CONTINUAL SELF-IMPROVEMENT WITH LIGHTWEIGHT EXPERIENTIAL LATENT MEMORIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models achieve strong reasoning performance by scaling inference-time compute, yet remain fundamentally stateless, discarding the rich, self-produced reasoning traces generated during this process. We investigate whether models can instead learn online from this experience, converting transient computation (reasoning traces) into persistent reusable knowledge, and without external supervision or access to future data. We show that In-Context Learning (ICL) over raw reasoning traces fails to generalize, reflecting a fundamental limitation of token-level reuse: individual traces lack the abstraction needed for transfer, even after refinement (e.g. self-reflection). In contrast, drawing inspiration from recent works on unsupervised reinforcement learning, we find that lightweight per-instance training with self-generated test-time signals as rewards could yield substantial gains, often surpassing full-dataset offline training, motivating a shift from raw traces to learned latent representations. Building on this insight, we propose an online method that distills inference-time compute spent on encountered problems into compact modular latent memories capturing the underlying reasoning structure. These memories are stored and retrieved for future inputs, enabling continual improvement while avoiding catastrophic forgetting through modular design. Importantly, our method is lightweight, parametrized as extremely small and lightly trained soft prompt memories ($\sim 0.001\%$ of model parameters), yet achieving performance competitive with full parametric updates and offline training. Across challenging reasoning benchmarks, our approach significantly outperforms textual ICL methods, enabling reliable self improvement from experience.

1 INTRODUCTION

Large language models (LLMs) have demonstrated remarkable reasoning capabilities (e.g., Jaech et al. (2024); Guo et al. (2025); Shao et al. (2024)), increasingly through extended inference-time compute and thinking Wei et al. (2022); Snell et al. (2025); Brown et al. (2024), yet they remain stateless after deployment, spending significant compute on each problem without ever accumulating experience or converting the compute spent into reusable knowledge for the future. As models are increasingly deployed in interactive, problem-solving settings, the reasoning traces they produce represent a rich and largely untapped source of self-generated data that the model can exploit to learn continuously. A natural question arises, how an LLM can self-improve online and learn from its experience by converting its own thinking process into reusable knowledge (we refer to hereafter as experiential memories). Recent work has pursued this question primarily through training-free methods based on in-context learning (ICL) Suzgun et al. (2025); Zhang et al. (2026a); Ouyang et al. (2025), which store and retrieve past reasoning traces to inform new predictions. While appealing in their simplicity, these works force reasoning traces storage through a textual bottleneck, by storing only what a model *said* rather than what it *computed*, discarding rich information about intermediate states, explored dead ends, confidence, and structural understanding. Recent work on latent reasoning Hao et al. (2024) demonstrates that models can "think" in continuous latent space more efficiently than in discrete token space, probing studies Belinkov et al. (2017); Geva et al. (2021) show that crucial information exists in hidden states that never surface to the text output. Chain-of-thought faithfulness work Lanham et al. (2023) reveals that CoT explanations often do not reflect actual model computations. We further show that current training-free approaches for experiential memories yield negligible improvements on challenging reasoning benchmarks. This failure reflects

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

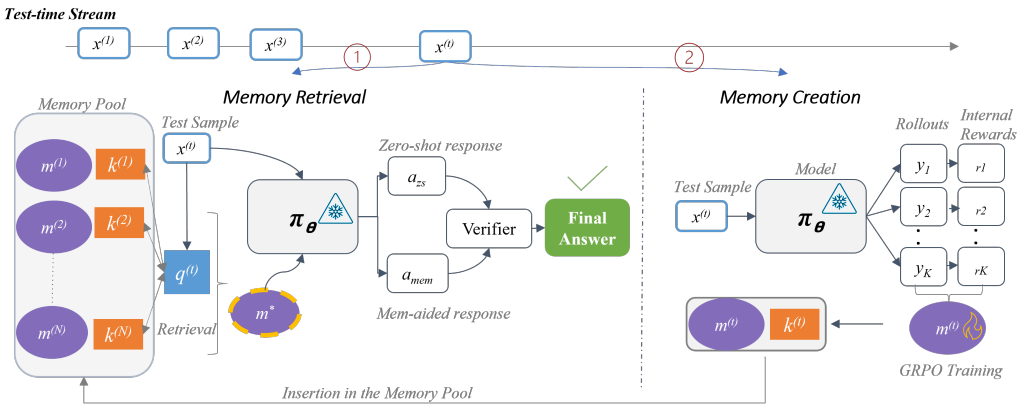


Figure 1: **Experiential Latent Memory** for continual self-improvement from a test-time stream. Memory retrieval: given a test sample, we query the most relevant memory. For reliability, a verifier routes between zero-shot and memory-augmented responses, selecting the higher-scoring output. Memory creation: whenever possible, we use additional test-time compute to train a lightweight modular memory for a given test sample with GRPO and internally derived rewards. Newly created memories are added to the memory pool for later reuse.

a fundamental limitation of storing reasoning traces or their textual abstraction: retrieving a past reasoning trace provides a sample of prior experience but not a meaningful learning signal, yielding shallow generalization. We thus posit, and empirically validate, that meaningful memories should instead be learned from the model's explored trajectories, successes and failures when trying to solve a given problem with test-time compute, thus internalizing the findings of the reasoning process rather than storing its surface-level output.

Towards answering the question on how to learn with self-generated signals, recent work on Reinforcement Learning (RL) uses internal reward signals to drive improvement without ground-truth labels, demonstrating meaningful gains from purely internal supervision Zuo et al. (2025); Zhao et al. (2025b); Jayalath et al. (2025); Zhao et al. (2025a). However, these approaches operate in an **offline** regime: they require access to the full test dataset upfront, training on it before gains on this set can be shown. Beyond this practical limitation and even more importantly, such methods have been recently shown to degrade as test-time data accumulate, essentially showing that they do not scale Zhang et al. (2025b); He et al. (2026) possibly due to *interfering self-supervisory signals injected into the main model*. A final drawback is that these approaches rely on full-model updates, potentially leading to forgetting, optimization instability, and substantial compute and memory overhead. The truly **online streaming setting**, where the model encounters each problem once and must efficiently improve continuously as experience accumulates and without prior data access, remains largely unexplored and unsolved. To the best of our knowledge, this is the first work that addresses the question of *how can an LLM internalize its experiences and reasoning explorations without external supervision in a streaming fashion, enabling steady self-improvement with no interference or forgetting of its existing capabilities?*

We begin with a key empirical observation: training on individual examples for a few steps using RL with internal self-generated rewards can substantially boost model's performance (generalization), even outperforming full-dataset offline training. In other words, when leveraging internal model feedback, learning from reasoning traces of individual samples can elicit generalization that matches or surpasses complete dataset offline training. Importantly, we show that introducing modular and efficient parameterization that leaves the main model intact can further improve the performance, while significantly reducing compute and memory requirements. To this end, we introduce an efficient online continual self-improvement method in which the model, after solving each problem, constructs a lightweight latent memory, parametrized as a soft prompt, distilling the test-time compute spent on this experience into a compact latent representation stored and retrieved for relevant future inputs.

Our approach addresses limitations of both lines of work. Unlike textual ICL, latent memories operate in the model's representational space, capturing abstract reasoning rather than surface patterns, leading to stronger generalization. Unlike offline RL methods, ***ELM* is designed for streaming settings** and

scales with accumulated experience, mitigating forgetting via modularity: memories are isolated and composed at retrieval time without interference. Our method is also highly lightweight, using very small memories ($\sim$0.001% of parameters) trained with few gradient steps, yet achieving performance competitive with full fine-tuning. Fig. 1 illustrates *ELM* (Experiential Latent Memory). Empirically, *ELM* outperforms zero-shot and textual ICL baselines on challenging reasoning benchmarks, shows strong cross-dataset transfer, and provides a scalable path to continual self-improvement by converting test-time compute into reusable knowledge. Our contributions are as follows: 1) We introduce latent experiential memories for efficient, modular instance-level learning. 2) We integrate them into an online protocol with per-sample storage and retrieval. 3) We demonstrate strong gains on challenging benchmarks with minimal overhead.

