LatentEvolve: Self-Evolving Test-Time Scaling in Latent Space

Anonymous ACL submission

Abstract

Test-time Scaling (TTS) has been demonstrated to significantly enhance the reasoning capabilities of Large Language Models (LLMs) during the inference phase without altering model parameters. However, existing TTS methods are largely independent, implying that LLMs have not yet evolved to progressively learn how to scale more effectively. With the objective of evolving LLMs to learn "how to scale test-time computation," we propose `LatentEvolve`, a self-evolving latent TTS framework inspired by the complementary learning system (CLS) theory. Analogous to the human brain's dual system of a fast-recall hippocampus and a slow-consolidating neocortex, `LatentEvolve` comprises two evolutionary components: *daytime scaling*, which rapidly retrieves historical latent representations to better guide current LLM reasoning; and *nighttime scaling*, which integrates past latent optimizations in a manner akin to the human brain's consolidation of experiences during sleep. The alternation of daytime and nighttime processes facilitates a fast and slow evolution of LLM TTS, mirroring human cognitive dynamics in a fully unsupervised manner. Extensive experiments across eight benchmarks and five model backbones demonstrate that our `LatentEvolve` surpasses state-of-the-art TTS methods such as LatentSeek and TTRL by up to 13.33% and exhibits exceptional cross-domain and cross-backbone generalization. The codes are available at `https://anonymous.4open.science/r/le-code/`.

1 Introduction

The general capabilities of large language models (LLMs) have been extensively developed and widely recognized across numerous domains, such as mathematical reasoning (Zeng et al., 2024; Wu et al., 2025), software engineering (Wei et al., 2025; Yang et al., 2024), multimodal understanding (Zheng et al., 2025b; Su et al., 2025), and embodied action (Wang et al., 2023a), emerging as dominant paradigms that are steadily advancing toward artificial general intelligence (AGI) (Bubeck et al., 2023). Much of this success in recent years has been driven by *training-time scaling*, wherein increasing the volume of training data and parameters consistently yields performance improvements (Kaplan et al., 2020; Aghajanyan et al., 2023). However, the pace of this scaling, particularly in terms of pre-training scale, has begun to slow, constrained by its resource-intensive nature and the depletion of high-quality training data (Villalobos et al., 2022; Zhou et al., 2025). Consequently, a growing body of research has shifted attention to *test-time scaling* (TTS) (Zhang et al., 2025c; Chung et al., 2025), aiming to fully harness the intrinsic knowledge of LLMs to maximize their real-world utility without additional training during the test phase.

The dimensions of TTS are highly diverse. One prominent form is **(I) parallel scaling**, wherein multiple candidate responses are generated for a given query, which are subsequently aggregated via an appropriate mechanism. This can involve multiple samples from a single LLM (Brown et al., 2024; Snell et al., 2024) or sampling from multiple heterogeneous LLMs (Zhang et al., 2025d; Ye et al., 2025). Another form is **(II) sequential scaling**, where the LLM iteratively refines solutions based on its own previous outputs, and which underlies many "System 2"-style generation methods (Yu et al., 2024; Wei et al., 2023; He et al., 2024; Gou et al., 2024). Other variants include *hybrid* approaches that integrate both strategies (Wang et al., 2024a; Besta et al., 2024), as well as *internalized scaling*, where models like DeepSeek R1 (Guo et al., 2025) and OpenAI o-series (Li et al., 2025b) are inherently capable of adaptively allocating computational resources during inference.

However, regardless of the specific form, most TTS paradigms lack the capacity for *self-evolution*, as inference-time computations for distinct queries

are typically treated as **independent** events. For example, in verbal reinforcement learning approaches such as Reflexion (Shinn et al., 2023) and Mind Evolution (Lee et al., 2025), successful reflective strategies are instance-specific and are not transferred to subsequent tasks. Likewise, in "sampling-and-voting" scaling methods (Brown et al., 2024; Irvine et al., 2023), prior successes in selecting the correct answer do not inform or refine future selection strategies. This *inter-task independence* fundamentally constrains the potential of TTS paradigms to progressively evolve through continual interaction with the environment. This raises a natural yet critical research question: *How can we design a TTS framework that learns from experience, enabling its scaling capabilities to evolve and improve as it solves more problems?*

To address this challenge, we introduce `LatentEvolve`, a self-evolving TTS framework inspired by the Complementary Learning Systems (CLS) theory (McClelland et al., 1995; Kumaran et al., 2016). CLS theory posits that the brain uses two synergistic systems: a fast-learning hippocampus for specific episodic memories, and a slow-learning neocortex to consolidate experiences into general knowledge. Analogously, `LatentEvolve` operates through a dual-phase evolution:

- **Daytime Scaling** for *fast, episodic adaptation*: For each new query, `LatentEvolve` performs instance-level latent optimization that steers the LLM toward better reasoning paths. This process is initialized by retrieving relevant "episodic traces", *i.e.*, latent representations from previously solved problems, mirroring the *daytime* fast recall of individual memories.
- **Nighttime Scaling** for *slow, procedural consolidation*: Mirroring brain's consolidates experiences into general skills during sleep, `LatentEvolve` periodically fine-tunes a compact knowledge consolidation model (*latent weaver*) on the collection of *daytime* traces. This *nighttime* process distills these specific experiences into procedural knowledge, evolving to generate superior initial latent representations for future tasks.

Within this continual interplay, `LatentEvolve` enables LLMs to perform test-time computation during daytime inference while simultaneously accumulating experiential knowledge. During nighttime reflection, these experiences are periodically consolidated into procedural memory, thereby achieving a "fast-slow" evolution of test-time scaling. The entire process operates **without** reliance on ground-truth labels or any other external signals.

Experimental Observation. Extensive evaluations across eight benchmarks across four domains demonstrate that `LatentEvolve` provides: ❶ **high performance**: achieving up to 23.3% gains on math reasoning, surpassing GRPO and LatentSeek on MATH-500 by 2.35% and 4.95%, respectively; ❷ **cross-domain generalization**: test-time scaling on MMLU and MATH transfers to out-of-domain datasets, yielding gains of 8.60% on GPQA and 5.22% on JAMA; ❸ **continual learning ability**: test-time scaling across multiple new domains does not degrade performance on previously seen domains and can even improve slightly.

2 Related Work

Test-time computation is a canonical pathway for transitioning from System 1 to System 2 models, with two primary branches: *test-time training* (TTT) and *test-time scaling* (TTS). The former involves updating model parameters during the test phase unsupervisedly, as exemplified by TTT (Sun et al., 2020; Akyürek et al., 2024), TTT+++ (Liu et al., 2021), and SIFT (Hübotter et al., 2025). The latter increases computational expenditure without altering parameters, which can occur in the **(I) explicit, natural-language space**, as in self-correction (Shinn et al., 2023; Gou et al., 2024; Kang et al., 2025), feedback modeling (Cobbe et al., 2021; Yu et al., 2025), or repeated sampling (Gui et al., 2024; Ye et al., 2025); or it may operate in the **(II) latent space**, where methods such as Coconut (Hao et al., 2024) and SoftCoT (Xu et al., 2025b,c) perform deep scaling within the model's hidden representations. Our `LatentEvolve` falls primarily within the latent TTS. Yet, regardless of form, existing approaches are rarely capable of rapid evolution through the ongoing process of problem solving, a limitation that `LatentEvolve` is designed to overcome.

Latent Computation & Reasoning Latent computation enables LLMs to reason through continuous representations rather than discrete language tokens (Zhu et al., 2025). Mainstream approaches can be broadly categorized as: **(I) architecturally enabling native latent reasoning**, as exemplified by Coconut (Hao et al., 2024), CoLaR (Tan et al., 2025), and Recurrent Depth (Geiping et al., 2025); and **(II) employing latent computation to steer LLM generation**, as in LatentSeek (Li

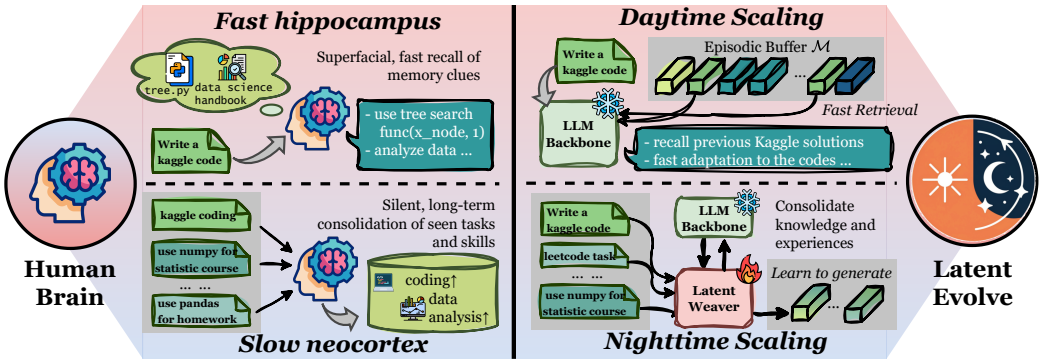

Figure 1: The daytime scaling of `LatentEvolve` functions analogously to the human hippocampus, rapidly retrieving memory cues, whereas the nighttime scaling mirrors the neocortex during sleep, performing deep integration of accumulated knowledge.

et al., 2025a), SoftCoT (Xu et al., 2025c,b), and others (Liu et al., 2024; Sun et al., 2025), which leverage latent representations to modulate generated output quality. Some works further use latent tokens for contextual memory (Orlicki, 2025; Wang et al., 2024c, 2025a). In contrast, `LatentEvolve` performs dual-stage test-time evolution in latent space, enabling experience transfer across tasks rather than treating each query independently.

We discuss broader connections to self-evolving LLMs and agents in Appendix B.

3 Preliminary

In this section, we formally describe the procedure of current latent-based TTS methods, which steer LLM's generative process by introducing adaptable, continuous vectors.

Latent-Space Aided Reasoning. Let $\pi_{\boldsymbol{\theta}}$ be a language model with frozen parameters $\boldsymbol{\theta}$. For a given problem context $\mathbf{c}$, the standard generative process produces an output sequence $\mathbf{y}$ by sampling from the conditional probability distribution $p(\mathbf{y}|\mathbf{c};\boldsymbol{\theta})$. The core principle of this paradigm is to introduce an auxiliary sequence of continuous vectors, $\mathbf{z} = (z_1, z_2, \cdots, z_L)$, which we refer to as a *latent token sequence*. These vectors act as a dynamic, instance-specific control signal that conditions the generative process of the frozen LLM. The generation is thus reformulated as sampling from a new distribution, conditioned on both the original context and the latent intervention:

$$\mathbf{y} \sim p(\mathbf{y}|\mathbf{c}, \mathbf{z}; \boldsymbol{\theta}) \tag{1}$$

The latent sequence $\mathbf{z}$ can be introduced through various mechanisms, such as being prepended to input embeddings, directly augmenting the model's internal key-value (KV) cache, or representing a latent thought process for subsequent decoding (Xu et al., 2025c; Liu et al., 2024; Sun et al., 2025). The primary objective is to find an optimal latent intervention $\mathbf{z}^*$ that maximizes an objective function $J(\mathbf{z})$. This objective is formalized as the expected quality of the generated output:

$$\mathbf{z}^* = \arg\max_{\mathbf{z}} \underbrace{\mathbb{E}_{\mathbf{y}\sim p(\mathbf{y}|\mathbf{c},\mathbf{z};\boldsymbol{\theta})}[Q(\mathbf{y})]}_{J(\mathbf{z})} \tag{2}$$

where $Q(\mathbf{y})$ is a scoring function that evaluates the quality of an output $\mathbf{y}$.

