Temporal Sampling for Forgotten Reasoning in LLMs

Yuetai Li[♣*] **Zhangchen Xu**[♣*] **Fengqing Jiang**[♣] **Bhaskar Ramasubramanian**[♠]
Luyao Niu[♣] **Bill Yuchen Lin**[♣] **Xiang Yue**[♢] **Radha Poovendran**[♣]

[♣]University of Washington [♢]Carnegie Mellon University [♠]Western Washington University

Abstract

Fine-tuning large language models (LLMs) is intended to improve their reasoning capabilities, yet we uncover a counterintuitive effect: models often forget how to solve problems they previously answered correctly during training. We term this phenomenon *Temporal Forgetting* and show that it is widespread across model sizes, fine-tuning methods (both Reinforcement Learning and Supervised Fine-Tuning), and multiple reasoning benchmarks. Our analysis reveals on average more than 20 % of final errors were once solved correctly at an earlier checkpoint. Inspired by the phenomenon of Temporal Forgetting, we proposed *Temporal Sampling*, a simple decoding strategy that draws outputs from multiple checkpoints along the training trajectory. This approach recovers forgotten solutions and leads to significant improvements in reasoning performance than final-ckpt-sampling only, gains from 4 to 19 points in Pass@k and consistent gains for majority-voting and Best-of-N across several benchmarks. Temporal sampling also outperforms strong baselines such as model merging. By leveraging the temporal diversity inherent in training, Temporal Sampling offers a practical, compute-efficient way to surface hidden reasoning ability and rethink how we evaluate LLMs.

1 Introduction

Fine-tuning large language models (LLMs) is expected to improve their reasoning ability (Luo et al., 2025; DeepSeek-AI et al., 2025; Zeng et al., 2025; Muennighoff et al., 2025; NovaSky, 2025; Jin et al., 2025). Yet, we uncover a surprising phenomenon: *models often forget how to solve problems they previously solved correctly during fine-tuning*. We refer to this systematic behavior as *Temporal Forgetting*.

Temporal Forgetting is not rare or model-specific. To quantify this phenomenon, we introduce a new metric: the Temporal Forgetting Score (P_{TFS}). P_{TFS} captures the percentage of questions in the benchmark that were answered correctly by *some* checkpoint during RL/ SFT but were ultimately answered incorrectly by the *final* checkpoint. Across Supervised Fine-Tuning (SFT) and Reinforcement Learning (RL) fine-tuning (Shao et al., 2024a; DeepSeek-AI et al., 2025; Zeng et al., 2025) of Qwen2.5 models (1.5B and 7B) on multiple reasoning benchmarks (AIME, AMC, OlympiadBench (He et al., 2024), MATH-500 (Hendrycks et al., 2021a), GPQA (Rein et al., 2024)), we find that on average more than 20% of final errors were once solved correctly at an earlier checkpoint. This pattern persists across different model sizes, architectures, and training approaches.

This metrics highlight a fundamental limitation in current evaluation methodologies. Standard metrics like Pass@k (Chen et al., 2021), Majority@k (Wang et al., 2023) and Best-of-N (Snell et al., 2024a), computed only on the final model, implicitly assume that checkpoint to be the model's most capable state. However, our findings reveal that many correct reasoning paths are transient, making final-checkpoint-only evaluation a narrow and often misleading lens. The significant Temporal Forgetting Score suggests that the reasoning potential of fine-tuned models are substantially underestimated when using only the final checkpoint.

Inspired by this, we proposed *Temporal Sampling*, a simple decoding strategy that samples completions across multiple checkpoints rather than just the final one, which is shown in Figure 1 (b). By spreading the sample budget across time, Temporal Sampling recovers forgotten solutions without retraining or ensembling. Temporal Sampling yields substantial improvements across diverse reasoning tasks. On benchmarks such as AIME2024, AMC, and AIME2025, we observe gains from 4 to

[0*]These authors contributed equally to this work.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 28309–28327

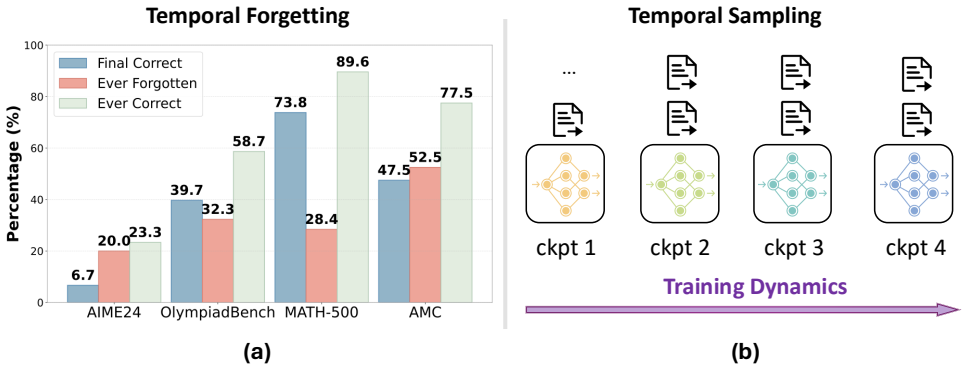

Figure 1: (a) We observed that during RL training of Qwen2.5-7B, many benchmark problems answered correctly by *some* intermediate checkpoint were ultimately incorrect in the *final* checkpoint. We term this phenomenon as **Temporal Forgetting**. (b) We proposed **Temporal Sampling**: This method utilizes training dynamics as a source of answer diversity by distributing inference samples across multiple distinct checkpoints from the training trajectory, rather than relying solely on the single final checkpoint.

19 points in Pass@k compared to final-checkpoint-only sampling, and consistent improvements in Majority@k and Best-of-N.

These findings suggest that true model competence may not reside in a single parameter snapshot, but rather in the collective dynamics of training itself. Temporal Sampling offers a practical and powerful way to reclaim lost reasoning ability, challenging the standard paradigm of using only the final model checkpoint for evaluation and deployment.

2 Temporal Forgetting: Correct Answers Emerge and Vanish in Training

2.1 Overall Performance Score cannot Tell Everything

To understand how RL or SFT alters a model's ability to correctly answer reasoning problems, we investigate instances where base models succeeded on questions but failed after fine-tuning. To quantify this, we introduce the **Lost Score**:

- P_{Lost} (**Lost Score**): The percentage of questions in a benchmark that were answered correctly by the base model but incorrectly by the model after fine-tuning.

This score specifically highlights the phenomenon where a model, despite any overall performance changes after fine-tuning, loses its correctness on certain problems it previously solved correctly. Note that overall performance scores cannot capture the statistical pattern reflected by P_{Lost}.