2 Related Work

RL for LLM Reasoning. Reinforcement learning with verifiable reward (RLVR) improves LLM reasoning via rule-based rewards Gao et al. (2024) or answer matching against ground truth Zeng et al. (2025). While typically applied at scale, recent work shows that strong reasoning can emerge from few examples, supported by a theoretical lower bound on the number of required training instances Zhang et al. (2026b) and empirical results with small or even single training instances Fatemi et al. (2025); Wang et al. (2025). However, these works do not provide mechanisms for selecting or exploiting such examples.

To reduce reliance on ground-truth labels, prior work explores internally derived rewards Xu et al. (2025); Zhao et al. (2025a); Jayalath et al. (2025); Zuo et al. (2025); Zhao et al. (2025b). Closest to ours, TTRL Zuo et al. (2025) generates rewards via majority voting, however, test-time data are collected offline and used to train the full model with GRPO Shao et al. (2024). In contrast, our approach demonstrates that efficient online training on test samples can yield performance gains that rival offline test-time training variants.

Memory of Past Experience for Improved Reasoning. Prior work stores test-time experience, typically as text (Suzgun et al., 2025; Zhang et al., 2026a; Ouyang et al., 2025), often requiring ground-truth supervision and large models. This textual bottleneck discards rich internal signals (e.g., intermediate states, uncertainty). LAG Chen et al. (2025) instead reuses KV activations, but remains limited without learning. Learned latent memory methods (e.g., MemGen Zhang et al. (2025a), FlashMem Hou et al. (2026)) rely on offline-trained modules and remain largely static at test-time. In contrast, we learn latent memories directly from individual experiences using internal feedback, enabling continual online adaptation and improved generalization.

Online Continual Learning. Our work is related to online continual learning (Aljundi et al., 2019), where models learn from a stream of data while accumulating knowledge over time. Classical approaches update a single model under supervised streams, where the main challenge is catastrophic forgetting of previously learned knowledge (French, 1999). A complementary line considers self-supervised continual learning, where supervisory signals are internally generated (Liang et al., 2025). Unlike these approaches, which still update a single model, we address forgetting by introducing experiential latent memories: experience is distilled into modular, individually parameterized units that are retrieved at test time. This design mitigates interference and enables compact, scalable knowledge accumulation, aligning with recent views on modular long-term memory (Dorovatas et al., 2026).

3 *ELM*: Continual Self-Improvement with *E*xperiential *L*atent *M*emories

Our goal is to enable a model of continuous self-improvement by distilling test-time compute into reusable experiential memories. These memories can be accumulated and retrieved to improve performance on relevant future test samples with less compute. To approach this goal, one has to answer two questions: (1) how to learn *efficiently and effectively* from a given sample using internal model feedback, and (2) how to leverage previously constructed memories to elevate performance on upcoming test samples. In the following, we describe our approach and key design choices.

3.1 How to Design Efficient and Effective Latent Memories?

An online self-improvement method must extract experience from each encountered problem that remains useful for future ones. Recent work on data selection for RL (Wang et al., 2025) shows that training on certain individual samples can yield substantial gains on a broader test set, suggesting that single test-time samples can likewise benefit future queries if exploited correctly. Since external feedback is typically unavailable at test time, we derive the training signal from the model itself, following TTRL (Zuo et al., 2025; Zhao et al., 2025b): given a test sample x, we draw N responses from the policy π_θ, take their majority-vote answer y^* as a pseudo-label, and assign each response the binary reward

$$R(\hat{y}_i, y^*) = \mathbb{1}[\hat{y}_i = y^*]. \tag{1}$$

These rewards are optimized with GRPO (Shao et al., 2024).

Applying this objective to the full model on every incoming sample, however, is impractical for two reasons: (1) per-sample full-model updates are computationally prohibitive, and (2) continual updates on a stream risk catastrophic forgetting (French, 1999) of both the model's original capabilities and previously acquired experience. We therefore do not inject experience into the model itself. Instead, we capture the experience from each test sample in an independent, modular *experiential memory* that is stored for later reuse. Isolating each sample's experience keeps the base model stable and minimizes interference between memories; to be reusable, memories must also be retrievable, i.e., efficiently recalled for future samples that can benefit from them. To this end, we parameterize each memory as a soft prompt (Lester et al., 2021).

Algorithm 1 Continual Self-Improvement with Experiential Latent Memories (*ELM*)

Input: Stream $\{x_i\}$, frozen policy π_θ
Initialize: memory pool $\mathcal{M} \leftarrow \emptyset$
Hyperparameters: #rollouts K, train steps S
for each incoming sample x_i **do**
  # Memory Retrieval
  if $|\mathcal{M}| > 0$ **then**
    $q_i = \mathbf{Embed}(x_i)$
    $m^* \leftarrow \mathbf{Retrieve}(\mathcal{M}, q_i)$
    # Response Selection
    $a_{\text{zs}} \sim \pi_\theta(\cdot \mid x_i)$
    $a_{\text{mem}} \sim \pi_\theta(\cdot \mid [m^*, x_i])$
    if $a_{\text{zs}} = a_{\text{mem}}$ **then**
      $\hat{y}_i \leftarrow a_{\text{zs}}$
    else
      $\hat{y}_i \leftarrow \text{Verifier}(a_{\text{zs}}, a_{\text{mem}})$
    end if
  else
    $\hat{y}_i \sim \pi_\theta(\cdot \mid x_i)$
  end if
  return $\hat{y}_i$ *# Response Returned*

  # Memory Creation
  $m^{(i)} \leftarrow \mathbf{Init}(x_i)$
  $m^{(i)} \leftarrow \mathbf{GRPO}(m_{x_i}, K, S)$
  $\mathcal{M} \leftarrow \mathcal{M} \cup \{m^{(i)}\}$
end for

Soft Prompts as Latent Experiential Memories. A soft prompt (Lester et al., 2021) is a sequence of k trainable vectors $\{p_j\}_{j=1}^k \subset \mathbb{R}^d$ prepended to the input embeddings, so that the model processes $[p_1, \ldots, p_k, x]$ while all other parameters stay frozen. We use it to define a memory $m_x \in \mathbb{R}^{k \times d}$ capturing the experience from sample x. This choice offers two properties: (1) *efficiency*, as each memory amounts to only $\sim$0.001% of the model parameters and the base model is never modified; and (2) *retrieval utility*, as the memory is initialized from the sample's text embedding and refined for only a few steps in the model's latent space, so it enables meaningful similarity comparisons with embeddings extracted from future queries.

Memory Creation. For each encountered input, we train its memory for a few GRPO steps with majority-voting rewards (Eq. 1) as internal supervision. This requires no ground-truth labels, and keeping the base model frozen provides a stable policy reference that limits the capture of spurious noise. In Sec. 4.1, we show that (1) individual memories can generalize strongly, rivaling offline full-dataset training; (2) their utility varies across samples; and (3) noisy memories that degrade performance exist, so handling them is essential for reliable self-improvement. We hypothesize that full-dataset training mixes noisy and beneficial signals, leading to averaged gains and to the instability or collapse reported for TTRL (Zhang et al., 2025b; He et al., 2026); isolating experience in individual memories instead avoids cross-sample interference and enables selective retrieval. Finally, our work can be seen as a first step towards approaching the framework proposed in the recent continual learning position paper (Dorovatas et al., 2026), where modular memory is viewed as an essential key to scalable continual learning agents.

3.2 How to Leverage Latent Memories for Elevating Test-Time Performance

Having described how to efficiently construct a memory that captures the experience gained from solving a given inquiry, we now turn to the question of how to reliably and effectively leverage previously accumulated memories at test time. We assume a pool of learned memories $\mathcal{M} = \{m^{(i)}\}_{i=1}^{N}$ constructed from previously encountered test samples $\{x^{(i)}\}$.