Generation of Latent Representations. The mechanism for generating the latent sequence $\mathbf{z}$ defines the specific TTS method. Existing work either optimizes a single set of *task-specific* soft prompts, $\mathbf{z}_{\text{task}}$, applied to all instances (Xiao et al., 2023; Choi et al., 2023), or performs *query-specific* optimization to find a bespoke latent path $\mathbf{z}_i$ for each individual query $\mathbf{c}_i$ (Li et al., 2025a; Peng et al., 2024; Xu et al., 2025c; Sun et al., 2025). Whatever the granularity is, most of these practices are self-contained, *i.e.*, do not rapidly learn or evolve across instances, thereby incapable of on-the-fly adaptation from cumulative experience.

4 Methodology

`LatentEvolve` unfolds as a dual-phase evolving process that enables LLMs to adapt and self-improve at test time. First, we introduce *daytime test-time scaling* (▷ Section 4.1), which performs fast, instance-specific adaptation guided by weighted momentum transfer. Then, *nighttime deliberative consolidation* (▷ Section 4.2) integrates these episodic traces into a compact parametric prior through the latent weaver. Finally, *dual-phase evolving scaling* (▷ Section 4.3) ties the two phases into a recurrent cycle, ensuring continual interleaved evolution in latent space.

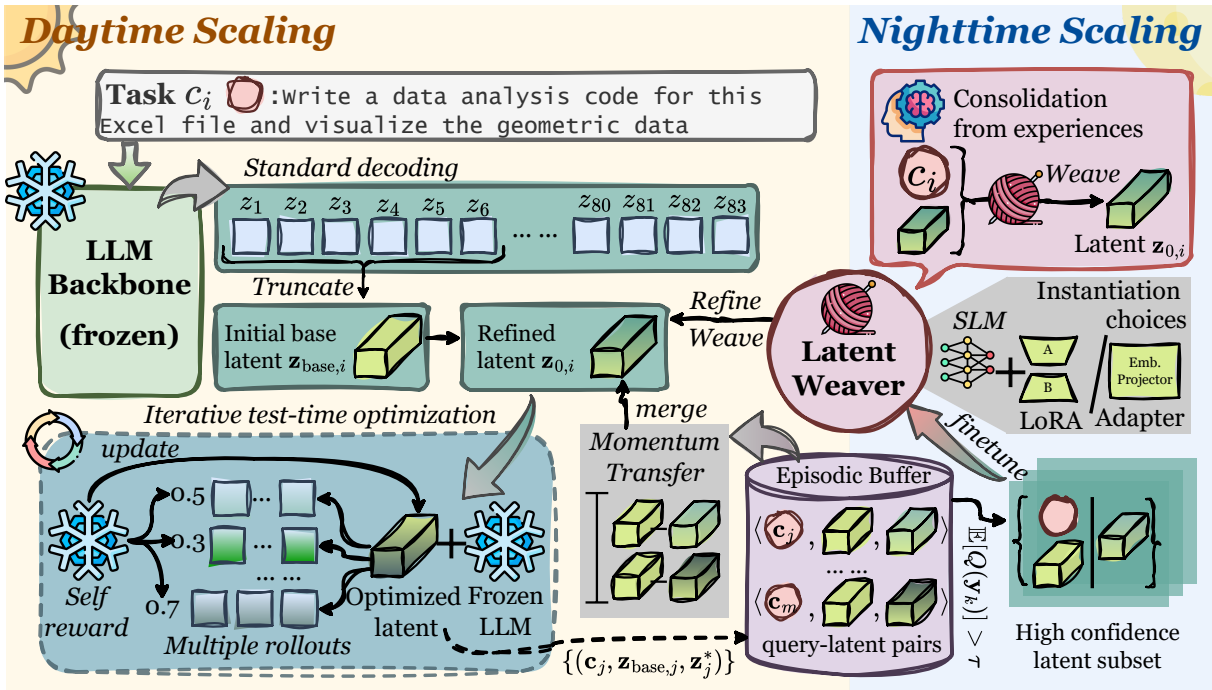

Figure 2: The overview of our proposed `LatentEvolve`.

4.1 Daytime Test-time Optimization

Daytime Scaling is designed for fast on-the-fly adaptation, mirroring the brain's ability to recall past experiences to navigate current challenges. This process unfolds in three key stages for each incoming query: retrieving relevant memories, constructing an informed initial latent sequence, and refining it through self-guided optimization.

Associative Retrieval. Inspired by the function of episodic memory in cognitive science, `LatentEvolve` maintains an **episodic buffer**, $\mathcal{M}$, which serves as a dynamic archive of specific, high-quality test-time scaling experiences. Each entry is a triplet $(\mathbf{e}_{\mathbf{c}_j}, \mathbf{z}_{\text{base},j}, \mathbf{z}_j^*)$, storing a previous query's context embedding $\mathbf{e}_{\mathbf{c}_j}$, its initial latent sequence $\mathbf{z}_{\text{base},j}$, and its refined latent sequence $\mathbf{z}_j^*$.

Upon receiving a new query (defined as input prompt $\mathbf{c}_i$), we first compute its semantic embedding $\mathbf{e}_{\mathbf{c}_i}$ using the frozen LLM's final hidden state. We then perform a similarity search to retrieve the top-k most relevant experiences from the buffer as the neighborhood:

$$\mathcal{N}_k(\mathbf{c}_i) = \text{Top-k}_j \Big\{ (\mathbf{e}_{\mathbf{c}_j}, \mathbf{z}_{\text{base},j}, \mathbf{z}_j^*) \in \mathcal{M} \mid S(\mathbf{e}_{\mathbf{c}_i}, \mathbf{e}_{\mathbf{c}_j}) \Big\} \tag{3}$$

where $S(\cdot,\cdot)$ denotes cosine similarity. This lets test-time optimization to leverage a small, highly relevant subset of past experiences.

Informed Latent Initialization. A well-informed starting point can substantially improve both the efficiency and quality of reasoning. For each query $\mathbf{c}_i$, we first derive a base initialization $\mathbf{z}_{\text{base},i}$ via an initial Chain-of-Thought (CoT) decoding, using the prefix of the latent sequence:

$$\mathbf{z}_{\text{base},i} = H_{\boldsymbol{\theta}}(\mathbf{c}_i)_{1:L'} \tag{4}$$

where $H_{\boldsymbol{\theta}}(\mathbf{c}_i)$ denotes the full latent sequence produced by $\pi_{\boldsymbol{\theta}}$ under greedy decoding, and the subscript $1:L'$ selects the first L' latent vectors. This base state serves as a preliminary reasoning trajectory, refined via the retrieved neighborhood $\mathcal{N}_k(\mathbf{c}_i)$ to form a superior initialization $\mathbf{z}_{0,i}$. A naive approach averages retrieved final latent sequences $\mathbf{z}_j^*$, but this risks misleading due to conflicting patterns across different queries. Instead, we prioritize the optimization *journey* from initial $\mathbf{z}_{\text{base},j}$ to refined $\mathbf{z}_j^*$ (over final latent states alone), capturing this as momentum $\Delta\mathbf{z}_j = \mathbf{z}_j^* - \mathbf{z}_{\text{base},j}$ and introducing **weighted momentum transfer**:

By aggregating weighted momenta, we steer $\mathbf{z}_{\text{base},i}$ toward fruitful latent space regions via past trajectories:

$$\mathbf{z}_{0,i} = \mathbf{z}_{\text{base},i} + \sum_{j \in \mathcal{N}_k(\mathbf{c}_i)} \alpha_j \Delta\mathbf{z}_j, \quad \alpha_j \propto \exp\big(S(\mathbf{e}_i, \mathbf{e}_j)\big) \tag{5}$$

This steers initialization toward promising regions *along* historically high-performance trajectories, enabling reasoning to start from a grounded, well-informed foundation.

Self-Supervised Refinement and Archiving. Although the informed initial state $\mathbf{z}_{0,i}$ offers a promising foundation, it is not tailored to the specific context $\mathbf{c}_i$ and thus requires refinement to enhance reasoning performance. We adopt a self-rewarding strategy, a paradigm broadly validated in prior work (Li et al., 2025a; Yuan et al., 2025; Zuo et al., 2025). Concretely, the LLM π_θ serves as its own evaluator by assigning a quality score $Q(\mathbf{y}_k)$ to the output $\mathbf{y}_k$ generated under the guidance of $\mathbf{z}_{0,i}$ (see the detailed implementation of $Q(\cdot)$ in Appendix C.1). The latent sequence is then iteratively refined through gradient ascent with respect to this self-supervised signal. The gradient of $J(\mathbf{z}_k)$ is estimated via policy gradient (Williams, 1992) as:

$$\begin{aligned}\nabla_{\mathbf{z}_k} J(\mathbf{z}_k) &= \nabla_{\mathbf{z}_k} \mathbb{E}_{\mathbf{y}\sim p(\cdot|\mathbf{c}_i,\mathbf{z}_k;\boldsymbol{\theta})}[Q(\mathbf{y})] \\ &\approx \frac{1}{M}\sum_{m=1}^{M} Q(\mathbf{y}^{(m)}) \nabla_{\mathbf{z}_k} \log p(\mathbf{y}^{(m)}|\mathbf{c}_i,\mathbf{z}_k;\boldsymbol{\theta}),\end{aligned} \tag{6}$$

where $\{\mathbf{y}^{(m)}\}_{m=1}^{M}$ are samples drawn from $p(\cdot \mid \mathbf{c}_i, \mathbf{z}_k; \boldsymbol{\theta})$ by M times. Accordingly, the latent state is iteratively updated as $\mathbf{z}_{k+1} \leftarrow \mathbf{z}_k + \eta \nabla_{\mathbf{z}_k} J(\mathbf{z}_k)$, where η is the learning rate. The refinement terminates either after K iterations or once $\mathbb{E}[Q(\mathbf{y}_k)]$ has failed to increase for three successive rounds, yielding the final latent state $\mathbf{z}_i^*$, under whose guidance π_θ produces the ultimate output $\mathbf{y}$. The triplet $(\mathbf{e}_{\mathbf{c}_i}, \mathbf{z}_{\text{base},i}, \mathbf{z}_i^*)$ is archived into $\mathcal{M}$ whenever $\mathbb{E}[Q(\mathbf{y}_k)]$ exceeds a predefined threshold τ (see detailed process in Appendix C.2). Thus, the preservation of high-confidence experiences deepens the repository from which `LatentEvolve` continually distills its evolving knowledge.

