Proactive Heuristic Synthesis (PHS): Addressing the Reactive Bottleneck through Latent Idle Consolidation in LLMs

Ali Zulfiqar
Department of Computer Science
FAST National University of Computer and Emerging Sciences
Lahore, Pakistan
l240658@lhr.nu.edu.pk

***Abstract*—Current Large Language Model (LLM) architectures are fundamentally reactive, operating within a "Prompt-Response" paradigm that leaves significant computational resources dormant during inter-prompt intervals. This paper introduces Proactive Heuristic Synthesis (PHS), an architectural framework that enables models to transition into a state of Latent Idle Consolidation. Unlike "thought-at-inference" methods (e.g., Quiet-STaR) which impose latency penalties, or static distillation methods (e.g., Fast Quiet-STaR) which lack continuous adaptability, PHS shifts the computational burden to asynchronous idle cycles. The framework utilizes a Regret-Based Replay mechanism defined by the counterfactual delta between initial inference failure and post-exploration success to target high-value optimization trajectories. Unlike recent replay methods such as SuRe, which prioritize high-perplexity samples for memory retention, PHS prioritizes high-regret samples for reasoning evolution. By autonomously navigating these associations, the model synthesizes novel heuristics verified via a Dual-Model Consensus engine to prevent generation-verification collapse. This paper establishes the mathematical formulation and architectural blueprint of PHS. We theoretically demonstrate that by shifting from perplexity-driven retention to regret-driven evolution, PHS provides an asymptotically optimal solution to the latency-reasoning trade-off. Furthermore, relying on recent bounds in recursive training, we formalize how our Dual-Model Consensus mathematically mitigates model collapse, offering a rigorous pathway to zero-latency self-evolution in LLMs.**

I. Introduction

THE paradigm of Large Language Models (LLMs) is currently undergoing a fundamental shift from "System-1" pattern matching to "System-2" deliberate reasoning [14]. While scaling laws for training capability have been well-established, recent advancements demonstrate that scaling test-time compute-allowing models to generate intermediate reasoning traces or "thoughts"-can yield performance gains that rival scaling model parameters [15], [16]. Frameworks such as Chain-of-Thought (CoT) [1] and Tree of Thoughts (ToT) have proven that explicit reasoning steps significantly enhance performance on complex logical tasks. However, these advancements have introduced a critical inefficiency in current AI architectures: the **Latency-Reasoning Trade-off**.

Current state-of-the-art methods for self-improvement and reasoning operate synchronously during user interaction. For instance, Quiet-STaR [2] trains models to generate rationale tokens implicitly at every step of generation, but this imposes a massive computational overhead, increasing inference latency by over $10\times$ even with short thought traces. Similarly, Forest-of-Thought (FoT) [3] scales reasoning by generating multiple trees during inference, forcing the user to wait while the model explores the solution space. These approaches treat the LLM as a purely reactive entity: it remains dormant until prompted, and then attempts to synthesize complex heuristics in the fleeting moments of user interaction. We define this architectural limitation as the **Reactive Bottleneck**.

In this paper, we introduce **Proactive Heuristic Synthesis (PHS)**, a novel framework that decouples reasoning scaling from inference latency by shifting the computational burden to **Latent Idle Consolidation**. Unlike existing methods that force a trade-off between reasoning depth and responsiveness, PHS utilizes the significant computational resources that lie dormant during inter-prompt intervals (system idle states). PHS enables the model to revisit previous queries asynchronously, exploring its latent space to synthesize optimized reasoning heuristics which are then integrated into the model's parametric memory via Low-Rank Adaptation (LoRA) [8].

A. The Limitations of Current Self-Improvement Paradigms

While self-improvement in LLMs is a vibrant research area, existing methodologies fail to address the Reactive Bottleneck due to three specific limitations:

1) **The Synchronization Problem (Inference vs. Idle Time):** Methods like Quiet-STaR [2] and FoT [3] are synchronous; they consume compute while the user waits. Although Fast Quiet-STaR [4] attempts to mitigate this by distilling thoughts into the model via curriculum learning, it remains a static training recipe rather than a continuous architectural lifecycle. PHS is the first framework to operationalize *asynchronous* self-improvement, treating the inter-prompt interval not as dead air, but as a cognitive "daydreaming" phase for structural adaptation.
2) **The Metric Misalignment (Surprise vs. Regret):** Architectures that utilize replay buffers, such as SuRe