Storing & Retrieving Memories. Each memory $m^{(i)}$ is stored with a key $k^{(i)}$, and given a new input we retrieve the single most relevant memory by cosine similarity between a query vector computed from the input and the stored keys. Owing to the latent nature of our memories, keys can be built in two ways, giving two retrieval variants: **input retrieval**, where the key mean-pools the soft prompt's learnable token embeddings and the query mean-pools the input embeddings; and **KV retrieval**, where the soft prompt and the input are passed through the model's attention layers, their key vectors are pooled per layer, and similarities are averaged across layers. Details are given in A.2.

Response Routing. As discussed, retrieval always returns a nearest neighbor, even when no relevant memory exists in the pool, and since memories are learned from self-supervised signals, even relevant ones may be noisy. As our offline analysis shows, memories improve performance on average but can turn a correct zero-shot answer into an incorrect one on specific inputs (Sec. 4.1). To guard against such regressions, we generate both a zero-shot response a_{zs} and a memory-conditioned response a_{mem}: if they agree, the shared answer is returned; if not, a response router selects one. Our modular design makes this mitigation lightweight, as the zero-shot path requires no memory at all.

Decoupling Memory Creation from Serving. *ELM* separates two compute regimes (Fig. 1). At *serving* time, the model answers each query using previously acquired memories, via greedy decoding, retrieval, and response routing. *Memory creation* takes place once per encountered sample, after the response is returned, and is where the additional test-time compute (multiple rollouts, majority voting, and a few GRPO steps) is spent. Unlike methods that apply self-consistency or search to every query, *ELM* thus amortizes this cost into a reusable memory. Our results suggest that this investment is crucial, particularly for base models, where training-free memories built without such compute provide limited gains (Sec. 4.2).
Algorithm 1 outlines the main steps of our full method, termed *ELM* (Experiential Latent Memory).

4 Experiments

This section validates the following questions: (1) Can training with internal rewards on individual samples yield generalization performance? (2) Is our memory parameterization effective? (3) Can ELM achieve significant gains over training-free baselines?

Experimental Setup. We present the experimental setup across both offline and online settings.
Benchmarks. We evaluate on four challenging reasoning benchmarks. For math, **MATH500** (Lightman et al., 2023) is a 500-problem subset of the MATH dataset (Hendrycks et al.), spanning competition-level problems across algebra, geometry, number theory, and related topics. **AMC23** (Li et al., 2024) and **AIME24** (Zhang & Math-AI, 2024) comprise problems from the 2023 AMC and 2024 AIME competitions respectively, representing challenging olympiad-level reasoning problems. Beyond math, we evaluate on **GPQA-Diamond** (Rein et al., 2024), a set of 198 graduate-level multiple-choice questions in biology, physics, and chemistry, which tests whether memories transfer to scientific reasoning outside math.
Models. We evaluate on three models. **Llama-3.1-8B-Instruct** (Grattafiori et al., 2024) is a general-purpose instruction-tuned model with no domain-specific mathematical training, while **Qwen2.5-Math-7B** (Yang et al., 2024) is a math-specialized base model of comparable size. To study the effect of scale, we additionally include **Qwen2.5-Math-1.5B** (Yang et al., 2024), a smaller model from the same family. Together, these models let us compare instruction-tuned vs. base models at similar scale (Llama-8B vs. Qwen-7B), and small vs. large models within the same family (Qwen-1.5B vs. Qwen-7B).
Implementation Details. For individual sample training, we apply 10 gradient steps per example. We use 8 rollouts per step and a maximum sequence length of 2048, without up-scaling the number

	Llama-3.1-8B-Instr.				Qwen2.5-Math-7B			
	#tr_params	MATH	AMC	AIME	#tr_params	MATH	AMC	AIME
Zero-shot CoT	–	45.0	20.0	3.3	–	67.0	50.0	10.0
GRPO (GT reward)	8B	**47.0**	27.5	6.7	7B	68.0	62.5	**20.0**
TTRL (Maj reward)	8B	45.6	27.5	**10.0**	7B	67.2	62.5	**20.0**
ELM offline	**0.08M**	45.0	**32.5**	**10.0**	**0.07M**	**73.4**	**65.0**	**20.0**
⌞ oracle ZS vs Mem	–	55.4	35.0	10.0	–	77.6	67.5	22.5
ELM-Full$_{\text{offline}}$	8B	46.0	27.5	6.7	7B	67.8	60.0	13.3
ELM-LoRA$_{\text{offline}}$	42M	46.2	30.0	6.7	40M	68.6	62.5	10.0

Table 1: Accuracy (greedy pass@1) across benchmarks when trained with GRPO using a majority voting (Maj) reward (TTRL formulation). Green rows correspond to full-dataset training, while Blue rows correspond to per-sample (modular) training. We also show results with ground truth (referred to as GRPO) rewards for the full model full-dataset training case.

of samples for majority voting reward estimation (unlike (Zuo et al., 2025)) [1]. In terms of **evaluation metrics**, we report greedy *pass@1*. Each soft prompt memory consists of 20 trainable tokens initialized from the embeddings of the input question. Refer to Appendix A for full implementation details.

4.1 Offline Experiments

To assess the generalization potential of training on individual samples with internal rewards, we introduce the following offline experimental protocol: for each sample in the evaluation set, we train a latent memory and then evaluate it on the full set. AMC23 and AIME24 are small enough to allow per-sample training across all examples; for these datasets, we run each experiment with three seeds to mitigate variance. MATH500 is expensive to evaluate exhaustively; instead, we sample multiple subsets using different seeds, train per-sample memories on each subset, and select the best-performing memories for evaluation on the full set. This setup corresponds to an offline variant of our method without retrieval or verification, which we denote as *ELM* offline.

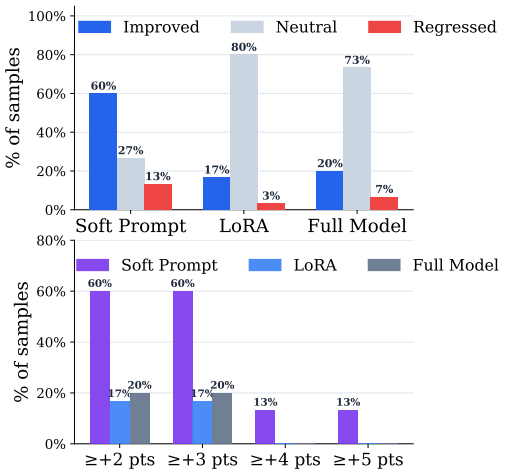


Figure 2: Memory design analysis on AIME24 with Llama. *Top*: % of training samples improving over zero-shot. *Bottom*: % exceeding score improvement thresholds.

As a comparison point, we also perform full-dataset offline training using either ground-truth rewards (GRPO-GT) or majority voting (TTRL) on the full model. Training is performed for 10, 30, and 80 epochs on MATH500, AMC23, and AIME24, respectively, following the setup of (Zuo et al., 2025). To ablate the choice of memory parameterization, we consider a LoRA-based parameterization (Hu et al., 2022) (*ELM*-LoRA$_{\text{offline}}$) [2], as well as full-model per-sample training (*ELM*-Full$_{\text{offline}}$). For the per-sample variants, we report the average performance across seeds of the best-performing memory. We stress that *ELM* offline is an *analysis*, not a deployable method: its goal is to measure how much a single, self-supervised memory can generalize to the rest of the dataset, i.e., the potential gain available from online experience. Accordingly, we report the best memory per setting (the one with the largest improvement over zero-shot across the evaluation set), which constitutes an upper bound, and complement it with analysis on the full distribution of memories to characterize how consistently memories improve or regress.

[1] We note that this serves as a constrained instantiation of the original TTRL setup which uses 64 samples for extracting the label and 32 for training. However, follow-up work (He et al., 2026) showed that 8 rollouts yields stable training.

[2] For LoRA-memories, we use rank $r = 16$ applied to all attention and MLP projections.

