information Processing Systems, volume 37, pages 37665–37691, Vancouver, Canada, 2024. Neural Information Processing Systems Foundation, Inc.

[8] Audrey Cheng, Shu Liu, Melissa Pan, Zhifei Li, Shubham Agarwal, Mert Cemri, Bowen Wang, Alexander Krentsel, Tian Xia, Jongseok Park, et al. Let the Barbarians In: How AI Can Accelerate Systems Performance Research. arXiv preprint arXiv:2512.14806, 2025.

[9] Shengkun Cui, Rahul Krishna, Saurabh Jha, and Ravishankar K. Iyer. PRAXIS: Integrating program analysis with observability for root-cause analysis. In Proceedings of the 56th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). IEEE, 2026. Accepted to appear.

[10] Yihe Deng and Paul Mineiro. Flow-dpo: Improving llm mathematical reasoning through online multi-agent learning. In Workshop on Mathematical Reasoning and AI (MATH-AI), NeurIPS, 2024.

[11] Erhu Feng, Wenbo Zhou, Zibin Liu, Le Chen, Yunpeng Dong, Cheng Zhang, Yisheng Zhao, Dong Du, Zhichao Hua, Yubin Xia, et al. Get experience from practice: LLM agents with record & replay. arXiv preprint arXiv:2505.17716, 2025.

[12] Rasmus Gahrn-Andersen. Informational resilience in the human cognitive ecology. Entropy, 25(9):1247, 2023.

[13] Guoxiu He, Xin Song, and Aixin Sun. Knowledge updating? no more model editing! just selective contextual reasoning. Journal of the ACM (JACM), 2025.

[14] Hangfeng He, Hongming Zhang, and Dan Roth. Rethinking with retrieval: Faithful large language model inference. arXiv preprint arXiv:2301.00303, 2022.

[15] Robert Hester, Janelle Madeley, Kevin Murphy, and Jason B Mattingley. Learning from errors: error-related neural activity predicts improvements in future inhibitory control performance. Journal of Neuroscience, 29(22):7158–7165, 2009.

[16] Jakob Hohwy. Attention and conscious perception in the hypothesis testing brain. Frontiers in Psychology, 3:96, 2012.

[17] Kelly Hong, Anton Troynikov, and Jeff Huber. Context rot: How increasing input tokens impacts llm performance. Technical report, Chroma, July 2025.

[18] Yuxin Jiang, Yufei Wang, Chuhan Wu, Wanjun Zhong, Xingshan Zeng, Jiahui Gao, Liangyou Li, Xin Jiang, Lifeng Shang, Ruiming Tang, Qun Liu, and Wei Wang. Learning to edit: Aligning llms with knowledge editing. In Proceedings of the 2024 Annual Meeting of the Association for Computational Linguistics (ACL), 2024.

[19] Bowen Jin, Chulin Xie, Jiawei Zhang, Kashob Kumar Roy, Yu Zhang, Zheng Li, Ruirui Li, Xianfeng Tang, Suhang Wang, Yu Meng, and Jiawei Han. Graph chain-of-thought: Augmenting large language models by reasoning on graphs. In Findings of the Association for Computational Linguistics: ACL 2024, pages 163–184, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[20] Rohan Kadekodi, Zhan Jin, Keisuke Kamahori, Yile Gu, Sean Khatiri, Noah Bayindirli, Sergey Gorbunov, and Baris Kasikci. Dualtune: Decoupled fine-tuning for on-device agentic systems. In Advances in Neural Information Processing Systems (NeurIPS), 2025.

[21] Argonne National Laboratory. ALCF AI Testbed. https://www.alcf.anl.gov/alcf-ai-testbed.

[22] LangChain. LangChain: Building applications with llms through composability. Python package. Accessed 2026-01-31.

[23] LangChain. LangGraph: Building stateful, multi-actor applications with llms. Python package. Accessed 2026-01-31.

[24] Daeyeol Lee, Hyojung Seo, and Min Whan Jung. Neural basis of reinforcement learning and decision making. Annual review of neuroscience, 35(1):287–308, 2012.

[25] Moxin Li, Wenjie Wang, Fuli Feng, Yixin Cao, Jizhi Zhang, and Tat-Seng Chua. Robust prompt optimization for large language models against distribution shifts. arXiv preprint arXiv:2305.13954, 2023.

[26] Hong Liu, Saisai Gong, Yixin Ji, Kaixin Wu, Jia Xu, and Jinjie Gu. Boosting LLM-based relevance modeling with distribution-aware robust learning. In Proceedings of the 33rd ACM International Conference on Information and Knowledge Management, pages 4718–4725, 2024.

[27] Na Liu, Liangyu Chen, Xiaoyu Tian, Wei Zou, Kaijiang Chen, and Ming Cui. From llm to conversational agent: A memory enhanced architecture with fine-tuning of large language models. arXiv preprint arXiv:2401.02777, 2024.

