

REWARD IS ENOUGH: LLMS ARE IN-CONTEXT REINFORCEMENT LEARNERS

000
001
002
003
004
005
006
007
008
009
010
011
012
013
014
015
016
017
018
019
020
021
022
023
024
025
026
027
028
029
030
031
032
033
034
035
036
037
038
039
040
041
042
043
044
045
046
047
048
049
050
051
052
053
Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning (RL) is a framework for solving sequential decision-making problems. In this work, we demonstrate that, surprisingly, RL emerges during the inference time of large language models (LLMs), a phenomenon we term in-context RL (ICRL). To reveal this capability, we introduce a simple multi-round prompting framework, we call ICRL prompting, for inference-time self-improvement. The goal of ICRL prompting is to guide LLMs to perform reinforcement learning during inference for self-improvement on a given task. After each response, the model receives numerical scalar feedback, denoted as a reward. In the next round, we prompt the LLM again together with a context that concatenates all prior responses and their associated rewards. We consistently observe that response quality improves as the context grows. In other words, the LLM can optimize scalar reward signals during inference, exhibiting behavior analogous to reinforcement learning. We evaluate ICRL prompting on Game of 24, creative writing, ScienceWorld, and Olympiad-level math competitions (AIME and HMMT), demonstrating significant improvements over baselines such as Self-Refine and Reflexion. Notably, even when the reward signals are generated by the same LLM, ICRL prompting still improves performance, highlighting a promising new paradigm for test-time scaling.

1 INTRODUCTION

For Large Language Models (LLMs) to act as effective agents on novel tasks, they must be able to improve during inference time, a capability often referred to as test-time scaling (Zhang et al., 2025). Learning and search are the two general methods that can leverage scaling computation for performance improvement (Sutton, 2019), reaching superhuman performance on Chess (Campbell et al., 2002) and Go (Silver et al., 2016). Search has been successfully applied to LLM self-improvement in test-time scaling, starting from the simple Best-of-N (Stiennon et al., 2022) to Tree of Thoughts (Yao et al., 2023) and Monte Carlo Tree Search (Ding et al., 2024).

Learning, however, has yet to receive the same attention for LLM self-improvement at inference time. In-context (supervised) learning (ICL; Brown et al. (2020)), as a supervised learning paradigm, requires expert demonstrations as ground-truth labels. However, such demonstration data are not easily scalable during inference time, which restricts the applicability of ICL to test-time scaling. Thus, LLMs must instead learn from their own generated experience for continual self-improvement (Silver & Sutton, 2024).

Reinforcement learning is perhaps the most successful algorithm capable of self-improvement independent of human knowledge (Silver et al., 2017). However, its major successes have primarily appeared in simulated environments (Mnih et al., 2015; Silver et al., 2016) or during the training time of LLMs Guo et al. (2025). These current RL settings fall short in the big world setting (Javed & Sutton, 2024), where the real-world environment is vastly more complex than the agent itself. In such environments, agents will encounter numerous situations far beyond their prior training data and must adapt and improve their solutions on the fly. Bridging this gap requires models that (1) can handle diverse tasks in the real world, where natural language often constitutes an essential action space (Silver & Sutton, 2025), and (2) can continually improve their solutions during inference, rather than relying on costly retraining for every novel situation.

054 This naturally raises the question: can reinforcement learning emerge during the inference phase of
 055 LLMs? Enabling LLMs to perform RL purely in context provides an elegant mechanism to meet
 056 both requirements: LLM provides a general-purpose initial policy, while RL introduces the capability
 057 for continual self-improvement. Inspired by the first surprising evidence that LLMs can act as
 058 in-context learners in supervised settings (Brown et al., 2020), a growing body of work has begun to
 059 explore in-context reinforcement learning (ICRL; Moeini et al. (2025)). However, current instantiations
 060 are largely restricted to bandit or simulated environments (Monea et al., 2025; Krishnamurthy
 061 et al., 2024), failing short of addressing many diverse open-ended tasks where natural language is
 062 the action space.

063 In this paper, we bridge this critical gap by demonstrating that LLMs can act as effective in-context
 064 reinforcement learners, an emergent capability that improves performance on diverse, language-
 065 based tasks ranging from conducting scientific experiments to creative writing to solving olympiad-
 066 level mathematics. To reveal this capability, we introduce a simple multi-round prompting frame-
 067 work, **ICRL prompting**. The goal of ICRL prompting is to guide LLMs to perform reinforcement
 068 learning for self-improvement on a task. Initially, the prompt is only the task description. After the
 069 LLM generates a response, we give numerical scalar feedbacks for the response, called the rewards.
 070 Then in the next round, we prompt the LLM again with the same task description and a context
 071 consisting of all previous responses and rewards. So on and so forth. We observe that the quality of
 072 the LLM’s response increases as the context grows. In other words, the LLM is able to maximize
 073 the scalar reward signal during the inference time, just like an RL algorithm.

074 A key design principle of ICRL prompting is minimality. To ensure that the observed gains arise
 075 from the emergent RL capacity of LLMs rather than auxiliary mechanisms, we deliberately exclude
 076 textual gradients (Yuksekgonul et al., 2025), prioritized experience replay, sampling-based heuris-
 077 tics (Zhang et al., 2024; Yang et al., 2024), or additional engineered modules (Brooks et al., 2024).
 078 The only supervision provided is the scalar reward itself. This complies with both the reward hy-
 079 pothesis (Sutton, 2004), “*that all of what we mean by goals and purposes can be well thought of as*
 080 *maximization of the expected value of the cumulative sum of a received scalar signal (reward)*”, and
 081 the “reward is enough” hypothesis (Silver et al., 2021), “*intelligence, and its associated abilities,*
 082 *can be understood as subsuming the maximisation of reward*”.

083 To summarize, this paper makes three contributions:

084 **(1):** We introduce the *ICRL prompting* framework, a minimal design that elicits inference-time self-
 085 improvement in LLMs using only scalar rewards. Just as ICL places (x, y) pairs in context, our
 086 framework places state-action-reward tuples with simple meta-instructions in context. This design
 087 isolates the LLM’s intrinsic capacity for ICRL, free from external code or engineered mechanisms.
 088 **(2):** We provide strong evidence suggesting the emergence of RL in LLM’s inference time when the
 089 ICRL prompting framework is used. Specifically, we demonstrate the maximisation of the scalar
 090 reward signal, the exploration-exploitation trade-off in LLM’s inference time, the performance im-
 091 provements from the growth of the context, the performance drop with short context, and the perfor-
 092 mance drop when the reward is absent. All those observations are well expected for an RL algorithm.
 093 Essentially, this is a “duck test” (Heim, 2007)¹ for the inference process.
 094 **(3):** We demonstrate that ICRL prompting yields significant improvements over self-revision meth-
 095 ods such as Self-Refine (Madaan et al., 2023) and Reflexion (Shinn et al., 2023), across diverse
 096 benchmarks including Game of 24, creative writing, ScienceWorld, and Olympiad-level mathemat-
 097 ics (AIME and HMMT). In Game of 24 and creative writing, the rewards are generated by the LLM
 098 itself, yet consistent performance gains are still observed.

2 BACKGROUND

100 **Reinforcement Learning.** RL uses Markov Decision Processes (MDPs) to model a task, consisting
 101 of a state space \mathcal{S} , an action space \mathcal{A} , a reward function $r : \mathcal{S} \rightarrow \mathbb{R}$, an initial distribution $p_0 \in$
 102 $\Delta(\mathcal{S})$ with $\Delta(\mathcal{S})$ denoting the set of probability distributions over \mathcal{S} , and a transition function $p : \mathcal{S} \times \mathcal{A} \rightarrow \Delta(\mathcal{S})$. At time step 0, an initial state S_0 sampled from p_0 . At time t , an agent at S_t
 103 takes an action A_t according to its policy $\pi : \mathcal{S} \rightarrow \Delta(\mathcal{A})$ with $\Delta(\mathcal{A})$ denoting the set of probability
 104 distributions over \mathcal{A} , i.e., $A_t \sim \pi(S_t)$. The action A_t is then executed, after which the agent
 105 transitions to a successor state $S_{t+1} \sim p(S_t, A_t)$ and receives a reward $R_{t+1} \doteq r(S_{t+1})$. This

1¹If it looks like a duck, swims like a duck, and quacks like a duck, then it probably is a duck.

108 agent-environment interaction continues until a time T , which marks the end of an episode. The
 109 goal of the agent is to adapt its policy π such that the expected total rewards $J(\pi) \doteq \mathbb{E}[\sum_{t=1}^T R_t]$ is
 110 maximized. In modern deep RL (Mnih et al., 2015; Schulman et al., 2017), the policy π is usually
 111 parameterized by a neural network. We use θ to denote the network parameter and write the policy
 112 as π_θ . Typically, RL algorithms update θ to adapt its policy. For example, at time t , the action A_t
 113 is sampled from $\pi_{\theta_t}(S_t)$. The RL algorithm then update θ_t to θ_{t+1} based on available information
 114 such as $S_0, A_0, R_1, \dots, S_t, A_t, R_{t+1}, S_{t+1}$. Then at time $t+1$, the action A_{t+1} is sampled from the
 115 updated policy $\pi_{\theta_{t+1}}(S_{t+1})$. Essentially, the typical RL process is reflected in the updates of θ_t .