Experiment Setup. We consider various existing SOTA model such as DeepScaleR-1.5B (Luo et al., 2025), OpenR1-7B (Face, 2025) and S1-32B (Muennighoff et al., 2025). Please see Appendix D.2 for the full list of evaluated models and their base models. We calculate the overall performance of various SOTA models after fine-tuning (denoted P_{FT}), the performance of their corresponding base model (denoted P_{Base}), and our proposed Lost Score (P_{Lost}). These evaluations were conducted on the OlympiadBench (He et al., 2024), MATH-500 (Hendrycks et al., 2021b), and GPQA (Rein et al., 2024) benchmarks. We excluded AIME2024 and AMC2023 from this particular analysis because the number of questions available in these datasets was insufficient for a meaningful comparison. To minimize variability arising from different sampling methods during evaluation, we employ greedy sampling following (Wei et al., 2022).

Results. Figure 2 demonstrates that although OpenR1-7B improves OlympiadBench performance from 42.5 to 56.6, a notable percentage of questions ($P_{Lost} = 9.2$) were correctly solved by the base model but incorrectly by the fine-tuned model. In Table 1, we present a comprehensive analysis of various SOTA models. We found that P_{Lost} could range from 6.1 to 16.0 points, with the average of 9.5 points. This implies that there are a considerable number of questions answered correctly by the base model but incorrectly after RL or SFT, in spite of the improvement of overall performance. Additionally, we demonstrate more experiments results regarding different sampling methods for various SOTA models, detailed results of which are included in Appendix D.2.

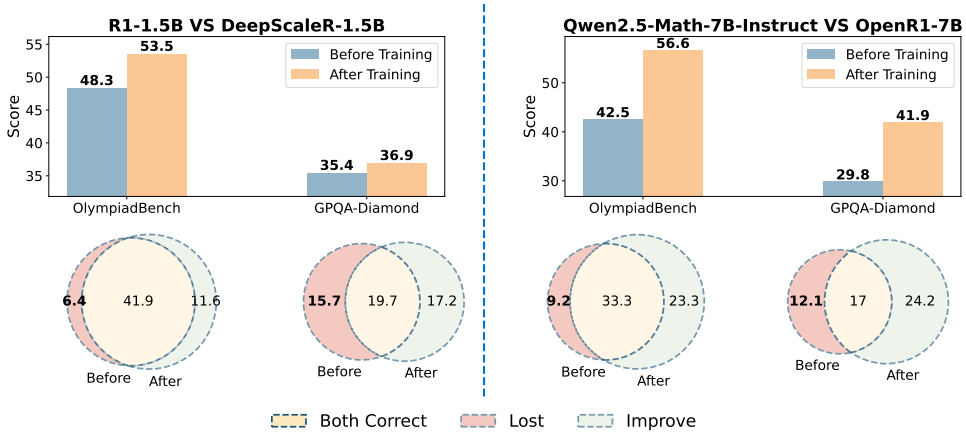

Figure 2: Fine-tuned models like DeepscaleR-1.5B (Luo et al., 2025) and OpenR1-7B (Face, 2025) outperform the base model overall but also forget many questions the base model answered correctly.

Table 1: Performance of the base model ($P_{\text{Base}} \uparrow$), the fine-tuned model ($P_{\text{FT}} \uparrow$) and the Lost Score ($P_{Lost} \downarrow$) for different SOTA models. We observed that in spite of the improvement of overall performance, the average P_{Lost} ranges from 6.1 to 16.0, which implies a high percentage of questions answered correctly by the base model is answered incorrectly after RL or SFT. To minimize variability caused by random fluctuations in model performance from diverse sampling, we employed greedy sampling following (Wei et al., 2022). Please see Appendix D.2 for more training details of each model.

Model	**OlympiadBench**			**MATH-500**			**GPQA-Diamond**			**Avg.** P_{Lost}
	P_{Base}	P_{FT}	P_{Lost}	P_{Base}	P_{FT}	P_{Lost}	P_{Base}	P_{FT}	P_{Lost}	
DeepScaleR-1.5B (Luo et al., 2025)	48.3	53.5	6.4	82.0	89.8	2.4	35.4	36.9	15.7	8.2
Still-1.5B (Team, 2025b)	48.3	48.4	8.6	82.0	83.8	5.0	35.4	34.8	17.2	10.3
S1.1-1.5B (Muennighoff et al., 2025)	18.7	11.7	11.1	46.2	37.6	19.2	23.2	16.2	17.7	16.0
II-thought-1.5B (Internet, 2025)	48.3	58.4	5.3	82.0	88.0	3.4	35.4	34.3	16.7	8.5
S1.1-3B (Muennighoff et al., 2025)	29.8	24.7	12.4	65.0	64.8	10.2	32.8	30.3	18.7	13.8
SmallThinker-3B	29.8	38.2	6.2	65.0	69.2	9.8	32.8	28.3	21.7	12.6
S1.1-7B (Muennighoff et al., 2025)	40.4	42.2	10.5	76.0	76.8	7.8	32.8	41.4	15.2	11.2
OpenR1-Qwen-7B (Face, 2025)	42.5	56.6	9.2	83.0	89.8	3.8	29.8	41.9	12.1	8.4
OpenThinker-7B (Team, 2025a)	40.4	48.7	8.1	76.0	85.0	4.2	32.8	43.9	13.6	8.6
S1-32B (Muennighoff et al., 2025)	49.8	60.1	4.3	81.6	89.6	3.2	43.9	55.1	13.1	6.9
Sky-T1-32B-Preview (NovaSky, 2025)	49.8	58.4	4.6	81.6	88.2	3.0	43.9	53.0	11.1	6.2
Bespoke-Stratos-32B	49.8	54.2	7.1	81.6	89.2	3.0	43.9	57.6	8.1	6.1
OpenThinker-32B (Team, 2025a)	49.8	61.2	8.0	81.6	91.4	2.8	43.9	59.1	11.1	7.3

2.2 Temporal Forgetting

To investigate how answer correctness evolves during post-training, we conducted SFT and RL on various base models, evaluating checkpoints at different training steps. We introduce two metrics to quantify the temporal dynamics: the **Ever Correct Score** and the **Temporal Forgetting Score**:

- P_{ECS} (**Ever Correct Score**): The percentage of questions in the benchmark that were answered correctly by *at least one* checkpoint saved during RL/SFT.
- P_{TFS} (**Temporal Forgetting Score**): The percentage of questions in the benchmark that were answered correctly at least once by *some* checkpoint during RL/SFT but were ultimately answered incorrectly by the final checkpoint. Mathematically, $P_{TFS} = P_{ECS} - P_{FT}$, where P_{FT} is the performance score of the fine-tuned model.