Results are reported in Tab. 1. As observed, training on individual samples can be highly effective across models and datasets, often yielding performance that exceeds full-dataset training, even with GRPO-GT. Furthermore, **among memory parameterizations, soft-prompt memories yield the most consistent generalization gains despite representing only $\sim 0.001\%$ of model parameters**. In contrast, full-model parameterization generalizes poorly, likely due to a mismatch between capacity and the limited training regime (single sample, 10 steps). Fig. 2 and 4 analyze these behaviors for AMC23 and AIME24: the top row shows the fraction of training samples that improve over zero-shot on the full evaluation set, while the bottom row reports the fraction exceeding given improvement thresholds. Overall, soft-prompt memories achieve the most consistent gains and are the only representation surpassing +3 points on the challenging AIME24 benchmark.

Memory Specialization. Tab. 1 reports, for *ELM* $_{\text{offline}}$, a variant of oracle selection between the zero-shot response and the memory-aided response for each test sample during eval, showing that it further improves overall performance. This serves as evidence motivating our retrieval and response routing components: although individual memories improve performance on average, they can degrade specific samples by turning a correct zero-shot prediction into an incorrect one.

Model	Flip 1→0	Overlap IoU
Qwen	18.5%	0.55
LLaMA	21.6%	0.37

Table 2: Negative flip rate (1→0) for top-5 memories and IoU between top-2 memories on AMC23.

To investigate this phenomenon, we show in Tab. 2 that the percentage of negative flips[3] for the top-5 memories on AMC23 is around 20% across models. In contrast, the Intersection over Union (IoU) between samples improved by the top-2 memories is moderate for both models. This indicates that each memory tends to improve a different subset of test samples, with only partial overlap.

This suggests that memories exhibit *sample-dependent specialization*, rather than a single memory being uniformly optimal. Consequently, effective retrieval is necessary to match test queries to appropriate memories, and a robust routing mechanism is required to reliably select between zero-shot and memory-augmented predictions in the online setting.

4.2 Online Experiments

Now we turn our focus to the primary goal of this work: a strictly online streaming scenario where the model encounters inputs one at a time, in order, with no prior access to the evaluation data. This distinguishes our setting from two commonly conflated paradigms: a) *test-time adaptation (Wang et al., 2020)*, which trains a model on a sample before the evaluation on the sample itself, and b) *unsupervised RL* methods (e.g., TTRL (Zuo et al., 2025)), which require the full test set upfront to generate training signal before evaluating on the same set.

Baselines. We compare against four experiential memory approaches: **(1) ICL** stores raw question–solution pairs (q, CoT) and retrieves the most similar example at test time to prompt the model in context. **(2) Reasoning Bank** Ouyang et al. (2025) solves each problem, performs a correctness-checking pass, and then a refinement pass to extract reusable reasoning strategies, which are stored in memory and retrieved to augment future queries. **(3) Dynamic Cheatsheet (DC)** Suzgun et al. (2025) maintains a single, evolving cheatsheet (prompt) of problem-solving insights that is updated after each problem and prepended to subsequent queries. **(4) LAG** Chen et al. (2025) uses latent representations extracted from the key-value (KV) activations of prior computations, retrieves them based on their textual representation, and prepends them to the query. For reference, we also report offline full-dataset training methods, namely GRPO and TTRL.[4].

As shown in Tab. 3, *ELM* is the strongest online method overall and matches or surpasses full offline training on most benchmarks [5]. On MATH500, *ELM* builds its memories entirely online, without prior dataset access, yet outperforms both GRPO and TTRL on both models, a setting where the long stream favors memory accumulation and where unsupervised RL (TTRL) starts to degrade (He et al., 2026). On AMC23 and AIME24, *ELM* improves over zero-shot by $+4.4$ to $+14.2$ points across all three models (e.g., $+14.2$ for Qwen2.5-Math-7B on AMC23, surpassing offline training).

[3]This rate is defined as $(\#\text{ negative flips})/(\#\text{ negative flips} + \#\text{ positive flips})$

[4]To ensure a fair comparison, in Appendix E we report results for a zero-shot baseline using two sampled responses with verifier-based selection, matching the additional compute introduced by our routing mechanism.

[5]In Appendix E.7, we report additionally the standard deviation of all online results.

	Llama-3.1-8B-Instr.				Qwen2.5-Math-7B				Qwen2.5-Math-1.5B			
	MATH	AMC	AIME	GPQA	MATH	AMC	AIME	GPQA	MATH	AMC	AIME	GPQA
Zero-shot CoT	45.0	20.0	3.3	21.7	67.0	50.0	10.0	24.7	63.2	45.0	10.0	18.2
Online												
ICL	41.2	20.0	2.2	19.5	24.0	25.0	13.3	16.5	25.8	12.5	2.2	8.9
Reasoning Bank	41.2	22.5	2.2	16.3	29.6	25.8	12.2	14.5	20.3	39.2	8.9	6.7
LAG	44.0	22.5	4.4	**22.2**	34.2	19.2	3.3	12.6	26.8	18.3	1.1	12.6
DC	39.1	19.2	2.2	13.6	32.1	42.5	16.7	6.6	37.4	29.2	4.4	5.6
ELM (input)	**50.9**	**29.2**	10.0	19.7	**72.0**	59.2	**18.9**	27.3	**68.0**	**50.0**	**14.4**	**24.7**
ELM (kv)	**50.9**	28.3	**12.2**	19.5	71.3	**64.2**	13.3	**27.6**	67.4	**50.0**	13.3	22.2
Offline												
GRPO	47.0	27.5	6.7	–	68.0	62.5	20.0	–	–	–	–	–
TTRL	45.6	27.5	10.0	–	67.2	62.5	20.0	–	–	–	–	–

Table 3: **Main Online Results.** Performance comparison (greedy pass@1) of *ELM* against training-free ICL baselines and full-offline training across datasets. Offline results are included for reference and are taken from the offline experiments in Table 1; they are therefore reported only for the two larger models on the math benchmarks.

The comparison is most striking for the base models, which remain the preferred foundation for many reasoning systems and RL-based post-training pipelines due to their adaptability and headroom for improvement, and thus an effective online self-improvement method should therefore not be limited to instruction-tuned models. Yet on Qwen2.5-Math-7B and Qwen2.5-Math-1.5B, all text-based memory methods (ICL, Reasoning Bank, DC) and LAG fall *below* zero-shot performance in nearly every setting (the only exceptions are on AIME24), i.e., they fail to learn from experience at all. We hypothesize that base models cannot reliably exploit memories expressed as prepended context. Latent memories, in contrast, steer them effectively: *ELM* improves over zero-shot on every benchmark for both Qwen models, including a $+4.8$ point gain on MATH500 for the 1.5B model, where every baseline degrades by at least 25 points. For the instruction-tuned Llama-3.1-8B, text-based baselines are more competitive (e.g., LAG is best on GPQA), but *ELM* still yields the most consistent gains across benchmarks. Thus, while *ELM* spends additional compute to construct memories, this investment appears necessary to obtain any benefit from experience in base models. Importantly, *this gap is not explained by our routing mechanism*: equipping the baselines with the same router recovers at most their zero-shot performance on the Qwen models, and they still fall short of *ELM* (Appendix E.4). Finally, both our retrieval variants perform comparably, with small differences across models and datasets.

Can *ELM* surpass its training signal? Since memories are trained on the model's own majority-vote answers, zero-shot maj@8 can be viewed as the supervision signal available to *ELM*. Tab. 13 (Appendix) shows that applying majority voting on top of *ELM* consistently exceeds this signal: the best *ELM* variant matches or surpasses zero-shot maj@8 in every setting, with gains of up to $+4.2$ points (Llama, MATH500) and $+2$–3 points on most benchmarks for both Qwen models. In contrast, training-free baselines generally fall below zero-shot maj@8, particularly for the base models. Although modest, these gains suggest that *ELM* can bootstrap beyond the quality of its own pseudo-labels, a key ingredient of genuine self-improvement.