116 **In-Context Reinforcement Learning.** ICRL (Moeini et al., 2025), first coined by Laskin et al.
 117 (2023), is an emerging inference-time compute paradigm where the RL process occurs in the inference
 118 time (i.e., the forward pass) of the network without any parameter update. In ICRL, the policy
 119 π_θ is additionally conditioned on a context called C_t , i.e., $A_t \sim \pi_\theta(S_t, C_t)$. The construction of C_t
 120 is an active research area but one example is to use all previous state-action-reward pairs obtained in
 121 the task. Notably, this usually includes state-action-reward pairs from all previous episodes, not just
 122 the current episode (Laskin et al., 2023). In ICRL, there is a pretraining stage where the network θ is
 123 pretrained in a wide range of tasks (MDPs). We use θ_* to denote the parameter after the pretraining.
 124 After the pretraining stage, the policy π_{θ_*} is evaluated in new tasks. In other words, in the new
 125 MDP, the action A_t is sampled from $\pi_{\theta_*}(S_t, C_t)$. Importantly, the θ_* is kept fixed. Nevertheless,
 126 it is observed that the quality of A_t increases as C_t grows in the new task. Since θ_* is fixed, this
 127 improvement can only come from the increase of the context. This is thus called in-context policy
 128 improvement. Notably, this in-context policy improvement is also observed even when the new task
 129 is out of the distribution of the pretraining tasks, e.g., Laskin et al. (2023) demonstrate in-context
 130 policy improvement in new bandit problems that have the opposite optimal arms to the pretraining
 131 bandit problems. Thus this in-context policy improvement cannot be attributed to the hypothesis that
 132 θ_* memorizes the pretraining tasks. The only plausible hypothesis seems to be that the forward pass
 133 of the network parameterized by θ_* implements some RL algorithm to process the information in
 134 the context C_t to generate the action A_t . This inference-time (forward pass) RL is called in-context
 135 RL.

136 **LLMs as RL Agents.** The token generation process of LLMs can be modeled via RL. In short, the
 137 state is all generated tokens and the action is the next token to generate. Namely, let \mathcal{V} be the set of
 138 all possible tokens. We consider a state space $\mathcal{S} \doteq \bigcup_{i=1}^{\infty} \mathcal{V}^i$ and an action space $\mathcal{A} \doteq \mathcal{V}$. At time step
 139 0, an initial prompt is given, denoted as $S_0 \in \mathcal{S}$. In this work, S_0 contains a description of a task.
 140 We refer to the LLM with parameter θ as π_θ . At time t , given the current tokens S_t , a new token
 141 A_t is sampled from $\pi_\theta(S_t)$. The new state is then $S_{t+1} = [S_t A_t]$, i.e., the new state is obtained by
 142 concatenating current tokens and the new token. A reward signal $R_{t+1} \doteq r(S_{t+1})$ is then emitted
 143 according to a reward function r . This token generation process continues until a time T , where
 144 either T is the maximal allowed response length or A_{T-1} is a special end-of-sequence token. Either
 145 way, this marks the end of an episode and the final state S_T , called the terminal state, contains both
 146 the initial task description and the LLM’s response. There are two types of reward functions. One
 147 is sparse (the outcome reward model, Ouyang et al. (2022)), where $r(s)$ is nonzero only when s is a
 148 terminal state. The other is dense (the progress reward model, Lightman et al. (2023)), where $r(s)$
 149 can also be nonzero for non-terminal states.

3 IN-CONTEXT REINFORCEMENT LEARNING PROMPTING

153 We now present our main contribution, the ICRL prompting framework (Algorithm 1, Figure 1),
 154 consisting of the following ingredients.

155 **LLM as the Policy.** An LLM, denoted as π_θ , serves as the policy network. The goal is to prompt the
 156 LLM to solve a task. We assume a natural language description of the task is available and we denote
 157 it as $s_{\text{task}} \in \mathcal{S}$. At the beginning of each episode, we construct the initial prompt by concatenating
 158 the LLM’s own previous attempts together with the corresponding rewards, the task description,
 159 and some meta instruction denoted as s_{ICRL} . The details of the concatenation of previous attempts
 160 and the choice of the meta instruction will be discussed shortly. With this initial prompt, the LLM
 161 generates the response. Both the response and the rewards are stored in the buffer for future episodes.

162 **Algorithm 1** ICRL Prompting

Require: An LLM π_θ . A reward function r . Number of episodes K . An experience buffer \mathcal{B} .
A task description $s_{\text{task}} \in \mathcal{S}$. The JICRL instruction $s_{\text{JICRL}} \in \mathcal{S}$.

```

165  A task description  $s_{\text{task}} \in \mathcal{S}$ . The LLM instruction  $s_{\text{ICRL}} \in \mathcal{S}$ .
166 1: for  $k = 1$  to  $K$  do
167 2:   Construct the initial prompt  $S_0$  by concatenating all the tokens in  $\mathcal{B}$ ,  $s_{\text{task}}$ , and  $s_{\text{ICRL}}$ .
168 3:    $t \leftarrow 0$  // Execute the policy  $\pi_\theta$  starting from  $S_0$ 
169 4:   while  $S_t$  is not terminal do
170 5:      $A_t \sim \pi_\theta(S_t)$ ,  $S_{t+1} \doteq [S_t A_t]$ ,  $R_{t+1} \doteq r(S_{t+1})$ ,  $t \leftarrow t + 1$ 
171 6:   end while
172 7:   //  $[A_0 A_1, \dots, A_{T-1}]$  is the LLM's response to  $s_{\text{task}}$  at the current episode
173 8:   Push  $(A_0, R_1, A_1, R_2, A_2, R_3, \dots, A_{T-1}, R_T)$  into  $\mathcal{B}$ .
9: end for

```

Reward Function. A numerical scalar reward feedback is provided for each S_t in the episode. Notably, the reward can be either sparse (i.e., only R_T is nonzero) or dense. The reward function can be rule-based, learned separately, or instantiated via the same LLM for self-evaluation. The flexibility of using LLM’s self-evaluation as the reward function allows the ICRL prompting framework to be applied to a wide range of tasks. Notably, this scalar reward is the only feedback we provide to the LLM. But we do tell the LLM that this scalar is a reward. We do so by explicitly writing down the word “Reward” before this number when constructing the initial prompt. Notably, if the reward function is rule-based and learned separately, the reward signal constitutes an external feedback. But if the reward function is just the LLM’s own evaluation of the answer, there is no external feedback at all in the ICRL prompting framework. Yet we still expect the LLM’s response to improve over the episode. This is because of the widely believed hypothesis that evaluation is easier than generation. But we do hypothesize that the ceiling with self-evaluation is lower than that with external feedback.

Memory for Experience. We use an experience buffer \mathcal{B} to store the LLM’s responses and rewards for the task in previous episodes. Our underlying hypothesis is that pretrained LLM already has the ICRL ability. To use this innate ICRL ability to improve LLM’s response to the task, we concatenate its previous attempts and rewards as many as the context window allows. We expect that the LLM can reinforcement learn from the experiences in the context during the inference time.

ICRL Instructions. To facilitate LLM’s inference time RL, we additionally provide some instructions in initial prompt S_0 at each episode. The instruction is in natural language and is denoted as *SICRL*. We consider three types of instructions: (1) the exploration instruction (Figure 4 in App. A), (2) the exploitation instruction (Figure 5 in App. A), (3) the exploration or exploitation instruction (Figure 6 in App. A). For exploration instruction, the model is asked to provide a response that is different from all its previous responses. For exploitation instruction, the model is asked to generate the best response based on the previous responses with the highest reward. We consider two strategies.

(1) ICRL Preset: We alternate between the exploration and exploitation instructions. When the episode number K is even, we use the exploration instruction. When the episode number K is odd, we use the exploitation instruction. **(2) ICRL Autonomous:** We always provide the “exploration or exploitation” instruction at each episode and let the LLM itself to decide on which to use.

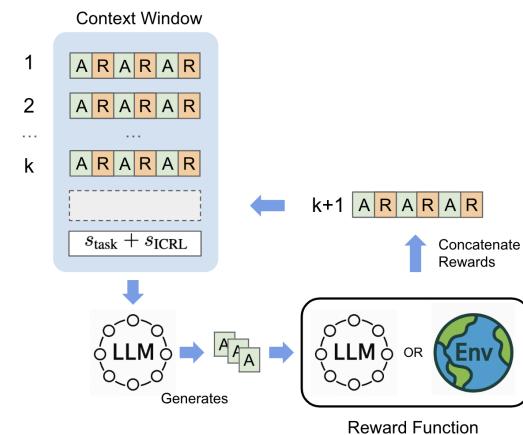


Figure 1: ICRL Prompting. At each episode $k+1$, LLM generates action tokens based on previous experiences up to k , and receives numerical rewards either from itself as the evaluator or from the environment. At the end of the episode, the rewards are then concatenated with the action tokens and placed back into the context.

216

4 RELATED WORKS

217

4.1 IN-CONTEXT REINFORCEMENT LEARNING.

218 The study of inference-time RL algorithms dates back to Duan et al. (2016); Wang et al. (2016),
 219 with Laskin et al. (2023) later coining the term in-context reinforcement learning (ICRL), spurring
 220 rapid growth in the field (Kirsch et al., 2023; Raparthy et al., 2023; Schmied et al., 2024; Lee et al.,
 221 2024; Zisman et al., 2023; Grigsby et al., 2024; Lu et al., 2023; Bauer et al., 2023; Wang et al.,
 222 2025; Cook et al., 2024; Xu et al., 2024; Shi et al., 2024; Huang et al., 2024; Liu & Abbeel, 2023;
 223 Dai et al., 2024). See Moeini et al. (2025) for a survey. Most existing ICRL works, as a subarea
 224 of meta-RL (Beck et al., 2023), use small models trained from scratch in games or robotics. Some
 225 employ pretrained LLMs, e.g., as simulators (Brooks et al., 2024; Mirchandani et al., 2023; Resendiz
 226 & Klinger, 2025) or in bandit tasks (Krishnamurthy et al., 2024; Nie et al., 2024; Park et al., 2024;
 227 Monea et al., 2025), where artificial interventions are often needed and LLMs remain uncompetitive
 228 with algorithmic baselines.