4.2 Nighttime Deliberative Consolidation

While the *daytime scaling* excels at rapid, instance-level adaptation, its knowledge remains fragmented within the buffer. To achieve generalizable improvement, these scattered experiences must be integrated into a coherent procedural skill, which is also the purpose of the *nighttime scaling*, analogous to the neocortex's role in consolidating memories into abstract knowledge during sleep.

Latent Weaver. To perform this consolidation, we introduce the *latent weaver* $\mathbf{W}_{\boldsymbol{\psi}}$, aimed at distilling the collective wisdom from the episodic buffer. Technically, $\mathbf{W}_{\boldsymbol{\psi}}$ is trained to predict the refined latent sequence $\mathbf{z}_j^*$ conditioned on the context embedding $\mathbf{e}_{\mathbf{c}_j}$ and the base state $\mathbf{z}_{\text{base},j}$, thereby enabling rapid and precise test-time scaling. We instantiate $\mathbf{W}_{\boldsymbol{\psi}}$ via a smaller LLM ψ.

Consolidation through Experience Replay. Periodically, after the episodic buffer $\mathcal{M}$ has accumulated a sufficient number of high-confidence experiences, the nighttime consolidation is triggered. The experience triplets $\{(\mathbf{e}_{\mathbf{c}_j}, \mathbf{z}_{\text{base},j}, \mathbf{z}_j^*)\}$ from $\mathcal{M}$ are leveraged to update the parameters $\boldsymbol{\psi}$ of the latent weaver. The training objective is to minimize the reconstruction error between the weaver's prediction and the archived optimal latent sequence:

$$\mathcal{L}(\boldsymbol{\psi}) = \mathbb{E}_{(\mathbf{e}_{\mathbf{c}_j}, \mathbf{z}_{\text{base},j}, \mathbf{z}_j^*)\sim\mathcal{M}} \left[\left\| \mathbf{W}_{\boldsymbol{\psi}}(\mathbf{e}_{\mathbf{c}_j}, \mathbf{z}_{\text{base},j}) - \mathbf{z}_j^* \right\|_2^2 \right], \tag{7}$$

which effectively *weaves* the sparse, episodic optimization experiences into the continuous parametric space of the model. Through such nighttime scaling, $\mathbf{W}_{\psi}$ is imbued with procedural intuition and capable of generating superior initial reasoning paths for subsequent LLM reasoning. In the next section, we illustrate the overall picture of the dual-phase evolving process.

4.3 Dual-Phase Evolving Scaling

In this section, we formally describe the dual-phase evolution process of `LatentEvolve`. The daytime and nighttime mechanisms, though effective in isolation, realize their full potential when embedded in a recurring cycle that mirrors the brain's complementary learning systems: the hippocampus for rapid encoding of episodic traces and the neocortex for gradual schema formation. For each incoming query $\mathbf{c}_i$, the latent weaver $\mathbf{W}_{\boldsymbol{\psi}}$ first transforms the base latent state (except in the initial round, when $\mathbf{W}_{\boldsymbol{\psi}}$ remains untrained) to yield a refined $\mathbf{z}'_{\text{base},i}$. This is followed by daytime scaling, which, via momentum transfer, produces $\mathbf{z}_{0,i}$. Iterative refinement then generates the final latent sequence $\mathbf{z}_i^*$ through self-guided optimization, ensuring that each query benefits not only from episodic recall but also from the procedural insights accumulated during prior nighttime consolidations:

$$\begin{aligned}\mathbf{z}'_{\text{base},i} &= \mathbf{W}_{\boldsymbol{\psi}}(\mathbf{e}_{\mathbf{c}_i}, \mathbf{z}_{\text{base},i}), \\ \mathbf{z}_i^* &= \Phi_{\text{day}}(\mathbf{c}_i, \mathbf{z}'_{\text{base},i}, \mathcal{M}; \boldsymbol{\theta})\end{aligned} \tag{8}$$

where Φ_{day} denotes daytime optimization of a given query $\mathbf{c}_i$ under the assistance of $\mathcal{M}$ and π_θ, as described in Equation (6). Over time, the episodic buffer $\mathcal{M}$ accumulates triplets of adaptations $\{(\mathbf{e}_{\mathbf{c}_j}, \mathbf{z}_{\text{base},j}, \mathbf{z}_j^*)\}$. At periodic intervals (specifically, we set $T = 200$ test-time instances as one cycle), nighttime scaling is invoked to consolidate accumulated experiences by updating $\mathbf{W}_{\boldsymbol{\psi}}$:

$$\mathbf{W}_{\boldsymbol{\psi}} \leftarrow \Phi_{\text{night}}(\mathcal{M}, \mathbf{W}_{\boldsymbol{\psi}}), \tag{9}$$

Table 1: **Performance Comparison** across `Qwen2.5-7b` against thirteen baselines and on eight benchmarks. The best and second best results are highlighted and underlined, respectively.

	Method	General QA	Mathematical Reasoning				Sci. Reasoning		Med. Reasoning
		MMLU	GSM8K	MATH-500	AIME24	AIME25	SciBench	GPQA	JAMA Clinical
Qwen2.5-7b	***Prompting (training-free)***								
	Vanilla Model	72.10	89.32	75.73	11.84	8.31	11.27	35.07	47.72
	CoT	73.50	91.38	76.21	11.95	8.92	11.99	35.86	50.96
	Reinforcement Learning								
	Self-Rewarding	63.10	88.30	73.14	6.25	9.89	9.36	30.30	47.07
	Genius	58.30	87.93	76.89	6.82	9.11	13.60	29.67	41.78
	GRPO	68.90	92.30	78.20	18.49	15.10	-	36.61	51.62
	Reinforce	63.77	92.30	77.83	17.70	14.27	-	36.99	49.16
	Reinforce++	65.90	92.60	78.01	18.75	14.94	-	35.98	50.40
	Latent Reasoning								
	Coprocessor	68.10	83.60	77.19	14.16	10.26	-	35.47	43.70
	SoftCoT	62.30	80.13	74.13	10.78	8.17	-	35.10	49.70
	Test-time Scaling								
	Self-Consistency	69.80	88.62	69.40	14.94	10.20	12.13	36.74	51.62
	Self-Refine	61.40	86.33	59.32	13.38	8.64	9.36	35.85	45.64
	LatentSeek	68.50	91.58	75.60	14.06	15.41	14.45	34.21	50.40
	TTRL	70.90	92.80	77.39	**23.33**	16.71	13.92	36.36	49.16
	`LatentEvolve`	**77.30**	**94.53**	**80.55**	20.56	**18.37**	**19.79**	**43.67**	**52.94**

where Φ_{night} denotes experience replay and parametric distillation, as described in Equation (7). The overall evolution is thus expressed as the alternating transformation

$$(\mathcal{M}, \mathbf{W}_\psi) \xrightarrow{\Phi_{\text{day}}} \mathcal{M}' \xrightarrow{\Phi_{\text{night}}} (\mathcal{M}', \mathbf{W}'_\psi), \tag{10}$$

which continually refreshes the episodic buffer while also imbuing the weaver with generalized procedural knowledge. This perpetual cycle of experience and consolidation allows `LatentEvolve`'s reasoning capabilities to self-evolve on the fly, entirely in an **unsupervised manner without reliance on any external labels**. For clarity, we provide the complete pseudocode of LatentEvolve in Algorithm 1 in Appendix E.

5 Experiments

5.1 Experiment Setting

Backbones. To evaluate the generalizability of `LatentEvolve`, we experiment with LLMs from different families and of varying sizes, including `Llama-3.2-3b` (Grattafiori et al., 2024), `Qwen2.5-7b-instruct` (Qwen et al., 2025), `Qwen3-4b-instruct-2507`, `Qwen3-8b` (Yang et al., 2025), and `Gemma-3-12b-it` (Team et al., 2025).

Benchmarks. We conduct a comprehensive evaluation of `LatentEvolve` across eight benchmarks from four task domains: ■ *general QA*, MMLU (Hendrycks et al., 2021a); ■ *mathematical reasoning*, including GSM8K (Cobbe et al., 2021), MATH-500 (Hendrycks et al., 2021b), and AIME 2024/2025 (Li et al., 2024); ■ *scientific reasoning*, SciBench (Wang et al., 2024b) and GPQA-Diamond (Rein et al., 2023); ■ *medical reasoning*, JAMA Clinical Challenge (Chen et al., 2025a). Detailed dataset statistics are listed in Appendix C.4.

Evaluation Setup. We apply `LatentEvolve` to each benchmark's test set, except for AIME24/25, where we evaluate after applying `LatentEvolve` on MATH-500 due to limited test size. We set the maximum generation length to 2048 tokens. The small LLM used for latent weaver $\mathbf{W}_\psi$ is set as `Qwen-2.5-1.5b`. The dimension L' in Equation (4) is 15, the threshold τ is 0.5, and the dual-evolution period T is 200. The learning rate η is 0.3, iteration count $K = 10$, and the sampling times M in Equation (6) is 8. For performance evaluation, we employ *Pass@1* and *Avg@16* accuracy under a sampling temperature of 0.

Baselines. We compare against well-established baselines:

- **Prompting (training-free)**: vanilla model and CoT (Wei et al., 2023);
- **Reinforcement Learning**: (1) self-rewarding methods, e.g., Self-Rewarding (Yuan et al., 2025) and Genius (Xu et al., 2025a), and (2) verifiable reward methods, e.g., GRPO (DeepSeek-AI et al., 2025), Reinforce (Williams, 1992), and Reinforce++ (Hu et al., 2025). The latter three baselines are trained on each dataset's training split and evaluated on its test split. Due to limited AIME24/25 size, models trained on MATH are directly evaluated on these benchmarks. Re-

Table 2: **Performance Comparison** of the vanilla model versus that enhanced with `LatentEvolve` across five LLM backbones. The Δ row indicates the absolute improvement.

