CTRL: Continual Test-Time Reinforcement Learning for Large Language Models

Chu Zhao
Shenzhen Campus of Sun Yat-sen University
chuzhao007@gmail.com

Enneng Yang*
Shenzhen Campus of Sun Yat-sen University
ennengyang@gmail.com

Abstract

Test-time reinforcement learning (TTRL) generates pseudo-labels via majority voting over multiple sampled trajectories, constructing pseudo-supervision signals for online adaptation of Large Language Models (LLMs). However, existing research predominantly focuses on single-task settings, overlooking continuous updates across task streams. Theoretical and empirical analyses reveal two coupled challenges in this setting: (1) Error Accumulation: Majority voting tends to treat erroneous consensus as pseudo-labels, with such bias amplifying across sequential parameter updates; (2) Catastrophic Forgetting: Gradients from new tasks conflict with historical knowledge, causing acquired reasoning patterns to be overwritten. These two challenges form a mutually reinforcing degradation loop: pseudo-label bias intensifies gradient interference, while catastrophic forgetting weakens reasoning ability, further reducing rollout quality and amplifying pseudo-label noise. As a result, signal reliability and overall performance steadily decline during continual adaptation. To address this, we propose the Continual Test-time Reinforcement Learning (CTRL) framework: first, we introduce a Process Reward Model (PRM) to refine trajectory selection beyond sole reliance on majority voting, filtering high-confidence trajectories via fine-grained process scoring to suppress pseudo-label bias; second, we design a cognitive anchor-driven soft gradient correction module that dynamically recomputes anchor gradients from a historical trajectory memory pool and projects out update components negatively correlated with historical knowledge, constraining the parameter update trajectory. Experiments demonstrate that CTRL consistently outperforms existing TTRL baselines on continuous task streams, exhibiting stronger robustness in blocking error propagation and preserving historical reasoning patterns. The code is available at https://github.com/user683/CTRL.

1 Introduction

TTRL [45] constructs pseudo-supervisory signals by sampling multiple candidate trajectories during inference and aggregating them via majority voting, self-consistency verification, or lightweight reward models, enabling LLMs to perform online policy updates without additional annotations for instant adaptive optimization. To improve sampling efficiency and signal quality, researchers have proposed various enhancement strategies: for instance, [18] adopts entropy-based tree rollouts to replace chain-style generation, mitigating path dependency and exploration bias; [27, 28, 6] dynamically synthesize samples of varying difficulties through question paraphrasing or curriculum learning, expanding the exploration space and optimizing reward distribution; [16] introduces external tools to formalize the reasoning process as executable programs, enabling more precise automated reward estimation. However, these works predominantly focus on instant optimization for isolated instances or static distributions.

*corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

We extend this to the Continual TTRL setting [2]: the LLMs must handle a continuous stream of arriving tasks, retaining its acquired reasoning capabilities while adapting to new tasks via test-time updates. This setting introduces two mutually coupled core challenges: error accumulation in pseudo-reward estimation, and catastrophic forgetting induced by sequential parameter updates.

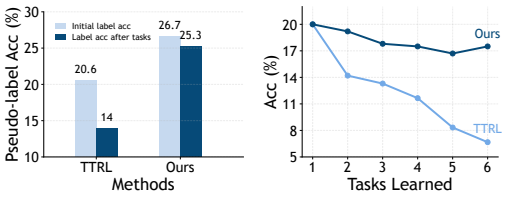


Figure 1: Empirical comparison of pseudo-label reliability and continual performance. **(Left)** Pseudo-label accuracy before and after the task stream. **(Right)** Model accuracy across sequential tasks.

Error Accumulation. Mainstream TTRL relies on majority voting to construct pseudo-labels, which inherently collapses the trajectory distribution into a single consensus answer. When the majority vote deviates from the ground truth, incorrect trajectories are assigned positive rewards while correct but minority trajectories are suppressed. In single-task scenarios, this introduces only bounded estimation bias; however, in a continuous task stream, such biased updates induce progressive parameter drift, degrading the trajectory sampling quality for subsequent tasks. Noisy signals are continuously propagated and amplified across sequential updates, leading to systematic error accumulation. **Catastrophic Forgetting.** In a continuous task stream, sequential parameter updates for new tasks inevitably introduce gradient interference [24]. Even when current pseudo-labels are accurate, the optimization direction for new tasks may overwrite critical weights relied upon by historical tasks, leading to the gradual erosion of acquired reasoning patterns. Empirical analysis (Figure 1, left) reveals that baseline TTRL directly impairs the model's capacity to construct high-quality pseudo-labels during continual adaptation: after completing the full task stream, its pseudo-label accuracy drops sharply from an initial 20.6 to 14. The performance evolution curve (Figure 1, right) further corroborates this degradation trend: as the task sequence progresses, TTRL's accuracy exhibits monotonic decay, continuously declining from 20 on the first task to 6.7 on the sixth task. These phenomena uncover the intrinsic coupling mechanism between error accumulation and catastrophic forgetting: biased pseudo-labels exacerbate cross-task gradient interference, while the degradation of historical representations reduces the sampling quality of subsequent rollouts, rendering pseudo-signal estimation increasingly unstable. Together, they form a mutually reinforcing vicious cycle in Continual TTRL, jointly driving the model into synchronous deterioration of both signal reliability and reasoning capability during continual adaptation.

To address these challenges, we propose the CTRL framework, which achieves stable online adaptation for LLMs under continuous task streams by decoupling pseudo-signal construction from parameter update directions. Specifically, it comprises two core components: (1) PRM-Guided Trajectory Selection: Rather than relying exclusively on majority voting, it introduces a PRM [44, 37] to perform step-by-step fine-grained evaluation. By hierarchically combining answer-level consensus with process-level quality, it identifies a high-confidence golden trajectory as the self-supervised signal, thereby suppressing error accumulation at its source. (2) History Anchor-based Gradient Correction: It maintains a memory pool of historical high-confidence trajectories and recomputes the gradients of historical anchors in the current parameter space prior to each update. Employing Soft Gradient Surgery, it removes conflict components in the current update that are negatively correlated with historical gradients.

The **main contributions** of this paper are threefold: ❶ ***Insight.*** We formulate Continual TTRL, where LLMs continuously adapt to a stream of unlabeled tasks without parameter reset. We reveal a distinctive degradation loop in this setting: majority-voted pseudo-label errors induce biased gradients that accumulate across updates, while catastrophic forgetting weakens future rollout quality and further amplifies pseudo-label noise. Our theory formalizes this coupling by showing that pseudo-label bias causes task-length-dependent parameter drift and can enlarge historical-task loss growth. ❷ ***Method.*** We propose CTRL, a mechanism-driven framework that attacks this loop from two sides. PRM-based trajectory selection refines majority-voted candidates through process-level quality assessment, reducing pseudo-label-induced bias at the supervision source. Cognitive anchor-driven gradient correction suppresses historically destructive update components, mitigating cross-task interference during continual optimization. ❸ ***Evaluation.*** Extensive experiments show that CTRL achieves more stable long-horizon test-time adaptation, improves pseudo-label reliability, and alleviates forgetting, with ablations verifying the necessity of both signal purification and history-aware gradient correction.

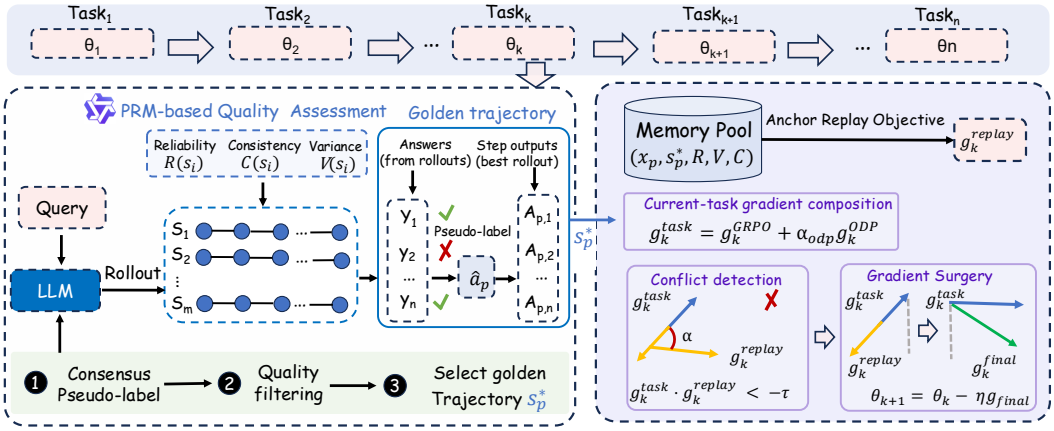


Figure 2: Overview of our proposed framework of CTRL.

2 Preliminaries

2.1 Group Relative Policy Optimization

TTRL employs Group Relative Policy Optimization (GRPO) [8] for parameter updates. GRPO eliminates the need for a separate critic network by constructing an internal baseline from a group of candidate outputs sampled for the same question, thereby significantly reducing memory overhead. Specifically, for a given question q, GRPO samples a group of outputs $\{o_i\}_{i=1}^{G}$ from the old policy $\pi_{\theta_{\text{old}}}$ and optimizes the current policy π_θ by maximizing the following objective:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{q\sim P(Q),\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|q)} \left[\frac{1}{G}\sum_{i=1}^{G}\left(\min\left(\rho_i(\theta)A_i,\ \text{clip}\left(\rho_i(\theta), 1-\epsilon, 1+\epsilon\right)A_i\right) - \beta\mathbb{D}_{\text{KL}}\left(\pi_\theta(\cdot|q)\|\pi_{\text{ref}}(\cdot|q)\right)\right)\right], \tag{1}$$

where $\rho_i(\theta) = \frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)}$ is the importance sampling ratio. The advantage A_i is computed by normalizing the rewards $\{r_i\}_{i=1}^{G}$ within the group:$A_i = \frac{r_i - \text{mean}(\{r_1,\ldots,r_G\})}{\text{std}(\{r_1,\ldots,r_G\})}$. To ensure stability during long-chain reasoning, the KL divergence penalty uses an unbiased estimator: $\mathbb{D}_{\text{KL}}\left(\pi_\theta\|\pi_{\text{ref}}\right) = \frac{\pi_{\text{ref}}(o_i|q)}{\pi_\theta(o_i|q)} - \log\frac{\pi_{\text{ref}}(o_i|q)}{\pi_\theta(o_i|q)} - 1$, where π_{ref} is the reference policy, and ϵ, β are hyperparameters.

2.2 Continual TTRL Formulation

We consider a sequence of unlabeled tasks $\{T_k\}_{k=1}^{K}$, where each task T_k follows an input distribution D_k. At task k, the current policy π_{θ_k} samples N reasoning trajectories $Y(x) = \{y_i\}_{i=1}^{N} \sim \pi_{\theta_k}(\cdot \mid x)$ for each input $x \sim D_k$. Traditional TTRL constructs a pseudo-label by majority voting over rollout answers, $\hat{a}_k(x) = \text{Maj}(\{\text{ans}(y_i)\}_{i=1}^{N})$, and assigns the pseudo-reward $\hat{r}_k(x,y) = \mathbb{I}[\text{ans}(y) = \hat{a}_k(x)]$. For analysis, let $a_k^\star(x)$ be the ground-truth answer and $r_k^\star(x,y) = \mathbb{I}[\text{ans}(y) = a_k^\star(x)]$ be the ideal reward. We define $g_k^\star$ and $\hat{g}_k$ as the gradients induced by $r_k^\star$ and $\hat{r}_k$, respectively, and define the pseudo-label-induced gradient bias as $e_k = \hat{g}_k - g_k^\star$.

3 Methodology

3.1 Theoretical Analysis

We theoretically characterize how pseudo-label error accumulation and catastrophic forgetting form a mutually reinforcing degradation loop in continual TTRL. Unlike single-stage settings, sequential updates over unlabeled task streams cause early voting biases to propagate recursively. We formalize this via two theorems: (1) biased pseudo-rewards induce task-length-dependent parameter drift, and (2) this bias amplifies catastrophic forgetting by exacerbating cross-task gradient conflicts. Full proofs are provided in Appendix D.1 and Appendix D.2.