Furthermore, to characterize how answer correctness changes between consecutive checkpoints, we define specific events: an answer is considered to "Forget" if it shifts from correct to incorrect, and "Improve" if it transitions from incorrect to correct. If the answer's correctness status (either correct or incorrect) remains unchanged across two consecutive checkpoints, it is labeled as "Both Correct/Wrong."

Experiment Setup. We performed GRPO (Shao et al., 2024b) on the Qwen2.5-7B, Qwen2.5-1.5B, and Qwen2.5-Math-7B models (Yang et al., 2024a,b). The training data consisted of 4k sam-

ples randomly selected from the DeepscaleR-40k dataset (Luo et al., 2025). Throughout the training of each model, we saved 8 checkpoints. We set the RL training parameters following (Luo et al., 2025), and detailed training script parameters can be found in Appendix C. For SFT, we utilized the same DeepscaleR-4k sampled data. We then employed QwQ-Preview-32B (Qwen Team, 2024) for rejection sampling to obtain correct responses (Dong et al., 2023), subsequently fine-tuning each model on this curated dataset. We evaluated the performance of various checkpoints from the training process on five benchmarks: AIME24, AMC, MATH-500, OlympiadBench, and GPQA-Diamond. To minimize variability caused by random fluctuations in model performance from diverse sampling, we employed greedy sampling following (Wei et al., 2022).

Results. Table 2 presents the Ever Correct Score P_{ECS} and Temporal Forgetting Score P_{TFS} of different models after RL or SFT. We observed that a substantial number of questions were correctly answered at some checkpoint during the training process but were answered incorrectly by the final checkpoint (measured by a significantly high P_{TFS}). Surprisingly, we found that P_{TFS} ranges from 6.4% to 56.1%, with average as high as 25 points. This implies that, on average, up to 25% of the questions in a benchmark were correctly solved by the model at some checkpoint during training but were incorrect in the final output. Please see Appendix D.3 for base model performance and more benchmark results including AIME24 and AMC.

We further evaluate models on TheoremQA (Chen et al., 2023), which is a free-form QA dataset with 800 questions spanning diverse natural science domains. The model must generate full reasoning traces and answers without selecting from pre-defined options. We report results of P_{ECS} and P_{TFS} for RL models on TheoremQA. Notably, over 20% of questions exhibit temporal forgetting, i.e., they were answered correctly by an earlier checkpoint but incorrectly by the final checkpoint. This demonstrates that the temporal forgetting phenomenon persists beyond multiple-choice settings of GPQA-D.

We further demonstrate the phenomenon of **Forgetting Dynamics**: model oscillates between correct and incorrect answers across training checkpoints. Please see Figure 11 in Appendix D.3 for more details.

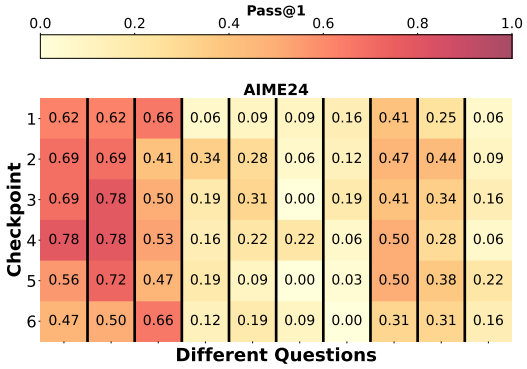

Figure 3: Pass rate distribution across training checkpoints on AIME24. Individual problems show varying pass rates over time. Temporal Sampling exploits these dynamics to improve answer diversity at inference.

Comparison with Catastrophic Forgetting. Catastrophic Forgetting (Luo et al., 2023) focuses on cross-domain degradation, while Temporal Forgetting in our work refers to in-domain degradation over time. Thus, the underlying problem formulations may differ in these two phenomenon. Additionally, in contrast to Catastrophic Forgetting where overall performance drops markedly, Temporal Forgetting focuses on changes in correctness at the individual question level, in spite of the improvement of overall performance. Thus temporal forgetting cannot be directly captured by the overall performance score only.

3 Temporal Sampling: Scaling Inference Compute over Checkpoints

3.1 Temporal Sampling

Inspired by the observed learning and forgetting dynamics during model training, we propose **Temporal Sampling**. Temporal Sampling utilizes the evolving state of the model across different training checkpoints as a source of diversity for answer generation at inference time. Specifically, instead of relying solely on the final checkpoint, k samples are generated by allocating the sampling budget across t distinct training checkpoints according to a chosen distribution strategy.

Temporal Sampling typically selects the t most recent available checkpoints, which are then ordered from latest (e.g., the final checkpoint) to the t-th latest. While various methods can be employed to distribute the k sampling attempts among these checkpoints, this paper primarily focuses on a round-robin allocation. In this approach, sampling

Table 2: Performance of fine-tuned models (P_{FT} ↑), the Ever Correct Score (P_{ECS} ↑), and the Temporal Forgetting Score (P_{TFS} ↓) of different models after GRPO or SFT. We observed both high P_{ECS} and P_{TFS}, which implies a high percentage of questions (more than 20% on average) are answered correctly at some checkpoint during training but are ultimately incorrect in the final checkpoint. Please see the base model performance and more benchmark results in Appendix D.3.

Model	OlympiadBench			MATH-500			GPQA-Diamond			Avg. P_{TFS}
	P_{FT}	P_{ECS}	P_{TFS}	P_{FT}	P_{ECS}	P_{TFS}	P_{FT}	P_{ECS}	P_{TFS}	
Qwen2.5-7B (GRPO)	39.7	58.7	19.0	73.8	89.6	15.8	33.8	74.7	40.9	25.2
Qwen2.5-7B (SFT)	40.1	55.8	15.7	69.8	86.6	16.8	25.3	81.4	56.1	29.5
Qwen2.5-1.5B (GRPO)	18.8	36.1	17.3	55.6	73.0	17.4	26.8	72.3	45.5	26.7
Qwen2.5-1.5B (SFT)	11.0	26.0	15.0	36.2	66.0	29.8	13.1	65.1	52.0	32.3
Qwen2.5-Math-7B (GRPO)	41.0	57.3	16.3	79.8	86.2	6.4	32.8	71.7	38.9	20.5
Qwen2.5-Math-7B (SFT)	43.9	62.9	19.0	76.4	90.4	14.2	30.8	79.8	49.0	27.4

Table 3: Temporal Forgetting on TheoremQA (Chen et al., 2023), a free-form QA dataset with 800 questions spanning diverse natural science domains. The model must generate full reasoning traces and answers without selecting from pre-defined options. This demonstrates that the temporal forgetting phenomenon persists beyond multiple-choice settings of GPQA-D.