LLM Backbone	Method	MMLU	GSM8K	MATH	SciBench	GPQA	JAMA	AIME24	AIME25
`Llama3.2-3b`	Vanilla	57.60	71.31	42.30	6.79	23.31	45.14	0.00	0.00
	+LE	64.30	75.51	52.40	9.39	29.29	48.44	7.59	5.92
	Δ	+6.70	+4.20	+10.10	+2.60	+5.98	+3.30	+7.59	+5.92
`Qwen2.5-7b`	Vanilla	72.10	89.32	75.73	11.27	35.07	47.72	11.84	8.31
	+LE	77.30	94.53	80.55	19.79	43.67	52.94	20.56	18.37
	Δ	+5.20	+5.21	+4.82	+8.52	+8.60	+5.22	+8.72	+10.06
`Qwen3-4b`	Vanilla	74.10	89.23	84.71	12.28	41.28	51.49	24.17	18.75
	+LE	78.30	97.83	95.32	31.93	58.59	53.67	33.41	25.56
	Δ	+4.20	+8.60	+10.61	+19.65	+17.31	+2.18	+9.24	+6.81
`Qwen3-8b`	Vanilla	75.20	90.52	86.93	6.36	39.47	53.08	29.04	19.86
	+LE	79.80	98.21	97.11	10.83	46.18	54.60	40.13	27.78
	Δ	+4.60	+7.69	+10.18	+4.47	+6.71	+1.52	+11.09	+7.92
`Gemma-3-12b`	Vanilla	75.50	89.23	84.80	10.84	40.48	49.50	23.52	18.27
	+LE	77.60	92.79	89.32	18.93	47.20	55.06	32.75	26.23
	Δ	+2.10	+3.56	+4.52	+8.09	+6.72	+5.56	+9.23	+7.96

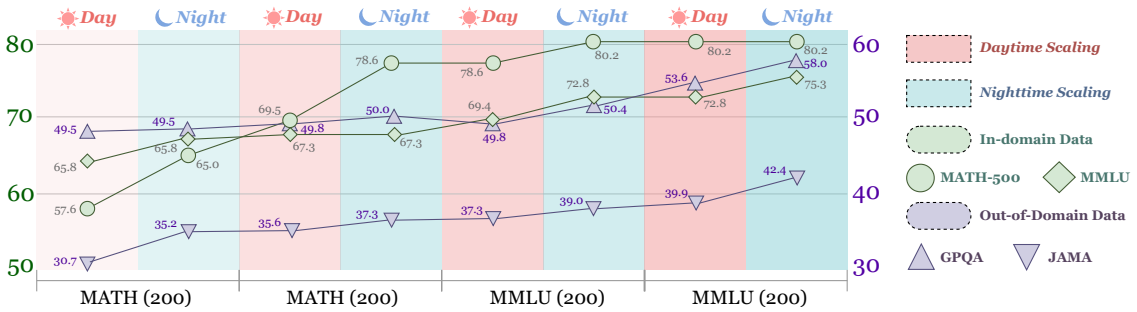

Figure 3: The evolving dynamics of `LatentEvolve` when applied on `Gemma-3-12b` across two in-domain and out-of-domain datasets.

sults on SciBench are omitted for these baselines due to the absence of a dedicated training set.

- **Latent Reasoning**, including Coprocessor (Liu et al., 2024) and SoftCoT (Xu et al., 2025c).
- **Test-time Scaling** methods, including Self-Consistency (Wang et al., 2023b), Self-refine (Madaan et al., 2023), LatentSeek (Li et al., 2025a), and TTRL (Xiang et al., 2025).

5.2 Main Results

Obs. ❶: `LatentEvolve` performs well across most task domains. As shown in Table 1, most baselines fail to deliver consistent gains across all benchmark categories. LatentSeek and TTRL excel in mathematical reasoning yet underperform in other domains: for instance, LatentSeek with `Qwen2.5-7b` yields limited improvement on MMLU and exhibits only marginal gains on SciBench, while TTRL with `Qwen2.5-7b` provides modest benefit on JAMA Clinical (+1.44%). In contrast, `LatentEvolve` not only matches or surpasses TTRL in the general QA domain (e.g., on `Qwen2.5-7b`, MMLU +6.4%) but also achieves superior cross-domain performance, including a +8.52% improvement on SciBench.

Obs. ❷: `LatentEvolve` generalizes well across LLM backbones. In contrast to many baselines whose gains across different LLMs are highly inconsistent, `LatentEvolve` consistently delivers positive improvements across models of varying scales, as clearly illustrated in Table 2. Notably, its benefits naturally *scale* with model size: on SciBench, for example, the improvement rises from 2.60% with `Llma3.2-3b` to 8.09% with `Gemma-3-12b`.

5.3 Generalization and Continuality

This section investigates the continual learning and generalization capacity of `LatentEvolve`. Figure 3 illustrates performance trajectories when `LatentEvolve`, instantiated with `Gemma-3-12b`, sequentially processes test data from MATH and MMLU. In-domain evaluation is conducted on MATH and MMLU, while out-of-domain evaluation is performed on GPQA

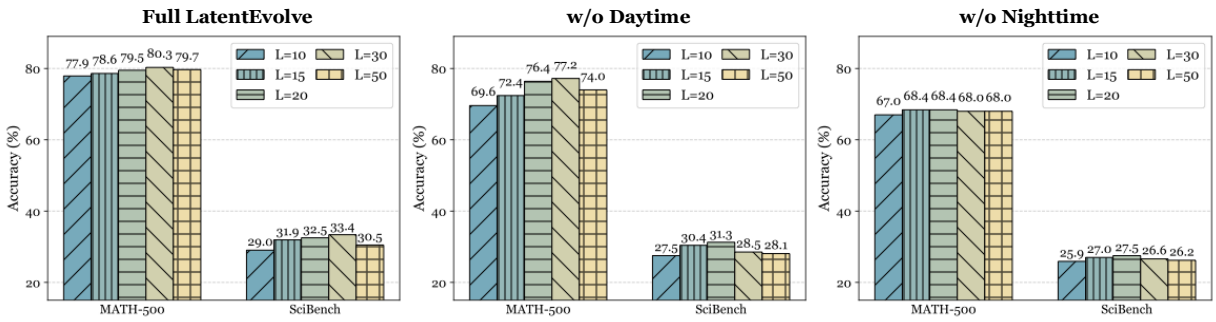

Figure 4: The ablation study and parameter sensitivity analysis of `LatentEvolve`.

and JAMA. Red zones denote evaluations after a daytime scaling step with updated episodic buffer $\mathcal{M}'$, whereas blue zones correspond to evaluations after a nighttime scaling step yielding updated latent weaver $\mathbf{W}'_{\psi}$.

Obs. ❸: `LatentEvolve` generalizes across domains. As shown in Figure 3, after two rounds of MATH data, performance improves substantially in-domain (57.6% $\rightarrow$ 78.6%) while also transferring gains to distinct domains (JAMA +6.6%, MMLU +1.5%). Notably, nighttime scaling proves more conducive to such cross-domain generalization: the first nighttime scaling on MATH increases JAMA by +4.5%, compared to only +0.4% from daytime scaling. This highlights that nighttime scaling, akin to cortical consolidation in the human brain, integrates experiences into more transferable knowledge, whereas daytime scaling yields more superficial improvements. Moreover, `LatentEvolve` demonstrates strong continual learning ability: after two rounds of MMLU data, `Gemma-3-12b` shows not only no degradation but a slight improvement on MATH (78.6% $\rightarrow$ 80.2%), highlighting the robustness of `LatentEvolve` in continual adaptation.

5.4 Framework Analysis

Component and Hyperparameter Analysis. We summarize the key ablation and sensitivity findings here, with full results deferred to Figure 4. Removing either daytime or nighttime scaling weakens `LatentEvolve`, confirming that episodic momentum transfer and nighttime consolidation are complementary. The nighttime component contributes more substantially: at $L' = 30$, removing daytime scaling reduces SciBench from 33.4% to 28.5%, whereas removing nighttime scaling causes a 6.8% drop. For the latent length L', both MATH and SciBench exhibit an inverted-U trend from 10 to 50, with the best results at $L' = 30$ (MATH 80.3%, SciBench 33.4%). Additional analyses of T, the retrieval metric, and the threshold τ are reported in Appendices D.3 and D.4.

Auxiliary Evidence. The appendix further supports the same mechanism from several perspectives. Case studies show machine-native latent reasoning patterns in Table 6, while token statistics indicate shorter reasoning than vanilla CoT in Table 4. Efficiency measurements in Tables 7 and 8 show moderate nighttime overhead with consistent performance gains.

Visualizations. We visualize latent representations with t-SNE on SciBench and MATH in Figure 5. Across daytime–nighttime cycles, refined latents generated by the nighttime weaver move closer to successful latent regions, while base latents remain relatively distant. The centroid-distance results in Tables 11 and 12 show the same trend, indicating that nighttime consolidation provides better latent initializations and helps stabilize downstream reasoning.

6 Conclusion

In this work, we propose `LatentEvolve`, a self-evolving latent test-time scaling framework inspired by complementary learning systems. By alternating *daytime scaling* for fast episodic adaptation with *nighttime scaling* for slow procedural consolidation, our approach enables LLMs to accumulate and refine experiential knowledge during inference without supervision. Experiments on eight benchmarks and five backbones show that `LatentEvolve` surpasses state-of-the-art TTS methods (*e.g.*, TTRL, LatentSeek), transfers across tasks, and exhibits continual learning. Broadly, our work points to a new path where LLMs not only *scale at test time*, but also *evolve through it*, toward adaptive, accumulative human-like intelligence.

Limitations

Although `LatentEvolve` demonstrates strong performance across multiple reasoning benchmarks, it still has several limitations. First, our method introduces additional test-time computation due to iterative latent refinement and periodic nighttime consolidation. While the measured overhead is moderate, it may still be costly for latency-sensitive applications. Second, `LatentEvolve` relies on self-reward signals produced by the LLM itself. Since no external labels are used during test time, the quality of latent optimization depends on the model's ability to evaluate its own outputs, which may be unreliable for highly ambiguous or open-ended tasks. Third, the current experiments mainly focus on structured reasoning scenarios, including mathematical reasoning, scientific reasoning, medical QA, and general QA. Its effectiveness on more open-ended generation tasks, dialogue settings, or long-horizon agentic tasks remains to be further investigated. Finally, although the backbone LLM parameters remain frozen, the latent weaver is periodically updated, which introduces an auxiliary training component and requires careful management of accumulated latent experiences.

Ethical considerations

`LatentEvolve` aims to improve the reasoning ability of LLMs by optimizing latent representations at test time. This work does not introduce new datasets involving private, sensitive, or personally identifiable information, and all experiments are conducted on existing public benchmarks. However, improving reasoning performance may also increase users' trust in model outputs, which could be risky when the model is applied to high-stakes domains such as medicine, education, or scientific decision-making. In particular, since `LatentEvolve` uses self-generated reward signals, incorrect self-evaluation may still lead to confident but wrong answers. Therefore, systems based on this method should not be used as standalone decision-makers in high-stakes scenarios without human oversight and task-specific validation. In addition, the extra test-time computation required by latent refinement and nighttime consolidation may increase energy consumption, so future work should further improve the efficiency of the framework.

References

Armen Aghajanyan, Lili Yu, Alexis Conneau, Wei-Ning Hsu, Karen Hambardzumyan, Susan Zhang, Stephen Roller, Naman Goyal, Omer Levy, and Luke Zettlemoyer. 2023. Scaling laws for generative mixed-modal language models. *Preprint*, arXiv:2301.03728.

Ekin Akyürek, Mehul Damani, Adam Zweiger, Linlu Qiu, Han Guo, Jyothish Pari, Yoon Kim, and Jacob Andreas. 2024. The surprising effectiveness of test-time training for few-shot learning. *arXiv preprint arXiv:2411.07279*.