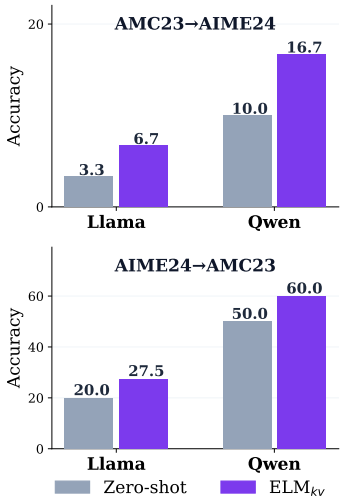


Figure 3: Cross-dataset transfer of *ELM* memories.

5 Ablations

In this section we ablate several design choices of our proposed method, specifically the memory design, the routing mechanism and the potential of transfer of memories across datasets. Further ablations (training objective, training steps, soft prompt length) and compute comparison between memory parameterizations are in Appendix E.

Memory Design: Soft Prompts vs. LoRA. Although LoRA is the standard choice for parameter-efficient adaptation, soft prompts are better suited for *ELM*. Offline, they generalize better than LoRA in 5 of 6 settings (Tab. 1). Online, LoRA memories, keyed by input-question activations stored at creation, yield no improvement over zero-shot (Appendix Tab. 6). Sec. D.1 traces this to retrieval misalignment: the top-retrieved LoRA memories tend to be those that perform poorly offline. Soft prompts avoid this issue, since they live in the embedding space used for retrieval. Beyond performance, soft prompts are also more compute- and memory-efficient, train faster, and incur lower inference overhead (Appendix D).[6]

Does *ELM* transfer across datasets? Since our main evaluation runs ELM online on test samples from a given dataset, we examine whether the resulting memories transfer across datasets. Fig. 3 shows transfer between AIME24 and AMC23, using pre-constructed memories from the other set, compared to the zero-shot baseline. Notably, we observe consistent gains on the target dataset, indicating that the learned memories generalize beyond a single test set.

Effect of response routing. Even without routing, always conditioning on the retrieved memory already improves over zero-shot ($+3.8$ points on average, Tab. 4, results on AMC23 and AIME24). Routing between the zero-shot and memory-conditioned answers further adds $+2.7$ to $+4.3$ points. This is expected, as a relevant memory may not always exist in the pool, particularly early in the stream when few memories have been accumulated, making unconditional reliance on retrieved memories noisy. Extended results, including the smaller model and an alternative Self-Judge router, are reported in Appendix E.5, where we find that routing gains scale with router quality.

6 Discussion and Limitations

Given the strong gains that large reasoning models obtain from increased test-time compute, we study how to distill such gains into reusable experience that improves future performance at lower cost. We introduce Experiential Latent Memory (*ELM*), a lightweight modular memory learned per sample with GRPO and internally derived rewards. We show that individual memories can generalize to unseen data and approach the performance of full offline training, but that memory quality varies and some memories can degrade otherwise correct responses. Leveraging the modularity of *ELM*, we propose an online protocol that retrieves relevant memories at inference and uses a response router to select between zero-shot and memory-conditioned answers. Across models and benchmarks, *ELM* yields consistent gains where training-free memories built from raw experience (e.g., solutions or extracted strategies) often fail, most notably on base models, where they typically fall below zero-shot performance. This suggests that reliable learning from experience requires investing compute at memory creation time, a cost that *ELM* pays once per sample, in the background, and amortizes over all future queries. Overall, this work is a first step toward models capable of continual self-improvement.

	Llama-8B	Qwen-7B
Zero-shot	12.9	31.7
input retr.		
Always retr.	15.3	35.9
+ Router	19.6	39.1
kv retr.		
Always retr.	17.2	36.1
+ Router	20.3	38.8

Table 4: Effect of response routing (pass@1, avg. over AMC23 and AIME24). Full results in App. E.5.

Several limitations remain. Memory creation adds compute per encountered sample, and serving requires a second decoding pass and a verifier call; while this overhead is amortized and modest, it may be prohibitive in latency- or resource-constrained settings. We do not address how to identify samples that lead to high-quality memories; preliminary experiments show no strong correlation between simple training-free statistics and memory utility. Although retrieval filters many irrelevant memories, some retrieved memories remain harmful, likely due to incorrect self-supervision; we rely on a PRM verifier for reliability, and our results show that weaker routers such as self-judging recover much less of the gain, leaving the design of more efficient routing strategies to future work. Due to computational constraints, we limit our analysis of what individual memories encode and of objectives beyond majority voting. Finally, important open questions remain concerning memory compositionality, and how memories should be updated, merged, evicted, or distilled into the model over time. We hope this work provides a foundation for reliable and efficient continual self-improvement.

[6] We also discuss Prefix Tuning as an alternative in Appendix B.

AI USE STATEMENT

In this work, we used generative AI tools for polishing the manuscript, including rephrasing and shortening text and formatting latex tables. We have not used generative AI tools for generating research ideas, designing the method, or running experiments, and the remaining required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools to assist in writing scripts that parse experiment logs and aggregate results (e.g., computing means over runs) for the tables. We have reviewed all AI-assisted work: all reported numbers were cross-checked by the authors against the raw experiment logs, and all AI-edited text was verified to accurately reflect our methods and results. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work uses only publicly available models and benchmarks and involves no human subjects or personal data. We are not aware of any conflicts of interest or other ethical concerns.

REPRODUCIBILITY STATEMENT

All experiments use publicly available models and benchmarks. The method is described in Sec.3 and Algorithm 1, and all experimental details, including hyperparameters, prompts, and random seeds, are provided in Appendix. We plan to publicly release our code upon acceptance.

REFERENCES

Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. *Advances in neural information processing systems*, 32, 2019.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. What do neural machine translation models learn about morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 861–872, 2017.

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

Peter Baile Chen, Yi Zhang, Dan Roth, Samuel Madden, Jacob Andreas, and Michael Cafarella. Log-augmented generation: Scaling test-time reasoning with reusable computation. *arXiv preprint arXiv:2505.14398*, 2025.

Vaggelis Dorovatas, Malte Schwerin, Andrew D Bagdanov, Lucas Caccia, Antonio Carta, Laurent Charlin, Barbara Hammer, Tyler L Hayes, Timm Hess, Christopher Kanan, et al. Modular memory is the key to continual learning agents. *arXiv preprint arXiv:2603.01761*, 2026.

Mehdi Fatemi, Banafsheh Rafiee, Mingjie Tang, and Kartik Talamadupula. Concise reasoning via reinforcement learning. *arXiv preprint arXiv:2504.05185*, 2025.

Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3 (4):128–135, 1999.

Jiaxuan Gao, Shusheng Xu, Wenjie Ye, Weilin Liu, Chuyi He, Wei Fu, Zhiyu Mei, Guangju Wang, and Yi Wu. On designing effective rl reward at training time for llm reasoning. *arXiv preprint arXiv:2410.15115*, 2024.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5484–5495, 2021.

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

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

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

Bingxiang He, Yuxin Zuo, Zeyuan Liu, Shangziqi Zhao, Zixuan Fu, Junlin Yang, Cheng Qian, Kaiyan Zhang, Yuchen Fan, Ganqu Cui, et al. How far can unsupervised rlvr scale llm training? *arXiv preprint arXiv:2603.08660*, 2026.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Yubo Hou, Zhisheng Chen, Tao Wan, and Zengchang Qin. Flashmem: Distilling intrinsic latent memory via computation reuse. *arXiv preprint arXiv:2601.05505*, 2026.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Dulhan Jayalath, Shashwat Goel, Thomas Foster, Parag Jain, Suchin Gururangan, Cheng Zhang, Anirudh Goyal, and Alan Schelten. Compute as teacher: Turning inference compute into reference-free supervision. *arXiv preprint arXiv:2509.14234*, 2025.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, et al. Measuring faithfulness in chain-of-thought reasoning. *arXiv preprint arXiv:2307.13702*, 2023.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 3045–3059, 2021.