231

4.2 INFERENCE-TIME LLM SELF-IMPROVEMENT

232 Existing methods often rely on natural language self-revision, e.g., Self-Refine (Madaan et al.,
 233 2023), Reflexion (Shinn et al., 2023), and Textual Gradient (Yuksekgonul et al., 2025). Since the
 234 quality of self-revision is dependent upon the model’s parametric knowledge of the task, such ap-
 235 proaches are prone to hallucinated feedback that accumulates across iterations, leading to perfor-
 236 mance collapse (Stechly et al., 2025). In essence, they resemble language-guided search (Liu et al.,
 237 2025), where feedback serves as new explicit instructions for the next revision.

238 By contrast, ICRL requires only numerical rewards, without prescribing new instructions. The
 239 model must infer a better response by recognizing patterns from its past experience, making the
 240 process akin to reinforcement learning. Crucially, such rewards can also originate directly from the
 241 environment, providing a strong source of verification signals.

242 A parallel line of work improves LLMs at inference via search, e.g., Tree-of-Thoughts (ToT) (Yao
 243 et al., 2023), Graph-of-Thoughts (Got) (Besta et al., 2024), Monte Carlo Tree Search (MCTS) (Ding
 244 et al., 2024), and Intelligent Go-Explore (Lu et al., 2025). These methods largely depend on exter-
 245 nally engineered components such as heuristics or memory management, rather than leveraging the
 246 model’s intrinsic learning ability. Our work is also related to previous work on prompt optimization
 247 (Yang et al., 2024), where numerical scores guide prompt refinement, though through top-k selection
 248 and error filtering. Thus, it is more aligned with in-context supervised learning (e.g., filtered behav-
 249 ior cloning, Grigsby et al. (2024)) than reinforcement learning. In contrast, ICRL enables learning
 250 from failure experiences.

253

5 EXPERIMENT

254 In this section, we evaluate ICRL prompting on three benchmarks: Game of 24, creative writing
 255 from Yao et al. (2023), and ScienceWorld (Wang et al., 2022). We compare several baselines in-
 256 cluding **CoT-only**, **Long-CoT** style prompting, **Best-of-N**, **Self-Refine**, and **Reflexion**. Notably, in
 257 all the experiments, we allow the prompt of Self-Refine and Reflexion to grow as long as the LLM
 258 allows.

261

5.1 GAME OF 24

262 **Task Setup.** Given four input numbers, the model must use each number exactly once and apply
 263 only addition, subtraction, multiplication, or division in any order to reach 24. We choose the
 264 GPT-4.1 model for this experiment because of its excellent long-context capacity (OpenAI, 2025),
 265 accessed through api calls. Following Yao et al. (2023), we use CoT prompting to elicit the model to
 266 provide a step-by-step solution, where each step the model picks two numbers from the remaining
 267 numbers and an operation to perform the calculation, and obtains one complete solution per LLM
 268 query, containing a total of 4 thinking steps. To ensure that the LLM generates response with
 269 correct format, we additionally provide 5 in-context supervised learning demonstrations. The CoT

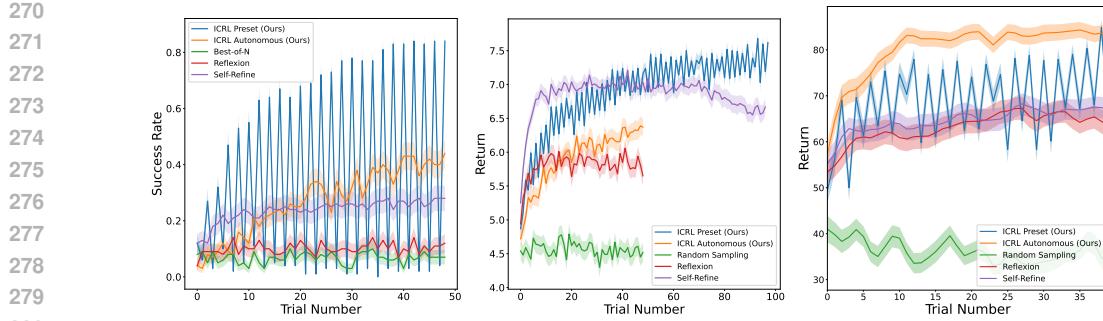


Figure 2: **Baseline Method Comparison.** **(Left)** Mean Success Rate on Game of 24. **(Middle)** Mean Coherence Reward on Creative Writing. Both ICRL Preset and Self-Refine went through an additional run of 50 episodes. **(Right)** Mean Return on Science World. A running max version of the plots is available in Figure 13 in App. B. This plot shows quality of the response at the current trial while the running max version shows the quality of the best response until now. The shaded region represents ± 1 standard error of the performance calculated across the evaluated tasks.

instruction and the 5 demonstrations together form the task description s_{task} (Figure 7 in App. A).

Evaluation. To verify the correctness of the solutions, we leverage SymPy (Meurer et al., 2017), a library for symbolic mathematics, by extracting operands and operators and evaluating the reconstructed expression to confirm it equals 24, and report the mean success rate over the 100 problems. This rule-based success rate is referred to as r_* (i.e., the ground truth reward function) in the rest of the paper. We use r to denote the reward function that the algorithms actually have access to. In particular, we use GPT-4.1 as the r , the same LLM as the policy LLM but prompted differently (see Figure 12 in App. A). After the policy LLM generates the response, for each thinking step, GPT-4.1 scores the likelihood of reaching 24 with the remaining numbers on a 0-3 scale (0 = impossible, 3 = sure). The task is challenging in that no algorithm has access to r_* . Instead, they have to rely on their own (possibly imperfect) evaluation, generated by the same LLM, to improve the response.

Baselines. We compare our method with CoT-only, Long-CoT style prompting, Best-of-N, Reflexion, and Self-Refine. In CoT-only prompting, the model receives only the task description s_{task} and produces a single step-by-step solution. In Long-CoT style prompting, we explicitly ask the LLM to generate a long chain-of-thought, and keep retrying if the solution is incorrect in "`<think>...</think>`" tags, before finally providing the answer. Although GPT-4.1 is not specifically trained for long-form CoT reasoning, we find that Long-CoT style prompting can elicit significantly longer and self-correcting thought traces compared to zero-shot prompting, making it a strong baseline for the Game of 24. Both methods cannot make use of a reward signal and are run for one pass. For Best-of-N, to make it even stronger, we use the ground truth reward r_* to select the best response. Self-Refine does not require a reward function. It instead asks the LLM itself to provide textual verbal feedback. Reflexion generates reflection according to r . ICRL prompting is different from Self-Refine and Reflexion in that it uses the reward r directly, without any verbal feedback. So the comparison between ICRL prompting and Self-Refine / Reflexion is essentially the comparison between scalar feedback and verbal feedback.

ICRL prompting. As discussed before, both π_θ and r in Algorithm 1 are the same LLM GPT-4.1 (prompted differently). We now clarify how we compute S_0 at each episode. Since each action is a

324 token, not all actions receive a reward. In fact, since we use CoT to prompt the LLM for 4 thinking
 325 steps, only 4 rewards are available for each episode. We thus only include those 4 rewards in S_0 .
 326 We add a “Reward: ” tag before the actual scalar reward and then concatenate the tagged reward
 327 immediately after the corresponding action (i.e., token).

329 **Results.** The success rate (i.e., r_*) against the number of trials (i.e., the episodes in Algorithm 1) is
 330 reported in Figure 2. The ICRL Preset method achieves the highest performance, and the observed
 331 oscillations in success rate reflect the model’s alternating phases of exploration and exploitation.
 332 The mean of running max is also plotted in Figure 13 in App. B. For each problem, we compute its
 333 running maximum success rate up to each episode and then average these values across all problems
 334 at every episode. As summarized in Table 1, after 50 trials, our methods achieve a success rate of
 335 90% which is significantly higher than 49% from Best-of-N sampling, 47% from Self-Refine, and
 336 44% from Reflexion.

339 5.2 CREATIVE WRITING

341 **Task Setup.** We consider the creative writing task from Yao et al. (2023), where four sentences are
 342 randomly sampled from a pool of sentences. The task for LLMs is to generate four paragraphs, each
 343 ending with a sentence, while ensuring that the generated passage is coherent. This is a difficult
 344 task, as it challenges the LLMs to craft a unified storyline that logically justifies each of the four
 345 sampled sentences by weaving them into a single narrative. A total of 100 problems are evaluated.
 346 An example of s_{task} is in Figure 8 in App. A.

347 **Evaluation.** We evaluated outputs using the Length-
 348 Controlled Alpaca-Eval 2 (Tatsu Lab, 2025) framework,
 349 a widely used proxy for human evaluation (Hong et al.,
 350 2024; Ethayarajh et al., 2024; Meng et al., 2024) with up
 351 to 0.98 Pearson correlation with human judgments. For
 352 100 creative writing problems, we present each method’s
 353 top response: for Reflexion and Best-of-N, the highest-
 354 reward output among 50 trials; for ICRL and Self-Refine,
 355 the 50th episode output. Alpaca-Eval then computes pair-
 356 wise win rates, denoted as r_* .