(Surprise-Driven Prioritised Replay) [5], focus on Continual Learning-preventing catastrophic forgetting. SuRe prioritizes samples based on "Surprise" (High Negative Log-Likelihood), replaying data the model finds unlikely. This aligns with recent findings in RLEP [6], where replaying verified successful trajectories improves downstream performance. However, effectively learning to reason requires optimizing for *Evolution*, not just *Retention*. PHS employs a **Regret-Based Replay** mechanism. Instead of prioritizing high-perplexity samples, PHS prioritizes high-regret samples-queries where a counterfactual search discovers a better reasoning path than the one originally served to the user.

3) **The Verification Gap (Heuristics vs. Samples):** Iterative self-improvement schemes often saturate quickly due to the collapse of the "generation-verification gap," where the model's ability to generate solutions outpaces its ability to verify them. Recent empirical studies have shown that LLMs fundamentally struggle to self-correct reasoning errors without external or frozen oracles [18]. PHS addresses this by synthesizing abstracted reasoning heuristics verified by a **Dual-Model Consensus** engine (a frozen Critic and the active Policy) before integration. This ensures that the model updates its weights based on verified logic rather than noisy self-generated data, preventing the model collapse observed in recursive training loops [7].

B. Contributions

By synthesizing the asynchronous nature of idle-time processing with the precision of regret-driven optimization, PHS transforms the LLM from a static tool into a continuously evolving cognitive system. Our specific contributions are:

- **Architecture:** We propose the PHS Framework, which utilizes idle GPU cycles to autonomously navigate latent associations, identifying and rectifying suboptimal reasoning paths without user latency.
- **Mechanism:** We introduce Regret-Based Heuristic Discovery, a selection metric defined by the optimization delta between initial inference failure and post-exploration success.
- **Stability:** We formalize a Dual-Model Consensus verification loop. Drawing on recent theoretical findings that self-verification prevents model collapse in recursive training [12], we theoretically guarantee the stability of our self-improvement loop.
- **Efficiency:** We provide a computational complexity analysis demonstrating that PHS decouples reasoning scaling from inference latency, achieving $O(\mathcal{C}_{\text{base}})$ user-facing latency independent of reasoning depth.

II. Related Work

Our work sits at the intersection of test-time compute scaling, continual learning via experience replay, and autonomous self-improvement. While recent advances have pushed the boundaries in each of these domains, they collectively face a **Reactive Bottleneck**-the inability to utilize dormant computational resources for structural adaptation outside of active user prompting.

A. Scaling Test-Time Compute and Inference Reasoning

Recent scaling laws indicate that increasing test-time compute can yield performance gains comparable to scaling pre-training parameters [15], [16]. This has led to "System-2" architectures [14] that generate intermediate reasoning traces. Methods like Chain-of-Thought (CoT) [1], Tree of Thoughts (ToT), and Forest-of-Thought (FoT) [3] explicitly scale compute by exploring multiple paths during inference. However, these are fundamentally synchronous, introducing significant latency penalties. Similarly, Quiet-STaR [2] generates "thought tokens" at every step, incurring a $10\times$ latency overhead. Fast Quiet-STaR [4] and S1 attempt to distill these reasoning capabilities into model weights, but remain static training recipes.

Recent closed-source reasoning systems such as OpenAI's o1 architecture [19] and DeepSeek-R1 [20] represent the current frontier of inference-time scaling, generating extended chain-of-thought traces prior to producing a final answer. While these systems demonstrate substantial performance gains on complex reasoning benchmarks, they remain fundamentally synchronous: the entire reasoning budget is consumed during the active user session, imposing latency proportional to reasoning depth. PHS is architecturally distinct in that it does not perform reasoning *at* inference time; instead, it performs reasoning *between* inference sessions, encoding the results of that reasoning directly into model weights. This allows PHS to achieve comparable reasoning depth at zero additional user-facing latency.

PHS differentiates itself by decoupling reasoning from inference. By shifting the discovery of heuristics to **asynchronous idle cycles**, PHS achieves the benefits of "deep thinking" without the runtime cost. This approach draws inspiration from GPU serving optimizations (e.g., vLLM), aiming to improve *computational goodput* by utilizing the idle gaps inherent in interactive sessions.