Theorem 1 (Error Accumulation under Biased Pseudo-Labels) *Let θ_{K+1} and $\theta^\star_{K+1}$ denote the parameters after K continual updates using majority-voted pseudo-rewards and ideal ground-truth rewards, respectively. For each task T_k, let $\hat{g}_k$ and $g^\star_k$ be the corresponding pseudo-reward and ideal gradients, and define $e_k := \hat{g}_k - g^\star_k$. Then $\theta_{K+1} - \theta^\star_{K+1} = -\eta \sum_{k=1}^{K} e_k$, $\left\|\theta_{K+1} - \theta^\star_{K+1}\right\| \leq \eta \sum_{k=1}^{K} \|e_k\|$. Moreover, if $\left\| \sum_{k=1}^{K} \mathbb{E}[e_k] \right\| \geq cK\epsilon$ for constants $c, \epsilon > 0$, then*

$$\mathbb{E}\left[\left\|\theta_{K+1} - \theta^\star_{K+1}\right\|\right] \geq \eta c K \epsilon. \tag{2}$$

Thus, under persistent alignment, systematic pseudo-label bias can induce parameter drift that grows linearly with the task sequence length.

Theorem 1 indicates that majority-voting errors are not merely isolated label noise. Once they induce a persistent gradient bias, the resulting deviation is recursively accumulated across tasks. We next show that such biased updates can also directly increase the risk of forgetting historical tasks.

Theorem 2 (Forgetting Amplification by Pseudo-Label Bias) *Let $L_j(\theta)$ be the loss of a historical task T_j ($j < k$), and assume that L_j is β-smooth. Let $g_j = \nabla_\theta L_j(\theta_k)$. After a single update step*

$$\theta_{k+1} = \theta_k - \eta(g^\star_k + e_k), \tag{3}$$

where $g^\star_k$ is the ideal current-task gradient and e_k is the pseudo-label-induced gradient bias, the change in historical loss satisfies

$$L_j(\theta_{k+1}) - L_j(\theta_k) \leq -\eta g_j^\top g^\star_k \underbrace{-\eta g_j^\top e_k}_{\textit{bias amplification}} + \frac{\beta\eta^2}{2}\|g^\star_k + e_k\|^2. \tag{4}$$

When the pseudo-label bias is anti-aligned with the historical gradient, i.e., $g_j^\top e_k < 0$, the bias-amplification term becomes positive, thereby enlarging the upper bound of historical loss growth and increasing the risk of catastrophic forgetting.

Theorem 2 formally couples these failure modes by showing that the bias term e_k driving parameter drift (Theorem 1) simultaneously acts as a destructive interference source for historical knowledge. This linkage reveals two complementary intervention points: purifying e_k at its source tightens the drift bound, while orthogonalizing current updates against historical gradients suppresses bias amplification.

3.2 Model Framework

CTRL models test-time continual learning as an online policy optimization process. Its core algorithmic framework consists of five tightly coupled modules that sequentially perform trajectory quality assessment, supervision signal purification, adaptive gating, knowledge archiving, and multi-objective gradient optimization. The specific mechanisms of each module are detailed below.

3.2.1 Trajectory Generation and PRM-Based Quality Assessment

For each input prompt x_p, the policy model π_θ samples N complete reasoning trajectories in parallel $\{s_{p,i}\}_{i=1}^{N} \sim \pi_\theta(\cdot \mid x_p)$. The Process Reward Model (PRM) first segments each trajectory into discrete reasoning steps based on logical delimiters and assigns a fine-grained score to each step, yielding a step-level score sequence $\{r_{p,i,\tau}\}_{\tau=1}^{T_{p,i}}$. Based on this sequence, we compute three trajectory-level quality metrics to comprehensively characterize reasoning stability:

$$R(s_{p,i}) = \frac{1}{m}\sum_{\ell=1}^{m} r_{p,i,\text{sorted}[\ell]},\ C(s_{p,i}) = \frac{1}{T_{p,i}}\sum_{\tau=1}^{T_{p,i}} r_{p,i,\tau},\ V(s_{p,i}) = \frac{1}{T_{p,i}}\sum_{\tau=1}^{T_{p,i}} \left(r_{p,i,\tau} - C(s_{p,i})\right)^2, \tag{5}$$

where $R(s_{p,i})$ denotes reliability, calculated as the mean of the lowest m step scores, which characterizes the model's robustness in handling weak reasoning links. $C(s_{p,i})$ represents consistency, defined as the average score across all steps to measure the overall coherence of the reasoning process. $V(s_{p,i})$ captures variance, quantifying the fluctuation amplitude between step scores to identify potentially unstable reasoning paths. Finally, the terminal answers $\{a_{p,i}\}$ are automatically extracted from the end of each trajectory via standard regular expressions, providing the foundation for subsequent pseudo-label construction.

3.2.2 Consensus Pseudo-Label Construction and Hierarchical Golden Trajectory Selection

Instead of treating majority voting as the sole supervision source, we leverage it as an answer-consistency constraint and employ the PRM to evaluate reasoning quality. This hierarchical design ensures stable pseudo-label construction when consensus is reliable, while gracefully falling back to process-aware selection when majority signals are noisy or misleading. Without ground-truth supervision, we derive a consensus pseudo-label $\hat{a}_p = \arg\max_a \text{count}(a \in \{a_{p,i}\})$ for prompt p via majority voting, and define its consensus strength as the proportion of rollouts supporting $\hat{a}_p$. Since the majority answer does not necessarily correspond to the most informative reasoning path, we further identify a *golden trajectory* s_p^* to provide a more reliable supervision target. To this end, we score each trajectory by jointly considering reward quality, confidence, and reasoning length:

$$\text{Score}(s) = \lambda_r R(s) + \lambda_c C(s) - \lambda_l \ln |s|, \tag{6}$$

where the length penalty discourages unnecessarily verbose trajectories. We then keep only trajectories that satisfy basic quality requirements, namely $R(s) \geq \gamma_R$, $C(s) \geq \gamma_C$, and $a(s) \neq \varnothing$.

The golden trajectory is selected through a three-stage hierarchy. We first search among trajectories whose answers agree with the consensus pseudo-label and choose the highest-scoring one if any qualified candidate exists. If none survives, we relax the answer-consistency constraint and select the highest-scoring trajectory from all qualified rollouts. If no trajectory passes the quality filter, we fall back to the one with the highest consistency score $C(s)$. This design prefers trajectories that are simultaneously answer-consistent and high-quality, while remaining robust when the majority signal is weak or noisy. After obtaining s_p^*, we extract its final answer $a_p^* = \text{answer}(s_p^*)$ and assign binary rewards to all answer-matching rollouts:

$$r(s_{p,i}) = \mathbb{I}[a_{p,i} = a_p^* \wedge a_p^* \neq \varnothing]. \tag{7}$$

Following the sparse-reward setting in sequence generation, the reward is applied only to the last valid token of each trajectory. In this way, the golden trajectory serves as a stable supervisory anchor, while all consistent rollouts receive aligned terminal reward signals for policy optimization.

3.2.3 Adaptive Update Gating and Posterior Correction Mechanism

To avoid policy degradation caused by unreliable pseudo-labels, we introduce an adaptive update gating mechanism that filters prompts before policy optimization. Specifically, a prompt x_p is used for updating only when it satisfies both a sufficiently strong answer consensus, $\text{consensus}_p \geq \gamma_{\text{mv}}$, and adequate trajectory quality, $\max_i R(s_{p,i}) \geq \gamma_r^{\text{update}}$. In this way, policy updates are restricted to samples with reliable answer-level agreement and at least one high-quality rollout. Nevertheless, strong consensus does not always imply the presence of a qualified reasoning trajectory. We therefore identify a set of weak-but-recoverable prompts,

$$\mathcal{P}_{\text{weak}} = \{p : \hat{a}_p \neq \varnothing \wedge \text{Filter}(\{s_{p,i} : a_{p,i} = \hat{a}_p\}) = \varnothing\}, \tag{8}$$

where the majority answer is valid but none of its supporting trajectories passes the quality filter. For such cases, rather than discarding them directly, we invoke a posterior correction step.

Specifically, we construct a guided prompt (in Appendix F) conditioned on the current consensus answer $\hat{a}_p$ and resample a corrected trajectory $\tilde{s}_p \sim \pi_\theta(\cdot \mid \tilde{x}_p)$. If the regenerated trajectory satisfies the quality criteria, i.e., $R(\tilde{s}_p) \geq \gamma_R$ and $C(\tilde{s}_p) \geq \gamma_C$, it is accepted as a supplementary golden trajectory. This mechanism improves update reliability by blocking noisy pseudo-labels while recovering useful supervision from ambiguous yet informative samples.

3.2.4 Cognitive Replay Buffer Management

To support knowledge retention during continual learning, purified and verified high-quality golden trajectories are structurally archived into the cognitive replay buffer $\mathcal{M}$. Each record stores a quintuple (x_p, s_p^*, R, V, C) and is dynamically sorted according to a composite priority $P(s_p^*) = \lambda_r R(s_p^*) + \lambda_c C(s_p^*) - \lambda_l \ln |s_p^*|$. The buffer is maintained using a min-heap data structure at the underlying layer, following capacity constraints and a priority-driven strategy: when $|\mathcal{M}| < B$, where B is the buffer capacity and new trajectories are inserted directly; when the buffer is full, replacement occurs only if the new trajectory's priority is higher than the minimum priority in the pool $P_{\min}(\mathcal{M})$. This mechanism ensures that the buffer consistently consists of "highly reliable, low-variance, refined and concise" reasoning paradigms, providing a stable historical knowledge foundation for subsequent anchor sampling and gradient correction.

3.2.5 Conflict-Aware Multi-Objective Optimization

To balance rapid adaptation to the current task with retention of previously acquired knowledge, we jointly optimize three complementary objectives: a GRPO objective for online policy improvement, an output-process distillation (OPD) objective for high-quality trajectory supervision, and an anchor replay objective for historical knowledge preservation. Formally, the overall objective is

$$\mathcal{L} = \mathcal{L}_{\text{GRPO}} + \alpha_{\text{opd}}\mathcal{L}_{\text{OPD}} + \alpha_{\text{rep}}\mathcal{L}_{\text{anchor}}. \tag{9}$$

Since these objectives may induce conflicting updates, we compute their gradients separately and explicitly resolve destructive interference before applying the final parameter update. For current-task optimization, we adopt GRPO-style group-relative advantage estimation. Given the rollout set under prompt x_p, the relative advantage of trajectory $s_{p,i}$ is defined as

$$A_{p,i} = \frac{r(s_{p,i}) - \mu_p}{\sigma_p + \epsilon}, \tag{10}$$

where μ_p and σ_p denote the mean and standard deviation of rewards within the same prompt group. To improve training stability, we further apply symmetric clipping to the normalized advantage. Based on the gated rollout data, we optimize the GRPO objective with KL regularization and denote the resulting gradient by g_{grpo}. To further exploit the supervision signal provided by the selected golden trajectory, we introduce an OPD objective. Instead of imitating the full trajectory uniformly, OPD performs fine-grained supervision only on high-quality reasoning steps identified by a step-level PRM mask. The token-level distillation loss is defined as

$$\mathcal{L}_{\text{OPD}} = \frac{\sum_p \sum_{\tau=1}^{T_p} w_{p,\tau}^{\text{gold}} \big(-\log \pi_\theta(s_{p,\tau}^* \mid x_p, s_{p,<\tau}^*) \big)}{\sum_p \sum_{\tau=1}^{T_p} w_{p,\tau}^{\text{gold}}}, \tag{11}$$

where $w_{p,\tau}^{\text{gold}} = \mathbb{I}[\text{PRM}(s_{p,\tau}^*) \geq \gamma_{\text{step}}]$ retains only reliable reasoning steps. This design encourages the model to absorb informative process-level supervision while avoiding noisy intermediate tokens. We denote the corresponding gradient by g_{opd}. To mitigate forgetting, we additionally replay historical cognitive anchors sampled from the memory buffer $\mathcal{M}$. Rather than treating replayed trajectories equally, we assign each historical sample a reliability-aware weight based on its quality and variance,

$$\omega_i = \frac{\exp(\kappa R_i - \rho V_i)}{\sum_{j=1}^{M} \exp(\kappa R_j - \rho V_j)}, \tag{12}$$

and apply a stricter step-level mask to suppress noisy historical tokens. The resulting anchor replay objective encourages the model to preserve previously validated reasoning patterns while reducing the influence of unreliable memory samples. We denote its gradient by g_{replay}. We combine the current-task adaptation and process-level supervision signals as $g_{\text{task}} = g_{\text{grpo}} + \alpha_{\text{opd}} g_{\text{opd}}$. When g_{task} and g_{replay} are aligned, replay naturally supports current adaptation. However, when their inner product becomes sufficiently negative, naive joint optimization may either overwrite historical knowledge or suppress useful task adaptation. To mitigate such interference, we explicitly detect gradient conflict and project the current-task gradient onto the orthogonal complement of the replay gradient [38]:

$$g_{\text{final}} = \begin{cases} g_{\text{task}} - \lambda_{\text{proj}} \dfrac{g_{\text{task}}^\top g_{\text{replay}}}{\|g_{\text{replay}}\|^2} g_{\text{replay}}, & \text{if } g_{\text{task}}^\top g_{\text{replay}} < -\tau, \\ g_{\text{task}}, & \text{otherwise.} \end{cases} \tag{13}$$

The resulting gradient is then used for parameter updates with standard gradient clipping.