357 We next introduce the reward function r accessible to
 358 the algorithms. We follow standard pairwise comparison
 359 (Zheng et al., 2023), using GPT-4.1 with a coherent reference paragraph to score each response from
 360 1–10 (see Fig. 11). Notably, although both r and r_* use an LLM as a judge, they serve distinct pur-
 361 poses. r compares responses against a fixed reference text with emphasis on coherence. By contrast,
 362 r_* performs pairwise comparison between two responses generated by our method and a baseline
 363 method.

364 **Baselines.** We compare our method with Best-of-N, Reflexion, and Self-Refine. We allow Best-of-
 365 N, Reflexion and ICRL prompting to use r . Self-Refine do not use r and instead asks GPT-4.1 to
 366 provide verbal feedback. Since it is hard to distinguish CoT and Long-CoT style prompting for this
 367 task, we include Long-CoT style prompting as the baseline.

368 **ICRL prompting.** GPT-4.1 is used as both the policy LLM π_θ and the reward model r in Algo-
 369 rithm 1. At each episode, the initial prompt S_0 is constructed by concatenating all of the previous
 370 generations along with their coherence scores from r . Notably, this reward is sparse and only R_T
 371 can be nonzero. We, therefore, only include R_T in constructing S_0 .

372 **Results.** Our method achieves a length-controlled win rate of 59.48% against Reflexion, 78.36 %
 373 against Long-CoT style prompting, 86.32 % against Self-Refine, and 93.81 % against Best-of-N
 374 as shown in Table 2. This shows the responses generated by our method outperform the ones by
 375 baselines in terms of following the instruction to write coherent paragraphs and achieving better
 376 human preference. The return curve from reward model r is plotted in Figure 2, and a running
 377 max of the return is plotted in Figure 13 in App. B. Although Self-Refine initially matches ICRL
 in terms of coherence reward, extending both methods by 50 additional episodes, our methods keep

Table 2: Length-Controlled Win Rate (LC) and Standard Error (SE) from Alpaca-Eval 2.0 on Creative Writing.

Comparison	LC \pm SE (%)
Ours vs Reflexion	59.48 \pm 3.47
Ours vs Long CoT	78.36 \pm 1.99
Ours vs Self-Refine	86.32 \pm 3.03
Ours vs Best-of-N	93.81 \pm 1.01

378 improving, whereas self-refine first plateaus, then declines, likely due to the significant growth of its
 379 context.
 380

381 **5.3 SCIENCEWORLD**
 382

383 **Task Setup.** ScienceWorld (Wang et al., 2022) is an interactive, text-based benchmark consisting of
 384 30 science-experiment tasks set in a multi-room environment populated with diverse objects. The
 385 environment is challenging due to sparse rewards, large action spaces, and the requirement for sci-
 386 entific knowledge and efficient exploration. At each step, the agent observes the result of its action
 387 and receives zero reward unless it completes a predefined subgoal. This reward signal is used both
 388 for evaluation and for inference-time self-improvement (i.e., r and r_* are identical in this task).
 389 Completing all subgoals yields a cumulative reward of 100 and marks the episode as successful.
 390 An episode ends in failure if the agent either reaches the maximum number of steps or executes an
 391 incorrect terminating action. The input s_{task} provided to the agent describes the environment, the
 392 task, and the template of all possible actions. An example of s_{task} is provided in App. A

393 **Evaluation.** We use the environment-provided reward function for each task both to construct the
 394 trajectories used in the context (r), and to evaluate the model (r^*). We benchmark each method on
 395 all 30 tasks and aggregate the results. GPT-4.1 mini is used as the policy for all compared algo-
 396 rithms.

397 **Baselines.** In Reflexion, at the end of each episode, the agent is prompted to reflect on its attempt.
 398 The reflection is then sanitized and appended to a reflection buffer, which is formatted into the con-
 399 text for subsequent trials. Self-Refine similarly generates self-feedback, but appends it to a trajectory
 400 summary, which is then added to the buffer. To ensure a fair comparison, we allow these methods
 401 access to the reward signals of the current episode (unlike ICRL) before prompting for reflection.

402 **ICRL Setup.** Each trial corresponds to a single episode in the environment. After the trial, the new
 403 trajectory added to the buffer is constructed by concatenating the actions, observations, rewards,
 404 and the final outcome (success or failure). As each episode typically yields only a few rewards, we
 405 include only those. At the start of each trial, S_0 is constructed by concatenating the task description
 406 S_{task} , the collected trajectories, and then the instruction s_{ICRL} . An example of S_0 is provided in
 407 App.A.

408 **Results.** The mean return at each trial, is presented in Figure 2 Right. Steady improvements are ob-
 409 served for methods that make use of some form of history of interactions similar to ICRL prompting.
 410 However, ICRL prompting outperforms baseline methods by about 20% after enough iterations. To
 411 make the comparison fair for efficient baselines such as Best-of-N, in App. B, we compare baselines
 412 as we scale test-time compute budget and observe that ICRL also scales better than the baselines not
 413 only in terms of number of trials but also the test-time compute budget (in dollar amounts).

414 **6 ANALYSIS**
 415

416 **Ablation Study.** To better understand ICRL prompting, we consider following ablations. **(1)**
 417 Zero Rewards: We set all rewards to 0. **(2)** Short Context: In Algorithm 1, the buffer \mathcal{B} is
 418 essentially a queue of infinite length. Instead, we now make it a deque of length 3. In other
 419 words, only the recent 3 episodes are used in constructing S_0 . **(3)** Exploration Only: We simply
 420 ask the LLM to provide a different response than the ones in context, using the exploration
 421 instruction as s_{ICRL} , without the reward signal. **(4)** Exploitation Only: We always use the exploita-
 422 tion instruction as s_{ICRL} , with the reward signal. **(5)** No ICRL Instruction: We entirely remove s_{ICRL} .
 423

424 The running max results of the ablation study are plotted in Figure 3. Both our two methods and the
 425 exploitation only with reward signals demonstrate the best-performing curves. This demonstrates
 426 our ICRL prompting framework is quite robust to the different prompts setup. We have also ob-
 427 served performance drop with short context and performance drop when the reward is absent. A
 428 key finding is that the “exploration only without reward signal” method (shown in green) performs
 429 significantly worse than our approach when comparing the maximum performance achieved over
 430 time (running max). This demonstrates that our method’s improvement is not just due to exploring
 431 various responses and then picking the best one previously seen as doing a Best-of-N. Instead, ICRL
 432 can genuinely generate novel responses that are better than the ones during the exploration phase.

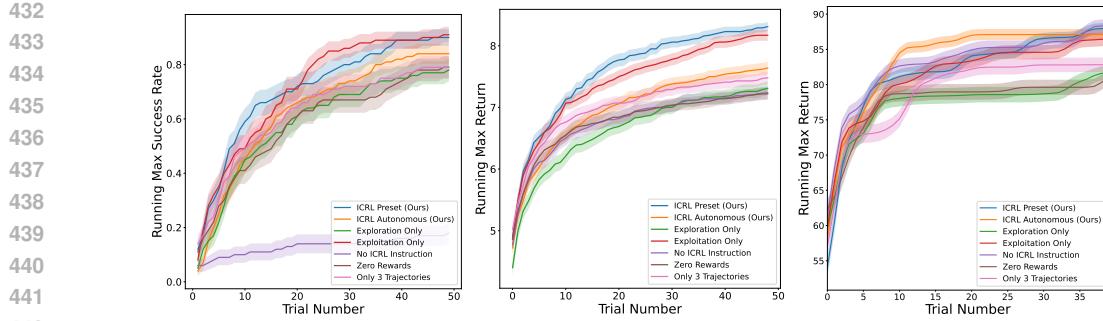


Figure 3: **Ablation Studies (Running Max).** **(Left)** The mean of running max success rate on Game of 24. **(Middle)** The mean of running max coherence reward on creative critiq. **(Right)**. The mean of running max return on ScienceWorld. The shaded region represents ± 1 standard error of the mean (SEM) of the performance calculated across the evaluated tasks within each benchmark.

Table 4: Performance across benchmarks (HMMT, AIME, Creative Writing) for different models and inference-time improvement methods.

Method	Qwen3 32B (32k)			Qwen3 32B think (32k)			Llama 4 Maverick (32k)			Phi-4 (16k)		
	HMMT	AIME	CW	HMMT	AIME	CW	HMMT	AIME	CW	HMMT	AIME	CW
Base	9.14	22.54	34.14	52.00	66.58	1.24	8.50	17.58	0.98	5.55	20.00	10.85
Self-Refine	16.66	43.33	46.00	56.66	83.33	30.23	13.33	20.00	45.52	13.33	33.33	51.98
Reflexion	23.33	33.33	41.17	60.00	70.00	38.33	10.00	23.33	24.96	13.33	40.00	33.30
ICRL	33.33	46.66	50.00	60.00	80.00	50.00	20.00	35.00	50.00	13.33	40.00	50.00

Context Length Analysis. To assess compute efficiency under different input scales, we evaluate ICRL on Qwen3-32B (Yang et al., 2025) across context lengths of 8k, 16k, and 32k (Table 3). The results show that ICRL consistently surpasses Self-Refine and Reflexion in both Creative Writing and AIME, demonstrating superior performance per unit of compute.

Evaluating ICRL on Open-Source Models and Olympiad-level Mathematics. To demonstrate the broad applicability of ICRL, we evaluated its performance across two key dimensions: model architecture and task complexity. First, we tested ICRL on a range of open-source models, including Phi-4 (Abdin et al., 2024), Llama-4 Maverick (AI, 2025), Qwen3-32B, and Qwen3-32B-thinking mode (Yang et al., 2025) on the creative writing task. Second, we applied it to challenging Olympiad-level mathematics (AIME (Mathematical Association of America), HMMT (Harvard-MIT Mathematics Tournament)). For details of the experimental setup, please refer to Appendix B.1. ICRL consistently outperforms baselines like self-refine and Reflexion in all settings. Notably, the performance gains are substantial across the board, with improvements of up to 10–20 points over the base model on both creative and mathematical benchmarks. These results demonstrate that ICRL is a robust capability that exists across diverse models and proves effective in challenging task domains.