B. Continual Learning and Experience Replay

To prevent catastrophic forgetting, Continual Learning (CL) relies on Experience Replay (ER) [21]. Recent innovations like SuRe (Surprise-Driven Prioritised Replay) [5] prioritize samples based on high Negative Log-Likelihood (NLL). However, existing replay methods optimize for *Retention* (minimizing perplexity), whereas PHS optimizes for *Evolution* (maximizing reasoning delta). Unlike SuRe, PHS utilizes a **Regret-Based Replay** mechanism, prioritizing queries where a counterfactual search discovers a superior reasoning path, aligning with parameter-efficient fine-tuning approaches [8] but applied to self-improvement rather than static adaptation. This aligns PHS more closely with self-supervised exploration than traditional data retention.

C. Autonomous Self-Improvement and Verification

The paradigm of "Self-Evolving" LLMs aims to bootstrap capabilities without human annotations. Frameworks like STaR [9], ReST, and ReGenesis [10], as well as standard RLHF pipelines [17], generate and filter reasoning traces for fine-tuning. Self-Refine [11] and CRITIC employ the model itself to critique outputs. However, subsequent empirical studies have proven that LLMs fundamentally struggle to self-correct reasoning errors without external or frozen oracles [18]. Theoretical work further warns that iterative self-improvement saturates quickly as the generator collapses into the verifier's blind spots [7].

To mitigate the **recursive training collapse** observed when models train on their own synthetic output, PHS employs a **Dual-Model Consensus** engine (utilizing a frozen Critic). Furthermore, unlike Instruction-Level Weight Shaping (ILWS) [13], which updates non-parametric system prompts, PHS distills verified heuristics directly into parametric memory via LoRA [8], ensuring long-term evolution without context-window bloat.

III. The PHS Framework

The Proactive Heuristic Synthesis (PHS) framework is an asynchronous architectural lifecycle designed to decouple reasoning scaling from user latency. Unlike synchronous methods that utilize compute during the active inference loop, PHS operationalizes *Latent Idle Consolidation*-a process of structural adaptation that occurs during inter-prompt intervals.

A. Architectural Components

The PHS architecture consists of four primary modules that manage the transition from reactive inference to proactive optimization:

1) The Regret-Log: During active user sessions, the system maintains a lightweight telemetry log of high-complexity queries. Queries are flagged for the Regret-Log if they meet specific criteria: high inference time, low confidence scores from the internal policy, or negative user feedback. Unlike standard replay buffers that store raw data for retention [21], the Regret-Log serves as a seed for asynchronous exploration.

2) Consensus-Driven Explorer: Upon detecting system idle states-defined operationally as periods during which GPU utilization falls below a configurable threshold ϵ_{idle} for a sustained watchdog interval Δt-the Explorer utilizes the Regret-Log to initiate a stochastic search within the model's latent space. It performs a counterfactual search, exploring alternative reasoning trajectories that were skipped during the latency-constrained inference phase.

3) Dual-Model Consensus Engine: To mitigate the *generation-verification gap* and prevent recursive training collapse, every synthesized heuristic must pass a verification gate. This engine employs a frozen "Critic" model to perform logical consistency checks on the newly discovered paths. A heuristic is only committed to the buffer if it achieves a consensus score $\tau > \tau_{\text{min}}$, where τ_{min} is a tunable hyperparameter set to 0.95 in our formulation pending empirical calibration on downstream reasoning benchmarks. The necessity of this verification gate is consistent with recent theoretical findings on recursive training collapse, which demonstrate that training on fully synthetic data without an independent verification signal leads to exponential error amplification [7], [12]. By enforcing that every synthesized heuristic is gated by a frozen Critic model, PHS explicitly satisfies the conditions required to maintain stability in fully synthetic training regimes, preventing the generation-verification gap from closing.

4) LoRA Integration Module: Verified heuristics are not stored as raw samples; instead, they are abstracted into generalized logical patterns and integrated into the model's weights via Low-Rank Adaptation (LoRA) [8]. This ensures that the model "internalizes" the reasoning without context-window bloat or the need for full-scale retraining.