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

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4582–4597, 2021.

Jian Liang, Ran He, and Tieniu Tan. A comprehensive survey on test-time adaptation under distribution shifts. *International Journal of Computer Vision*, 133(1):31–64, 2025.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The twelfth international conference on learning representations*, 2023.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.

Siru Ouyang, Jun Yan, I Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T Le, Samira Daruki, Xiangru Tang, et al. Reasoningbank: Scaling agent self-evolving with reasoning memory. *arXiv preprint arXiv:2509.25140*, 2025.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=Ti67584b98.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025.

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

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726*, 2020.

Yiping Wang, Qing Yang, Zhiyuan Zeng, Liliang Ren, Liyuan Liu, Baolin Peng, Hao Cheng, Xuehai He, Kuan Wang, Jianfeng Gao, et al. Reinforcement learning for reasoning in large language models with one training example. *arXiv preprint arXiv:2504.20571*, 2025.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Fangzhi Xu, Hang Yan, Chang Ma, Haiteng Zhao, Qiushi Sun, Kanzhi Cheng, Junxian He, Jun Liu, and Zhiyong Wu. 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)*, pp. 13153–13167, 2025.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, et al. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. *arXiv preprint arXiv:2503.18892*, 2025.

Guibin Zhang, Muxin Fu, and Shuicheng Yan. Memgen: Weaving generative latent memory for self-evolving agents. *arXiv preprint arXiv:2509.24704*, 2025a.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, et al. Agentic context engineering: Evolving contexts for self-improving language models. *The Fourteenth International Conference on Learning Representations*, 2026a.

Yanzhi Zhang, Zhaoxi Zhang, Haoxiang Guan, Yilin Cheng, Yitong Duan, Chen Wang, Yue Wang, Shuxin Zheng, and Jiyan He. No free lunch: Rethinking internal feedback for llm reasoning. *arXiv preprint arXiv:2506.17219*, 2025b.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2024, 2024.

Yunjian Zhang, Sudong Wang, Yang Li, Peiran Xu, Conghao Zhou, Xiaoyue Ma, Jianing Li, and Yao Zhu. Resource-efficient reinforcement for reasoning large language models via dynamic one-shot policy refinement. *arXiv preprint arXiv:2602.00815*, 2026b.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 10495–10516, 2025c.

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

Xuandong Zhao, Zhewei Kang, Aosong Feng, Sergey Levine, and Dawn Song. Learning to reason without external rewards. *arXiv preprint arXiv:2505.19590*, 2025b.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang, Xinwei Long, Ermo Hua, et al. Ttrl: Test-time reinforcement learning. *arXiv preprint arXiv:2504.16084*, 2025.

A Additional Experimental Details

All models are trained using the TRL GRPO Trainer (Hugging-Face) with the DR-GRPO (Liu et al., 2025) loss variant. As mentioned, we use a group size of G=8 completions per prompt, a maximum completion length of 2048 tokens, and a KL penalty of β=0.01. We use the AdamW fused optimizer with a constant learning rate of 10^{-4} for soft prompt and LoRA memories, 10^{-3} for Prefix Tuning and 10^{-6} for full model training, with a global batch size of 64 and no gradient accumulation. For Qwen models we use a sampling temperature of 0.8 and 0.7 for Llama. All training is performed on a single H200 GPU, with evaluations conducted on an A100 GPU using the `lm-eval-harness` library. The offline experiments on AMC23 and AIME24 require around 12 hours to complete, while for online, extracting memories (training) requires around 5 hours for AMC23 and AIME24, and 1 day for MATH500, and then evaluation needs 2h for the small ones and around 10 hours for MATH500. For the *offline* experiment on MATH500 subsets we use the seeds $[17, 42, 123, 456, 789, 1234, 5678, 9999]$, with a subset of 80 which we split into 40 for training and 40 for evaluation. For the *online* experiments we use 3 seeds, namely $[\text{None}, 17, 42]$ and we report the mean over them (None simply means we use the original dataset ordering). We use a Process Reward Model (PRM) as the verifier, Qwen2.5-Math-PRM-7B (Zhang et al., 2025c).

Online Experiments. We always retrieve the top-1 memory for each input. The verifier model we use is a Process Reward Model (PRM), specifically Qwen2.5-Math-PRM-7B (Zhang et al., 2025c). Finally, for each online experiment we run 3 seeds and we report average performance for each method.

A.1 Prompts

We use a simple zero-shot chain-of-thought prompt across all experiments:

```
Please reason step by step, and put your final answer within \boxed{answer}.
{question}
```

For Llama-3.1-8B-Instruct, this prompt is wrapped using the model's default chat template.

A.2 Memory Retrieval Details

We assume a pool of learned memories $\mathcal{M} = \{m^{(i)}\}_{i=1}^{N}$, each associated with a key constructed at memory creation.

Input retrieval. Given an input x, we compute a query $q = \text{mean}_t(\mathbf{E}(x)) \in \mathbb{R}^D$, where $\mathbf{E}(x)$ is the input embedding matrix and the mean is over the sequence dimension. Keys $k^{(i)}$ are constructed analogously, and we select

$$m^* = \underset{m^{(i)} \in \mathcal{M}}{\text{argmax}} \, \text{sim}\left(q, k^{(i)}\right).$$

KV retrieval. For each layer $l \in \{1, \ldots, L\}$, we pool attention activations across heads and sequence positions to obtain a layer-wise query $q^{(l)} \in \mathbb{R}^D$; at memory creation, the key vectors of each layer are pooled in the same way to obtain $k^{(i,l)}$. Similarities are computed per layer and averaged:

$$m^* = \underset{m^{(i)} \in \mathcal{M}}{\text{argmax}} \frac{1}{L} \sum_{l=1}^{L} \text{sim}\left(q^{(l)}, k^{(i,l)}\right).$$

B Prefix Tuning Parametrization

For completeness, we also compare against Prefix Tuning Li & Liang (2021) as an additional memory parameterization. Prefix Tuning prepends trainable continuous vectors directly to the keys and values of each transformer attention layer, offering a deeper form of conditioning than soft prompts. We initialize them from the key-value activations of the question (similar idea with the soft prompts). As shown in Tab. 5, Prefix Tuning underperforms overall soft prompts, while being more parameter-heavy, further supporting our choice of soft prompt memories for ***ELM***.

	#Trainable Params	AMC23	AIME24
Llama-3.1-8B-Instr.			
ELM-SoftPrompt$_{\text{offline}}$	0.07M	32.5	10
ELM-LoRA$_{\text{offline}}$	42M	30	6.7
ELM-PrefixTuning$_{\text{offline}}$	6.5M	30	10
Qwen2.5-Math-7B			
ELM-SoftPrompt$_{\text{offline}}$	0.07M	65	20
ELM-LoRA$_{\text{offline}}$	42M	62.5	10
ELM-PrefixTuning$_{\text{offline}}$	4M	55	13.3

Table 5: **Memory Design.** Average accuracy (greedy pass@1) across 3 seeds on AMC23 and AIME24.

	AMC23	AIME24
Llama		
$\hookrightarrow$ ZS	20	3.3
ELM-LoRA	20	3.3
Qwen		
$\hookrightarrow$ ZS	50	10
ELM-LoRA	50	10

Table 6: Average performance of *ELM* with LoRA memory parametrization (greedy pass@1) across 3 seeds on AMC23 and AIME24. As we discuss (Misaligned Retrieval), the top-retrieved memories in the online experiment are the ones that do not improve over base in the offline experiment, thus we see no improvement over baseline here.

C Additional Offline Statistics

We extend here Fig. 2 of the main paper to include both models and AMC23. Results are in Fig. 4.