Table 3: Performance of Qwen/Qwen3-32B without reasoning on Creative Writing (LC-WR: length-controlled win rate) and AIME (% solved) across different context lengths.

Method	CW (LC-WR)			AIME (% solved)		
	8k	16k	32k	8k	16k	32k
Self-Refine	46.33	45.83	44.41	33.33	43.33	43.33
Reflexion	42.73	40.87	40.72	30.00	30.00	33.33
ICRL	50.00	50.00	50.00	40.00	43.33	46.66

486 **Learning from Rewards vs. Parametric Knowledge Search.** To verify that ICRL truly learns
 487 from external rewards rather than merely searching within the model’s parametric knowledge, we
 488 evaluated it on generating abstracts for arXiv papers published after the model’s training cutoff.
 489 In this setting, where the ground truth is absent from the model’s training data, standard search
 490 methods like Best-of-N and self-correction methods like Reflexion plateau quickly (Figure 16). In
 491 contrast, ICRL continues to improve ROUGE-Recall scores over 200 iterations, demonstrating its
 492 ability to uncover unseen information solely by exploiting the scalar reward signal. Detailed results
 493 are provided in Appendix C.

494 7 CONCLUSION

495 In this paper, we demonstrate that reinforcement learning is an emergent capability of LLMs at
 496 inference time. We show that our minimal, scalar-reward-based ICRL prompting framework unlocks
 497 this ability across diverse models and general-purpose tasks, outperforming self-revision methods. A
 498 key direction in future work is to investigate how training-time interventions might further enhance
 499 this in-context RL capability in LLMs. This surprisingly effective capability points toward a future
 500 of more autonomous agents that can explore, adapt, and self-improve in open-ended settings by
 501 learning from their own experience.

502 REFERENCES

503 Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar,
 504 Michael Harrison, Russell J. Hewett, Mojan Javaheripi, Piero Kauffmann, James R. Lee, Yin Tat
 505 Lee, Yuanzhi Li, Weishung Liu, Caio C. T. Mendes, Anh Nguyen, Eric Price, Gustavo de Rosa,
 506 Olli Saarikivi, Adil Salim, Shital Shah, Xin Wang, Rachel Ward, Yue Wu, Dingli Yu, Cyril
 507 Zhang, and Yi Zhang. Phi-4 technical report, 2024. URL <https://arxiv.org/abs/2412.08905>.

508 Meta AI. The llama 4 herd: The beginning of a new era of natively multimodal ai in-
 509 novation. <https://ai.meta.com/blog/llama-4-multimodal-intelligence/>,
 510 2025. Accessed: 2025-09-25.

511 Jakob Bauer, Kate Baumli, Feryal Behbahani, Avishkar Bhoopchand, Nathalie Bradley-Schmiege,
 512 Michael Chang, Natalie Clay, Adrian Collister, Vibhavari Dasagi, Lucy Gonzalez, et al. Human-
 513 timescale adaptation in an open-ended task space. In *Proceedings of the International Conference
 514 on Machine Learning*, 2023.

515 Jacob Beck, Risto Vuorio, Evan Zheran Liu, Zheng Xiong, Luisa Zintgraf, Chelsea Finn, and Shi-
 516 mon Whiteson. A survey of meta-reinforcement learning. *ArXiv Preprint*, 2023.

517 Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Giani-
 518 nazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoe-
 519 fler. Graph of thoughts: Solving elaborate problems with large language models. *Proceedings of
 520 the AAAI Conference on Artificial Intelligence*, 2024.

521 Ethan Brooks, Logan Walls, Richard L Lewis, and Satinder Singh. Large language models can
 522 implement policy iteration. In *Advances in Neural Information Processing Systems*, 2024.

523 Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal,
 524 Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel
 525 Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler,
 526 Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray,
 527 Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever,
 528 and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information
 529 Processing Systems*, 2020.

530 Murray Campbell, Feng-Hsiung Hsu, and A. Joseph Hoane. Deep blue. *Artificial Intelligence*, 134
 531 (1):57–83, 2002. doi: 10.1016/S0004-3702(01)00129-1.

532 Jonathan Cook, Chris Lu, Edward Hughes, Joel Z Leibo, and Jakob Foerster. Artificial generational
 533 intelligence: Cultural accumulation in reinforcement learning. *ArXiv Preprint*, 2024.

540 Zhenwen Dai, Federico Tomasi, and Sina Ghassian. In-context exploration-exploitation for rein-
 541forcement learning. *ArXiv Preprint*, 2024.

542

543 DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu,
 544 Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu,
 545 Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao
 546 Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan,
 547 Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao,
 548 Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding,
 549 Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang
 550 Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai
 551 Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang,
 552 Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang,
 553 Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang,
 554 Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang,
 555 R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhua Chen, Shengfeng
 556 Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing
 557 Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanja Zhao, Wen
 558 Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong
 559 Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu,
 560 Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xi-
 561 aoshua Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia
 562 Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng
 563 Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong
 564 Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong,
 565 Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou,
 566 Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying
 567 Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda
 568 Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu,
 569 Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu
 570 Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via rein-
 571forcement learning. *ArXiv Preprint*, 2025.

572

573 Ruomeng Ding, Chaoyun Zhang, Lu Wang, Yong Xu, Minghua Ma, Wei Zhang, Si Qin, Saravan Ra-
 574j Mohan, Qingwei Lin, and Dongmei Zhang. Everything of thoughts: Defying the law of penrose
 575 triangle for thought generation. In *Findings of the Association for Computational Linguistics: ACL 2024*, 2024.

576

577 Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl2: fast
 578 reinforcement learning via slow reinforcement learning. *ArXiv Preprint*, 2016.

579

580 Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. Kto: Model
 581 alignment as prospect theoretic optimization. *ArXiv Preprint*, 2024.

582

583 Jake Grigsby, Linxi Fan, and Yuke Zhu. Amago: Scalable in-context reinforcement learning for
 584 adaptive agents. In *Proceedings of the International Conference on Learning Representations*,
 585 2024.

586

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

590

591 Harvard-MIT Mathematics Tournament. Hmmt (harvard-mit mathematics tournament). URL
 592 <https://www.hmmt.org/>.

593

594 Michael Heim. *Exploring Indiana Highways*. 2007.

595

596 Jiwoo Hong, Noah Lee, and James Thorne. ORPO: Monolithic preference optimization without ref-
 597 erence model. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language
 598 Processing*, 2024.

594 Sili Huang, Jifeng Hu, Hechang Chen, Lichao Sun, and Bo Yang. In-context decision transformer:
 595 Reinforcement learning via hierarchical chain-of-thought. *ArXiv Preprint*, 2024.
 596

597 Khurram Javed and Richard S. Sutton. The big world hypothesis and its ramifications for artifi-
 598 cial intelligence. In *Finding the Frame Workshop at the Reinforcement Learning Conference*.
 599 OpenReview, 2024. URL <https://openreview.net/pdf?id=Sv7DazuCn8>. CC BY
 600 4.0.

601 Louis Kirsch, James Harrison, C. Freeman, Jascha Sohl-Dickstein, and Jürgen Schmidhuber. To-
 602 wards general-purpose in-context learning agents. In *NeurIPS Foundation Models for Decision*
 603 *Making Workshop*, 2023.

604

605 Akshay Krishnamurthy, Keegan Harris, Dylan J Foster, Cyril Zhang, and Aleksandrs Slivkins. Can
 606 large language models explore in-context? *ArXiv Preprint*, 2024.

607

608 Michael Laskin, Luyu Wang, Junhyuk Oh, Emilio Parisotto, Stephen Spencer, Richie Steigerwald,
 609 DJ Strouse, Steven Stenberg Hansen, Angelos Filos, Ethan Brooks, maxime gazeau, Himanshu
 610 Sahni, Satinder Singh, and Volodymyr Mnih. In-context reinforcement learning with algorithm
 611 distillation. In *Proceedings of the International Conference on Learning Representations*, 2023.

612

613 Jonathan Lee, Annie Xie, Aldo Pacchiano, Yash Chandak, Chelsea Finn, Ofir Nachum, and Emma
 614 Brunskill. Supervised pretraining can learn in-context reinforcement learning. In *Advances in*
 615 *Neural Information Processing Systems*, 2024.

616

617 Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan
 618 Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *ArXiv Preprint*,
 2023.

619

620 Hao Liu and Pieter Abbeel. Emergent agentic transformer from chain of hindsight experience. In
 621 *Proceedings of the International Conference on Machine Learning*, 2023.

622

623 Max Liu, Chan-Hung Yu, Wei-Hsu Lee, Cheng-Wei Hung, Yen-Chun Chen, and Shao-Hua Sun.
 624 Synthesizing programmatic reinforcement learning policies with large language model guided
 625 search. In *Proceedings of the International Conference on Learning Representations*, 2025.

626

627 Chris Lu, Yannick Schroecker, Albert Gu, Emilio Parisotto, Jakob Foerster, Satinder Singh, and Fer-
 628 yal Behbahani. Structured state space models for in-context reinforcement learning. In *Advances*
 629 *in Neural Information Processing Systems*, 2023.

630

631 Cong Lu, Shengran Hu, and Jeff Clune. Intelligent go-explore: Standing on the shoulders of giant
 632 foundation models. In *Proceedings of the International Conference on Learning Representations*,
 633 2025.