[28] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. Transactions of the Association for Computational Linguistics, 12:157–173, 2024.

[29] Joao Moura. CrewAI: Orchestrating role-playing, autonomous ai agents. Python package. Accessed 2026-01-31.

[30] Timothy H Murphy and Dale Corbett. Plasticity during stroke recovery: from synapse to behaviour. Nature reviews neuroscience, 10(12):861–872, 2009.

[31] OpenAI. Introducing codex, May 2025. Accessed 2026-01-31.

[32] Ravi Pandya, Madison Bland, Duy P Nguyen, Changliu Liu, Jaime Fernández Fisac, and Andrea Bajcsy. From refusal to recovery: A control-theoretic approach to generative AI guardrails. arXiv preprint arXiv:2510.13727, 2025.

[33] Haoran Qiu, Weichao Mao, Chen Wang Hubertus Franke, Zbigniew T Kalbarczyk, Tamer Basar, and Ravishankar K Iyer. On the promise and challenges of foundation models for learning-based cloud systems management. In Workshop on Machine Learning for Systems at NeurIPS, volume 2023, 2023.

[34] Haoran Qiu, Weichao Mao, Archit Patke, Shengkun Cui, Chen Wang, Hubertus Franke, Zbigniew T Kalbarczyk, Tamer Başar, and Ravishankar K Iyer. FLASH: Fast model adaptation in ML-centric cloud platforms. Proceedings of Machine Learning and Systems, 6:524–544, 2024.

[35] Dana Rad, Monica Maier, Zorica Triff, and Radiana Marcu. Entangled autopoiesis: Reframing psychotherapy and neuroscience through cognitive science and systems engineering. Brain Sciences, 15(10):1032, 2025.

[36] Traian Rebedea, Razvan Dinu, Makesh Narsimhan Sreedhar, Christopher Parisien, and Jonathan Cohen. Nemo guardrails: A toolkit for controllable and safe LLM applications with programmable rails. In Proceedings of the 2023 conference on empirical methods in natural language processing: system demonstrations, pages 431–445, 2023.

[37] Yeonju Ro, Haoran Qiu, Íñigo Goiri, Rodrigo Fonseca, Ricardo Bianchini, Aditya Akella, Zhangyang Wang, Mattan Erez, and Esha Choukse. Sherlock: Reliable and efficient agentic workflow execution. arXiv preprint arXiv:2511.00330, 2025.

[38] Emily Singer. In natural networks, strength in loops. Quanta Magazine, 2013.

[39] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. arXiv preprint arXiv:2408.03314, 2024.

[40] Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel Ni, Heung-Yeung Shum, and Jian Guo. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In International Conference on Learning Representations (ICLR), 2024.

[41] Xingyu Tan, Xiaoyang Wang, Qing Liu, Xiwei Xu, Xin Yuan, and Wenjie Zhang. Paths-over-graph: Knowledge graph empowered large language model reasoning. In Guodong Long, Michale Blumestein,

Yi Chang, Liane Lewin-Eytan, Zi Helen Huang, and Elad Yom-Tov, editors, Proceedings of the ACM on Web Conference 2025 (WWW 2025), Sydney, NSW, Australia, 28 April 2025- 2 May 2025, pages 3505–3522. ACM, 2025.

[42] Aayush Atul Verma, Amir Saeidi, Shamanthak Hegde, Ajay Therala, Fenil Denish Bardoliya, Nagaraju Machavarapu, Shri Ajay Kumar Ravindhiran, Srija Malyala, Agneet Chatterjee, Yezhou Yang, et al. Evaluating multimodal large language models across distribution shifts and augmentations. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 5314–5324, 2024.

[43] Jaswanth Reddy Vulchi and Eric Ackerman. Exploring owasp top 10 security risks in LLMs with practical testing and prevention. 2024.

[44] Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does LLM safety training fail? Advances in Neural Information Processing Systems, 36:80079–80110, 2023.

[45] Yicheng Xiao, Lin Song, Rui Yang, Cheng Cheng, Yixiao Ge, Xiu Li, and Ying Shan. Lora-gen: Specializing large language models via online lora generation. In International Conference on Machine Learning (ICML), 2025.

[46] Ke Yang, Yao Liu, Sapana Chaudhary, Rasool Fakoor, Pratik Chaudhari, George Karypis, and Huzefa Rangwala. Agentoccam: A simple yet strong baseline for LLM-based web agents. arXiv preprint arXiv:2410.13825, 2024.

[47] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In The eleventh international conference on learning representations, 2022.

[48] Alex L. Zhang, Tim Kraska, and Omar Khattab. Recursive language models, 2026.

[49] Boyang Zhang, Yicong Tan, Yun Shen, Ahmed Salem, Michael Backes, Savvas Zannettou, and Yang Zhang. Breaking agents: Compromising autonomous LLM agents through malfunction amplification. In Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 34952–34964, 2025.