Model	TheoremQA			
	P_{Base}	P_{FT}	P_{ECS}	P_{TFS}
Qwen2.5-7B (GRPO)	36	37.4	59.3	21.9
Qwen2.5-Math-7B (GRPO)	35.8	37.1	57.5	20.4

commences with the latest checkpoint for the first sample, the next latest for the second, and so on, cycling through the ordered sequence. This procedure defaults to conventional sampling (from only the final checkpoint) when $t = 1$. Please see more experiment results in Appendix D.4 for choosing checkpoints in different orders.

3.2 Metric $Pass@k|t$

To better measure the performance of Temporal Sampling, we introduce a new metric, $Pass@k|t$. This metric is defined as the probability of obtaining at least one correct answer when k samples are drawn from t checkpoints. Although samples may be drawn in various ways, in what follows we adopt a round-robin manner: we first give the formal definition of $Pass@k|t$ under this distribution way and then derive the unbiased estimator.

Definition. Let $r_{i,j}$ denote the $Pass@1$ rate (i.e., the probability of correctness with a single sample) for the j-th checkpoint on the i-th problem. We define

$$Pass@k|t = \mathop{\mathbb{E}}_{i}\left\{1 - \prod_{j=1}^{t}(1 - r_{i,j})^{k_j}\right\}$$

where $\sum_j k_j = k$ and $\{k_j\}$ is the *Balanced Integer Partition* of k on t (Andrews and Eriksson, 2004):

$$k_j = \begin{cases} \lfloor k/t \rfloor + 1 & \text{if } j \leq (k \pmod t) \\ \lfloor k/t \rfloor & \text{if } j > (k \pmod t) \end{cases}$$

Note that if $t = 1$, this reduces to the standard definition of $Pass@k$ (Chen et al., 2021).

Unbiased Estimation. We provide an unbiased estimator for $Pass@k|t$. Let N be the total number of candidate samples generated for evaluation from each checkpoint j on a problem i. Let $C_{i,j}$ be the number of correct samples among these N candidates for problem i from checkpoint j. The unbiased estimation can be expressed as:

$$Pass@k|t = \mathop{\mathbb{E}}_{i}\left\{1 - \prod_{j=1}^{t}\left(\frac{\binom{N-C_{i,j}}{k_j}}{\binom{N}{k_j}}\right)\right\}$$

The proof of this estimator's unbiased nature is provided in Appendix B.

3.3 Experiment Setup

To evaluate the efficacy of Temporal Sampling, we conducted experiments on benchmarks including AIME2024, AMC2023, and AIME2025. We utilized GRPO to fine-tune the Qwen-7B-Base model on the DeepScaleR-4k dataset, following the training settings detailed in (Luo et al., 2025). For each problem, we generated 64 responses using diverse sampling with a temperature of 0.6, top-p of 0.95, and a maximum token length of 16384 (Yue et al., 2025).

We saved 8 checkpoints during the RL training phase, which constituted the checkpoint pool for our Temporal Sampling. As baselines, we considered the standard $Pass@k$ (Chen et al., 2021) and Maj@k (self-consistency, also known as majority

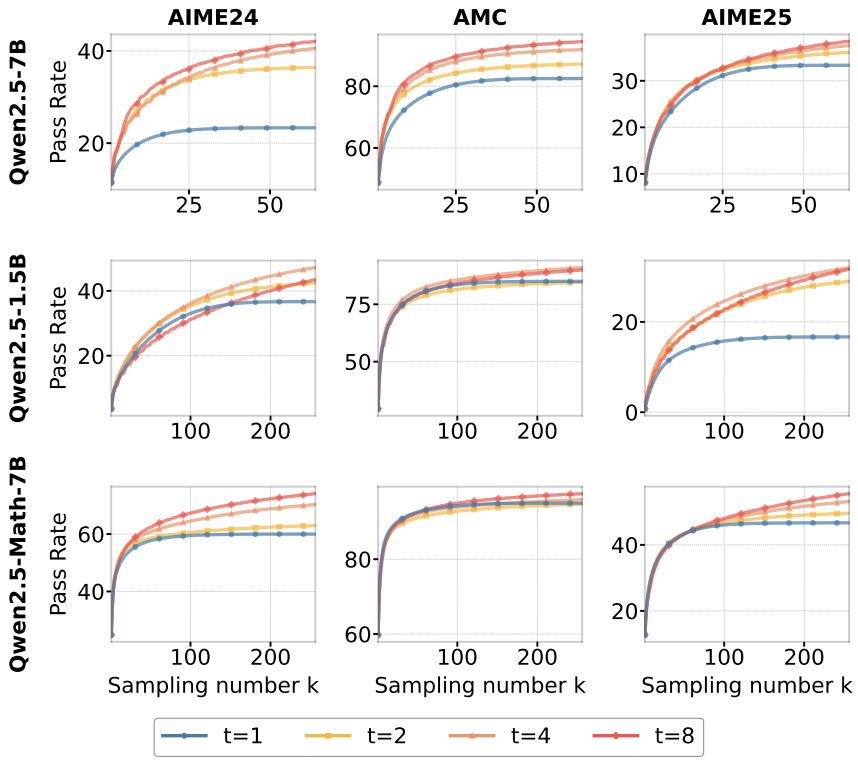

Figure 4: Pass@k for different numbers of checkpoints t on the AIME2024, AMC, and AIME2025 benchmarks when using Temporal Sampling. The case $t = 1$ represents the baseline of standard $Pass@k$ sampling on the final checkpoint. Our proposed Temporal Sampling for Qwen2.5-7B with $t = 8$ outperforms the baseline by more than 19, 13, and 4 percentage points on AIME2024, AMC, and AIME2025, respectively, when sampling 64 responses.

voting) (Wang et al., 2023). For Maj@k, we followed the Majority Voting (Wang et al., 2023) by generating k samples and selecting the most frequent answer as the final model output. We denote our Temporal Sampling variants as $Pass@k|t$ and Maj@$k|t$. For Best-of-N (BoN) sampling, we follow (Snell et al., 2024a) and select answers with the highest score given by the reward model as the final output. When $t = 1$, Pass@$k|t$ Maj@$k|t$, and BoN with temporal sampling are equivalent to the baseline settings that samples only on the final checkpoint.

3.4 Temporal Sampling Achieves Higher Sampling Performance

Figure 4 demonstrates that Temporal Sampling achieves higher sampling performance (as measured by $Pass@k|t$) compared to the baseline of sampling only on the final checkpoint, under identical computational budgets. These advantages are consistently observed across the AIME2024, AIME2025, and AMC benchmarks. For instance $Pass@k|8$ of Qwen2.5-7B results in a pass rate that is over 19 percentage points higher than that of sampling only on the final checkpoint on AIME24 when $k = 64$. The enhanced efficiency of Temporal Sampling is further highlighted by its ability to reach a 22.5% pass rate with only $k = 5$ samples, a level that requires $k = 64$ samples for $t = 1$.