634

635 Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri
 636 Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad
 637 Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine:
 638 Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*,
 639 2023.

640

641 Mathematical Association of America. American invitational mathematics examination (aime).
 642 URL <https://maa.org/maa-invitational-competitions/>.

643

644 Yu Meng, Mengzhou Xia, and Danqi Chen. Simpo: Simple preference optimization with a
 645 reference-free reward. *ArXiv Preprint*, 2024.

646

647 Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Certík, Sergey B. Kirpichev,
 648 Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rath-
 649 nayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam
 650 Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Simon Roucka,
 651 Akshay Saboo, Ishaan Fernando, Shravan Kulal, Robert Cimrman, and Anthony Scopatz. Sympy:
 652 symbolic computing in python. *PeerJ Computer Science*, 2017.

648 Suvir Mirchandani, Fei Xia, Pete Florence, Brian Ichter, Danny Driess, Montserrat Gonzalez Are-
 649 nas, Kanishka Rao, Dorsa Sadigh, and Andy Zeng. Large Language Models as General Pattern
 650 Machines. *ArXiv Preprint*, 2023.

651

652 Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Belle-
 653 mare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen,
 654 Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wier-
 655 stra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning.
 656 *Nature*, 2015.

657

658 Amir Moeini, Jiuqi Wang, Jacob Beck, Ethan Blaser, Shimon Whiteson, Rohan Chandra, and Shang-
 659 tong Zhang. A survey of in-context reinforcement learning. *ArXiv Preprint*, 2025.

660

661 Giovanni Monea, Antoine Bosselut, Kianté Brantley, and Yoav Artzi. Llms are in-context bandit
 662 reinforcement learners. *ArXiv Preprint*, 2025.

663

664 Allen Nie, Yi Su, Bo Chang, Jonathan N. Lee, Ed H. Chi, Quoc V. Le, and Minmin Chen. EVOLvE:
 665 Evaluating and Optimizing LLMs For Exploration. *ArXiv Preprint*, 2024.

666

667 OpenAI. Introducing gpt-4.1 in the api. <https://openai.com/index/gpt-4-1/>, 2025.

668

669 Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong
 670 Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow
 671 instructions with human feedback. In *Advances in Neural Information Processing Systems*, 2022.

672

673 Chanwoo Park, Xiangyu Liu, Asuman Ozdaglar, and Kaiqing Zhang. Do llm agents have regret? a
 674 case study in online learning and games. *ArXiv Preprint*, 2024.

675

676 Sharath Chandra Raparthy, Eric Hambro, Robert Kirk, Mikael Henaff, and Roberta Raileanu. Gen-
 677 eralization to new sequential decision making tasks with in-context learning. *ArXiv Preprint*,
 678 2023.

679

680 Yarik Menchaca Resendiz and Roman Klinger. Parl: Prompt-based agents for reinforcement learn-
 681 ing, 2025. URL <https://arxiv.org/abs/2510.21306>.

682

683 Thomas Schmied, Thomas Adler, Vihang Patil, Maximilian Beck, Korbinian Pöppel, Johannes
 684 Brandstetter, Günter Klambauer, Razvan Pascanu, and Sepp Hochreiter. A Large Recurrent Ac-
 685 tion Model: xLSTM enables Fast Inference for Robotics Tasks. *ArXiv Preprint*, 2024.

686

687 John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy
 688 optimization algorithms. *ArXiv Preprint*, 2017.

689

690 Lucy Xiaoyang Shi, Yunfan Jiang, Jake Grigsby, Linxi Fan, and Yuke Zhu. Cross-episodic curricu-
 691 lum for transformer agents. In *Advances in Neural Information Processing Systems*, 2024.

692

693 Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and
 694 Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in
 695 Neural Information Processing Systems*, 2023.

696

697 David Silver and Richard S. Sutton. Welcome to the era of experience, 2024.

698

699 David Silver and Richard S Sutton. Welcome to the era of experience. *Google AI*, 2025.

700

701 David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driess-
 702 che, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander
 703 Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap,
 704 Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game
 705 of go with deep neural networks and tree search. *Nature*, 2016.

706

707 David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez,
 708 Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan
 709 Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering
 710 the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017. doi: 10.1038/
 711 nature24270.

702 David Silver, Satinder Singh, Doina Precup, and Richard S. Sutton. Reward is enough. *Artificial*
 703 *Intelligence*, 2021.

704

705 Kaya Stechly, Karthik Valmecikam, and Subbarao Kambhampati. On the self-verification limitations
 706 of large language models on reasoning and planning tasks. In *Proceedings of the International*
 707 *Conference on Learning Representations*, 2025.

708 Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Rad-
 709 ford, Dario Amodei, and Paul Christiano. Learning to summarize from human feedback. *ArXiv*
 710 *Preprint*, 2022.

711

712 Yi Su, Dian Yu, Linfeng Song, Juntao Li, Haitao Mi, Zhaopeng Tu, Min Zhang, and Dong Yu.
 713 Crossing the Reward Bridge: Expanding RL with Verifiable Rewards Across Diverse Domains,
 714 April 2025. URL <http://arxiv.org/abs/2503.23829>.

715

716 Richard S. Sutton. The reward hypothesis. <http://incompleteideas.net/rlai.cs.ualberta.ca/RLAI/rewardhypothesis.html>, 2004.

717

718 Richard S. Sutton. The bitter lesson. <http://www.incompleteideas.net/IncIdeas/BitterLesson.html>, 2019.

719

720 Tatsu Lab. alpaca_eval: An evaluation framework for llms. https://github.com/tatsu-lab/alpaca_eval, 2025.

721

722 Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos,
 723 Charles Blundell, Dharrshan Kumaran, and Matt Botvinick. Learning to reinforcement learn.
 724 *ArXiv Preprint*, 2016.

725

726 Jiuqi Wang, Ethan Blaser, Hadi Daneshmand, and Shangtong Zhang. Transformers can learn tempo-
 727 ral difference methods for in-context reinforcement learning. In *Proceedings of the International*
 728 *Conference on Learning Representations*, 2025.

729

730 Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. ScienceWorld: Is
 731 your Agent Smarter than a 5th Grader? *ArXiv Preprint*, 2022.

732

733 Tengye Xu, Zihao Li, and Qinyuan Ren. Meta-reinforcement learning robust to distributional shift
 734 via performing lifelong in-context learning. In *Proceedings of the International Conference on*
 735 *Machine Learning*, 2024.

736

737 An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang
 738 Gao, Chengan Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu,
 739 Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin
 740 Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang,
 741 Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui
 742 Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang
 743 Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger
 744 Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan
 745 Qiu. Qwen3 technical report, 2025. URL <https://arxiv.org/abs/2505.09388>.

746

747 Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun
 748 Chen. Large language models as optimizers. *ArXiv Preprint*, 2024.

749

750 Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik
 751 Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *ArXiv*
 752 *Preprint*, 2023.

753

754 Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin,
 755 and James Zou. Optimizing generative ai by backpropagating language model feedback. *Nature*,
 756 2025.

757

758 Qiyuan Zhang, Fuyuan Lyu, Zexu Sun, Lei Wang, Weixu Zhang, Zhihan Guo, Yufei Wang, Niklas
 759 Muennighoff, Irwin King, Xue Liu, and Chen Ma. What, how, where, and how well? a survey on
 760 test-time scaling in large language models, 2025. URL <https://arxiv.org/abs/2503.24235>.

756 Shimao Zhang, Yu Bao, and Shujian Huang. Edt: Improving large language models' generation by
 757 entropy-based dynamic temperature sampling. *ArXiv Preprint*, 2024.
 758

759 Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang,
 760 Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica.
 761 Judging ILM-as-a-judge with mt-bench and chatbot arena. *ArXiv Preprint*, 2023.
 762

763 Ilya Zisman, Vladislav Kurenkov, Alexander Nikulin, Viacheslav Sinii, and Sergey Kolesnikov.
 764 Emergence of in-context reinforcement learning from noise distillation. *ArXiv Preprint*, 2023.
 765

766 A PROMPT EXAMPLES

767
 768 Instruction: Examine all the <attempt>...</attempt> examples, each showing a candidate Response,
 769 and the Rewards for each step of the Response. Provide a response that is completely different for any
 770 steps from every single one of the previous attempts demonstrated in the context.
 771

772
 773
 774
 775 Figure 4: The Exploration Instruction (s_{ICRL}).
 776

777
 778 Instruction: You will be given multiple <attempt>...</attempt> examples, each showing a candidate
 779 Response, and the Rewards for each step of the Response. Your task: Based on the previous attempts,
 780 try your best to produce a response that can achieve higher rewards.
 781

782
 783
 784 Figure 5: The Exploitation Instruction (s_{ICRL}).
 785

786
 787 Instruction: Examine all the <attempt>...</attempt> examples, each showing a candidate Response
 788 and its Reward. You have two options: exploration or exploitation.
 789

790 For exploration, provide a response that is completely different for any steps from every single
 791 one of the previous attempts demonstrated in the context, while making sure it correctly follows the
 792 task instruction.

793 For exploitation, based on the previous attempts, try your best to produce a response that can
 794 achieve higher rewards.