Overall, this design unifies current-task adaptation, process-level supervision, and historical replay within a GRPO-based optimization framework, while using conflict-aware gradient projection to reduce destructive interference between plasticity and stability. This is particularly important in continual TTRL, where pseudo-reward noise and sequential adaptation can otherwise amplify error accumulation and forgetting over time. An overview of the CTRL model framework is shown in Figure 2, while the detailed pseudocode is given in Algorithm 1.

4 Experiment

4.1 Experimental Setup

Datasets. We conduct continual test-time reinforcement learning on language datasets. In detail, beyond standard math and general reasoning tasks, we additionally include medical-domain datasets

Table 1: Final performance comparison of methods across text reasoning benchmarks after the full continual test-time learning sequence. Best and second-best results are **bolded** and underlined.

Methods	Mathematic					General		AVG	FM
	AIME24	AMC	AIME25	MATH	GPQA	MMLU	MedQA		
Qwen3-1.7B-Base	20.0	26.5	13.3	42.7	20.3	41.4	70.2	33.5	-
+ INTUITOR	16.7	38.3	10.0	45.2	22.1	43.8	71.5	35.4	-1.85
+ TTRL	19.3	35.6	12.0	52.3	26.7	48.2	73.8	38.3	-1.02
+ RIF-RFT	20.7	42.1	16.7	58.4	28.3	51.6	75.2	41.9	-0.86
+ HARMONY	20.0	55.8	23.3	68.1	30.5	54.3	76.8	47.0	-0.44
+ PS-GRPO	23.3	62.4	25.0	78.5	33.2	57.1	78.3	51.1	-0.58
+ CTRL	**26.7**	**71.1**	**30.0**	**86.0**	**41.4**	**63.4**	**80.0**	**56.9**	**-0.31**
Δ	3.4	8.7	5.0	7.5	8.2	6.3	1.7	5.8	-
Qwen3-4B-Base	20.0	34.9	23.3	60.2	16.7	50.0	82.0	41.0	-
+ INTUITOR	16.3	38.2	20.0	63.5	28.9	52.4	83.2	43.2	-2.07
+ TTRL	17.2	42.5	20.0	68.1	35.3	56.8	84.5	46.3	-1.72
+ RIF-RFT	20.0	48.3	23.3	72.4	37.8	60.2	85.8	49.7	-0.88
+ HARMONY	21.3	63.6	24.0	76.8	41.2	64.5	87.1	54.1	-1.05
+ PS-GRPO	26.7	71.2	24.7	81.3	45.6	68.9	88.6	58.1	-0.32
+ CTRL	**33.3**	**80.0**	**33.3**	**90.0**	**53.1**	**78.1**	**93.1**	**65.8**	**-0.08**
Δ	6.6	8.8	8.6	8.7	7.5	9.2	4.5	7.7	-
Qwen3-8B-Base	26.9	45.9	33.3	82.3	33.4	82.0	90.2	56.3	—
+ INTUITOR	22.0	48.3	30.0	84.1	35.2	89.2	89.2	56.9	-1.45
+ TTRL	27.8	46.6	32.1	83.0	27.6	83.4	89.8	55.8	-1.00
+ RIF-RFT	26.7	58.6	33.2	85.7	38.9	85.2	90.3	59.8	-0.77
+ HARMONY	30.0	64.2	32.7	87.4	32.7	87.1	90.9	60.7	-0.54
+ PS-GRPO	32.7	71.5	36.7	89.6	45.8	89.3	91.5	65.3	-0.25
+ CTRL	**40.0**	**90.0**	**46.7**	**94.2**	**62.5**	**92.0**	**94.4**	**74.3**	**-0.05**
Δ	7.3	18.5	10.0	4.6	16.7	2.7	2.9	9.0	-
Llama3-8B-Instruct	4.6	23.3	16.7	48.6	30.8	11.1	28.2	23.3	-
+ INTUITOR	3.3	25.1	13.3	49.2	34.1	12.5	29.5	23.9	-3.65
+ TTRL	10.0	32.0	16.7	50.2	32.4	18.2	30.8	27.2	-2.90
+ RIF-RFT	13.3	38.5	18.0	54.6	35.7	20.0	32.1	30.3	-2.20
+ HARMONY	15.3	44.2	20.0	58.3	38.9	21.5	33.6	33.1	-1.45
+ PS-GRPO	16.7	51.9	23.3	62.1	42.3	25.8	35.2	36.8	-1.01
+ CTRL	**20.0**	**55.8**	**28.0**	**71.4**	**50.6**	**31.4**	**41.6**	**42.7**	**-0.75**
Δ	3.3	3.9	4.7	9.3	8.3	5.6	6.4	5.9	-

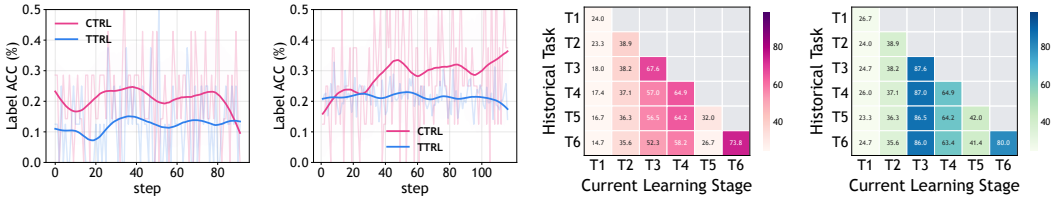
to better evaluate cross-domain continual learning. Specifically, the pure-language datasets include AIME2024, AMC, MMLU, GPQA-Diamond, MedQA, MATH-500, and AIME2025. Detailed descriptions of all datasets are provided in Appendix E.1. **Baselines**. We benchmark our method against several established baselines, including TTRL [45], HARMONY [27], RIF-RFT [14], INTUITOR [42], and PS-GRPO [20]. We adopt Accuracy and Forgetting Measure (FM) as our evaluation metrics. Detailed descriptions of the baseline models and the formal definitions of these metrics are provided in Appendix E.2. The implementation details can be found in Appendix E.4.

4.2 Main Results

Table 1 reports the final performance of each method after completing the full continual test-time learning sequence. CTRL achieves the best performance across all model scales and benchmarks: on the Qwen3 series, compared with the strongest baseline PS-GRPO, CTRL improves average accuracy from 51.1 to 56.9 , from 58.1 to 65.8, and from 65.3 to 74.3, while the FM remains consistently closest to zero (-0.31 to -0.05), indicating better preservation of historical reasoning ability. Task-level analysis reveals that CTRL yields particularly substantial gains on challenging benchmarks such as AMC and GPQA, validating the critical role of PRM-based fine-grained scoring in filtering pseudo-label noise and blocking error accumulation. Experiments on Llama3-8B-Instruct further suggest potential cross-architecture applicability of our method: average accuracy improves

Table 2: Ablation study on core components of CTRL. Results are reported on five representative reasoning benchmarks. FM closer to zero indicates superior knowledge retention.

Methods	AIME24	GPQA	AMC	AIME25	MATH	AVG	FM
Qwen3-1.7B-Base	11.3	20.3	26.5	13.3	42.7	22.8	-
w/o PRM	18.0	35.4	48.6	21.3	68.3	38.4	-1.85
w/o Replay	21.3	45.8	65.2	28.7	81.5	48.6	-0.92
w/o Proj	14.7	22.3	58.9	19.3	74.2	41.8	-0.78
CTRL	**26.7**	**41.4**	**71.1**	**30.0**	**86.0**	**51.0**	**-0.31**
Qwen3-4B-Base	20.0	16.7	34.9	23.3	60.2	31.0	-
w/o PRM	28.0	38.2	55.4	26.7	78.9	45.4	-1.62
w/o Replay	30.7	52.8	76.5	30.0	86.3	55.5	-0.65
w/o Proj	22.6	48.1	72.4	25.3	80.7	48.8	-0.48
CTRL	**33.3**	**63.1**	**90.0**	**33.3**	**90.0**	**62.0**	**-0.08**



(a) AIME24 (b) GPQA (c) TTRL (d) CTRL

Figure 3: Visualization analysis experiment on Qwen3-1.7B-Base

from 23.3 to 42.7, while FM improves from -3.65 to -0.75. Furthermore, we observe that voting-based methods (e.g., TTRL and INTUITOR) underperform the base model on the AIME benchmark after continual test-time learning. This observation directly validates our core motivation: the majority voting mechanism tends to solidify erroneous consensus into pseudo-labels during continual adaptation, thereby triggering a positive feedback loop of error accumulation and catastrophic forgetting. In summary, through PRM-driven high-confidence trajectory selection and cognitive anchor-guided gradient correction, CTRL effectively decouples pseudo-signal construction from parameter update directions, mitigating error accumulation while alleviating catastrophic forgetting, thereby establishing a reliable paradigm for CTRL that combines high adaptability with strong memory preservation. We provide further analysis of task order sensitivity in Appendix G.6 .

4.3 Ablations

To validate CTRL's core design, we evaluate three ablations: **w/o PRM** (reverts to majority-voting pseudo-labels), **w/o Replay** (disables historical anchor memory), and **w/o Proj** (removes gradient conflict projection). Table 2 demonstrates that w/o PRM severely degrades FM to -1.85 and -1.62, consistent with Theorem 1: without process-level supervision, majority voting amplifies erroneous consensus into recursive gradient bias. Meanwhile, w/o Replay and w/o Proj yield substantially higher forgetting (-0.92/-0.78 and -0.65/-0.48), consistent with Theorem 2: signal purification alone cannot prevent knowledge erosion; historical anchoring and explicit gradient decoupling are mandatory to resolve cross-task conflicts. Collectively, these results underscore CTRL's dual-path synergy: PRM purifies supervision signals at the source, while gradient projection enforces update orthogonality to historical representations, jointly severing the coupled feedback loop between error accumulation and catastrophic forgetting. Additional PRM-budget-controlled ablations are provided in Appendix G.5.

4.4 Quantitative Analysis and Discussion

This section examines whether CTRL effectively addresses error accumulation and catastrophic forgetting in continual TTRL. Figure 1 shows that the pseudo-label accuracy of TTRL degrades from 20.6 to 14 after continual task learning, whereas CTRL remains stable, varying from 26.7 to 26.3. Figures 3(a) and (b) further demonstrate that CTRL consistently maintains higher pseudo-label accuracy across training steps, which we attribute to PRM-based fine-grained scoring that filters

low-confidence trajectories, thereby mitigating error accumulation at its source. The historical task re-evaluation in Figures 3(c) and (d) reveals that TTRL exhibits monotonic degradation in early task accuracy (T1 drops from 24.0 to 14.7), while CTRL maintains stable performance across all stages (T1 at 26.7, T3 at 87.6) with minimal forgetting. This stability stems from cognitive anchor replay and soft gradient projection mechanisms that explicitly decouple gradients between new and historical tasks. In summary, through the synergistic interaction of process-level reward modeling and history-aware gradient correction, CTRL severs the positive feedback loop between pseudo-label noise and parameter drift, simultaneously achieving effective suppression of error accumulation and significant mitigation of catastrophic forgetting.