3.5 Temporal Sampling Improves Performance of Inference-Time Scaling

Figure 5 demonstrates that Temporal Sampling markedly enhances the performance of majority voting (measured by $Maj@k|t$). Across the AIME2024, AIME2025, and AMC benchmarks, employing a greater number of checkpoints (t) within the Temporal Sampling framework leads to improved accuracy compared to the baseline Maj@k only sampling on the final checkpoint under identical computational budgets. Specifically, at $k = 64$, $Maj@k|8$ achieves an accuracy exceeding 21, substantially outperforming the 13% accuracy of the baseline.

Figure 6 demonstrates the effectiveness of Temporal Sampling when combined with Best-of-N (BoN) decoding on the AIME2024, AMC, and AIME2025 benchmarks. We use Qwen2.5-Math-PRM-72B following (Zhang et al., 2025) as the process reward model. The results clearly show that

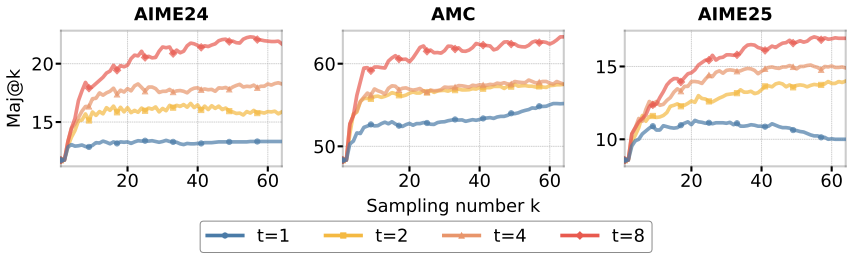

Figure 5: Maj@k (Majority voting) for different numbers of checkpoints t on the AIME2024, AMC, and AIME2025 benchmarks using Temporal Sampling. The case $t = 1$ represents the baseline of standard majority voting sampling on the final checkpoint. Our proposed Temporal Sampling with $t = 8$ checkpoints outperforms the baseline by more than 8, 7, and 7 percentage points on AIME2024, AMC, and AIME2025, respectively, when sampling 64 responses.

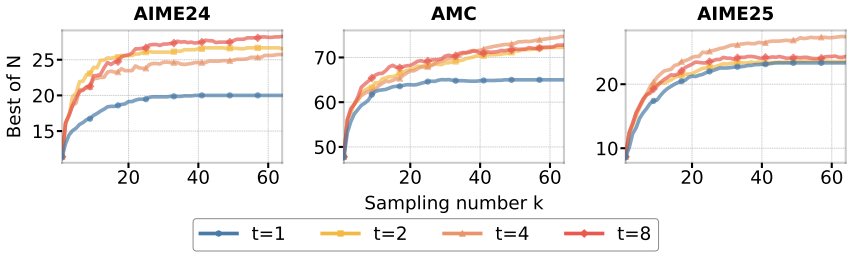

Figure 6: Best-of-N decoding on the AIME2024, AMC, and AIME2025 benchmarks using Temporal Sampling. The case $t = 1$ represents the baseline of standard BoN sampling on the final checkpoint. Our proposed Temporal Sampling with $t = 8$ checkpoints outperforms the baseline by more than 7, 8, and 1 percentage points on AIME2024, AMC, and AIME2025, respectively, when sampling 64 responses.

Temporal Sampling with $t = 8$ checkpoints significantly outperforms the baseline ($t = 1$), achieving improvements of more than 7, 8, and 1 percentage points across the three benchmarks when sampling $k = 64$ responses. We present more results of Best-of-N sampling with different reward models in Appendix D.1.

3.6 More Ablations

Comparison with Model Merge. We merged the last t checkpoints of training to form a single model and compared it against temporal sampling over the same t checkpoints. As shown in Table 4, temporal sampling consistently outperforms checkpoint merging across all benchmarks (AIME2024, AMC, AIME2025), particularly at larger sampling sizes (pass@64 and maj@64). For example, when t = 4: on pass@64, temporal sampling exceeds checkpoint merging by 10.7, 9.5, and 7.6 on AIME2024, AMC, and AIME2025, respectively. On maj@64, the gains are 3.4, 4.6, and 1.0, respectively. These results suggest that sampling diverse responses from temporally spaced checkpoints yields greater performance than simply averaging weights.

Comparison with Mixture of Models (MoM). We evaluate our proposed Temporal Sampling against the Mixture of Models, which combines outputs from different foundation models to answer each question collaboratively. To compare sampling efficiencies, we construct a model pool containing three models: our RL-trained final checkpoint (Qwen2.5-7B-Base), Llama 3.1-8B, and DeepSeek-Math-7B-Instruct. We apply Temporal Sampling (with $t = 3$) and the mixture strategy by sampling in a round-robin manner over the pool, then measure the majority voting performance Maj@k. As shown in Figure 7, Temporal Sampling achieves higher sampling performance than the mixture of models under the same computational budget. At Maj@64, Temporal Sampling outperforms the mixture approach by over 4, 9, and 9 points on the AIME24, AMC, and AIME25 benchmarks, respectively.

Ablation on Different Models We trained an RL model using a non-Qwen-based DeepSeek-Math-7B, and applied Temporal Sampling. We show that Temporal Sampling provides consistent gains on non-Qwen models. Please see Appendix D.5 for more information.

Latency Overheads of Temporal Sampling. We show that reasonably allocating GPU resources could make the latency overheads of temporal sam-

Table 4: Temporal Sampling V.S. Model Merge on AIME24, AIME25 and AMC. We observed that temporal sampling consistently outperforms checkpoint merging across all benchmarks, particularly at larger sampling sizes (pass@64 and maj@64). When t = 4 and k = 64, temporal sampling exceeds checkpoint merging by 10.7, 9.5, and 7.6 on AIME2024, AMC, and AIME2025, respectively.