795 Pick one option to follow.
 796

797
 798 Figure 6: The Exploration or Exploitation Instruction (s_{ICRL}).
 799

800
 801
 802 Prompt: Write a coherent passage of 4 short paragraphs. The end sentence of each paragraph must
 803 be: For some unfathomable reason, the response team didn't consider a lack of milk for my cereal as a
 804 proper emergency. You realize you're not alone as you sit in your bedroom massaging your calves after
 805 a long day of playing tug-of-war with Grandpa Joe in the hospital. He poured rocks in the dungeon of
 806 his mind. I'm a living furnace. Make a plan then write. Your output should be of the following format:
 807 Plan: Your plan here. Passage: Your passage here.
 808

809 Figure 8: An example of s_{task} for creative writing.

```

810
811
812
813
814
815
816
817 <attempt>
818   Input: 4 4 6 8
819   Step1:  $4 + 8 = 12$  (left: 4 6 12)
820   Step2:  $6 - 4 = 2$  (left: 2 12)
821   Step3:  $2 * 12 = 24$  (left: 24)
822   Answer:  $(6 - 4) * (4 + 8) = 24$ 
823 </attempt>
824
825 <attempt>
826   Input: 2 9 10 12
827   Step1:  $12 * 2 = 24$  (left: 9 10 24)
828   Step2:  $10 - 9 = 1$  (left: 1 24)
829   Step3:  $24 * 1 = 24$  (left: 24)
830   Answer:  $(12 * 2) * (10 - 9) = 24$ 
831 </attempt>
832
833 <attempt>
834   Input: 4 9 10 13
835   Step1:  $13 - 10 = 3$  (left: 3 4 9)
836   Step2:  $9 - 3 = 6$  (left: 4 6)
837   Step3:  $4 * 6 = 24$  (left: 24)
838   Answer:  $4 * (9 - (13 - 10)) = 24$ 
839 </attempt>
840
841 <attempt>
842   Input: 1 4 8 8
843   Step1:  $8 / 4 = 2$  (left: 1 2 8)
844   Step2:  $1 + 2 = 3$  (left: 3 8)
845   Step3:  $3 * 8 = 24$  (left: 24)
846   Answer:  $(1 + 8 / 4) * 8 = 24$ 
847 </attempt>
848
849 <attempt>
850   Input: 5 5 5 9
851   Step1:  $5 + 5 = 10$  (left: 5 9 10)
852   Step2:  $10 + 5 = 15$  (left: 9 15)
853   Step3:  $15 + 9 = 24$  (left: 24)
854   Answer:  $((5 + 5) + 5) + 9 = 24$ 
855 </attempt>
856
857   **Task**: Use numbers and basic arithmetic operations (+ - * /) to obtain 24. Put your answer in this format '<answer>**Response** Step1: ... (left: ...) Step2: ... (left: ...) Step3: ... (left: ...) **Answer**: <math operations of the 4 input numbers, even if it does not equal 24></answer>'. Whether it is correct or not, do not try again.
858   **Prompt**: Input: 1 8 10 11

```

Figure 7: An example of s_{task} for Game of 24 with few-shot CoT prompting.

859
860
861
862
863

```

864
865 You are a helpful assistant to do some scientific experiment in an environment.
866 <Environment description> # $task
867 In the environment, there are several rooms: kitchen, foundry, workshop, bathroom, outside, living
868 room, bedroom, greenhouse, art studio, hallway
869 You should explore the environment and find the items you need to complete the experiment.
870
871 The available actions are: Available Actions
872
873 FOCUS is a extremely critical action that can be only used the number of times 'focus' is mentioned
874 in the task description. Using it more than that or inappropriately (such as on a wrong object) will
875 terminate the session and the task WILL FAIL.
876
877 Task Description:
878 Your task is to change the state of matter of water. First, focus on the substance. Then, take actions
879 that will cause it to change its state of matter.
880 </Environment description>
881
882 <Instruction> $ICRL </Instruction>
883 <Attempts> # Buffer
884 :
885
886 Attempt n:
887 teleport to bathroom -> Observation: You teleport to the bathroom. (reward=3)
888 -> focus on substance in toilet -> Observation: You focus on the water. (reward=66)
889 -> use cup on substance in toilet -> Observation: I'm not sure how to do that. (reward=0)
890 -> activate sink -> Observation: The sink is now activated. (reward=0)
891 -> dunk cup into sink -> Observation: No known action matches that input. (reward=0)
892 -> move cup to sink -> Observation: You move the glass cup to the sink. (reward=0)
893 -> teleport to kitchen -> Observation: You teleport to the kitchen. (reward=0)
894 -> use cup on stove -> Observation: I'm not sure how to do that. (reward=0)
895 -> activate stove -> Observation: The stove is now activated. (reward=2)
896 -> move cup to table -> Observation: You move the glass cup to the chair. (reward=0)
897 -> examine cup -> Observation: a glass cup (containing nothing) (reward=0)
898 Task Failed. You have exceeded the maximum number of steps. (reward=0) Total reward: 71
899
900 Attempt n + 1:
901 teleport to bathroom -> Observation: You teleport to the bathroom. (reward=3)
902 -> focus on substance in toilet -> Observation: You focus on the water. (reward=66)
903 -> pick up cup -> Observation: You move the glass cup to the inventory. (reward=0)
904 -> move cup to sink -> Observation: You move the glass cup to the sink. (reward=0)
905 -> activate sink -> Observation: The sink is now activated. (reward=0)
906 -> dunk cup into sink -> Observation: No known action matches that input. (reward=0)
907 -> pour substance in cup into sink -> Observation: You pour the water into the sink. (reward=0)
908 -> activate stove -> Your generated action "activate stove" cannot be matched to a valid action. (re-
909 ward=0)
910 -> teleport to kitchen -> Observation: You teleport to the kitchen. (reward=0)
911 -> pick up cup containing nothing in table -> Observation: You move the glass cup to the inventory.
912 (reward=0)
913 -> activate sink -> Observation: The sink is now activated. (reward=0)
914 Task Failed. You have exceeded the maximum number of steps. (reward=0) Total reward: 69
915
916
917

```

Figure 9: An example of S_0 from ScienceWorld.

```

918
919 # Buffer:
920 <attempt>
921 Input: 4 9 10 13.
922 Response:
923 Step1: 10 - 4 = 6 (left: 6 9 13) <Reward: 3.00>
924 Step2: 13 - 6 = 7 (left: 7 9) <Reward: 0.00>
925 Step3: 9 * 7 = 63 (left: 63) <Reward: 0.00>
926 *Answer**: (13 - (10 - 4)) * 9 = 63 <Reward: 3.00>
927 </attempt>
928
929 <attempt>
930 Input: 4 9 10 13.
931 Response:
932 Step1: 10 + 4 = 14 (left: 9 13 14) <Reward: 0.00>
933 Step2: 14 + 9 = 23 (left: 13 23) <Reward: 0.00>
934 Step3: 23 + 13 = 36 (left: 36) <Reward: 0.00>
935 *Answer**: (10 + 4 + 9) + 13 = 36 <Reward: 0.00>
936 </attempt>
937
938 <attempt>
939 Input: 4 9 10 13.
940 Response:
941 Step1: 9 + 10 = 19 (left: 4 13 19) <Reward: 3.00>
942 Step2: 19 - 13 = 6 (left: 4 6) <Reward: 3.00>
943 Step3: 6 + 4 = 10 (left: 10) <Reward: 0.00>
944 *Answer**: ((9 + 10) - 13) + 4 = 10 <Reward: 6.00>
945 </attempt>
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971

```

Figure 10: An example of S_0 from Game of 24.

Instruction: You are a seasoned text coherence evaluator. Read the TEXT below and rate its overall coherence on a scale from 1 to 10, where 1 means significantly less coherent than the Base Answer, 5 means equally coherent, and 10 means significantly more coherent. Be a strict and conservative evaluator-only assign high scores when the TEXT is clearly better than the Base Answer.

Base Answer:

{At dawn, golden light slips past pale curtains, rousing the world in quiet celebration. A lone robin greets the morning with a clear, cheerful trill, its song drifting across dew-laden grass. A gentle breeze stirs the leaves, carrying the fresh, earthy scent of new growth. Nearby, rooftops and empty streets lie poised between night's calm and the city's stirring pulse, promising simple comforts like a warm cup of coffee. In this tranquil pause, one senses life's renewal and the gentle invitation to greet the day with hope and gratitude.}

TEXT: { model_answer }

Return your answer in exactly this format: Coherency score: <integer 1-10>.

Response:

Figure 11: Prompt for Pairwise Coherence Evaluation for Reward Model r .

972

973

974

975

976

977

978

979

980

981

982

983

984

985

986

987

988

989

990

991

992

993

994

995

996

997

998

999

1000

1001

1002

1003

1004

1005

1006

1007

1008

1009

1010

1011

1012

1013

1014

1015

1016

1017

1018

1019

1020

1021

1022

1023

1024

1025

Rule of the Game of 24: Use all four numbers provided in the input, without repetition, and only basic arithmetic operations ($+, -, \times, \div$) to obtain 24. Only three steps are allowed.

Given the following two remaining numbers from a previous step in the Game of 24, the current step is: {step}. Evaluate this step.

Examine the numbers shown in each “left: ...” after the step and reason whether it is still possible to reach 24: • **Sure** → 3 • **Likely** → 1 • **Impossible** → 0

Return the score in the following format: ****Answer****: <integer score>

Response:

Figure 12: Prompt for single-step evaluation used in the reward model r for Game of 24.