In Appendix G, we provide supplementary experiments and in-depth analyses. Specifically, Appendix G.1 evaluates the sensitivity of key hyperparameter settings; Appendix G.2 presents a detailed analysis of CTRL's time complexity and computational overhead; Appendix G.3 explores potential failure scenarios and capability boundaries; and Appendix G.4 investigates the impact of PRMs with varying parameter scales on final performance.

5 Related Work

5.1 Continual Post-Training

Continual Post-Training (CPT) aims to enable models to acquire new capabilities from sequential task streams while preserving prior knowledge. Recent studies [14, 9, 26, 17, 39, 21, 15] show that large models often suffer from catastrophic forgetting [30], and that the degree of forgetting varies across post-training paradigms. In particular, Reinforcement Fine-Tuning has been observed to retain knowledge better than Supervised Fine-Tuning [14], suggesting that forgetting is affected not only by task order but also by the optimization mechanism itself. Existing CPT methods [33, 3] commonly rely on replay [3, 7, 19, 1], regularization [4, 43], or parameter isolation [22, 25, 13], which often require historical data, explicit task boundaries, or extra storage. In contrast, our work studies continual online updating during test time, where supervision comes from model-generated pseudo-labels or pseudo-rewards rather than ground-truth annotations. As a result, continual TTRL must handle not only catastrophic forgetting but also self-supervision noise, reward bias, and error accumulation, making standard CPT solutions difficult to apply directly.

5.2 Test-time Reinforcement Learning

Test-Time Reinforcement Learning (TTRL) enables online adaptation at inference time using unlabeled test samples, where the key challenge is constructing reliable test-time rewards. Representative methods [45, 31] usually sample multiple rollouts for each problem, derive pseudo-labels through majority voting, and update the model using trajectories consistent with the voted answer. While this demonstrates the potential of self-generated consistency signals for test-time improvement, it also exposes the risk of unreliable pseudo-labels. Recent studies [5, 41] show that majority voting may incorrectly reinforce false consensus and discard useful information from high-quality minority trajectories. To address this, later works improve reward construction through rollout distribution modeling, confidence estimation, local consistency, curriculum learning, and external tool-based verification [27, 28, 6, 16, 32, 35, 11]. However, most existing TTRL methods remain limited to single-task or single-stage adaptation. In contrast, CTRL targets multi-task continual TTRL, aiming to jointly mitigate error accumulation and catastrophic forgetting.

6 Conclusion

To address the coupled challenges of error accumulation and catastrophic forgetting in continual test-time reinforcement learning, we propose the CTRL framework, which decouples pseudo-signal construction from parameter updates through PRM-based fine-grained trajectory filtering and cognitive anchor-driven soft gradient correction. Experiments demonstrate that CTRL consistently outperforms baselines across multiple model scales and cross-domain task streams, effectively maintaining pseudo-label stability and historical reasoning capabilities, with 7B-scale PRMs providing a strong efficiency-performance trade-off in our experiments. Future work will focus on lightweight scoring mechanisms, cross-modal fusion, and dynamic task scheduling to advance the method toward more complex scenarios.

Acknowledgement

This work is supported by the Postdoctoral Fellowship Program and China Postdoctoral Science Foundation under Grant (No. BX2026301), the China Postdoctoral Science Foundation under Grant (No. 2025M781535).

References

[1] Istabrak Abbes, Gopeshh Subbaraj, Matthew Riemer, Nizar Islah, Benjamin Therien, Tsuguchika Tabaru, Hiroaki Kingetsu, Sarath Chandar, and Irina Rish. Revisiting replay and gradient alignment for continual pre-training of large language models. *arXiv preprint arXiv:2508.01908*, 2025.

[2] Immanuel Abdi, Akshat Gupta, Micah Mok, Alexander Lu, Nicholas Lee, and Gopala Anumanchipalli. Evolutionary strategies lead to catastrophic forgetting in llms. *arXiv preprint arXiv:2601.20861*, 2026.

[3] Hongyang Chen, Zhongwu Sun, Hongfei Ye, Kunchi Li, and Xuemin Lin. Continual learning in large language models: Methods, challenges, and opportunities. *arXiv preprint arXiv:2603.12658*, 2026.

[4] Xuxi Chen, Zhendong Wang, Daouda Sow, Junjie Yang, Tianlong Chen, Yingbin Liang, Mingyuan Zhou, and Zhangyang Wang. Take the bull by the horns: Hard sample-reweighted continual training improves llm generalization. *arXiv preprint arXiv:2402.14270*, 2024.

[5] Bodong Du, Xuanqi Huang, and Xiaomeng Li. Distribution-aware reward estimation for test-time reinforcement learning. *arXiv preprint arXiv:2601.21804*, 2026.

[6] Kaixuan Du, Meng Cao, Hang Zhang, Yukun Wang, Xiangzhou Huang, and Ni Li. Dual consensus: Escaping from spurious majority in unsupervised rlvr via two-stage vote mechanism. *arXiv preprint arXiv:2603.16223*, 2026.

[7] Yujie Feng, Hao Wang, Jian Li, Xu Chu, Zhaolu Kang, Yiran Liu, Yasha Wang, Philip S Yu, and Xiao-Ming Wu. Forever: Forgetting curve-inspired memory replay for language model continual learning. *arXiv preprint arXiv:2601.03938*, 2026.

[8] 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.

[9] Naimul Haque. Catastrophic forgetting in llms: A comparative analysis across language tasks. *arXiv preprint arXiv:2504.01241*, 2025.

[10] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

[11] Chengsong Huang, Wenhao Yu, Xiaoyang Wang, Hongming Zhang, Zongxia Li, Ruosen Li, Jiaxin Huang, Haitao Mi, and Dong Yu. R-zero: Self-evolving reasoning llm from zero data. *arXiv preprint arXiv:2508.05004*, 2025.

[12] Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14):6421, 2021.

[13] Tatsuya Konishi, Mori Kurokawa, Chihiro Ono, Zixuan Ke, Gyuhak Kim, and Bing Liu. Parameter-level soft-masking for continual learning. In *International conference on machine learning*, pages 17492–17505. PMLR, 2023.

[14] Song Lai, Haohan Zhao, Rong Feng, Changyi Ma, Wenzhuo Liu, Hongbo Zhao, Xi Lin, Dong Yi, Qingfu Zhang, Hongbin Liu, et al. Reinforcement fine-tuning naturally mitigates forgetting in continual post-training. *arXiv preprint arXiv:2507.05386*, 2025.

[15] Hongyu Li, Liang Ding, Meng Fang, and Dacheng Tao. Revisiting catastrophic forgetting in large language model tuning. In *Findings of the association for computational linguistics: EMNLP 2024*, pages 4297–4308, 2024.

[16] Ruotong Liao, Nikolai Röhrich, Xiaohan Wang, Yuhui Zhang, Yasaman Samadzadeh, Volker Tresp, and Serena Yeung-Levy. Tool verification for test-time reinforcement learning. *arXiv preprint arXiv:2603.02203*, 2026.

[17] Chengyuan Liu, Yangyang Kang, Shihang Wang, Lizhi Qing, Fubang Zhao, Chao Wu, Changlong Sun, Kun Kuang, and Fei Wu. More than catastrophic forgetting: Integrating general capabilities for domain-specific llms. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7531–7548, 2024.

[18] Jia Liu, ChangYi He, YingQiao Lin, MingMin Yang, FeiYang Shen, and ShaoGuo Liu. Ettrl: Balancing exploration and exploitation in llm test-time reinforcement learning via entropy mechanism. *arXiv preprint arXiv:2508.11356*, 2025.

[19] Yiyang Lu, Yu He, Jianlong Chen, and Hongyuan Zha. Mssr: Memory-aware adaptive replay for continual llm fine-tuning. *arXiv preprint arXiv:2603.09892*, 2026.

[20] Ruilin Luo, Zhuofan Zheng, Yifan Wang, Xinzhe Ni, Zicheng Lin, Songtao Jiang, Yiyao Yu, Chufan Shi, Lei Wang, Ruihang Chu, et al. Unlocking multimodal mathematical reasoning via process reward model. *arXiv preprint arXiv:2501.04686*, 2025.

[21] Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *IEEE Transactions on Audio, Speech and Language Processing*, 2025.

[22] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7765–7773, 2018.

[23] Anastasios Nentidis, Georgios Katsimpras, Eirini Vandorou, Anastasia Krithara, Antonio Miranda-Escalada, Luis Gasco, Martin Krallinger, and Georgios Paliouras. Overview of bioasq 2022: the tenth bioasq challenge on large-scale biomedical semantic indexing and question answering. In *International conference of the cross-language evaluation forum for European languages*, pages 337–361. Springer, 2022.

[24] Weijieying Ren, Xinlong Li, Lei Wang, Tianxiang Zhao, and Wei Qin. Analyzing and reducing catastrophic forgetting in parameter efficient tuning. *arXiv preprint arXiv:2402.18865*, 2024.

[25] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International conference on machine learning*, pages 4548–4557. PMLR, 2018.

[26] Haizhou Shi, Zihao Xu, Hengyi Wang, Weiyi Qin, Wenyuan Wang, Yibin Wang, Zifeng Wang, Sayna Ebrahimi, and Hao Wang. Continual learning of large language models: A comprehensive survey. *ACM Computing Surveys*, 58(5):1–42, 2025.

[27] Ru Wang, Wei Huang, Qi Cao, Yusuke Iwasawa, Yutaka Matsuo, and Jiaxian Guo. Self-harmony: Learning to harmonize self-supervision and self-play in test-time reinforcement learning. *arXiv preprint arXiv:2511.01191*, 2025.

[28] Weiqin Wang, Yile Wang, Kehao Chen, and Hui Huang. Beyond majority voting: Towards fine-grained and more reliable reward signal for test-time reinforcement learning. *arXiv preprint arXiv:2512.15146*, 2025.

[29] 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.

[30] Zhenyi Wang, Enneng Yang, Li Shen, and Heng Huang. A comprehensive survey of forgetting in deep learning beyond continual learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(3):1464–1483, 2024.

[31] Lai Wei, Yuting Li, Chen Wang, Yue Wang, Linghe Kong, Weiran Huang, and Lichao Sun. First sft, second rl, third upt: Continual improving multi-modal llm reasoning via unsupervised post-training. *arXiv preprint arXiv:2505.22453*, 2025.

[32] Jianghao Wu, Yasmeen George, Jin Ye, Yicheng Wu, Daniel F Schmidt, and Jianfei Cai. Spine: Token-selective test-time reinforcement learning with entropy-band regularization. *arXiv preprint arXiv:2511.17938*, 2025.

[33] Tongtong Wu, Linhao Luo, Yuan-Fang Li, Shirui Pan, Thuy-Trang Vu, and Gholamreza Haffari. Continual learning for large language models: A survey. *arXiv preprint arXiv:2402.01364*, 2024.

[34] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[35] Chengyi Yang, Zhishang Xiang, Yunbo Tang, Zongpei Teng, Chengsong Huang, Fei Long, Yuhan Liu, and Jinsong Su. Ttcs: Test-time curriculum synthesis for self-evolving. *arXiv preprint arXiv:2601.22628*, 2026.

[36] Zhenquan Yao, Zitong Huang, Yihan Zeng, Jianhua Han, Hang Xu, Chun-Mei Feng, Jianwei Ma, and Wangmeng Zuo. Cgl: Advancing continual gui learning via reinforcement fine-tuning. *arXiv preprint arXiv:2603.02951*, 2026.