Method	Task	t	Pass@k					Maj@k				
			@4	@8	@16	@32	@64	@4	@8	@16	@32	@64
Merge	AIME24	2	17.6	21.4	25.2	28.4	29.8	12.2	13.0	13.5	13.6	13.5
Temporal Sampling	AIME24	2	21.7	26.5	31.3	34.9	36.5	13.3	14.9	15.3	15.8	16.0
Merge	AMC	2	68.4	74.9	79.4	81.8	82.4	53.0	54.0	55.0	55.8	56.9
Temporal Sampling	AMC	2	70.1	76.6	81.7	85.4	87.2	54.4	55.7	56.4	56.9	57.1
Merge	AIME25	2	16.6	22.1	26.6	29.1	29.9	7.7	8.4	9.5	10.3	11.2
Temporal Sampling	AIME25	2	18.7	24.5	29.7	33.5	36.1	10.7	11.5	12.4	13.3	13.6
Merge	AIME24	4	19.2	23.5	27.0	29.1	29.9	13.4	14.2	15.3	15.2	14.9
Temporal Sampling	AIME24	4	21.3	25.9	30.9	36.3	40.6	13.8	15.8	16.8	17.6	18.3
Merge	AMC	4	67.9	74.4	79.2	81.7	82.4	53.7	55.1	55.5	55.1	52.5
Temporal Sampling	AMC	4	70.6	78.4	84.9	89.5	91.9	54.2	56.4	56.9	57.4	57.1
Merge	AIME25	4	19.7	24.6	28.3	29.8	30.0	9.9	10.9	11.2	12.7	13.5
Temporal Sampling	AIME25	4	18.4	24.3	29.7	34.2	37.6	10.5	12.2	13.3	14.0	14.5

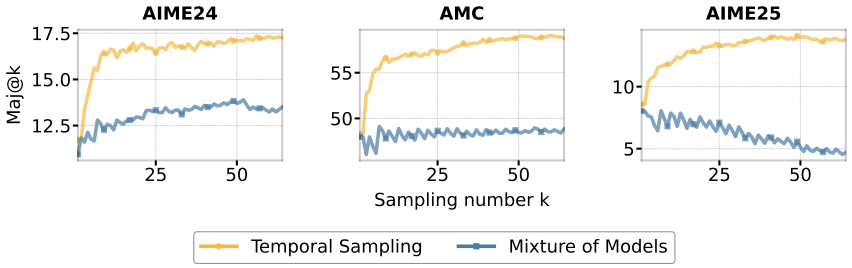

Figure 7: Maj@k comparison between Temporal Sampling ($t = 3$) and a Mixture of Models (MoM) approach on the AIME2024, AMC, and AIME2025 benchmarks. For MoM, the model pool included the Qwen2.5-7B-Base final RL checkpoint, Deepseek-Math-7B-Instruct, and Llama-3.1-8B-Instruct. Temporal Sampling outperforms the MoM approach by more than 4, 9, and 9 percentage points on AIME2024, AMC, and AIME2025, respectively, when sampling 64 responses.

Table 5: Comparison of latency overheads between temporal forgetting and sampling on final CKPT.

Exp	Run	Latency (in seconds)		
		AIME24	AIME25	AMC
Final CKPT Sampling	1	404	327	332
	2	382	342	346
	3	377	292	341
	Avg	388	320	340
Temporal Sampling (t=4)	1	400	358	371
	2	385	349	358
	3	412	365	384
	Avg	399	357	371
Temporal Sampling (t=8)	1	431	367	356
	2	364	326	408
	3	389	380	375
	Avg	395	358	380

pling similar with that of sampling on final CKPT. We use a total of 64 samples per question, 4 A100 GPUs and conduct the following experiments on the 7B model. We run each setting for three times and measure end-to-end latency (in seconds):

(1) Sampling on final checkpoint: 1 model with data parallel on 4 GPUs.

(2) Temporal Sampling ($t = 4$): 4 models on 4 GPUs without data parallel.

(3) Temporal Sampling ($t = 8$): 8 models on 4 GPUs with 2 CKPT per GPU (50% GPU utilization).

As shown in the Table 5, most of time temporal sampling has the similar latency overheads than sampling only on final CKPT and the maximum additional overheads is within 40 seconds.

4 Conclusion

In this paper, we observed the phenomenon of Temporal Forgetting: many correct solutions emerge transiently during training but are absent in the final model. Inspired by this phenomenon, we propose Temporal Sampling, a simple inference-time method that samples from multiple training checkpoints to recover forgotten solutions. This approach consistently improves reasoning performance by 4-19 points in Pass@k across benchmarks.

These findings suggest that true model competence may not reside in a single parameter snapshot, but rather in the collective dynamics of training

itself. Temporal Sampling offers a practical and powerful way to reclaim lost reasoning ability, challenging the standard paradigm of using only the final model checkpoint for evaluation and deployment.

Limitations

The experimental foundation of our work focuses on GRPO (Shao et al., 2024b) and SFT frameworks. While we believe our findings can generalize to other training methodologies, including on-policy approaches like PPO (Schulman et al., 2017), and off-policy techniques such as DPO (Rafailov et al., 2024). We have not empirically validated this hypothesis. When implementing Temporal Sampling, we focus on round-robin allocation strategies for distributing the k sampling attempts across t checkpoints. Alternative distribution approaches represent a promising avenue that we reserve for subsequent research.

Ethical Considerations

This work investigates LLM reasoning for math related tasks. It does not involve human subjects, user studies, or the collection of personally identifiable information. The dataset and benchmarks used in our experiments are publicly available. We do not introduce or endorse any applications that could cause harm or be misused. This paper does not present any ethical concerns.

Acknowledgment

This work is partially supported by the Office of Naval Research (ONR) under grant N0014-23-1-2386, the Air Force Office of Scientific Research (AFOSR) under grant FA9550-23-1-0208, and the National Science Foundation (NSF) AI Institute for Agent-based Cyber Threat Intelligence and Operation (ACTION) under grant IIS 2229876. Results presented in this paper were partially obtained using the Chameleon testbed (Keahey et al., 2020) supported by the National Science Foundation.

This work is supported in part by funds provided by the National Science Foundation, Department of Homeland Security, and IBM. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF or its federal agency and industry partners.

References

George E Andrews and Kimmo Eriksson. 2004. *Integer partitions*. Cambridge University Press.

Juhan Bae, Wu Lin, Jonathan Lorraine, and Roger Grosse. 2024. Training data attribution via approximate unrolled differentiation. *Preprint*, arXiv:2405.12186.

Edward Beeching, Lewis Tunstall, and Sasha Rush. 2024. Scaling test-time compute with open models.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Wenhu Chen, Ming Yin, Max Ku, Pan Lu, Yixin Wan, Xueguang Ma, Jianyu Xu, Xinyi Wang, and Tony Xia. 2023. TheoremQA: A theorem-driven question answering dataset. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

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

Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. 2023. Raft: Reward ranked finetuning for generative foundation model alignment. *Preprint*, arXiv:2304.06767.

Hugging Face. 2025. Open r1: A fully open reproduction of deepseek-r1.