D Additional Comparison of Soft Prompts against LoRA

D.1 Performance of LoRA in the online setting

Misaligned Retrieval. To further understand the online failure of LoRA memories, we analyze retrieval statistics on AMC23 with Qwen. The three most frequently retrieved memories account for roughly 72% of all retrievals: M13 (36%), which is neutral in the offline setting with no improvement over zero-shot; M10 (23%), which improves only a single sample; and M24 (13%), which regresses in the offline setting. This confirms that the retrieval mechanism systematically favors memories that fail to generalize, highlighting a fundamental misalignment between retrieval and utility that explains the lack of online improvement observed for LoRA.

D.2 Compute and Memory Overhead

In the following table we show the number of trainable parameters, as well as the seconds per GRPO step of LoRA and Soft Prompts. Soft Prompt memories are significantly lighter.

E Additional Comparisons

E.1 Objective and Training Steps

Due to space limitations in the main paper we present here the ablation table of traininig objective and number of training steps:

	# Train Params	Train Latency (s/step)
Llama		
↪ *ELM*	0.08M	70
↪ *ELM*-LoRA	42M	140
Qwen		
↪ *ELM*	0.07M	48
↪ *ELM*-LoRA	40M	91

Table 7: Efficiency comparison of Soft Prompts vs. LoRA memory design.

	AMC23	AIME24
Zero-shot	50	10
ELM	65	20
ELM-2 S	62.5	16.7
ELM-SFT	62.5	13.3

Table 8: The effect of objective and only 2 training steps on *ELM* (with the large Qwen model).

Objective. We use GRPO with majority voting, converting self-generated outputs into rewards. Unlike SFT on greedy responses, which can reinforce errors in the absence of ground truth, GRPO exploits additional test-time compute to produce stronger training signals, leading to more robust generalization (Tab. 8).

Effect of Training Steps. In the main experiments, we train memories for 10 steps, as a good trade-off between convergence and training time. In Tab. 8, we report results with only 2 steps, showing smaller but consistent gains over the zero-shot baseline.

E.2 Number of Soft Prompt Tokens

We ablate the number of soft prompt tokens per memory in the offline setting on AMC23 with Qwen2.5-Math-7B (Tab. 9). Performance is stable for small memories (5–20 tokens) but degrades sharply as the number of tokens grows, while the computational cost increases. Since each memory is trained on a single sample for only a few steps, a compact soft prompt acts as a regularizer and generalizes better, whereas larger ones overfit.

#Soft prompt tokens	5	20	50	100
AMC23 (pass@1)	67.5	65.0	52.5	40.0

Table 9: Effect of the number of soft prompt tokens (offline, Qwen2.5-Math-7B, AMC23).

E.3 Zero-shot Baseline with *verifier@2*

To ensure fair comparison with our method, as mentioned in the main paper, here we present the results of the base model when sampling 2 responses and using the verifier to select between them, thus matching the extra computational overhead of our method. As we see in Tab. 10 this baseline elevates the performance of greedy zero-shot (as more test-time compute is spent) but on average lags behind *ELM*.

E.4 Training-free memories with retrieval vs zero-shot selection

In this section, we isolate the effect of the **memory representation** by pairing the training-free baselines (ICL, Reasoning Bank, DC, and LAG) with our *response routing* mechanism. At inference time, each method produces both a zero-shot and a memory-conditioned response, and the final answer is selected by the same PRM router used for *ELM*. This controls for the routing strategy and allows a direct comparison of the memories themselves.

Tab. 11 reports the results across all models and benchmarks. Routing benefits every method, confirming that it is a general component rather than one tailored to *ELM*. However, routing cannot

	AMC23	AIME24
Llama		
↪ ZS	20	3.3
verifier@2	25	4.2
Qwen		
↪ ZS	50	10
verifier@2	57	16

Table 10: Performance of base model zero-shot greedy vs *verifier@2*.

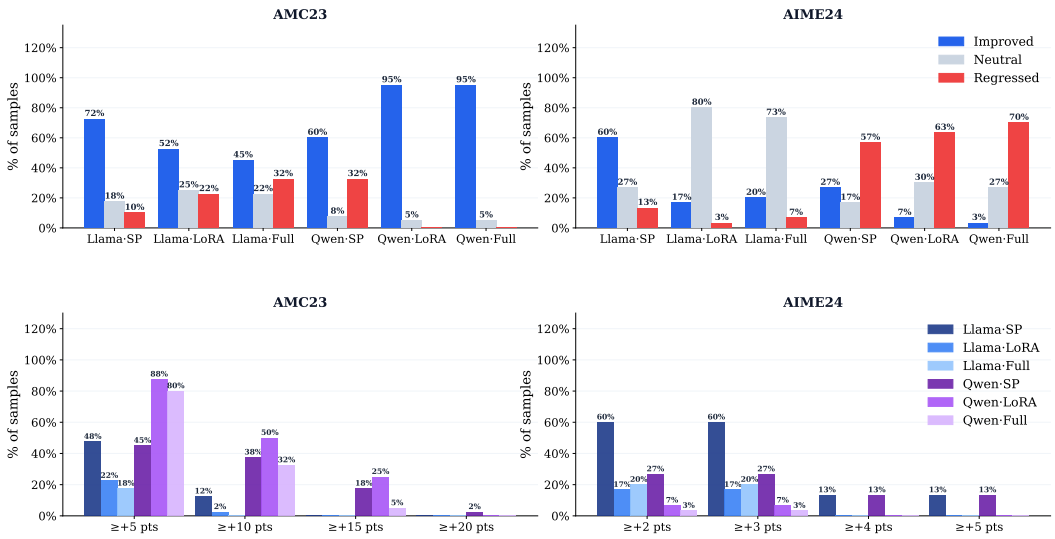


Figure 4: Statistics of *ELM* on AMC23 and AIME24 across memory designs.

compensate for memories that the model cannot exploit. On the base models, routed baselines at best recover zero-shot performance and remain well below it on MATH500 (e.g., the best routed baseline reaches 62.8 on Qwen2.5-Math-7B vs. 65.6 zero-shot and 72.0 for *ELM*), indicating that the verifier must frequently fall back to the zero-shot answer. The picture differs for the instruction-tuned Llama-3.1-8B: here, routed text-based memories become competitive and even outperform *ELM* on MATH500 (52.9 vs. 50.9) and GPQA (27.4 vs. 19.7), while *ELM* remains clearly ahead on AIME24 (12.2 vs. 5.6). Together, these results sharpen the main paper's conclusion: text-based memories are useful when the model can follow them, whereas learned latent memories remain effective even without instruction-following capabilities.

E.5 Response Router Ablation

In the main paper, a PRM verifier selects between the zero-shot response and the memory-conditioned response. We chose it because it is simple and efficient (forward passes only, no additional generations) and reflects the widespread use of verifier-based selection. However, our framework depends on *having* a routing mechanism (motivated in Tabs. 3 and 12), not on this particular implementation. To test this, we evaluate a fundamentally different router, **Self-Judge**, in which the base model itself adjudicates between its two candidate answers, a strategy also used in related work (Ouyang et al., 2025).

Given a problem x with zero-shot answer a_{ZS} and memory-conditioned answer a_{Retr}, we construct the prompt:

```
Problem: {x} Two students solved this problem and got
different answers.
(A) \boxed{a_Retr}
(B) \boxed{a_ZS}
```

	Llama-8B		Qwen-Math-7B		Qwen-Math-1.5B	
	AMC	AIME	AMC	AIME	AMC	AIME
Zero-shot CoT	20.0	3.3	50.0	10.0	45.0	10.0
Online (with response routing)						
ICL	28.3	4.4	37.5	11.1	41.7	8.9
Reasoning Bank	23.3	5.6	50.0	12.2	45.0	10.0
LAG	**29.2**	4.4	44.2	7.8	36.7	10.0
DC	26.7	4.4	48.3	13.3	37.5	4.4
ELM (input)	**29.2**	10.0	59.2	**18.9**	**50.0**	**14.4**
ELM (kv)	28.3	**12.2**	**64.2**	13.3	**50.0**	13.3

Table 11: Online performance (greedy pass@1) on AMC23 and AIME24 with response routing applied to all methods.