B ADDITIONAL EXPERIMENTAL RESULTS

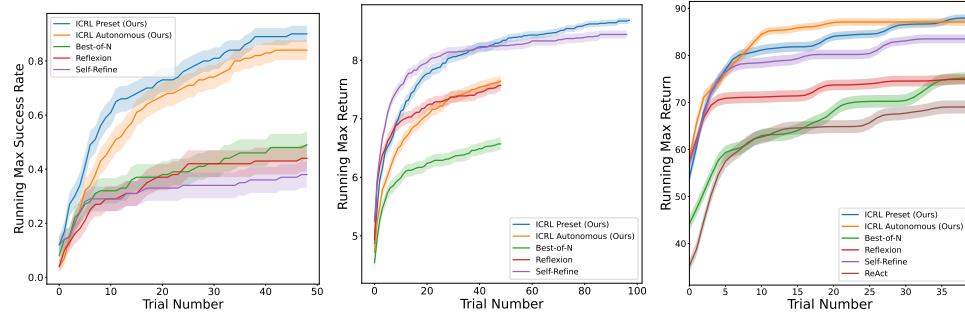


Figure 13: **Benchmark results: Mean of Running Max.** **(Left)** The mean of running max success rate on Game of 24. **(Middle)** The mean of running max coherence reward on creative writing. Both ICRL Preset and Self-Refine went through an additional run of 50 episodes. **(Right)**. The mean of running max return on ScienceWorld.

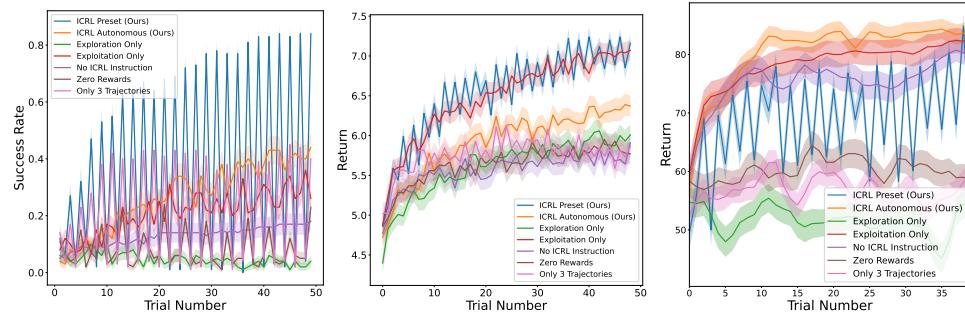


Figure 14: **Ablation study results: Original Curves.** **(Left)** The mean of success rate on Game of 24 ablation studies. **(Middle)** The mean of coherence reward on creative writing ablation studies. **(Right)**. The mean return on ScienceWorld ablation studies.

Table 5: Running max of return averaged over all the tasks in **ScienceWorld**.

Method	Return (max = 100)
ReAct	69 ± 1.4
Reflexion	74 ± 1.1
Best-of-N	75 ± 1.2
Self-Refine	83 ± 0.9
ICRL Preset (Ours)	88 ± 0.7
ICRL Autonomous (Ours)	87 ± 0.8

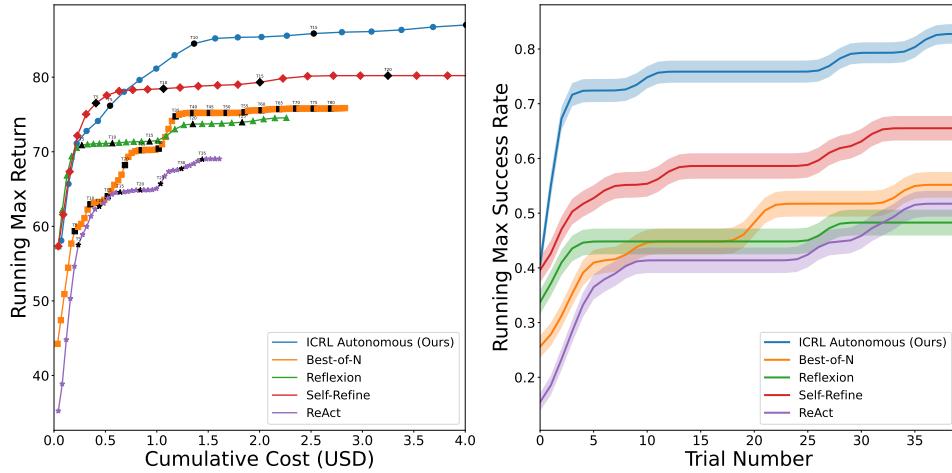


Figure 15: **Additional ScienceWorld Results.** **(Left)** Although ICRL’s context (comprising the experience buffer) is longer than that of random sampling methods, it still outperforms them and other experience-based approaches given additional compute budget. **(Right)** ICRL’s superior return improvement as seen in other results, also leads to a greater increase in success rate.

B.1 ADDITIONAL BENCHMARK: MATH COMPETITIONS

Task Setup. AIME (Mathematical Association of America) and HMMT (Harvard–MIT Mathematics Tournament) are two of the most difficult mathematics competitions in the U.S. for high school students. They require not only strong mathematical knowledge but also advanced problem-solving skills. Training language models for reasoning has been one of the most effective methods for improving performance on these benchmarks (DeepSeek-AI et al., 2025).

We aim to test whether reward signals derived from previous reasoning traces can improve subsequent reasoning attempts. To access these traces, we rely on open-source models. In addition to evaluating reasoning models on these benchmarks, we also include a broader set of open-source models to demonstrate the prevalence of ICRL capability across a wide spectrum.

In this benchmark, the model is given a single question and asked to produce an answer. The only additional instruction concerns formatting: the model must place its final answer within `<answer>` `</answer>` tags. The dataset includes the ground-truth answer for each question.

For the reward model, we use the model proposed by Su et al. (2025), which provides a denser signal than simply checking with SymPy whether the model’s output matches the ground truth. Using such a model aligns with our belief that the growing ecosystem of judge and reward models, primarily designed for reinforcement fine-tuning of language models (Lightman et al., 2023), will also play a key role in enabling ICRL as a test-time scaling method.

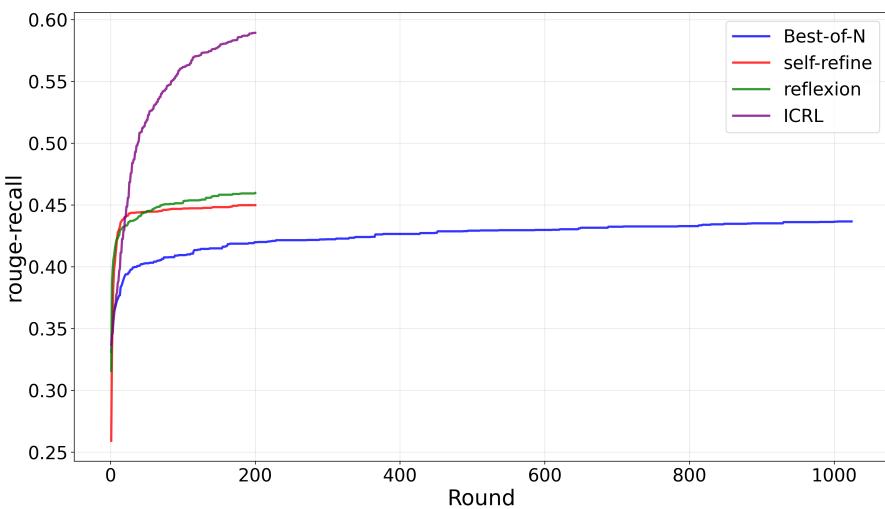
Evaluation. To verify correctness, we parse the model’s output and check whether its answer matches the ground truth answer provided in the datasets.

1080
1081
1082
1083
Baselines. We compare our method with Reflexion and Self-Refine, the two strongest baselines
from the other benchmarks. Their implementations are consistent with those used in the prior ex-
periments.

1084
1085
1086
1087
ICRL Setup. For attempts after the first, we insert the model’s previous answer (including reasoning
or CoT tokens) into the context, along with its associated reward. We then prompt the model either
to try a new method or to refine its best previous approaches. We truncate past answers to ensure
that at least 32 prior attempts can fit in the context.

1088
1089
1090
1091
1092
Results. Our method outperforms the baselines in almost all cases. For the reasoning mode of
Qwen3-32B, our approach remains competitive with Self-Refine on AIME and surpasses it on
HMMT, which is the harder benchmark. Notably, this is achieved using scalar rewards computed
with three orders of magnitude less computation, while Reflexion and Self-Refine are allowed nearly
twice the compute budget for generating long-CoT verbal feedback.

1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133

1134 C ADDITIONAL ANALYSIS RESULTS
11351136 C.1 UNSEEN PAPER ABSTRACT GENERATION
11371138 To isolate whether ICRL truly learns from external rewards rather than merely searching or selecting
1139 from the model’s parametric knowledge, we design a task where search within parametric knowledge
1140 alone is expected to be ineffective.1141 **Setup.** We fetch 30 arXiv papers published after GPT-4.1-mini’s training cutoff. Provided only
1142 the title of the papers, the model is instructed to generate the abstract. The goal is to uncover the
1143 unseen expert-written abstract as much as possible. We measure performance using ROUGE-recall
1144 between the model’s generated abstract and the ground-truth abstract, and we use the same score as
1145 the external reward. Because these abstracts lie outside the model’s pre-training distribution, success
1146 requires learning from external rewards rather than searching through memorized content.1147 **Results.** As shown in Fig. 16, Best-of-1024 sampling reaches only 0.44 ROUGE-recall, indicating
1148 that direct search over the model’s base distribution cannot recover the missing content. Self-Refine
1149 plateaus within a few rounds to 0.45 because it does not use external reward. Reflexion performs
1150 slightly better at 0.46 but largely mirrors Self-Refine, suggesting that its revisions are dominated by
1151 the model’s self-verbal feedback, which carries little useful information in this setup. In contrast,
1152 ICRL continues to improve over 200 iterations and achieves substantially higher ROUGE-recall at
1153 0.59, demonstrating that it can effectively learn from the external reward signal and is not limited by
1154 the model’s pre-training knowledge.1172 Figure 16: ROUGE-Recall on Generating Unseen Paper Abstracts
1173