Xidong Feng, Ziyu Wan, Muning Wen, Stephen Marcus McAleer, Ying Wen, Weinan Zhang, and Jun Wang. 2023. Alphazero-like tree-search can guide large language model decoding and training. *Preprint*, arXiv:2309.17179.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. 2024. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *Preprint*, arXiv:2402.14008.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021a. Measuring mathematical problem solving with the math dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1.

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

Intelligent Internet. 2025. Ii-thought : A large-scale, high-quality reasoning dataset.

Mingyu Jin, Weidi Luo, Sitao Cheng, Xinyi Wang, Wenyue Hua, Ruixiang Tang, William Yang Wang, and Yongfeng Zhang. 2025. Disentangling memory and reasoning ability in large language models. *Preprint*, arXiv:2411.13504.

Jikun Kang, Xin Zhe Li, Xi Chen, Amirreza Kazemi, Qianyi Sun, Boxing Chen, Dong Li, Xu He, Quan He, Feng Wen, and 1 others. 2024. MindStar: Enhancing math reasoning in pre-trained llms at inference time. *arXiv preprint arXiv:2405.16265*.

Kate Keahey, Jason Anderson, Zhuo Zhen, Pierre Riteau, Paul Ruth, Dan Stanzione, Mert Cevik, Jacob Colleran, Haryadi S. Gunawi, Cody Hammock, Joe Mambretti, Alexander Barnes, François Halbach, Alex Rocha, and Joe Stubbs. 2020. Lessons learned from the chameleon testbed. In *Proceedings of the 2020 USENIX Annual Technical Conference (USENIX ATC '20)*. USENIX Association.

Maxim Khanov, Jirayu Burapacheep, and Yixuan Li. 2024. ARGS: Alignment as reward-guided search. In *International Conference on Learning Representations (ICLR)*.

Kimi Team. 2025. Kimi k1.5: Scaling reinforcement learning with llms. *Preprint*, arXiv:2501.12599.

Ananya Kumar, Aditi Raghunathan, Robbie Jones, Tengyu Ma, and Percy Liang. 2022. Fine-tuning can distort pretrained features and underperform out-of-distribution. *Preprint*, arXiv:2202.10054.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, and 4 others. 2024. Tulu 3: Pushing frontiers in open language model post-training. *Preprint*, arXiv:2411.15124.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Li Erran Li, Raluca Ada Popa, and Ion Stoica. 2025. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl. Notion page. Accessed 2025-10-05.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. 2023. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *arXiv preprint arXiv:2308.08747*.

Rohin Manvi, Anikait Singh, and Stefano Ermon. 2024. Adaptive inference-time compute: Llms can predict if they can do better, even mid-generation. *arXiv preprint arXiv:2410.02725*.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *Preprint*, arXiv:2501.19393.

NovaSky. 2025. Sky-T1: Train your own o1 preview model within $450. Accessed: 2025-01-09.

OpenAI. 2024. Learning to reason with llms.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *Preprint*, arXiv:2203.02155.

Qwen Team. 2024. Qwq: Reflect deeply on the boundaries of the unknown.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Preprint*, arXiv:2305.18290.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2024. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

Yi Ren, Shangmin Guo, Wonho Bae, and Danica J. Sutherland. 2023. How to prepare your task head for finetuning. *Preprint*, arXiv:2302.05779.

Yi Ren and Danica J. Sutherland. 2025. Learning dynamics of llm finetuning. *Preprint*, arXiv:2407.10490.

Nikhil Sardana, Jacob Portes, Sasha Doubov, and Jonathan Frankle. 2024. Beyond chinchilla-optimal: Accounting for inference in language model scaling laws. In *International Conference on Machine Learning (ICML)*, volume 235, pages 43445–43460.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *Preprint*, arXiv:1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, and 1 others. 2024a. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, YK Li, Y Wu, and Daya Guo. 2024b. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2024. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*.

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

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

OpenThoughts Team. 2025a. Open Thoughts. https://open-thoughts.ai.

RUCAIBox STILL Team. 2025b. Still-3-1.5b-preview: Enhancing slow thinking abilities of small models through reinforcement learning.

Ziyu Wan, Xidong Feng, Muning Wen, Stephen Marcus Mcaleer, Ying Wen, Weinan Zhang, and Jun Wang. 2024. AlphaZero-like tree-search can guide large language model decoding and training. In *International Conference on Machine Learning (ICML)*, volume 235, pages 49890–49920.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations (ICLR)*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. 2024. Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models. *arXiv preprint arXiv:2408.00724*.

Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, James Xu Zhao, Min-Yen Kan, Junxian He, and Michael Xie. 2023. Self-evaluation guided beam search for reasoning. In *Advances in Neural Information Processing Systems*, volume 36, pages 41618–41650. Curran Associates, Inc.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 22 others. 2024a. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. 2024b. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pages 11809–11822.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, and 16 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *Preprint*, arXiv:2503.14476.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. 2025. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *arXiv preprint arXiv:2504.13837*.

Weihao Zeng, Yuzhen Huang, Wei Liu, Keqing He, Qian Liu, Zejun Ma, and Junxian He. 2025. 7b model and 8k examples: Emerging reasoning with reinforcement learning is both effective and efficient. `https://hkust-nlp.notion.site/simplerl-reason`. Notion Blog.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025. The lessons of developing process reward models in mathematical reasoning. *arXiv preprint arXiv:2501.07301*.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand. Association for Computational Linguistics.

A Related Work

Reinforcement learning for LLM. Reinforcement Learning (RL) has rapidly become a cornerstone for extending the capabilities of LLMs across various applications. Although it was first employed to align model behavior with human preferences through approaches like Reinforcement Learning from Human Feedback (RLHF) (Ouyang et al., 2022), its role now encompasses reasoning on complex tasks (Kimi Team, 2025; DeepSeek-AI, 2025; Lambert et al., 2024). For example, DeepSeek-R1 applied RL directly to a base "zero" LLM (DeepSeek-AI, 2025), and Kimi K1.5 augmented this framework with multimodal reasoning and verbosity control (Kimi Team, 2025). In particular, Reinforcement Learning has gained traction in areas such as mathematics and programming, where reward signals can be defined by clear, rule-based criteria like answer matching (Lambert et al., 2024; Shao et al., 2024b; Chen et al., 2021; DeepSeek-AI, 2025; Feng et al., 2023; Snell et al., 2024b; Xie et al., 2023; Wan et al., 2024). Advances in optimization, such as specialized PPO variants (e.g., VinePPO (Feng et al., 2023)) and stabilized GRPO algorithms (e.g., DAPO (Yu et al., 2025)), have simplified reward design, making RL more practical. Our work shifts focus from static performance gains of RL to the evolution of answer correctness over the procedure of RL training. We harness these temporal fluctuations as the diversity source to increase inference-time performance.