	Llama-8B		Qwen-Math-7B		Qwen-Math-1.5B	
	AMC	AIME	AMC	AIME	AMC	AIME
Zero-shot CoT	20.0	3.3	50.0	10.0	45.0	10.0
ELM (input retr.)						
Always retrieve	21.7	8.9	55.0	16.7	44.2	8.9
+ Self-Judge router	23.3	2.2	52.5	16.7	42.5	8.9
+ PRM router	29.2	10.0	59.2	18.9	50.0	14.4
ELM (kv retr.)						
Always retrieve	23.3	11.1	60.0	12.2	40.0	11.1
+ Self-Judge router	25.8	3.3	55.0	14.4	41.7	10.0
+ PRM router	28.3	12.2	64.2	13.3	50.0	13.3

Table 12: Effect of response routing on AMC23 and AIME24 (greedy pass@1). *Always retrieve* conditions every query on the retrieved memory; the router variants choose between the zero-shot and memory-conditioned answers using either the model itself as judge (*Self-Judge*) or a PRM verifier (*PRM*).

```
Only one answer is mathematically correct.  The correct
answer is:
```

We score the model's preference by the log-probability gap between the next tokens "A" and "B". To remove position bias, we query the model twice with the answers swapped and average:

$$\ell_1 = \log p(\mathrm{A}) - \log p(\mathrm{B}) \;\; (a_{\text{Retr}} \text{ in slot A}), \qquad \ell_2 = \log p(\mathrm{A}) - \log p(\mathrm{B}) \;\; (a_{\text{Retr}} \text{ in slot B}),$$
$$s = \tfrac{1}{2}(\ell_1 - \ell_2),$$

selecting a_{Retr} if $s > 0$ and a_{ZS} otherwise. This router requires no additional model, no generation, and no training signal beyond the base model's own token-level preferences.

As shown in Tab. 12, Self-Judge can improve over always retrieving in some settings, while the PRM router dominates throughout, consistent with the benefit of routing scaling with router quality.

E.6 Majority@8 Results

Tab. 13 reports maj@8 results for all methods, models, and benchmarks, complementing the greedy pass@1 results in the main paper. These results underlie the discussion in the main paper (*"Can ELM surpass its training signal?"*): the best *ELM* variant matches or surpasses zero-shot maj@8, i.e., its own training signal, in every setting, while training-free baselines generally fall below it.

E.7 Statistical Reporting

In this section we report the mean along with the full standard deviation of all online results over three data-stream orderings (Table 14), for completeness. As expected, the variance is higher in the smaller datasets.

	Llama-3.1-8B-Instr.				Qwen2.5-Math-7B				Qwen2.5-Math-1.5B			
	MATH	AMC	AIME	GPQA	MATH	AMC	AIME	GPQA	MATH	AMC	AIME	GPQA
Zero-shot (maj@8)	56.8	34.3	7.1	29.2	75.9	72.0	30.0	28.4	72.6	60.2	16.9	25.3
Online												
ICL	52.5	29.2	5.6	**30.3**	45.7	33.3	20.0	19.7	35.7	25.0	5.6	12.1
Reasoning Bank	55.8	23.3	4.4	26.4	50.4†	56.7	22.2	20.2	43.5	49.2	13.3	15.5
LAG	56.0	29.2	**7.8**	29.8	52.3	26.7	3.3	14.1	35.4	33.3	2.2	17.5
DC	52.1	29.2	**7.8**	18.4	49.2	45.0	23.3	9.3	56.9	42.5	15.6	11.4
ELM (input)	**61.0**	35.0	**7.8**	28.1	78.6	70.8	**32.2**	30.8	74.4	**62.5**	17.8	25.3
ELM (kv)	60.7	**36.7**	6.7	29.8	**78.7**	**72.5**	**32.2**	**31.0**	**75.0**	**62.5**	**20.0**	**27.8**

Table 13: **Maj@8**. Online performance comparison across models and datasets.

	Llama-3.1-8B-Instr.				Qwen2.5-Math-7B				Qwen2.5-Math-1.5B			
	MATH	AMC	AIME	GPQA	MATH	AMC	AIME	GPQA	MATH	AMC	AIME	GPQA
Zero-shot CoT	45.0	20.0	3.3	21.7	67.0	50.0	10.0	24.7	63.2	45.0	10.0	18.2
Online												
ICL	$41.2_{\pm 0.6}$	$20.0_{\pm 2.0}$	$2.2_{\pm 1.6}$	$19.5_{\pm 1.0}$	$24.0_{\pm 1.2}$	$25.0_{\pm 3.5}$	$13.3_{\pm 2.7}$	$16.5_{\pm 2.1}$	$25.8_{\pm 0.4}$	$12.5_{\pm 2.0}$	$2.2_{\pm 1.6}$	$8.9_{\pm 0.6}$
Reasoning Bank	$41.2_{\pm 1.5}$	$22.5_{\pm 2.0}$	$2.2_{\pm 1.6}$	$16.3_{\pm 1.6}$	$29.6_{\pm 1.4}$	$25.8_{\pm 3.1}$	$12.2_{\pm 1.6}$	$14.5_{\pm 1.7}$	$20.3_{\pm 2.6}$	$39.2_{\pm 6.2}$	$8.9_{\pm 1.6}$	$6.7_{\pm 1.4}$
LAG	$44.0_{\pm 0.9}$	$22.5_{\pm 2.0}$	$4.4_{\pm 1.6}$	$\mathbf{22.2}_{\pm 1.0}$	$34.2_{\pm 1.5}$	$19.2_{\pm 4.2}$	$3.3_{\pm 2.7}$	$12.6_{\pm 1.0}$	$26.8_{\pm 1.0}$	$18.3_{\pm 6.6}$	$1.1_{\pm 1.6}$	$12.6_{\pm 1.9}$
DC	$39.1_{\pm 0.8}$	$19.2_{\pm 1.2}$	$2.2_{\pm 1.6}$	$13.6_{\pm 1.1}$	$32.1_{\pm 1.7}$	$42.5_{\pm 7.1}$	$16.7_{\pm 7.2}$	$6.6_{\pm 1.5}$	$37.4_{\pm 0.9}$	$29.2_{\pm 3.1}$	$4.4_{\pm 1.6}$	$5.6_{\pm 0.4}$
ELM (input)	$\mathbf{50.9}_{\pm 0.7}$	$\mathbf{29.2}_{\pm 2.4}$	$10.0_{\pm 0.0}$	$19.7_{\pm 1.2}$	$\mathbf{72.0}_{\pm 0.7}$	$59.2_{\pm 1.2}$	$\mathbf{18.9}_{\pm 1.6}$	$27.3_{\pm 0.4}$	$\mathbf{68.0}_{\pm 0.8}$	$\mathbf{50.0}_{\pm 4.1}$	$\mathbf{14.4}_{\pm 1.6}$	$\mathbf{24.7}_{\pm 0.4}$
ELM (kv)	$\mathbf{50.9}_{\pm 0.5}$	$28.3_{\pm 5.1}$	$\mathbf{12.2}_{\pm 1.6}$	$19.5_{\pm 1.3}$	$71.3_{\pm 0.7}$	$\mathbf{64.2}_{\pm 1.2}$	$13.3_{\pm 4.7}$	$\mathbf{27.6}_{\pm 0.2}$	$67.4_{\pm 0.7}$	$\mathbf{50.0}_{\pm 2.0}$	$13.3_{\pm 2.7}$	$22.2_{\pm 0.5}$
Offline												
GRPO	47.0	27.5	6.7	–	68.0	62.5	20.0	–	–	–	–	–
TTRL	45.6	27.5	10.0	–	67.2	62.5	20.0	–	–	–	–	–

Table 14: **Main online results with standard deviations.** Greedy pass@1, reported as mean$_{\pm \text{std}}$ over three data-stream orderings. Zero-shot decoding is greedy and hence deterministic. Offline results are taken from Table 1 and reported for reference only.