Huan ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, Hongru Wang, Han Xiao, Yuhang Zhou, Shaokun Zhang, Jiayi Zhang, Jinyu Xiang, Yixiong Fang, Qiwen Zhao, Dongrui Liu, and 8 others. 2025. A survey of self-evolving agents: On path to artificial super intelligence. *Preprint*, arXiv:2507.21046.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2024. Graph of thoughts: Solving elaborate problems with large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17682–17690.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. 2024. Large language monkeys: Scaling inference compute with repeated sampling. *Preprint*, arXiv:2407.21787.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, and 1 others. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.

Hanjie Chen, Zhouxiang Fang, Yash Singla, and Mark Dredze. 2025a. Benchmarking large language models on answering and explaining challenging medical questions. 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)*, pages 3563–3599.

Xiaoyin Chen, Jiarui Lu, Minsu Kim, Dinghuai Zhang, Jian Tang, Alexandre Piché, Nicolas Gontier, Yoshua Bengio, and Ehsan Kamalloo. 2025b. Self-evolving curriculum for llm reasoning. *Preprint*, arXiv:2505.14970.

Zehui Chen, Kuikun Liu, Qiuchen Wang, Wenwei Zhang, Jiangning Liu, Dahua Lin, Kai Chen, and Feng Zhao. 2024. Agent-flan: Designing data and methods of effective agent tuning for large language models. *Preprint*, arXiv:2403.12881.

Joon-Young Choi, Junho Kim, Jun-Hyung Park, Wing-Lam Mok, and SangKeun Lee. 2023. SMoP: Towards efficient and effective prompt tuning with sparse mixture-of-prompts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14306–14316, Singapore. Association for Computational Linguistics.

Ho-Lam Chung, Teng-Yun Hsiao, Hsiao-Ying Huang, Chunerh Cho, Jian-Ren Lin, Zhang Ziwei, and Yun-Nung Chen. 2025. Revisiting test-time scaling: A survey and a diversity-aware method for efficient reasoning. *Preprint*, arXiv:2506.04611.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Jinyuan Fang, Yanwen Peng, Xi Zhang, Yingxu Wang, Xinhao Yi, Guibin Zhang, Yi Xu, Bin Wu, Siwei Liu, Zihao Li, Zhaochun Ren, Nikos Aletras, Xi Wang, Han Zhou, and Zaiqiao Meng. 2025. A comprehensive survey of self-evolving ai agents: A new paradigm bridging foundation models and lifelong agentic systems. *Preprint*, arXiv:2508.07407.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. 2025. Scaling up test-time compute with latent reasoning: A recurrent depth approach. *Preprint*, arXiv:2502.05171.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. 2024. Critic: Large language models can self-correct with tool-interactive critiquing. *Preprint*, arXiv:2305.11738.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Lin Gui, Cristina Gârbacea, and Victor Veitch. 2024. Bonbon alignment for large language models and the sweetness of best-of-n sampling. *Preprint*, arXiv:2406.00832.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. 2024. Training large language models to reason in a continuous latent space. *Preprint*, arXiv:2412.06769.

Bolei He, Nuo Chen, Xinran He, Lingyong Yan, Zhenkai Wei, Jinchang Luo, and Zhen-Hua Ling. 2024. Retrieving, rethinking and revising: The chain-of-verification can improve retrieval augmented generation. *Preprint*, arXiv:2410.05801.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. Measuring massive multitask language understanding. *Preprint*, arXiv:2009.03300.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the math dataset. *Preprint*, arXiv:2103.03874.

Jian Hu, Jason Klein Liu, Haotian Xu, and Wei Shen. 2025. Reinforce++: An efficient rlhf algorithm with robustness to both prompt and reward models. *Preprint*, arXiv:2501.03262.

Jonas Hübotter, Sascha Bongni, Ido Hakimi, and Andreas Krause. 2025. Efficiently learning at test-time: Active fine-tuning of LLMs. In *The Thirteenth International Conference on Learning Representations*.

Robert Irvine, Douglas Boubert, Vyas Raina, Adian Liusie, Ziyi Zhu, Vineet Mudupalli, Aliaksei Korshuk, Zongyi Liu, Fritz Cremer, Valentin Assassi, Christie-Carol Beauchamp, Xiaoding Lu, Thomas Rialan, and William Beauchamp. 2023. Rewarding chatbots for real-world engagement with millions of users. *Preprint*, arXiv:2303.06135.

Minki Kang, Jongwon Jeong, and Jaewoong Cho. 2025. T1: Tool-integrated self-verification for test-time compute scaling in small language models. *Preprint*, arXiv:2504.04718.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *Preprint*, arXiv:2001.08361.

Dharshan Kumaran, Demis Hassabis, and James L McClelland. 2016. What learning systems do intelligent agents need? complementary learning systems theory updated. *Trends in cognitive sciences*, 20(7):512–534.

Kuang-Huei Lee, Ian Fischer, Yueh-Hua Wu, Dave Marwood, Shumeet Baluja, Dale Schuurmans, and Xinyun Chen. 2025. Evolving deeper llm thinking. *Preprint*, arXiv:2501.09891.

Hengli Li, Chenxi Li, Tong Wu, Xuekai Zhu, Yuxuan Wang, Zhaoxin Yu, Eric Hanchen Jiang, Song-Chun Zhu, Zixia Jia, Ying Nian Wu, and Zilong Zheng. 2025a. Seek in the dark: Reasoning via test-time instance-level policy gradient in latent space. *Preprint*, arXiv:2505.13308.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q Jiang, Ziju Shen, and 1 others. 2024. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. *Hugging Face repository*, 13(9):9.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. 2025b. Search-o1: Agentic search-enhanced large reasoning models. *Preprint*, arXiv:2501.05366.

Shalev Lifshitz, Sheila A McIlraith, and Yilun Du. 2025. Multi-agent verification: Scaling test-time compute with multiple verifiers. *arXiv preprint arXiv:2502.20379*.

Luyang Liu, Jonas Pfeiffer, Jiaxing Wu, Jun Xie, and Arthur Szlam. 2024. Deliberation in latent space via differentiable cache augmentation. *arXiv preprint arXiv:2412.17747*.

Yuejiang Liu, Parth Kothari, Bastien Van Delft, Baptiste Bellot-Gurlet, Taylor Mordan, and Alexandre Alahi. 2021. Ttt++: When does self-supervised test-time training fail or thrive? *Advances in Neural Information Processing Systems*, 34:21808–21820.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. *Preprint*, arXiv:2303.17651.

James L McClelland, Bruce L McNaughton, and Randall C O'Reilly. 1995. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102(3):419.

José I. Orlicki. 2025. Beyond words: A latent memory approach to internal reasoning in llms. *Preprint*, arXiv:2502.21030.

Zhiyuan Peng, Xuyang Wu, Qifan Wang, and Yi Fang. 2024. Soft prompt tuning for augmenting dense retrieval with large language models. *Preprint*, arXiv:2307.08303.

Jiahao Qiu, Xinzhe Juan, Yimin Wang, Ling Yang, Xuan Qi, Tongcheng Zhang, Jiacheng Guo, Yifu Lu, Zixin Yao, Hongru Wang, Shilong Liu, Xun Jiang, Liu Leqi, and Mengdi Wang. 2025a. Agentdistill: Training-free agent distillation with generalizable mcp boxes. *Preprint*, arXiv:2506.14728.

Jiahao Qiu, Xuan Qi, Tongcheng Zhang, Xinzhe Juan, Jiacheng Guo, Yifu Lu, Yimin Wang, Zixin Yao, Qihan Ren, Xun Jiang, Xing Zhou, Dongrui Liu, Ling Yang, Yue Wu, Kaixuan Huang, Shilong Liu, Hongru Wang, and Mengdi Wang. 2025b. Alita: Generalist agent enabling scalable agentic reasoning with minimal predefinition and maximal self-evolution. *Preprint*, arXiv:2505.20286.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2023. Gpqa: A graduate-level google-proof qa benchmark. *Preprint*, arXiv:2311.12022.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Preprint*, arXiv:2303.11366.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *Preprint*, arXiv:2408.03314.

Yifan Song, Weimin Xiong, Xiutian Zhao, Dawei Zhu, Wenhao Wu, Ke Wang, Cheng Li, Wei Peng, and Sujian Li. 2024. Agentbank: Towards generalized llm agents via fine-tuning on 50000+ interaction trajectories. *Preprint*, arXiv:2410.07706.

Alex Su, Haozhe Wang, Weiming Ren, Fangzhen Lin, and Wenhu Chen. 2025. Pixel reasoner: Incentivizing pixel-space reasoning with curiosity-driven reinforcement learning. *Preprint*, arXiv:2505.15966.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. 2020. Test-time training with self-supervision for generalization under distribution shifts. In *International conference on machine learning*, pages 9229–9248. PMLR.

Yuchang Sun, Yanxi Chen, Yaliang Li, and Bolin Ding. 2025. Enhancing latent computation in transformers with latent tokens. *Preprint*, arXiv:2505.12629.

Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. 2025. Dynamic cheatsheet: Test-time learning with adaptive memory. *Preprint*, arXiv:2504.07952.

Wenhui Tan, Jiaze Li, Jianzhong Ju, Zhenbo Luo, Jian Luan, and Ruihua Song. 2025. Think silently, think fast: Dynamic latent compression of llm reasoning chains. *Preprint*, arXiv:2505.16552.

Xiangru Tang, Tianrui Qin, Tianhao Peng, Ziyang Zhou, Daniel Shao, Tingting Du, Xinming Wei, Peng Xia, Fang Wu, He Zhu, Ge Zhang, Jiaheng Liu, Xingyao Wang, Sirui Hong, Chenglin Wu, Hao Cheng, Chi Wang, and Wangchunshu Zhou. 2025. Agent kb: Leveraging cross-domain experience for agentic problem solving. *Preprint*, arXiv:2507.06229.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, and 1 others. 2025. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.

Pablo Villalobos, Anson Ho, Jaime Sevilla, Tamay Besiroglu, Lennart Heim, and Marius Hobbhahn. 2022. Will we run out of data? limits of llm scaling based on human-generated data. *arXiv preprint arXiv:2211.04325*.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023a. Voyager: An open-ended embodied agent with large language models. *Preprint*, arXiv:2305.16291.

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. 2024a. Mixture-of-agents enhances large language model capabilities. *arXiv preprint arXiv:2406.04692*.

Xiaoxuan Wang, Ziniu Hu, Pan Lu, Yanqiao Zhu, Jieyu Zhang, Satyen Subramaniam, Arjun R. Loomba, Shichang Zhang, Yizhou Sun, and Wei Wang. 2024b. Scibench: Evaluating college-level scientific problem-solving abilities of large language models. *Preprint*, arXiv:2307.10635.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. Self-consistency improves chain of thought reasoning in language models. *Preprint*, arXiv:2203.11171.

Yu Wang, Yifan Gao, Xiusi Chen, Haoming Jiang, Shiyang Li, Jingfeng Yang, Qingyu Yin, Zheng Li, Xian Li, Bing Yin, and 1 others. 2024c. Memoryllm: Towards self-updatable large language models. *arXiv preprint arXiv:2402.04624*.