B. Mathematical Formulation of Regret

We define the PHS optimization objective not by perplexity minimization (as in SuRe [5]), but by **Evolutionary Regret**. Let x be a query from the Regret-Log. We define the initial reasoning trajectory as y_{init} and the optimized trajectory discovered during idle-time exploration as y_{opt}. The Regret R for a given query x is formalized as:

$$R(x) = \mathcal{V}(y_{opt}) - \mathcal{V}(y_{init}) \tag{1}$$

where $\mathcal{V}$ is a value function derived from the frozen Critic, operationalized as a scalar reward signal. Concretely, $\mathcal{V}(y)$ is computed as the log-probability-weighted correctness score assigned by the frozen Critic model $\mathcal{M}_{\text{critic}}$ to a candidate trajectory y:

$$\mathcal{V}(y) = \mathcal{M}_{\text{critic}}(y \mid x) \tag{2}$$

where $\mathcal{M}_{\text{critic}}$ outputs a scalar score in $[0, 1]$, with higher scores indicating greater logical consistency and reasoning completeness as judged by the Critic. PHS prioritizes trajectories with the highest $R(x)$, ensuring that the model focuses its "daydreaming" compute on areas with the maximum potential for reasoning growth.

C. Operational Lifecycle

The PHS lifecycle follows a continuous three-stage loop:

1) **Observation:** Log high-friction queries during inference.
2) **Consolidation (Asynchronous):** Synthesize and verify optimized heuristics during idle cycles using the Regret-Based metric.
3) **Internalization:** Periodically merge LoRA weights to the base model, ensuring the next inference session benefits from prior "reflections."

IV. Theoretical Analysis and Complexity Bounds

We formalize Proactive Heuristic Synthesis (PHS) as an asynchronous, dual-phase lifecycle. Unlike standard LLMs which remain static between updates ($t_{train} \ll t_{inference}$), PHS exploits system idle time to minimize the *Generation-Verification Gap* through latent exploration.

Let $\mathcal{M}_\theta$ be the primary language model parameterized by θ. We define the operational timeline as a sequence of intervals $T = \{\tau^{(1)}_{active}, \tau^{(1)}_{idle}, \tau^{(2)}_{active}, \dots\}$, where τ_{active} represents user-facing inference latency and τ_{idle} represents dormant computational cycles.

A. Regret-Based Experience Accumulation

During an active interval τ_{active}, for a user prompt x, the model generates a reactive response $y_{react} \sim \mathcal{M}_\theta(y|x)$. We store the tuple (x, y_{react}) in a temporary **Candidate Buffer** ($\mathcal{B}_{cand}$).

During the subsequent idle interval τ_{idle}, the model retrieves inputs from $\mathcal{B}_{cand}$ and initiates a high-compute search (e.g., Monte Carlo Tree Search) to find a superior reasoning trajectory y_{opt}. We define **Counterfactual Regret** (R_{CF}) not as the loss of the ground truth, but as the optimization delta:

$$R_{CF}(x) = \mathcal{V}(y_{opt}) - \mathcal{V}(y_{react}) \tag{3}$$

where $\mathcal{V}(\cdot)$ is a value function derived from our Dual-Model Consensus engine. If $R_{CF}(x) > \delta$, the model "regrets" its initial fast response, and the sample is prioritized for heuristic distillation. Here, $\delta \geq 0$ is a tunable regret threshold hyperparameter that controls the sensitivity of the replay selection process. A higher δ restricts distillation to only the most significantly suboptimal responses, whereas $\delta \to 0$ admits all queries where any improvement was found. In practice, δ is calibrated per deployment context to balance the breadth of heuristic coverage against the computational cost of idle-time exploration.

B. Heuristic Distillation via Idle-LoRA

The objective of PHS is to distill the heavy compute used to find y_{opt} into the model's weights, ensuring future reactive passes approximate y_{opt} without the latency cost. This is achieved via LoRA updates during τ_{idle}. We minimize a regret-weighted cross-entropy loss:

$$\mathcal{L}_{PHS}(\theta) = \mathbb{E}_{x \sim \mathcal{B}_{heuristic}} \left[R_{CF}(x) \cdot \mathcal{L}_{CE}(\mathcal{M}_\theta(x), y_{opt}) \right] \tag{4}$$

High-regret examples-where the model identifies a significant reasoning failure-drive the gradient updates more aggressively, focusing parameter adaptation on high-friction logical patterns.