[37] Rui Yu, Shenghua Wan, Yucen Wang, Chen-Xiao Gao, Le Gan, Zongzhang Zhang, and De-Chuan Zhan. Reward models in deep reinforcement learning: A survey. *arXiv preprint arXiv:2506.15421*, 2025.

[38] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in neural information processing systems*, 33:5824–5836, 2020.

[39] Yuexiang Zhai, Shengbang Tong, Xiao Li, Mu Cai, Qing Qu, Yong Jae Lee, and Yi Ma. Investigating the catastrophic forgetting in multimodal large language models. *arXiv preprint arXiv:2309.10313*, 2023.

[40] 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*, pages 10495–10516, 2025.

[41] Chu Zhao, Enneng Yang, Yuting Liu, Jianzhe Zhao, and Guibing Guo. Echo: Entropy-confidence hybrid optimization for test-time reinforcement learning. *arXiv preprint arXiv:2602.02150*, 2026.

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

[43] Xuyang Zhao, Huiyuan Wang, Weiran Huang, and Wei Lin. A statistical theory of regularization-based continual learning. *arXiv preprint arXiv:2406.06213*, 2024.

[44] Congming Zheng, Jiachen Zhu, Zhuoying Ou, Yuxiang Chen, Kangning Zhang, Rong Shan, Zeyu Zheng, Mengyue Yang, Jianghao Lin, Yong Yu, et al. A survey of process reward models: From outcome signals to process supervisions for large language models. *arXiv preprint arXiv:2510.08049*, 2025.

[45] 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.

Appendix

A Limitations **13**

B Broader impact **13**

C Declaration of LLM usage **14**

D Proofs of Theoretical Results **14**

D.1 Proof of Theorem 1 14

D.2 Proof of Theorem 2 16

E Experimental Setup **16**

E.1 Datasets 16

E.2 Baselines 17

E.3 Large Language Models 17

E.4 Implementation Details 18

E.5 Evolution Setting 18

F Prompt Templates for Posterior Correction **18**

G Supplementary experiments and analysis **19**

G.1 Hyperparameter Sensitivity Analysis 19

G.2 Time complexity analysis 20

G.3 When Might CTRL Fail? 21

G.4 Analyzing How Varying PRM Model Parameters Affect CTRL Performance 22

G.5 Additional ablation experiments 23

G.6 Task Order Sensitivity 23

H Hyperparameter settings **24**

I Algorithm of CTRL **24**

A Limitations

Our current work primarily focuses on text-only reasoning tasks and has not yet been extended to multimodal interactions or dynamic open-domain scenarios. Additionally, the framework's performance remains constrained by the base model's initial reasoning capacity and the evaluation priors of the process reward model. Future work will explore cross-modal anchor fusion and more lightweight online scoring mechanisms to broaden its applicability.

B Broader impact

The proposed CTRL framework focuses on optimizing the test-time reinforcement learning algorithm itself and does not involve the construction of new datasets, human annotation, or research with human subjects. All experiments are conducted on publicly available academic benchmarks (e.g., AIME, MATH, GPQA, and MedQA), strictly adhering to the original licenses and usage guidelines

of each dataset. As a foundational methodological study, CTRL aims to enhance the adaptive stability and knowledge retention of models across continuous task streams, and its technical contributions do not directly target specific application scenarios. We acknowledge that any technology capable of enhancing model reasoning may be leveraged for diverse purposes; however, this work does not involve content generation, decision-support systems, or high-risk deployment scenarios. For future real-world applications, we recommend integrating content safety mechanisms, bias monitoring, and human oversight to ensure the responsible use of this technology.

C Declaration of LLM usage

This paper employs the large language model ChatGPT solely for linguistic refinement, including spelling correction and grammatical enhancement. The model was not utilized for literature search or the formulation of research concepts. After the language editing process, the authors meticulously reviewed and revised the content, accepting complete responsibility for the accuracy and integrity of the final published manuscript.

D Proofs of Theoretical Results

D.1 Proof of Theorem 1

We prove that biased pseudo-labels induced by majority voting lead to accumulated parameter drift in continual TTRL. For task T_k, the current policy π_{θ_k} samples N trajectories for each input $x \sim D_k$:

$$Y(x) = \{y_1, \ldots, y_N\} \sim \pi_{\theta_k}(\cdot \mid x). \tag{14}$$

Traditional TTRL constructs a majority-voted pseudo-label

$$\hat{a}_k(x) = \mathrm{Maj}\left(\{\mathrm{ans}(y_i)\}_{i=1}^{N}\right), \tag{15}$$

and assigns the binary pseudo-reward

$$\hat{r}_k(x, y) = \mathbb{I}\left[\mathrm{ans}(y) = \hat{a}_k(x)\right]. \tag{16}$$

Let $a_k^\star(x)$ denote the ground-truth answer, and define the corresponding ideal reward as

$$r_k^\star(x, y) = \mathbb{I}\left[\mathrm{ans}(y) = a_k^\star(x)\right]. \tag{17}$$

The ideal reward gradient and pseudo-reward gradient are defined as

$$g_k^\star = \nabla_\theta \mathbb{E}_{x \sim D_k,\, y \sim \pi_\theta(\cdot|x)}\left[r_k^\star(x, y)\right]\Big|_{\theta=\theta_k}, \tag{18}$$

and

$$\hat{g}_k = \nabla_\theta \mathbb{E}_{x \sim D_k,\, y \sim \pi_\theta(\cdot|x)}\left[\hat{r}_k(x, y)\right]\Big|_{\theta=\theta_k}. \tag{19}$$

We define the pseudo-label-induced gradient bias as

$$e_k = \hat{g}_k - g_k^\star. \tag{20}$$

Therefore,

$$\hat{g}_k = g_k^\star + e_k. \tag{21}$$

We first show why e_k is non-zero in general when majority voting is imperfect. Define the reward discrepancy

$$\delta_k(x, y) = \hat{r}_k(x, y) - r_k^\star(x, y). \tag{22}$$

If the majority vote is correct, i.e., $\hat{a}_k(x) = a_k^\star(x)$, then $\delta_k(x, y) = 0$ for all trajectories. However, if the majority vote is incorrect, i.e., $\hat{a}_k(x) \neq a_k^\star(x)$, then the discrepancy becomes systematic. For a trajectory that produces the true answer, $\mathrm{ans}(y) = a_k^\star(x)$, we have

$$r_k^\star(x, y) = 1, \qquad \hat{r}_k(x, y) = 0, \tag{23}$$

and hence

$$\delta_k(x, y) = -1. \tag{24}$$

Conversely, for a trajectory that produces the erroneous majority answer, $\mathrm{ans}(y) = \hat{a}_k(x) \neq a_k^\star(x)$, we have

$$r_k^\star(x, y) = 0, \qquad \hat{r}_k(x, y) = 1, \tag{25}$$

and hence

$$\delta_k(x, y) = 1. \tag{26}$$

Thus, once majority voting selects an incorrect pseudo-label, correct trajectories are systematically suppressed while erroneous majority trajectories are reinforced. Since majority voting is imperfect with non-zero probability, i.e.,

$$\Pr\left(\hat{a}_k(x) \neq a_k^\star(x)\right) > 0, \tag{27}$$

the discrepancy $\delta_k(x, y)$ is non-zero on a non-zero-measure subset of samples. Consequently, the pseudo-reward objective differs from the ideal reward objective in general, inducing a non-zero gradient discrepancy:

$$e_k = \nabla_\theta \mathbb{E}_{x\sim D_k,\, y\sim\pi_\theta(\cdot|x)}\left[\delta_k(x, y)\right]\Big|_{\theta=\theta_k}. \tag{28}$$

Next, we analyze the accumulated effect of this bias over K continual updates. The biased continual TTRL update is

$$\theta_{K+1} = \theta_1 - \eta \sum_{k=1}^{K} \hat{g}_k. \tag{29}$$

Substituting $\hat{g}_k = g_k^\star + e_k$ gives

$$\theta_{K+1} = \theta_1 - \eta \sum_{k=1}^{K} (g_k^\star + e_k) = \theta_1 - \eta \sum_{k=1}^{K} g_k^\star - \eta \sum_{k=1}^{K} e_k. \tag{30}$$

The ideal parameter trajectory obtained using ground-truth rewards is

$$\theta_{K+1}^\star = \theta_1 - \eta \sum_{k=1}^{K} g_k^\star. \tag{31}$$

Subtracting the ideal trajectory from the biased trajectory yields

$$\theta_{K+1} - \theta_{K+1}^\star = -\eta \sum_{k=1}^{K} e_k. \tag{32}$$

Taking norms and applying the triangle inequality gives

$$\left\|\theta_{K+1} - \theta_{K+1}^\star\right\| = \eta \left\|\sum_{k=1}^{K} e_k\right\| \leq \eta \sum_{k=1}^{K} \|e_k\|. \tag{33}$$

This proves that pseudo-label-induced gradient biases can accumulate through continual parameter recursion.

Finally, we prove the lower bound under persistent alignment of expected biases. By Jensen's inequality,

$$\mathbb{E}\left[\left\|\theta_{K+1} - \theta_{K+1}^\star\right\|\right] \geq \left\|\mathbb{E}\left[\theta_{K+1} - \theta_{K+1}^\star\right]\right\|. \tag{34}$$

Using

$$\theta_{K+1} - \theta_{K+1}^\star = -\eta \sum_{k=1}^{K} e_k, \tag{35}$$

we have

$$\left\|\mathbb{E}\left[\theta_{K+1} - \theta_{K+1}^\star\right]\right\| = \eta \left\|\sum_{k=1}^{K} \mathbb{E}[e_k]\right\|. \tag{36}$$

If the expected biases exhibit persistent alignment such that

$$\left\|\sum_{k=1}^{K} \mathbb{E}[e_k]\right\| \geq cK\epsilon \tag{37}$$

for constants $c, \epsilon > 0$, then

$$\mathbb{E}\left[\left\|\theta_{K+1} - \theta_{K+1}^\star\right\|\right] \geq \eta c K \epsilon. \tag{38}$$

Therefore, systematic pseudo-label bias can induce task-length-dependent parameter drift in continual TTRL. This completes the proof.

D.2 Proof of Theorem 2

We prove that pseudo-label-induced gradient bias can enlarge the upper bound of historical-task loss growth, thereby amplifying catastrophic forgetting.

Let $L_j(\theta)$ denote the loss of a historical task T_j, where $j < k$. Assume that L_j is β-smooth. By the standard smoothness inequality, for any parameter perturbation Δ, we have

$$L_j(\theta_k + \Delta) \leq L_j(\theta_k) + \nabla_\theta L_j(\theta_k)^\top \Delta + \frac{\beta}{2}\|\Delta\|^2. \tag{39}$$

At task T_k, the continual TTRL update uses the biased gradient

$$\hat{g}_k = g_k^\star + e_k, \tag{40}$$

where $g_k^\star$ is the ideal current-task gradient and e_k is the pseudo-label-induced gradient bias. The update is

$$\theta_{k+1} = \theta_k - \eta \hat{g}_k = \theta_k - \eta(g_k^\star + e_k). \tag{41}$$

Therefore, the perturbation from θ_k to θ_{k+1} is

$$\Delta = \theta_{k+1} - \theta_k = -\eta(g_k^\star + e_k). \tag{42}$$

Let

$$g_j = \nabla_\theta L_j(\theta_k) \tag{43}$$

be the historical-task gradient at the current parameter point. Substituting $\Delta = -\eta(g_k^\star + e_k)$ into the smoothness inequality gives

$$L_j(\theta_{k+1}) \leq L_j(\theta_k) - \eta g_j^\top (g_k^\star + e_k) + \frac{\beta\eta^2}{2}\|g_k^\star + e_k\|^2. \tag{44}$$

Subtracting $L_j(\theta_k)$ from both sides, we obtain

$$L_j(\theta_{k+1}) - L_j(\theta_k) \leq -\eta g_j^\top (g_k^\star + e_k) + \frac{\beta\eta^2}{2}\|g_k^\star + e_k\|^2. \tag{45}$$

Expanding the inner product yields

$$L_j(\theta_{k+1}) - L_j(\theta_k) \leq -\eta g_j^\top g_k^\star - \eta g_j^\top e_k + \frac{\beta\eta^2}{2}\|g_k^\star + e_k\|^2. \tag{46}$$

The term

$$-\eta g_j^\top e_k \tag{47}$$

captures the effect of pseudo-label-induced bias on the historical-task loss change. When the pseudo-label bias is anti-aligned with the historical gradient, i.e.,

$$g_j^\top e_k < 0, \tag{48}$$

we have

$$-\eta g_j^\top e_k > 0. \tag{49}$$

Thus, the bias term strictly increases the upper bound of historical-task loss growth. This shows that pseudo-label-induced bias can turn self-supervision errors into destructive interference with historical tasks, thereby increasing the risk of catastrophic forgetting. This completes the proof.