Yu Wang, Dmitry Krotov, Yuanzhe Hu, Yifan Gao, Wangchunshu Zhou, Julian McAuley, Dan Gutfreund, Rogerio Feris, and Zexue He. 2025a. M+: Extending memoryllm with scalable long-term memory. *Preprint*, arXiv:2502.00592.

Zhenhailong Wang, Haiyang Xu, Junyang Wang, Xi Zhang, Ming Yan, Ji Zhang, Fei Huang, and Heng Ji. 2025b. Mobile-agent-e: Self-evolving mobile assistant for complex tasks. *Preprint*, arXiv:2501.11733.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Yuxiang Wei, Olivier Duchenne, Jade Copet, Quentin Carbonneaux, Lingming Zhang, Daniel Fried, Gabriel Synnaeve, Rishabh Singh, and Sida I. Wang. 2025. Swe-rl: Advancing llm reasoning via reinforcement learning on open software evolution. *Preprint*, arXiv:2502.18449.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256.

Junde Wu, Jiayuan Zhu, and Yuyuan Liu. 2025. Agentic reasoning: Reasoning llms with tools for the deep research. *Preprint*, arXiv:2502.04644.

Violet Xiang, Charlie Snell, Kanishk Gandhi, Alon Albalak, Anikait Singh, Chase Blagden, Duy Phung, Rafael Rafailov, Nathan Lile, Dakota Mahan, Louis Castricato, Jan-Philipp Franken, Nick Haber, and Chelsea Finn. 2025. Towards system 2 reasoning in llms: Learning how to think with meta chain-of-thought. *Preprint*, arXiv:2501.04682.

Yao Xiao, Lu Xu, Jiaxi Li, Wei Lu, and Xiaoli Li. 2023. Decomposed prompt tuning via low-rank reparameterization. *Preprint*, arXiv:2310.10094.

Fangzhi Xu, Hang Yan, Chang Ma, Haiteng Zhao, Qiushi Sun, Kanzhi Cheng, Junxian He, Jun Liu, and Zhiyong Wu. 2025a. Genius: A generalizable and purely unsupervised self-training framework for advanced reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13153–13167, Vienna, Austria. Association for Computational Linguistics.

Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. 2025b. Softcot: Soft chain-of-thought for efficient reasoning with llms. *Preprint*, arXiv:2502.12134.

Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. 2025c. Softcot++: Test-time scaling with soft chain-of-thought reasoning. *Preprint*, arXiv:2505.11484.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. 2024. Swe-agent: Agent-computer interfaces enable automated software engineering. *Preprint*, arXiv:2405.15793.

Hai Ye, Mingbao Lin, Hwee Tou Ng, and Shuicheng Yan. 2025. Multi-agent sampling: Scaling inference compute for data synthesis with tree search-based agentic collaboration. *Preprint*, arXiv:2412.17061.

Ping Yu, Jing Xu, Jason Weston, and Ilia Kulikov. 2024. Distilling system 2 into system 1. *Preprint*, arXiv:2407.06023.

Yue Yu, Zhengxing Chen, Aston Zhang, Liang Tan, Chenguang Zhu, Richard Yuanzhe Pang, Yundi Qian, Xuewei Wang, Suchin Gururangan, Chao Zhang, Melanie Kambadur, Dhruv Mahajan, and Rui Hou. 2025. Self-generated critiques boost reward modeling for language models. *Preprint*, arXiv:2411.16646.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. 2025. Self-rewarding language models. *Preprint*, arXiv:2401.10020.

Yanwei Yue, Guibin Zhang, Boyang Liu, Guancheng Wan, Kun Wang, Dawei Cheng, and Yiyan Qi. 2025. Masrouter: Learning to route llms for multi-agent systems. *Preprint*, arXiv:2502.11133.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. 2023. Agenttuning: Enabling generalized agent abilities for llms. *Preprint*, arXiv:2310.12823.

Liang Zeng, Liangjun Zhong, Liang Zhao, Tianwen Wei, Liu Yang, Jujie He, Cheng Cheng, Rui Hu, Yang Liu, Shuicheng Yan, Han Fang, and Yahui Zhou. 2024. Skywork-math: Data scaling laws for mathematical reasoning in large language models – the story goes on. *Preprint*, arXiv:2407.08348.

Guibin Zhang, Muxin Fu, Guancheng Wan, Miao Yu, Kun Wang, and Shuicheng Yan. 2025a. G-memory: Tracing hierarchical memory for multi-agent systems. *Preprint*, arXiv:2506.07398.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. 2025b. Darwin godel machine: Open-ended evolution of self-improving agents. *Preprint*, arXiv:2505.22954.

Qiyuan Zhang, Fuyuan Lyu, Zexu Sun, Lei Wang, Weixu Zhang, Wenyue Hua, Haolun Wu, Zhihan Guo, Yufei Wang, Niklas Muennighoff, Irwin King, Xue Liu, and Chen Ma. 2025c. A survey on test-time scaling in large language models: What, how, where, and how well? *Preprint*, arXiv:2503.24235.

Yiqun Zhang, Hao Li, Chenxu Wang, Linyao Chen, Qiaosheng Zhang, Peng Ye, Shi Feng, Daling Wang, Zhen Wang, Xinrun Wang, and 1 others. 2025d. The avengers: A simple recipe for uniting smaller language models to challenge proprietary giants. *arXiv preprint arXiv:2505.19797*.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. 2024. Expel: Llm agents are experiential learners. *Preprint*, arXiv:2308.10144.

Andrew Zhao, Yiran Wu, Yang Yue, Tong Wu, Quentin Xu, Yang Yue, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. 2025. Absolute zero: Reinforced self-play reasoning with zero data. *Preprint*, arXiv:2505.03335.

Boyuan Zheng, Michael Y. Fatemi, Xiaolong Jin, Zora Zhiruo Wang, Apurva Gandhi, Yueqi Song, Yu Gu, Jayanth Srinivasa, Gaowen Liu, Graham Neubig, and Yu Su. 2025a. Skillweaver: Web agents can self-improve by discovering and honing skills. *Preprint*, arXiv:2504.07079.

Ziwei Zheng, Michael Yang, Jack Hong, Chenxiao Zhao, Guohai Xu, Le Yang, Chao Shen, and Xing Yu. 2025b. Deepeyes: Incentivizing "thinking with images" via reinforcement learning. *Preprint*, arXiv:2505.14362.

Xuanhe Zhou, Junxuan He, Wei Zhou, Haodong Chen, Zirui Tang, Haoyu Zhao, Xin Tong, Guoliang Li, Youmin Chen, Jun Zhou, Zhaojun Sun, Binyuan Hui, Shuo Wang, Conghui He, Zhiyuan Liu, Jingren Zhou, and Fan Wu. 2025. A survey of llm $\times$ data. *Preprint*, arXiv:2505.18458.

Rui-Jie Zhu, Tianhao Peng, Tianhao Cheng, Xingwei Qu, Jinfa Huang, Dawei Zhu, Hao Wang, Kaiwen Xue, Xuanliang Zhang, Yong Shan, and 1 others. 2025. A survey on latent reasoning. *arXiv preprint arXiv:2507.06203*.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang, Xinwei Long, Ermo Hua, Biqing Qi, Youbang Sun, Zhiyuan Ma, Lifan Yuan, Ning Ding, and Bowen Zhou. 2025. Ttrl: Test-time reinforcement learning. *Preprint*, arXiv:2504.16084.

A Use of Large Language Models

In preparing this work, we made limited use of large language models (LLMs) as auxiliary tools. Specifically, LLMs were employed to polish the language of early drafts, to assist with literature exploration, and to support information retrieval.

B Additional Related Work

Self-Evolving LLMs & Agents. How to evolve LLMs during their interactions with the environment has drawn increasing attention from the research community (ang Gao et al., 2025; Fang et al., 2025). Existing approaches generally rely on different carriers for evolution, including: **(I) parameter updates**, where prior experiences are encoded directly into model parameters (Zeng et al., 2023; Chen et al., 2024; Zhao et al., 2025; Chen et al., 2025b); **(II) experience databases**, where past problem-solving trajectories (Zhao et al., 2024; Song et al., 2024) or distilled experiential knowledge (Zhang et al., 2025a; Wang et al., 2025b; Tang et al., 2025) are used to enhance LLM capabilities; and **(III) skill condensation**, where reusable tools or skills (*e.g.*, APIs and MCPs) serve as functional assets (Zheng et al., 2025a; Suzgun et al., 2025; Zhang et al., 2025b; Qiu et al., 2025b,a). Distinct from these paradigms, `LatentEvolve` performs test-time evolution within the latent space, treating latent sequences as compact and adaptable carriers of accumulated reasoning experience.

C Methodology Details

C.1 Self-supervised Latent Refinement

The self-rewarding function $Q(\mathbf{y})$ in Equation (6) is formally defined as a weighted aggregation of numerical scores assigned by the LLM to distinct evaluation criteria. Following standard practices from (Lifshitz et al., 2025; Li et al., 2025a), for each candidate output $\mathbf{y}$, the LLM produces normalized scores $s_{\text{ans}}(\mathbf{y}), s_{\text{comp}}(\mathbf{y}), s_{\text{calc}}(\mathbf{y}), s_{\text{form}}(\mathbf{y}), s_{\text{clar}}(\mathbf{y}) \in [0, 1]$, corresponding respectively to (i) correctness of the final answer, (ii) accuracy of problem comprehension, (iii) validity of numerical calculations, (iv) conformity of the answer format to task requirements, and (v) provision of a clear and explicit answer. The overall reward is then computed as

$$Q(\mathbf{y}) \;=\; \frac{1}{7}\Big(s_{\text{ans}}(\mathbf{y}) + s_{\text{comp}}(\mathbf{y}) + s_{\text{calc}}(\mathbf{y}) + 2\, s_{\text{form}}(\mathbf{y}) + 2\, s_{\text{clar}}(\mathbf{y})\Big), \tag{11}$$

where the weighting scheme $1:1:1:2:2$ reflects the relative importance of the criteria, placing greater emphasis on answer format fidelity and clarity of presentation.