C. Theoretical Guarantee of Zero-Latency Evolution

Current *System-2* reasoning methods [14], such as Forest-of-Thought (FoT) [3], Quiet-STaR [2], or OpenAI's o1 architecture [19], scale test-time compute synchronously. Let N be the number of reasoning steps or trees, and L be the depth. The user-facing inference cost is mathematically bound to the reasoning complexity:

$$C_{\text{sync}} = O(N \cdot L)$$

This creates an upper bound on real-world deployment usability.

In contrast, PHS decouples the generation of the optimal trajectory y_{opt} from the active user session. During τ_{active}, the cost is strictly the base model's forward pass, independent of reasoning depth L:

$$C^{\text{PHS}}_{\text{active}} = O(\mathcal{C}_{\text{base}})$$

where $\mathcal{C}_{\text{base}}$ denotes the fixed cost of a single forward pass through $\mathcal{M}_\theta$, independent of N or L. The heavy computation $O(N \cdot L)$ is shifted entirely to the dormant idle budget B_{idle}.

Therefore, as reasoning depth $L \to \infty$, synchronous methods become mathematically unusable for real-time interaction, whereas PHS maintains constant user-facing latency bounded by $\mathcal{C}_{\text{base}}$.

This demonstrates that PHS is the asymptotically optimal architecture for scaling reasoning depth in deployed real-time environments.

D. Asynchronous Update Schedule

To ensure stability, we implement a **Double-Buffer Parameter Swap**:

- **Active Weights (θ_{active}):** Used for user inference; frozen during active sessions.
- **Shadow Weights (θ_{shadow}):** Updated by the optimizer during idle time.

When τ_{idle} concludes, we perform Polyak averaging: $\theta_{active} \leftarrow \alpha\theta_{shadow} + (1-\alpha)\theta_{active}$, ensuring a smooth integration of new heuristics without disrupting the live session. The mixing coefficient $\alpha \in (0, 1)$ controls the rate of weight transition and is treated as a deployment hyperparameter. The merge schedule itself is governed by a **Heuristic Accumulation Threshold** $\mathcal{H}_{\text{min}}$: weight integration is triggered when the verified heuristic buffer $\mathcal{B}_{\text{heuristic}}$ accumulates at least $\mathcal{H}_{\text{min}}$ validated entries, ensuring that each LoRA update is informed by a statistically meaningful batch of reasoning improvements rather than individual noisy samples. In low-traffic deployments where idle cycles are abundant, $\mathcal{H}_{\text{min}}$ can be set conservatively high to maximize update quality. In high-traffic deployments, a time-based fallback trigger ensures merges occur at least once per session regardless of buffer size.

V. Limitations and Discussion

While the mathematical formulation of Proactive Heuristic Synthesis (PHS) demonstrates asymptotic optimality for scaling test-time compute without user-facing latency penalties, translating this framework into real-world deployment introduces several practical and theoretical constraints.

A. The "Scarce Idle Time" Constraint in Edge Environments

The core assumption of Latent Idle Consolidation is the availability of dormant computational cycles τ_{idle} between active prompts τ_{active}. While this holds true for cloud-based inference servers experiencing fluctuating utilization troughs, it poses a challenge for mobile or edge deployments.

In resource-constrained environments (e.g., smartphones or IoT devices), "idle" time is often intentionally throttled to preserve battery life or thermal limits. If the available idle compute budget B_{idle} is insufficient to complete the latent tree

TABLE I
ARCHITECTURAL COMPARISON OF REASONING FRAMEWORKS

Feature	Quiet-STaR	Forest-of-Thought	SuRe	PHS (Ours)
Compute Phase	Inference (Synchronous)	Inference (Synchronous)	Post-Training (Online)	Idle Time (Asynchronous)
User Latency	High ($> 10\times$ overhead)	High (Tree Search)	Zero (Deployment)	Zero (Asymptotically)
Optimization Target	Next-Token Prediction	Best Path Selection	Memory Retention (NLL)	Evolution (Max Regret)
Collapse Resistant	N/A	N/A	Yes	Yes (Dual-Model Critic)
Lifecycle	Static Training	Static Inference	Continuous	Continuous

search and verifier passes, PHS defaults to the base model's reactive capabilities.