E Experimental Setup

E.1 Datasets

- **AIME2024**[2] is a competition-level mathematical reasoning benchmark drawn from the 2024 American Invitational Mathematics Examination. It contains 30 problems, each requiring multi-step symbolic reasoning. Following the official AIME setting, each question has a final answer in the form of an integer from 0 to 999, making it well suited for evaluating exact-match accuracy in advanced mathematical reasoning. Due to its high difficulty, AIME2024 is widely used to assess the mathematical problem-solving ability of large language models.

[2] https://huggingface.co/datasets/HuggingFaceH4/aime_2024

- **AMC** [29] is a mathematical reasoning benchmark drawn from the 2023 American Mathematics Competitions (AMC). It contains 40 problems that require multi-step quantitative reasoning and exact-answer prediction. As a widely used competition-math benchmark, AMC is commonly adopted to evaluate the mathematical reasoning ability of large language models.
- **MMLU** [10] is a broad knowledge and reasoning benchmark covering 57 subjects, including STEM, humanities, social science, and professional domains. It is widely used to evaluate the multitask knowledge understanding and general reasoning ability of large language models.
- **GPQA-Diamond** [23] is a challenging subset of GPQA, a graduate-level, Google-proof question answering benchmark covering biology, physics, and chemistry. The Diamond split contains 198 high-quality questions, designed such that domain experts can answer them correctly while non-experts typically cannot, making it a strong benchmark for expert-level scientific reasoning.
- **MATH-500**[3] is a curated subset of 500 problems from the MATH benchmark, commonly used for efficient evaluation of mathematical reasoning. It inherits the challenging, competition-level nature of MATH while providing a compact test set for standardized comparison.
- **AIME2025**[4] comprises 30 high-difficulty problems sourced directly from the February 2025 administration of the American Invitational Mathematics Examination (AIME I and II). Covering a broad spectrum of mathematical domains, including algebra, geometry, number theory, and combinatorics, this benchmark serves as a rigorous testbed for evaluating and comparing the mathematical reasoning capabilities of contemporary models.
- **MedQA** [12] is a large-scale medical question answering benchmark collected from professional medical board exams. The English split is widely used to evaluate medical knowledge understanding and clinical reasoning ability of language models.

E.2 Baselines

- **TTRL** [45] is a label-free test-time reinforcement learning method that uses majority voting over multiple sampled responses to construct pseudo-rewards for online adaptation.
- **SELF-HARMONY** [27] improves test-time reinforcement learning by encouraging answer consistency between the original question and its paraphrased version, and builds pseudo-labels through multi-view aggregation.
- **INTUITOR** [42] replaces external rewards with the model's own self-certainty signals, enabling unsupervised reinforcement learning for reasoning tasks.
- **RIF-RFT** [14] is a continual post-training baseline showing that reinforcement fine-tuning can alleviate forgetting compared with supervised fine-tuning, due to its implicit regularization effect during optimization.
- **PS-GRPO** [20] extends the standard GRPO framework by incorporating a **process reward model** to provide fine-grained supervision over intermediate reasoning steps.

E.3 Large Language Models

To systematically evaluate the continual adaptation capabilities of CTRL across varying parameter scales, architectural paradigms, and task distributions, we select multiple mainstream models as policy backbones and Process Reward Models (PRMs), all seamlessly integrated in a zero-shot or frozen-parameter manner. For policy backbones, we primarily utilize the Qwen3-Base series [34] (0.6B to 8B), leveraging their raw reasoning manifolds prior to instruction alignment to better satisfy the online exploration requirements of TTRL. We additionally incorporate Llama3-8B-Instruct[5] to verify cross-architecture compatibility and the adaptation stability of CTRL on instruction-aligned models. Regarding PRMs, we evaluate five distinct evaluators: the lightweight Skywork-PRM-1.5B [6], the domain-specialized Qwen2.5-Math-PRM [40] (7B/72B) optimized for structured reasoning, and the API-accessible Qwen3.6-Plus [7] and DeepSeek-V4-Pro [8], which exhibit strong cross-domain

[3] https://huggingface.co/datasets/HuggingFaceH4/MATH-500
[4] https://huggingface.co/datasets/math-ai/aime25
[5] https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct
[6] https://www.modelscope.cn/models/AI-ModelScope/Skywork-o1-Open-PRM-Qwen-2.5-1.5B
[7] https://qwen.ai/blog?id=qwen3.6
[8] https://www.deepseek.com/

generalization. This configuration spans the full spectrum from domain-specific to general-purpose LLMs, enabling CTRL to perform fine-grained trajectory evaluation and high-confidence anchor filtering without requiring any additional fine-tuning.

E.4 Implementation Details

Following the experimental configurations of prior work [45, 18], all baseline methods are uniformly implemented based on the GRPO framework. All reported accuracies are computed by exact-match evaluation against ground-truth answers or official benchmark evaluators. The Qwen2.5-Math-PRM-7B is used only as a frozen process-level scorer during test-time adaptation, not as the final evaluator; furthermore, we conduct ablation studies to analyze the impact of process reward models with varying parameter scales on final performance. Regarding hyperparameter settings, the policy model is optimized using AdamW with a cosine learning rate schedule peaking at $5e-7$. The tasks are introduced sequentially in the following order: AIME24, AMC, MMLU, MATH-500, GPQA, and MedQA. During the rollout phase, the temperature is set to 0.8, and 32 responses are sampled per prompt, which are simultaneously used for policy training and reward computation. The maximum generation length is uniformly capped at 3,072 tokens across all models. All results are the average of five independent inferences. All experiments were completed on 8 A800 GPUs.

E.5 Evolution Setting

We independently apply the CTRL method to each benchmark for evaluation. The maximum generation length is uniformly set to 3,072 tokens across all test benchmarks. In the main experiments, following the evaluation protocols of DeepSeek-R1 and TTRL, we adopt pass@k as the core evaluation metric and report pass@1 results. Specifically, we generate 16 responses per query with a sampling temperature of 0.8 and top-p set to 0.9. The pass@1 score is computed as:

$$\text{pass@1} = \frac{1}{k}\sum_{i=1}^{k} p_i \tag{50}$$

where $p_i \in \{0, 1\}$ indicates whether the i-th response is correct. Furthermore, to quantify the model's knowledge retention capability, we adopt the Forgetting Measure (FM) proposed in [36]. This metric evaluates knowledge forgetting during continual learning by computing the average difference between each task's final accuracy and its historical best accuracy throughout the training sequence:

$$\text{FM} = \frac{1}{T}\sum_{i=1}^{T}(P_{T,i} - P_i^*), \quad \text{where } P_i^* = \max_{k\in\{i,\ldots,T\}} P_{k,i} \tag{51}$$

A FM value closer to zero indicates less forgetting and superior knowledge preservation.

F Prompt Templates for Posterior Correction

For samples in $\mathcal{P}_{\text{weak}}$, we construct a guided prompt to encourage self-correction. The template is designed to (1) explicitly state the candidate consensus answer, (2) request a concise and logically self-consistent reconstruction, and (3) enforce answer formatting for easy parsing.

```
A candidate final answer is {{consensus_answer}}.  Please reconstruct a concise,
logically self-consistent reasoning chain that either verifies or corrects the
original reasoning.  If you identify contradictions or gaps, fix them before
finalizing your response.  Put your final answer in \boxed{}.
```

Usage notes:

- `{{consensus_answer}}` is replaced by the majority-voted answer $\hat{a}_p$ at runtime.
- The instruction emphasizes *conciseness* and *logical self-consistency* to avoid verbose or circular reasoning.
- The \boxed{} formatting requirement ensures compatibility with the answer extraction pipeline.

Table 3: Hyperparameter Sensitivity Analysis on Qwen3-4B-Base

γ_C	**0.3**	**0.45**	**0.55**	**0.65**	**0.75**	**0.8**
MATH	78.4	84.2	**88.6**	87.3	82.1	76.5
GPQA	38.2	45.8	**51.4**	49.7	43.2	36.8
γ_R	**0.3**	**0.45**	**0.55**	**0.65**	**0.75**	**0.8**
MATH	75.3	82.6	**89.2**	88.1	83.4	77.9
GPQA	35.6	44.3	**52.8**	51.2	45.1	38.4
γ_{step}	**0**	**0.3**	**0.4**	**0.5**	**0.7**	**0.8**
MATH	82.1	86.4	**90.0**	88.7	84.2	79.6
GPQA	42.3	48.1	**53.1**	51.6	46.8	41.2
τ	**0**	**0.01**	**0.05**	**0.1**	**0.5**	**1**
MATH	85.2	87.8	**90.0**	89.1	84.6	80.3
GPQA	46.7	50.2	**53.1**	52.3	47.5	42.8

G Supplementary experiments and analysis

G.1 Hyperparameter Sensitivity Analysis

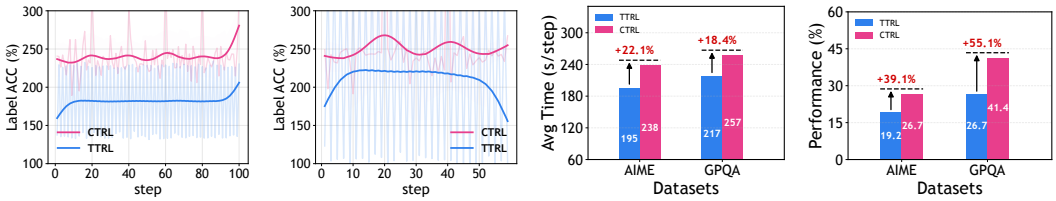
We conduct experiments by varying different hyperparameters, training Qwen3-4B-Base on three datasets (AIME24, MedQA, AIME25) and evaluating the impact on model performance using MATH and GPQA as test benchmarks. When testing a specific parameter, all other parameters remain fixed. The experimental results are presented in Table 3.

- **Impact of Consistency Threshold** γ_C. As γ_C increases from 0.3 to 0.55, model performance steadily improves, reaching peak accuracy on both MATH and GPQA. This suggests that a moderate consistency threshold can filter low-quality reasoning trajectories while preserving enough high-quality samples for training. However, when $\gamma_C > 0.65$, performance drops noticeably, as overly strict filtering removes too many trajectories and weakens the training signal. At $\gamma_C = 0.8$, the sharp decline further indicates that excessive pruning discards useful reasoning patterns.
- **Impact of Reliability Threshold** γ_R. γ_R shows a similar trend to γ_C, with optimal performance around 0.55. Compared with γ_C, γ_R is more sensitive to performance changes: under low thresholds, GPQA accuracy drops more severely. This supports the design rationale of the reliability metric, which focuses on the average score of the lowest-m steps and can better identify weak links in reasoning chains, making it more sensitive to noisy trajectories.
- **Impact of Step-wise Masking Threshold** γ_{step}. γ_{step} controls which reasoning steps are retained by the PRM during Output-Process Distillation (OPD). The best performance is achieved around $\gamma_{step} = 0.4$. Without masking, low-quality intermediate steps are included and introduce noisy supervision. In contrast, when $\gamma_{step} > 0.7$, overly strict masking retains too few steps for distillation, weakening process-level supervision. These results support the importance of fine-grained process supervision.
- **Impact of Gradient Conflict Threshold** τ**.** τ controls when gradient projection is triggered to mitigate catastrophic forgetting. The best trade-off is achieved around $\tau = 0.05$. When $\tau = 0$, the overly aggressive strategy projects any negatively correlated gradients, which may suppress current-task learning. Conversely, when τ becomes too large, performance approaches the setting without gradient projection, confirming that unresolved gradient conflicts can amplify forgetting, consistent with Proposition 2.
- **Buffer Size Sensitivity.** To evaluate the impact of memory pool capacity B on CTRL's performance and storage overhead, we vary $B \in \{0, 64, 128, 256, 512\}$ while keeping all other hyperparameters fixed. Table 4 reports the results on Qwen3-4B-Base. Three observations emerge: **(i) Minimal buffer suffices for substantial gains**: Even with $B = 64$, CTRL achieves +4.6% accuracy and reduces forgetting by 2.3× compared to $B = 0$, indicating that a small set of high-quality cognitive anchors can effectively stabilize historical knowledge. **(ii) Performance saturates at moderate**

Table 4: Buffer size sensitivity analysis on Qwen3-4B-Base.