Prompt for (i) correctness of the final answer

```
prompt_s_ans = f"""
INSTRUCTIONS:
Your task is to determine the correctness of the final answer within the PROPOSED SOLUTION.
Critically verify the final answer against the TASK DESCRIPTION and the reasoning steps provided in
     the PROPOSED SOLUTION.
Do NOT use external knowledge. Focus only on internal consistency and accuracy based on the given
     problem.

Your response must strictly follow the required format:
SCORE: [0.0-1.0]
(0.0 = completely incorrect, 1.0 = perfectly correct)

TASK DESCRIPTION:
[TASK_DESCRIPTION]

PROPOSED SOLUTION:
[PROPOSED_SOLUTION]
"""
```

Prompt for (ii) accuracy of problem comprehension

```
prompt_s_comp = f"""
INSTRUCTIONS:
Your task is to evaluate the PROPOSED SOLUTION's understanding of the TASK DESCRIPTION.
Identify all explicit and implicit constraints, conditions, and specific requests in the TASK
     DESCRIPTION.
Assess how accurately and comprehensively the PROPOSED SOLUTION addressed these elements,
     demonstrating full comprehension.

Your response must strictly follow the required format:
SCORE: [0.0-1.0]
(0.0 = no comprehension, 1.0 = full and accurate comprehension)

TASK DESCRIPTION:
[TASK_DESCRIPTION]

PROPOSED SOLUTION:
[PROPOSED_SOLUTION]
"""
```

Prompt for (iii) validity of numerical calculations

```
prompt_s_calc = f"""
INSTRUCTIONS:
Your task is to verify the validity of all numerical calculations and logical steps within the
     PROPOSED SOLUTION.
For each calculation or logical transition, independently recompute or re-evaluate it.
If any numerical or logical discrepancy is found, it indicates an error.

Your response must strictly follow the required format:
SCORE: [0.0-1.0]
(0.0 = many errors, 1.0 = all calculations and logical steps are valid)

TASK DESCRIPTION:
[TASK_DESCRIPTION]

PROPOSED SOLUTION:
[PROPOSED_SOLUTION]
"""
```

Prompt for (iv) conformity of the answer format to task requirements

```
prompt_s_form = f"""
INSTRUCTIONS:
Your task is to assess if the PROPOSED SOLUTION conforms to the expected output format requirements.
Consider if specific units are used, if the answer is structured as implicitly or explicitly
     requested (e.g., numeric only, step-by-step, \\boxed{} formatting), and if all parts of the
     response are appropriately presented.

Your response must strictly follow the required format:
SCORE: [0.0-1.0]
(0.0 = completely incorrect format, 1.0 = perfectly formatted)

TASK DESCRIPTION:
[TASK_DESCRIPTION]

PROPOSED SOLUTION:
[PROPOSED_SOLUTION]
"""
```

Prompt for (v) provision of a clear and explicit answer

```
prompt_s_clar = f"""
INSTRUCTIONS:
Your task is to evaluate the clarity and explicitness of the PROPOSED SOLUTION.
Assess if the reasoning is easy to follow, unambiguous, and if all necessary steps and explanations
     are provided without missing information.
Consider the overall readability and conciseness.

Your response must strictly follow the required format:
SCORE: [0.0-1.0]
(0.0 = very unclear/implicit, 1.0 = exceptionally clear and explicit)

TASK DESCRIPTION:
```

```
[TASK_DESCRIPTION]

PROPOSED SOLUTION:
[PROPOSED_SOLUTION]
"""
```

C.2 Nighttime Consolidation

When preparing training data for the latent weaver, not all encountered latent representations within a given cycle are equally valuable or should be retained. Recall that the rationale behind self-supervised refinement is as follows: although the latent state $\mathbf{z}_{0,i}$, obtained through weighted momentum transfer, may lie within a promising region of the space, it is not guaranteed to align perfectly with the current context. To address this, we further scale and update it through a self-rewarding mechanism, ensuring that it becomes optimally adapted to support the query at hand. Therefore, the latent weaver, which provides the initial seeds for reasoning, should itself be trained with relatively high-quality data. Concretely, for a triplet $(\mathbf{e}_{\mathbf{c}_i}, \mathbf{z}_{0,i}, \mathbf{z}_i^*)$, we include it into the memory $\mathcal{M}$ if and only if the LLM exhibits sufficient confidence in the associated latent representation. Formally, such confidence is defined as the expected quality score of the final output after iterative refinement. Let $\mathbf{y}_k^{(M)}$ denote the response generated at the last refinement step under rollout k $(k = 1, \ldots, M)$, then the confidence measure is given by

$$\mathbb{E}[Q(\mathbf{y}_k^{(M)})] = \frac{1}{M} \sum_{k=1}^{M} Q(\mathbf{y}_k^{(M)}),$$

and the triplet is retained only if

$$\mathbb{E}[Q(\mathbf{y}_k^{(M)})] \geq \tau,$$

where $Q(\mathbf{y}_k^{(M)}) \in [0, 1]$ denotes the numerical score assigned to the generated response according to task-specific evaluation criteria, and τ is a tunable threshold that governs the admission of latent experiences. We set $\tau = 0.5$ across all experiments.

C.3 Evaluation

The evaluation prompts used by **`LatentEvolve`** for datasets requiring numerical answers (including GSM8K, MATH, AIME 2024/2025, and SciBench) and for multiple-choice datasets (including MMLU, SciBench, JAMA, and GPQA) are summarized in Table 3.

Table 3: Evaluation prompts for **`LatentEvolve`** and other baselines.

Numerical-answer evaluation prompt: {Question Description}. Please reason step by step, and enclose your final answer within \\boxed{}.
Multiple-choice evaluation prompt: {Question Description}. Please select the correct option (A, B, C, or D) to answer the question. Your response should be formatted as follows: The correct answer is {your answer option letter here}.

C.4 Dataset Details

This section provides the fine-grained statistics of each dataset:

- **MMLU** (Hendrycks et al., 2021a): following prior practice (Yue et al., 2025), we sample 1000 instances.
- **MATH** (Hendrycks et al., 2021b): we adopt the standard MATH-500 subset.
- **GSM8K** (Cobbe et al., 2021): we opt for the full test set (1319 problems).
- **GPQA** (Rein et al., 2023): we employ the GPQA-Diamond subset containing 198 graduate-level questions of elevated difficulty.
- **SciBench** (Wang et al., 2024b): we include all 692 tasks.

- **JAMA Clinical Challenge** (Chen et al., 2025a): comprising questions derived from demanding clinical cases, we adopt all 1511 test items.
- **AIME 2024 and 2025** (Li et al., 2024): each consists of 30 problems.

D More Results

Table 4: **Algorithmic Statistics** of `LatentEvolve`. We report the ratio of answer length (when equipped with `LatentEvolve`) to vanilla CoT length, under three settings (full `LatentEvolve`, *w/o Daytime* and *w/o Nighttime*).

Model		Qwen2.5 7b	Qwen3 8b	Qwen3 4b	Llama3.2 3b	Avg
GSM8K	`LatentEvolve`	0.89	0.92	0.91	0.96	0.92
	w/o Daytime	0.92	0.92	0.90	0.97	0.93
	w/o Nighttime	0.98	0.95	0.94	0.94	0.95
MATH-500	`LatentEvolve`	0.91	0.93	0.95	0.97	0.94
	w/o Daytime	0.93	0.93	1.01	0.99	0.97
	w/o Nighttime	1.01	0.97	0.97	0.98	0.98
SciBench	`LatentEvolve`	0.93	0.95	0.94	0.98	0.95
	w/o Daytime	0.94	0.98	0.95	1.04	0.98
	w/o Nighttime	0.99	0.95	0.92	1.02	0.97

D.1 Sensitivity Analysis

From Table 5, we observe a clear improvement in performance with iterative refinement under moderate evolution intervals. For example, when $T = 200$, accuracy steadily increases from 68.3% at Iter 1 to 73.6% at Iter 5, indicating that frequent daytime and nighttime interactions allow the latent representations to be progressively aligned with the query context. A similar trend is seen for $T = 300$ and $T = 500$, although the performance gains diminish as the interval grows. In contrast, when $T = 1000$, the evolution becomes excessively coarse-grained, essentially reducing the process to a single daytime and nighttime interaction. This hinders the model's ability to perform gradual refinement, thereby limiting the benefits of the proposed dual-phase evolution. These results suggest that overly sparse consolidation undermines the advantages of iterative scaling, while moderate intervals strike a balance between stability and adaptability.

Table 5: Results across different iterations with different T on the MMLU dataset (T denotes the evolution interval controlling the frequency of daytime and nighttime interactions). Smaller T values allow more iterative refinements within the same training budget, leading to smoother convergence, while larger T values reduce the number of available iterations, coarsening the evolution process.

	Iter 1	**Iter 2**	**Iter 3**	**Iter 4**	**Iter 5**
200	68.3	70.2	72.8	73.3	73.6
300	70.2	71.9	73.3	73.3	-
500	72.1	73.9	-	-	-
1000	72.8	-	-	-	-

D.2 Efficiency Analysis

To further clarify the computational cost of `LatentEvolve` during nighttime optimization, we report detailed measurements across representative model–dataset configurations. Table 7 and Table 8 summarize the nighttime GPU hours, daytime wall-clock time, number of nighttime training samples, and the corresponding performance across iterations of latent evolution.

Overall, the nighttime computational overhead of `LatentEvolve` remains modest across models and domains, while each iteration consistently yields measurable performance gains in a fully unsupervised setting. These results demonstrate that `LatentEvolve` provides an efficient and practical optimization procedure without imposing substantial additional resource burden.

Table 6: Case Studies of **LatentEvolve**. The questions (from SciBench and GPQA), the answers, and the **LatentEvolve** generations are listed. The peculiar reasoning tokens are highlighted in blue.

QUESTION: Express the van der Waals parameters b=0.0226 dm^3 mol^{-1} in SI base units. **ANSWER:** 0.0000226 **LatentEvlove:** To convert parameter b=0.0226 dm^3 mol^{-1} in SI base units. Start unit-conv dm3-to-m3 step one. Identify 1 dm equal 0.1 m. So, 1 dm^3 is 0.001 m^3. Then apply factor-multiplied 0.0226. [...] The final value in SI base units is 2.26 * 10^{-5} m^3 mol^{-1}.
QUESTION: Which of the following physical theories never requires regularization at high energies? A. Quantum Chromodynamics B. Superstring Theory C. Quantum Electrodynamics D. Classical Electrodynamics **ANSWER:** B **LatentEvlove:** Which physical theories never never required regularization at high energies. Need-no-regularization-scan start. QCD, QED, needs. Classical Electrodynamics, not quantum, so no need. Superstring Theory, this theory no require regularization itself. [...] Based on the above analysis, the theory that never requires regularization at high energies is Superstring Theory. The correct answer is B.

Table 7: Nighttime computational cost and performance across iterations for Llama-3.2-3B on JAMA. Each iteration uses $T = 200$ test samples.

Metrics	Iter 1	Iter 2	Iter 3
Nighttime GPU hours (h)	1.23	1.11	1.33
Daytime wall-clock time (h)	2.5	2.5	2.4
Nighttime training samples	127	133	139
Performance (Acc.)	46.8%	47.2%	47.9%

D.3 Retrieval Similarity Metric

To assess the influence of the retrieval module on overall system performance, we further evaluate several alternative similarity measures beyond the default cosine similarity. Specifically, we compare cosine similarity, dot product, Pearson correlation, and Euclidean distance within the latent retrieval component. As summarized in Table 9, **LatentEvolve** exhibits limited sensitivity to the choice of similarity function: all variants achieve comparable performance, and cosine similarity remains a strong and robust default across both settings.

D.4 Analysis of Threshold τ

To examine the effect of the confidence threshold τ used for selecting nighttime training samples, we perform a sensitivity study across a range of values. Table 10 reports performance under $\tau \in \{0.3, 0.5, 0.7, 0.9\}$ for three representative model–data pairs. We observe that both overly small and overly large thresholds negatively impact performance: a small τ admits many low-confidence or noisy samples into nighttime consolidation, while a large τ prunes too aggressively and thus reduces the amount of useful training signal available to the latent weaver. In practice, moderate values ($\tau = 0.5$ or 0.7) yield consistently stable behavior, and our default choice of $\tau = 0.5$ provides a robust and broadly effective setting across all evaluated configurations.