A related concern is the computational cost of the idle-time search itself. Monte Carlo Tree Search, as proposed for the Consensus-Driven Explorer, can require substantial wall-clock time depending on tree depth L and branching factor N. In high-throughput cloud deployments where inter-prompt intervals may be as short as one to three seconds, a full MCTS pass may not complete within a single idle window. PHS addresses this through a **preemptible search design**: the Explorer checkpoints its search state at each node expansion, allowing synthesis to be paused immediately upon detection of an incoming prompt (ϵ_{idle} threshold exceeded) and resumed in the subsequent idle window. This ensures that even fragmented idle budgets contribute incrementally to heuristic discovery without blocking active inference. Future work must investigate dynamic budget allocation algorithms capable of optimally scheduling these fragmented synthesis windows across variable-length idle intervals.

B. Critic Initialization and the Cold Start Problem

PHS relies on a Dual-Model Consensus engine to prevent generation-verification collapse, but this introduces a dependency on the initial competence of the frozen Critic model.

If the system is deployed without prior exposure to a domain, the Regret-Log $\mathcal{B}_{\text{cand}}$ will be empty, and the model will lack the necessary counterfactual data required to begin self-improvement, creating a classic "cold start" problem.

Furthermore, the Critic must be sufficiently capable from the outset to accurately score candidate trajectories. Addressing this challenge requires an initial "warm-up" phase, potentially utilizing a larger oracle model for initial distillation, or bootstrapping the Critic through supervised fine-tuning (SFT) on ground-truth reasoning traces before transitioning to autonomous PHS.

C. LoRA Rank Sensitivity and Representation Capacity

The integration of synthesized heuristics relies on periodic Low-Rank Adaptation (LoRA) [8]. However, the efficacy of this integration is highly sensitive to the chosen rank r.

If the rank is too low, the update matrix ΔW may lack the representational capacity required to capture complex multi-step logical heuristics. Conversely, setting r too high risks destabilizing the pre-trained weights, potentially causing catastrophic forgetting of previously learned abstractions.

Determining the optimal "intrinsic rank" required specifically for reasoning heuristics, as opposed to standard knowledge injection, remains a critical open theoretical question for continuous learning architectures.

D. Qualitative Comparison to Existing Paradigms

To contextualize the theoretical advantages of PHS against the current landscape of LLM reasoning enhancements, Table I provides a qualitative architectural comparison.

Foundational experience replay mechanisms [21] and existing replay architectures for LLMs, such as SuRe [5], utilize Negative Log-Likelihood (NLL) to prioritize samples. As demonstrated in Table I, PHS is the only framework that simultaneously achieves continuous weight updates, theoretical resistance to model collapse, and zero user-facing latency. This fundamentally distinguishes it from inference-blocking methods such as Quiet-STaR and Forest-of-Thought, as well as retention-focused continual learning methods like SuRe.

VI. CONCLUSION

In this paper, we introduced the theoretical framework for Proactive Heuristic Synthesis (PHS), addressing the reactive bottleneck inherent in current Large Language Model architectures.

By formalizing the mechanism of Latent Idle Consolidation, we demonstrated how the heavy computational burden of System-2 reasoning [14] can be shifted to asynchronous idle cycles. Our complexity analysis theoretically guarantees that PHS decouples reasoning depth from user-facing latency, achieving $O(\mathcal{C}_{\text{base}})$ active inference costs independent of reasoning depth, while scaling reasoning capabilities through an idle-time compute budget.

Furthermore, we mathematically distinguished our Regret-Based Replay mechanism from standard perplexity-based continual learning methods [5], [21], arguing that the transition toward autonomous cognition requires optimizing for counterfactual evolution rather than simple data retention.

By integrating a Dual-Model Consensus engine, the PHS architecture is theoretically bounded against the recursive model collapse [7], [12] that often affects iterative self-improvement systems.

While this paper establishes the structural and mathematical blueprint for continuous autonomous cognition, comprehensive empirical validation across large-scale reasoning benchmarks and diverse hardware environments remains an essential direction for future work.

Ultimately, PHS provides a foundational architecture capable of transforming artificial intelligence systems from static prompt-response tools into continuously evolving cognitive

systems that proactively think, reflect, and improve during periods of inactivity.