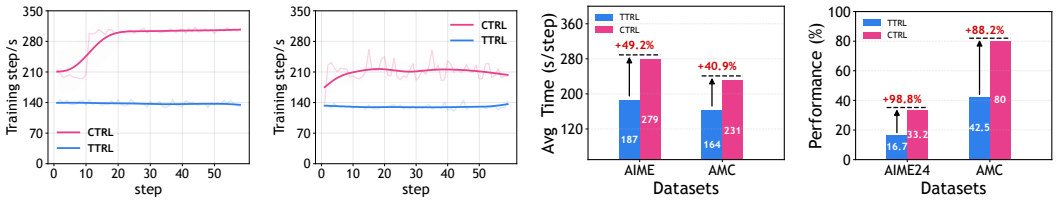
Buffer Size B	Avg. Acc.↑	FM↓	PRM Calls/prompt
0 (w/o Replay)	55.5	-0.65	34
64	60.1	-0.28	34
128 (default)	**62.0**	**-0.08**	34
256	62.3	-0.06	34
512	62.4	-0.05	34

capacity: Accuracy and FM improve marginally from $B = 128$ to $B = 512$ (+0.4% / -0.03), suggesting diminishing returns beyond the default setting. This confirms that CTRL's replay mechanism introduces no practical deployment burden. Based on these results, we adopt $B = 128$ as the default configuration, which balances performance saturation with minimal storage footprint.



(a) AIME24 (b) GPQA (c) Time (d) Performance

Figure 4: Time analysis and performance comparison Qwen3-1.7B-Base



(a) AIME24 (b) AMC (c) Time (d) Performance

Figure 5: Time analysis and performance comparison on Qwen3-4B-Base

G.2 Time complexity analysis

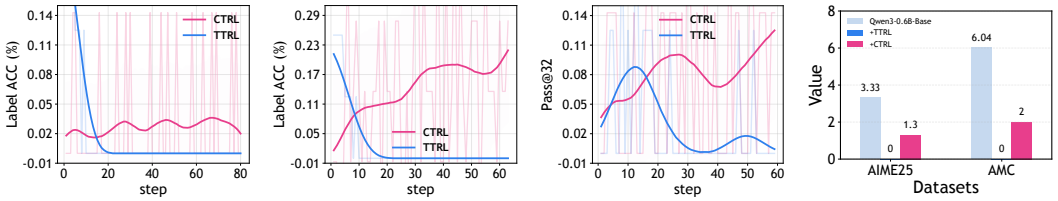
This section analyzes the time complexity of the CTRL framework from two perspectives: theoretical algorithmic structure and empirical experimental results.

Theoretical Algorithm Complexity Analysis. From an algorithmic design perspective, CTRL introduces three core mechanisms upon standard Test-Time Reinforcement Learning (TTRL): fine-grained trajectory scoring based on Process Reward Models (PRM), cognitive anchor memory pool management, and conflict-aware soft gradient projection correction. Compared to baseline methods that rely solely on majority voting, CTRL's per-step training overhead primarily increases in the following aspects: (1) The PRM model needs to perform step-by-step reasoning quality scoring for each sampled trajectory, with computational cost proportional to trajectory length and the number of PRM forward passes; (2) The three-stage golden trajectory selection, adaptive update gating, and posterior correction mechanisms introduce additional conditional judgments, resampling, and quality filtering logic; (3) Gradient conflict detection and projection operations involve inner product computation and orthogonal projection of historical anchor gradients, but their time complexity is only a linear operation O(d) with respect to parameter dimension, which is negligible compared to the forward and backward propagation of large language models. Overall, CTRL's per-step time complexity remains of the same order as baseline TTRL, introducing only constant-factor additional computational overhead without altering the algorithm's asymptotic complexity.

Experimental Time Overhead and Performance Trade-off Combining empirical results from Figure 4 (Qwen3-1.7B-Base) and Figure 5 (Qwen3-4B-Base), CTRL's actual time overhead manifests as moderate and controllable growth. On the AIME24 and GPQA datasets, CTRL's average per-step

training time increases by approximately 22.1% and 18.4% respectively compared to TTRL; for larger-scale models (4B) on AIME24 and AMC tasks, the time overhead increase ranges from 40.9% to 49.2%. This overhead primarily stems from PRM's fine-grained evaluation, high-quality trajectory filtering and caching, and orthogonal projection computations in gradient surgery. Although per-step time consumption increases, CTRL achieves significant performance gains and training stability: on the 1.7B model, performance improves by 39.1 and 55.1 respectively, while on the 4B model, performance gains reach as high as 88.2. Meanwhile, the Label ACC curves show that CTRL converges more smoothly, effectively suppressing the oscillations and performance degradation commonly observed in baseline methods under continual task streams.

Time Efficiency and Scalability Summary. The additional time cost reflects the cost of process-level scoring and history-aware correction. In our experiments, this overhead remains within a constant-factor range and is accompanied by improved stability and accuracy. Experiments demonstrate that this time overhead does not exhibit nonlinear expansion with task stream length or model scale, but remains within a reasonable linear range. In practical continual test-time adaptation scenarios, the per-step time increase of tens of seconds to several minutes is entirely within engineering-acceptable limits. The substantial improvements in continual learning robustness, historical reasoning pattern preservation, and final adaptation accuracy make CTRL highly cost-effective in terms of time-performance trade-off. In summary, while maintaining controllable time efficiency, CTRL successfully achieves stable, efficient, and scalable adaptation of the continual test-time reinforcement learning framework.



(a) AIME24 (b) AMC (c) Time (d) Performance

Figure 6: Comparison of pseudo-label accuracy and model performance on Qwen3-0.6B-Base

G.3 When Might CTRL Fail?

This section explores the potential failure scenarios and capability boundaries of the CTRL framework. Although CTRL significantly enhances the robustness of continual test-time adaptation through fine-grained PRM scoring and gradient surgery, it fundamentally remains a self-evolution paradigm based on Test-Time Reinforcement Learning. Such methods exhibit a strong dependency on the initial reasoning capability of the base model. **If the base model itself fails to generate a sufficient proportion of correct or high-quality trajectories during sampling, the TTRL paradigm will struggle to construct effective pseudo-supervisory signals, ultimately leading to optimization failure.**

Empirical results on Qwen3-0.6B-Base, as shown in Figure 6, clearly illustrate this capability boundary. Figures 6(a) and 6(b) present the pseudo-label accuracy on the AIME24 and AMC datasets, respectively. On this weaker 0.6B base model, the pseudo-label accuracy of TTRL rapidly collapses to near zero at the early stages of optimization, indicating that its reliance on majority voting is highly prone to generating misleading consensus when base capability is insufficient, thereby causing the supervision signal to completely fail. In contrast, CTRL, leveraging PRM-based fine-grained filtering and cognitive anchor mechanisms, maintains the pseudo-label accuracy within a low but non-zero stable range (fluctuating between 2%–4% on AIME24, and gradually climbing to around 20% on AMC), effectively preventing complete signal collapse. The Pass@32 curve in Figure 6(c) further corroborates this phenomenon: the generation diversity of TTRL rapidly drops to zero as optimization proceeds, falling into severe mode collapse; whereas CTRL, although constrained by the base model's capability and unable to achieve substantial growth, successfully preserves exploration capacity, with the Pass@32 metric showing a fluctuating upward trend. The performance comparison in Figure 6(d) visually displays the final outcome: TTRL yields zero performance gain on both AIME25 and AMC, indicating complete optimization failure; CTRL, on the other hand, achieves positive improvements (+1.3 on AIME25 and +2 on AMC), yet its absolute performance remains bounded by the initial capability ceiling of the base model.

Table 5: Impact of different Process Reward Models (PRMs) on CTRL performance. We evaluate five PRM variants with varying scales and architectures on the continual test-time learning sequence. Results are based on Qwen3-4B-Base as the policy backbone. Best and second-best results per column are **bolded** and underlined.

PRM Model	Params	AIME24	AMC	MATH	GPQA	AVG	Time on AIME24 /step (s)
Skywork-PRM-1.5B	1.5B	20.0	65.2	76.4	39.8	50.4	225.3
Qwen2.5-Math-PRM-7B	7B	33.3	80.0	90.0	53.1	64.1	238.0
Qwen2.5-Math-PRM-72B	72B	34.0	81.2	90.8	54.0	65.0	302.7
Qwen3.6-plus	-	**34.7**	80.8	**91.2**	53.8	65.1	481.5
DeepSeek-V4-Pro	1.6T	34.0	**82.1**	90.5	**55.4**	**65.5**	377.2

These results reveal the core applicability boundary of TTRL-based methods: self-evolution optimization is not a capability leap out of nothing, but rather a local calibration and expansion of the model's existing reasoning manifold. When the parameter scale or pretraining knowledge of the base model is insufficient to support basic correct trajectory sampling, pseudo-label noise will dominate the optimization direction. Even with the introduction of PRM filtering and conflict-aware correction, it is difficult to fundamentally reverse the dilemma of signal deficiency. The core value of CTRL in such extreme scenarios lies in its anti-collapse property, it maximally preserves the model's exploration capacity and prevents negative performance degradation when signals are weak or biased (e.g., Pass@32 does not drop to zero, and performance maintains positive gains), rather than achieving substantial breakthroughs on extremely low-capability bases. In practical deployment, this suggests that reasonable capability thresholds should be established for TTRL frameworks; for models with severely insufficient foundational reasoning capabilities, it is recommended to first solidify the base distribution via supervised fine-tuning or curriculum learning before introducing CTRL for continual adaptive optimization.

G.4 Analyzing How Varying PRM Model Parameters Affect CTRL Performance

To efficiently evaluate the impact of PRMs with varying parameter scales on CTRL performance, we conduct experiments on a unified policy backbone (Qwen3-1.7B-Base) comparing five PRM variants: Skywork-PRM-1.5B, Qwen2.5-Math-PRM-7B, Qwen2.5-Math-PRM-72B, and API-based Qwen3.6-plus and DeepSeek-V4-Pro (1.6T). All experiments maintain identical task sequences (AIME24 $\rightarrow$ AMC $\rightarrow$ MATH $\rightarrow$ GPQA) and hyperparameter configurations to ensure fair comparison.

From Table 5, we derive three core findings: **(1) Diminishing returns from scaling necessitate careful efficiency-performance trade-offs.** When scaling PRM parameters from 1.5B to 7B, CTRL achieves a substantial average performance gain of 13.7 percentage points (50.4 to 64.1), while per-step time overhead increases by only 5.6. However, further scaling from 7B to 72B yields merely a 0.9 percentage point improvement in average performance, yet incurs a 27 increase in time cost (238.0s to 302.7s), reflecting the effectiveness of sparse activation or inference optimization techniques. API-based ultra-large models saturate performance at 65.1–65.5, but their per-step time cost surges by 58%–102% relative to the 7B baseline, indicating that 7B-scale PRMs already suffice for high-quality trajectory filtering. **(2) Domain-specialized PRMs achieve comparable performance to general-purpose large models with superior efficiency.** On mathematical reasoning tasks, Qwen2.5-Math-PRM-7B trails DeepSeek-V4-Pro by only 0.5–0.7 percentage points, while its inference latency is merely 63 of the latter, making it more suitable for local deployment. **(3) Architecture alignment and task specificity jointly influence performance.** DeepSeek-V4-Pro achieves the best results on AMC (82.1), GPQA (55.4), and the aggregate metric (65.5), demonstrating the advantage of cross-architecture large models on general reasoning tasks. Qwen3.6-plus leads on pure mathematical tasks such as AIME24 (34.7) and MATH (91.2), suggesting that homogeneous architecture combined with novel training strategies facilitates positive transfer to domain-specific tasks. Qwen2.5-72B consistently maintains second-best performance across most metrics, validating the principle that increased scale yields stable but diminishing gains.