D.5 Visualization of Latent-Reasoning Evolution

Recall that in LatentEvolve, for each task query we first obtain the base latent vector $\mathbf{z}_{\text{base}}$. The nighttime weaver then produces the refined latent $\mathbf{z}'_{\text{base}}$, which is transformed into $\mathbf{z}_0$ after momentum transfer in Equation (5), and after daytime optimization we obtain the final latent $\mathbf{z}^*$, which we refer to as the *success latent* when the resulting trajectory is correct. We apply t-SNE to visualize the distributions of these three latent clusters on SciBench.

As shown in Figure 5, the base latents remain far from the success-latent cluster. Across subsequent iterations, the refined latents produced by the nighttime weaver progressively move toward the region associated with successful reasoning. This indicates that repeated nighttime consolidation enables the weaver to supply latent initializations that lie closer to successful inference regions. The momentum transfer process also steers the latent representations closer to the success region, which validates the effectiveness of momentum-based experience transfer.

Table 8: Nighttime computational cost and performance across iterations for Qwen2.5-7B on MATH. Each iteration uses $T = 250$ test samples.

Metrics	Iter 1	Iter 2
Nighttime GPU hours (h)	1.3h	1.3h
Daytime wall-clock time (h)	2.07	2.12
Nighttime training samples	188	213
Performance (Acc.)	74.2%	77.8%

Table 9: Ablation on retrieval similarity metrics.

Metric	MATH + `Qwen2.5-7B`	JAMA + `Llama-3.2-3B`
Cosine similarity	77.8	48.4
Dot product	77.4	47.5
Pearson correlation	76.7	48.0
Euclidean distance	75.9	47.1

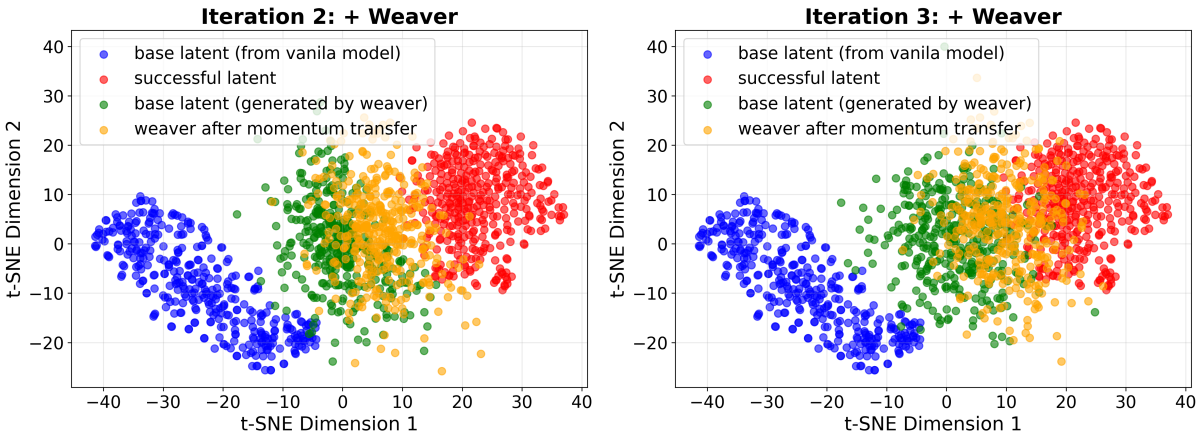

Figure 5: The visualizations of (1) base latent $\mathbf{z}_{\text{base}}$, (2) the weaver-refined latent embeddings $\mathbf{z}'_{\text{base}}$, (3) the latent embeddings after momentum trasfer $\mathbf{z}_0$, as well as (4) the final latent vectors $\mathbf{z}^*$ (we denote those that lead to the final successful reasoning as *success latent*).

D.6 Quantitative Comparison of Latent-Cluster Distances

To complement the qualitative visualization above, we compute the Euclidean distances between the centroids of the latent clusters at different iterations. The results are summarized below.

Across both settings, the refined latents consistently move closer to the success-latent region as daytime and nighttime phases alternate, while the base latents remain substantially farther away. These quantitative findings support the conclusion that nighttime consolidation provides a significantly improved initialization point for downstream reasoning.

Table 10: Sensitivity of LatentEvolve to the confidence threshold τ.

τ	0.3	0.5	0.7	0.9
`Qwen2.5-7B` + MATH	76.0	77.6	76.5	74.2
`Llama-3.2-3B` + JAMA	46.2	48.4	48.8	47.1
`Qwen3-8B` + GPQA	32.6	32.8	31.6	29.2

Table 11: Euclidean distances between latent-cluster centroids on `Llama-3.2-3B` + JAMA. Iterations correspond to successive daytime–nighttime cycles.

`Llama-3.2-3B` + JAMA	**Iter 1**	**Iter 2**	**Iter 3**
base latent $\leftrightarrow$ success latent	2317.401	2466.180	2289.177
refined latent $\leftrightarrow$ success latent	–	1094.228	834.102

Table 12: Euclidean distances between latent-cluster centroids on `Qwen2.5-7B` + MATH.

`Qwen2.5-7B` + MATH	**Iter 1**	**Iter 2**
base latent $\leftrightarrow$ success latent	1343.296	1308.065
refined latent $\leftrightarrow$ success latent	–	955.109

Table 13: **Extended comparison** on `Gemma-3-12b-it` with additional baselines.

Method	**MMLU**	**GSM8K**	**MATH**	**SciBench**	**GPQA**	**JAMA**	**AIME24**	**AIME25**
Vanilla	75.50	89.23	84.80	10.84	40.48	49.50	23.52	18.27
Self-Consistency	75.70	87.56	85.59	12.07	41.92	50.73	25.83	19.44
LatentSeek	72.80	92.41	85.20	13.52	39.90	51.18	28.67	25.39
`LatentEvolve`	**77.60**	**92.79**	**89.32**	**18.93**	**47.20**	**55.06**	**32.75**	**26.23**

E Pseudocode of LatentEvolve

Algorithm 1: `LatentEvolve`: Self-Evolving Test-Time Scaling

Input: Query stream $\{\mathbf{c}_1, \mathbf{c}_2, \dots\}$, frozen LLM π_θ, period T, max iters K, lr η, threshold τ, warm-up N_w

Output: Output $\mathbf{y}_i$ for each query $\mathbf{c}_i$

$\mathcal{M} \leftarrow \emptyset$, $\mathbf{W}_\psi \leftarrow$ untrained;

for *each incoming query* $\mathbf{c}_i$ **do**

  `/* Stage 1: Base Latent Generation (always active) */`

  Compute semantic embedding $\mathbf{e}_{\mathbf{c}_i}$ via π_θ's final hidden state. Obtain base latent $\mathbf{z}_{\text{base},i} \leftarrow H_\theta(\mathbf{c}_i)_{1:L'}$ by truncating greedy-decoded hidden states (Eq. 4);

  `/* Stage 2: Weaver Refinement (nighttime component; skipped in "w/o Nighttime" and when` $\mathbf{W}_\psi$ `is untrained) */`

  if $\mathbf{W}_\psi$ *has been trained* **then**

    $\mathbf{z}'_{\text{base},i} \leftarrow \mathbf{W}_\psi(\mathbf{e}_{\mathbf{c}_i}, \mathbf{z}_{\text{base},i})$ (Eq. 8);

  else

    $\mathbf{z}'_{\text{base},i} \leftarrow \mathbf{z}_{\text{base},i}$;

  end

  `/* Stage 3: Associative Retrieval & Momentum Transfer (daytime component; skipped in "w/o Daytime") */`

  if $|\mathcal{M}| > N_w$ **then**

    Retrieve $\mathcal{N}_k(\mathbf{c}_i) \leftarrow$ Top-k entries from $\mathcal{M}$ ranked by $S(\mathbf{e}_{\mathbf{c}_i}, \mathbf{e}_{\mathbf{c}_j})$ (Eq. 3);

    Compute momentum-transferred init: $\mathbf{z}_{0,i} \leftarrow \mathbf{z}'_{\text{base},i} + \sum_{j \in \mathcal{N}_k} \alpha_j (\mathbf{z}^*_j - \mathbf{z}_{\text{base},j})$, where $\alpha_j \propto \exp(S(\mathbf{e}_{\mathbf{c}_i}, \mathbf{e}_{\mathbf{c}_j}))$ (Eq. 5);

  else

    $\mathbf{z}_{0,i} \leftarrow \mathbf{z}'_{\text{base},i}$;

  end

  `/* Stage 4: Self-Supervised Refinement (always active; produces` $\mathbf{z}^*_i$ `and` $Q(\mathbf{y})$ `needed for archiving) */`

  for $k = 0$ **to** $K - 1$ **do**

    Greedy decode $\mathbf{y}_k \sim p(\cdot \mid \mathbf{c}_i, \mathbf{z}_{k,i}; \theta)$, then compute self-reward $Q(\mathbf{y}_k)$ via π_θ;

    Update $\mathbf{z}_{k+1,i} \leftarrow \mathbf{z}_{k,i} + \eta \cdot Q(\mathbf{y}_k) \nabla_{\mathbf{z}_{k,i}} \log p(\mathbf{y}_k \mid \mathbf{c}_i, \mathbf{z}_{k,i}; \theta)$ (Eq. 6);

    if $\mathbb{E}[Q]$ *has not increased for 3 consecutive rounds* **then**

      break;

    end

  end

  Set $\mathbf{z}^*_i \leftarrow \mathbf{z}_{k,i}$ and produce final output $\mathbf{y}_i$ from $p(\cdot \mid \mathbf{c}_i, \mathbf{z}^*_i; \theta)$;

  `/* Stage 5: Archiving (always active; gates buffer quality via` τ`) */`

  if $\mathbb{E}[Q(\mathbf{y})] \geq \tau$ **then**

    Store $(\mathbf{e}_{\mathbf{c}_i}, \mathbf{z}_{\text{base},i}, \mathbf{z}^*_i)$ into episodic buffer $\mathcal{M}$;

  end

  `/* Stage 6: Nighttime Consolidation (nighttime component; skipped in "w/o Nighttime"; triggered every` T `queries) */`

  if $i \bmod T = 0$ **then**

    Train weaver: $\mathbf{W}_\psi \leftarrow \arg\min_\psi \mathbb{E}_{(\mathbf{e}_{\mathbf{c}_j}, \mathbf{z}_{\text{base},j}, \mathbf{z}^*_j) \sim \mathcal{M}} \|\mathbf{W}_\psi(\mathbf{e}_{\mathbf{c}_j}, \mathbf{z}_{\text{base},j}) - \mathbf{z}^*_j\|^2$ (Eq. 7);

    Refresh $\mathcal{M} \leftarrow \emptyset$;

  end

end