REFERENCES

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

[2] E. Zelikman, G. Harik, Y. Shao, V. Jayasiri, N. Haber, and N. D. Goodman, "Quiet-STaR: Language models can teach themselves to think before speaking," *arXiv preprint arXiv:2403.09629*, 2024.

[3] Z. Bi, K. Han, C. Liu, Y. Tang, and Y. Wang, "Forest-of-Thought: Scaling test-time compute for enhancing LLM reasoning," *arXiv preprint arXiv:2412.09078*, 2024.

[4] W. Huang, Y. Xiong, X. Ye, Z. Deng, H. Chen, Z. Lin, and G. Ding, "Fast Quiet-STaR: Thinking Without Thought Tokens," *arXiv preprint arXiv:2505.17746*, 2025.

[5] H. Hazard, Z. Fountas, M. A. Benfeghoul, A. Oomerjee, J. Wang, and H. Bou-Ammar, "SuRe: Surprise-Driven Prioritised Replay for Continual LLM Learning," *arXiv preprint arXiv:2511.22367*, 2025.

[6] H. Zhang, J. Fu, J. Zhang, K. Fu, Q. Wang, F. Zhang, and G. Zhou, "RLEP: Reinforcement Learning with Experience Replay for LLM Reasoning," *arXiv preprint arXiv:2507.07451*, 2025.

[7] S. Alemohammad, J. Casco-Rodriguez, L. Luzi, A. I. Humayun, H. Babaei, D. LeJeune, A. Siahkoohi, and R. G. Baraniuk, "Self-consuming generative models go mad," *arXiv preprint arXiv:2307.01850*, 2023.

[8] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-rank adaptation of large language models," *arXiv preprint arXiv:2106.09685*, 2021.

[9] E. Zelikman, Y. Wu, J. Mu, and N. D. Goodman, "STaR: Bootstrapping reasoning with reasoning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 15476–15488, 2022.

[10] X. Peng, C. Xia, X. Yang, C. Xiong, C.-S. Wu, and C. Xing, "ReGenesis: LLMs can grow into reasoning generalists via self-improvement," in *International Conference on Learning Representations (ICLR)*, 2025.

[11] A. Madaan, N. Tandon, P. Gupta, S. Hallinan, L. Gao, S. Wiegreffe, U. Alon et al., "Self-refine: Iterative refinement with self-feedback," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[12] S. Fu, Y. Wang, Y. Chen, L. Shen, and D. Tao, "Self-verification provably prevents model collapse in recursive synthetic training," in *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025.

[13] R. Costa, "Instruction-Level Weight Shaping: A Framework for Self-Improving AI Agents," *arXiv preprint arXiv:2509.00251*, 2025.

[14] D. Kahneman, *Thinking, Fast and Slow*. Farrar, Straus and Giroux, 2011.

[15] B. Brown, J. Juravsky, R. Ehrlich, R. Clark, Q. V. Le, C. Ré, and A. Mirhoseini, "Large language monkeys: Scaling inference compute with repeated sampling," *arXiv preprint arXiv:2407.21787*, 2024.

[16] C. Snell, J. Lee, K. Xu, and A. Kumar, "Scaling LLM test-time compute optimally can be more effective than scaling model parameters," *arXiv preprint arXiv:2408.03314*, 2024.

[17] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray et al., "Training language models to follow instructions with human feedback," *Advances in Neural Information Processing Systems*, vol. 35, pp. 27730–27744, 2022.

[18] J. Huang, X. Chen, S. Mishra, H. S. Zheng, A. W. Yu, X. Song, and D. Zhou, "Large language models cannot self-correct reasoning yet," in *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

[19] OpenAI, A. Jaech, A. Kalai, A. Lerer, A. Richardson, A. El-Kishky et al., "OpenAI o1 system card," *arXiv preprint arXiv:2412.16720*, 2024.

[20] D. Guo et al., "DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning," *arXiv preprint arXiv:2501.12948*, 2025.

[21] D. Rolnick, A. Ahuja, J. Schwarz, T. P. Lillicrap, and G. Wayne, "Experience replay for continual learning," *Advances in Neural Information Processing Systems*, vol. 32, 2019.