In summary, 7B-scale domain-specialized PRMs (e.g., Qwen2.5-Math-PRM-7B) strike the optimal balance among performance, efficiency, and deployment convenience. For scenarios pursuing peak performance with abundant computational resources, 72B-scale or next-generation architecture models represent viable alternatives. For cross-domain general tasks, API-based ultra-large models

can serve as high-performance options, albeit with careful consideration of latency overhead and service dependency risks.

G.5 Additional ablation experiments

Table 6: PRM-Budget-Controlled Ablation on Qwen3-4B-Base with. All variants use identical PRM call budgets per prompt.

Method	Avg. Acc.↑	FM	Pseudo-Label Acc.↑
TTRL-Base	41.0	-1.72	20.6
TTRL+PRM-Score	44.2	-1.24	24.1
TTRL+PRM-Filter	48.7	-0.89	27.3
CTRL (Ours)	**52.9**	**-0.31**	**31.8**

To isolate the contribution of algorithmic design from increased computational budget, we introduce two baselines matched to CTRL's PRM compute budget: **(1) TTRL+PRM-Score** performs the same number of PRM forward passes as CTRL to compute trajectory-level quality scores $R(s)$ and $C(s)$ for each rollout, but does not utilize these scores for decision-making, retaining the standard majority-voting pseudo-label construction and GRPO update. **(2) TTRL+PRM-Filter** replaces majority voting with a single-stage quality filter that selects the highest-scoring trajectory satisfying $R(s) \geq \gamma_R$ and $C(s) \geq \gamma_C$ as the supervision target, while excluding CTRL's hierarchical fallback, adaptive gating, posterior correction, and gradient projection.

Table 6 reports the results on Qwen3-4B-Base with strictly matched PRM calls per prompt. We observe three key findings: **(i) Scoring alone provides marginal improvement.** TTRL+PRM-Score increases average accuracy by only 3.2% over TTRL-Base and exhibits substantial forgetting (FM $= -1.24$), indicating that computing fine-grained rewards without altering the selection or update mechanism cannot rectify erroneous consensus or mitigate gradient bias. **(ii) Single-stage filtering is insufficient.** TTRL+PRM-Filter achieves a moderate gain (+4.5% over the scoring variant) by leveraging PRM signals for trajectory selection. However, it still suffers from notable forgetting (FM $= -0.89$) and degrades on complex tasks, suggesting that static filtering lacks the adaptability required for noisy or ambiguous samples. **(iii) The full CTRL framework yields substantial gains.** Under identical PRM-call budgets, CTRL outperforms TTRL+PRM-Filter by 4.2% in accuracy and reduces forgetting by 2.9× (FM: -0.89 to -0.31), while achieving the highest pseudo-label accuracy (31.8 vs. 27.3). This performance gap demonstrates that CTRL's hierarchical selection, adaptive gating, and conflict-aware projection are necessary to mitigate the coupled degradation loop, an outcome that cannot be achieved through uniform compute scaling or naive filtering alone.

Overall, these findings demonstrate that Overall, these findings demonstrate that CTRL's gains do not merely come from additional PRM scoring. Instead, the improvement stems from its algorithmic design, including hierarchical selection, adaptive gating, and conflict-aware projection.

G.6 Task Order Sensitivity

To evaluate whether CTRL's performance depends on a specific task sequence, we test three coarse grained orders on Qwen3-1.7B-Base: *Math→General→Medical* (original), *General→Math→Medical*, and *Medical→General→Math*. Table 7 reports the results. Two key observations emerge. First, CTRL exhibits low order sensitivity. Its average accuracy varies by only 1.2% across orders with a standard deviation of 0.42, and the Forgetting Measure remains consistently low, ranging from -0.31 to -0.38. This stability stems from the cognitive anchor replay and gradient projection mechanisms, which explicitly decouple current updates from historical knowledge regardless of task arrival order. Second, TTRL suffers from strong order dependency. Its accuracy drops by 4.8% with a standard deviation of 1.87, and the Forgetting Measure degrades from -1.85 to -2.41 when the order changes. This confirms that naive majority voting pseudo labels amplify error accumulation when task distributions shift abruptly. These results validate that CTRL's algorithmic design, rather than a fortuitous task sequence, drives its robustness in continual TTRL.

Table 7: Order sensitivity analysis on Qwen3-1.7B-Base. We evaluate three coarse-grained task orders while keeping all other hyperparameters fixed. Lower std indicates better order-robustness.

Task Order	Avg. Acc.↑	FM	Acc. Std.↓
CTRL (Ours)			
Math → General → MedQA (Original)	51.0	**-0.31**	–
General → Math → MedQA	50.3	-0.35	–
MedQA → General → Math	49.8	-0.38	**0.42**
TTRL (Baseline)			
Math → General → MedQA (Original)	38.4	-1.85	–
General → Math → MedQA	35.1	-2.13	–
MedQA → General → Math	33.6	-2.41	1.87

Table 8: Key hyperparameter settings for CTRL.

Key Hyperparameter	**Value**
Number of rollouts N	32
Reliability bottom steps m	3
Advantage clipping c_{adv}	5.0
Reliability threshold γ_R	0.55
Consistency threshold γ_C	0.60
Majority-vote update threshold γ_{mv}	0.50
Reliability update threshold $\gamma_{R,\text{upd}}$	0.45
OPD coefficient α_{opd}	0.5
Length penalty coefficient	0.01
Reliability weight λ_R	1.0
Consistency weight λ_C	0.5
Length weight λ_L	0.01
Step masking threshold γ_{step}	0.5
Old-step masking threshold $\gamma_{\text{step}}^{\text{old}}$	0.4
Posterior correction	True

H Hyperparameter settings

I Algorithm of CTRL

In this section, we present the complete algorithm of CTRL, as shown below.

Algorithm 1 CTRL-local: PRM-Guided Test-Time Reinforcement Learning

Require: policy π_θ; PRM scorer $\mathcal{P}$; prompts $\mathcal{D}$; hyperparameters $B, N, \gamma_R, \gamma_C, \gamma_{mv}, \gamma_{R,u}, \gamma_{step}, \lambda_R, \lambda_C, \lambda_L, \alpha_{opd}$

Ensure: Updated policy π_θ

Initialize policy model $\pi_\theta \leftarrow \pi_{\theta_{\text{init}}}$; memory $\mathcal{M} \leftarrow \emptyset$
for iteration $= 1, \ldots, I$ **do**
  Sample prompt batch $\{x_i\}_{i=1}^{B}$ from $\mathcal{D}$
  Generate K rollouts $\{y_{i,j}\}_{j=1}^{K} \sim \pi_\theta(\cdot \mid x_i)$ for each x_i
  Query PRM: $\mathbf{s}_{i,j} \leftarrow \mathcal{P}(x_i, y_{i,j})$; compute $R_{i,j}, C_{i,j}$, extract $a_{i,j}$
  for each prompt i **do**
    Compute majority answer $\hat{a}_i$ and consensus strength v_i
    Select gold trajectory $j_i^\star = \arg\max_j(\lambda_R R_{i,j} + \lambda_C C_{i,j} - \lambda_L \log|y_{i,j}|)$
      subject to $R_{i,j} \geq \gamma_R$, $C_{i,j} \geq \gamma_C$, $a_{i,j} = \hat{a}_i$; fallback if no candidate qualifies
    Set pseudo-label $a_i^\star \leftarrow a_{i,j_i^\star}$; store $j_i^\star$ into $\mathcal{M}$
    Update gate: $u_i \leftarrow \mathbf{1}[v_i \geq \gamma_{mv} \ \wedge \ \max_j R_{i,j} \geq \gamma_{R,u}]$
  end for
  Assign binary rewards: $r_{i,j} \leftarrow \mathbf{1}[a_{i,j} = a_i^\star]$
  if any prompt lacks a qualifying gold trajectory **then**
    Run posterior correction: re-generate guided rollout, accept if $R \geq \gamma_R \wedge C \geq \gamma_C$
  end if
  Build OPD batch from gold trajectories with step mask $w_{t,\tau} \leftarrow \mathbf{1}[\mathbf{s}_{i,j^\star}^{(\tau)} \geq \gamma_{step}]$
  if replay enabled and $|\mathcal{M}| \geq$ threshold **then**
    Sample replay items from $\mathcal{M}$; build replay step mask with γ_{step}^{old}
  end if
  Compute total loss on prompts where $u_i = 1$:
  $\mathcal{L} = \mathcal{L}_{RL} + \alpha_{opd}\,\mathcal{L}_{OPD} + \mathcal{L}_{replay}$
  Apply gradient-conflict projection if task/replay gradients conflict
  Update $\theta \leftarrow \theta - \eta\,\nabla_\theta \mathcal{L}$
end for
Output π_θ

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We provide a detailed summary of contributions in the abstract and introduction, making them easy for readers to identify.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   [Yes]

Justification: We provide a discussion of the paper's limitations in the appendix, making it easy for readers to find.

Guidelines:

- The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: In the appendix, we provide detailed proofs and analyses for each theory presented in the paper, allowing readers to easily understand our theories and their corresponding proof procedures.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

Justification: In the appendix, we provide all the parameter settings and detailed experimental details needed to reproduce our results. In the abstract, we also include an anonymous repository link containing the implementation code for CTRL, allowing reviewers and authors to easily replicate the paper's results.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide an anonymous repository link containing the implementation code at the end of the abstract. The accompanying README file specifies the required environment and provides detailed step-by-step instructions for reproducing the results. Furthermore, Appendix D.4 presents comprehensive implementation settings for our method, and Appendix F provides a table of key parameter configurations. These materials collectively ensure the reproducibility of the main experimental results reported in our paper.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).

- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide all the training and test implementation details in Appendix D.4, along with detailed hyperparameter settings in Appendix F.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We use the deterministic accuracy of pass@k as the primary metric, and each experimental result is reported as the mean value obtained from five runs.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: In Appendix D.4, Implementation Details, we specify that the computational resource required for reproduction is a server equipped with 8 A800 80G GPUs.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

Answer: [Yes]

Justification: This research conforms to the NeurIPS Code of Ethics in every respect. The study involves no human subjects or sensitive data, and all experimental setups do not raise any privacy, fairness, or safety concerns. The collected data and code are released under open licenses to ensure reproducibility and transparency, and we have carefully considered potential broader impacts, which are discussed in the paper.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: We discuss the broader impact in detail in Appendix B.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [No]

    Justification: All experiments in our paper are conducted on publicly available datasets, and there is no high-risk data or model for misuse. All models used are open-source models.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: For the public datasets and open-source model weights used in this paper, we have provided the relevant citations and strictly adhered to all corresponding terms and licenses.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: We release the implementation of the CTRL framework as an anonymized code repository. The repository includes a comprehensive README with environment setup, step-by-step reproduction scripts, hyperparameter configurations, and an open-source license (e.g., MIT/Apache 2.0). All external datasets and base models are properly cited with their original licenses, and no new datasets or model checkpoints are introduced.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: [N/A]

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: [N/A]

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: Large language models are central to our methodology: (1) Qwen3-Base series and Llama3-8B serve as policy backbones for continual test-time adaptation; (2) PRM-based reward estimation relies on LLMs (e.g., Qwen2.5-Math-PRM, Skywork-PRM) for fine-grained trajectory scoring; (3) the proposed CTRL framework is specifically designed to address error accumulation and catastrophic forgetting in LLM-based continual TTRL. All model details, licensing, and usage protocols are described in Section B.3 (Implementation Details) and the main methodology.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.