Inference Time Scaling. Expanding the computational budget available during inference has become a powerful lever for squeezing extra performance out of large language models, giving rise to an ever-growing family of test-time scaling (TTS) techniques (OpenAI, 2024). The field has seen a variety of approaches to leverage this. Established techniques include sampling-driven methods like majority voting (Wang et al., 2023) or best-of-N (Sardana et al., 2024), which generate many candidate answers and select the most persuasive one. More intricate are search-based algorithms such as Tree-of-Thoughts (ToT) explorations (Yao et al., 2023) and Monte-Carlo tree search (MCTS) (Xie et al., 2023; Khanov et al., 2024; Wan et al., 2024). Such approaches often build upon the development of sophisticated verifiers and may integrate process-based reward signals directly into search methods (Kang et al., 2024; Wu et al., 2024; Snell et al., 2024b). To further enhance efficiency and adaptiveness, other techniques include self-evaluation mechanisms for judicious compute allocation (Manvi et al., 2024) and diversity-aware search tactics, sometimes referred to as Test-Time Scaling (TTS) with diversity, to reduce redundant sampling and explore a wider solution space (Beeching et al., 2024).

Learning Dynamics. Learning dynamics analyze model behavior during training, such as explaining "aha moments" (DeepSeek-AI, 2025), and challenges in fine-tuning generalization (e.g., (Kumar et al., 2022; Ren et al., 2023)). These works focus on the training process itself and offer novel perspectives on how models learn and develop capabilities. Other research analyzes the step-wise decomposition of how influence accumulates among different potential responses for both instruction and preference tuning in LLMs (Ren and Sutherland, 2025). This detailed analytical framework, offering hypothetical explanations for why specific types of hallucination are strengthened post-finetuning. From the data perspective, Training Data Attribution (TDA) (Bae et al., 2024) identifies influential training examples to explain model predictions. Orthogonal to these works, we empirically investigate the dynamic fluctuations in answer correctness across diverse reasoning tasks, and harness the learning dynamics as a source of answer diversity to widen the sampling space and performance.

B Proof of Unbiased Estimation

We provide a formal proof that our proposed estimator for Pass@$k|t$ is unbiased. The Pass@$k|t$ metric measures the probability of obtaining at least one correct answer when samples are drawn from multiple checkpoints in a round-robin manner. The following proof establishes the statistical validity of our evaluation framework, ensuring that our empirical measurements accurately reflect the true performance of **Temporal Sampling** across different checkpoints.

Theorem 1. *Denote $r_{i,j}$ as the Pass@1 rate for the j-th checkpoint on problem i, $C_{i,j}$ as the number of correct samples among N candidates for problem i from checkpoint j. Let*

$$P_i = 1 - \prod_{j=1}^{t} (1 - r_{i,j})^{k_j}$$

denote the probability of obtaining at least one correct answer when k samples are drawn from t

checkpoints for problem i, (i.e., Pass@$k|t$), where k_j is determined by the balanced integer partition of k on t:

$$k_j = \begin{cases} \lfloor k/t \rfloor + 1 & \text{if } j \leq (k \pmod t) \\ \lfloor k/t \rfloor & \text{if } j > (k \pmod t) \end{cases}$$

We have

$$\hat{P}_i = 1 - \prod_{j=1}^{t} \left(\frac{\binom{N-C_{i,j}}{k_j}}{\binom{N}{k_j}} \right)$$

is an unbiased estimator of P_i, i.e., $\mathbb{E}[\hat{P}_i] = P_i$.

Proof. For a single checkpoint j on problem i, we consider the probability of obtaining no correct solutions when sampling k_j solutions without replacement from N total samples. Given that $C_{i,j}$ of these N samples are correct, this probability follows the hypergeometric distribution:

$$P(X_{i,j} = 0) = \frac{\binom{N-C_{i,j}}{k_j}}{\binom{N}{k_j}}$$

For Pass@$k|t$, we succeed if at least one sample across all checkpoints is correct. The probability of failure (no correct solutions from any checkpoint) is:

$$P(\text{failure}) = \prod_{j=1}^{t} P(X_{i,j} = 0) = \prod_{j=1}^{t} \frac{\binom{N-C_{i,j}}{k_j}}{\binom{N}{k_j}}$$

Thus, our estimator for the success probability is:

$$\hat{P}_i = 1 - \prod_{j=1}^{t} \frac{\binom{N-C_{i,j}}{k_j}}{\binom{N}{k_j}}$$

To prove this estimator is unbiased, we need to show that $\mathbb{E}[\hat{P}_i] = P_i$. We first prove that:

$$\mathbb{E}\left[\frac{\binom{N-C_{i,j}}{k_j}}{\binom{N}{k_j}} \right] = (1 - r_{i,j})^{k_j}$$

Since $C_{i,j}$ follows a binomial distribution $B(N, r_{i,j})$, we have:

$$\mathbb{E}\left[\frac{\binom{N-C_{i,j}}{k_j}}{\binom{N}{k_j}} \right] \tag{1}$$

$$= \sum_{c=0}^{N} \frac{\binom{N-c}{k_j}}{\binom{N}{k_j}} \cdot \binom{N}{c} r_{i,j}^{c} (1 - r_{i,j})^{N-c} \tag{2}$$

We can simplify the coefficient:

$$\frac{\binom{N-c}{k_j}}{\binom{N}{k_j}} \cdot \binom{N}{c} \tag{3}$$

$$= \frac{(N-c)!}{k_j!(N-c-k_j)!} \cdot \frac{k_j!(N-k_j)!}{N!} \cdot \frac{N!}{c!(N-c)!} \tag{4}$$

$$= \binom{N-k_j}{c} \tag{5}$$

Substituting this back:

$$\mathbb{E}\left[\frac{\binom{N-C_{i,j}}{k_j}}{\binom{N}{k_j}} \right] = \sum_{c=0}^{N-k_j} \binom{N-k_j}{c} r_{i,j}^{c} (1 - r_{i,j})^{N-c} \tag{6}$$

$$= (1 - r_{i,j})^{k_j} \sum_{c=0}^{N-k_j} \binom{N-k_j}{c} r_{i,j}^{c} (1 - r_{i,j})^{N-k_j-c} \tag{7}$$

The summation represents the binomial expansion of $(r_{i,j} + (1 - r_{i,j}))^{N-k_j} = 1^{N-k_j} = 1$, yielding:

$$\mathbb{E}\left[\frac{\binom{N-C_{i,j}}{k_j}}{\binom{N}{k_j}} \right] = (1 - r_{i,j})^{k_j} \tag{8}$$

Since the samples from different checkpoints are independent, we have:

$$\mathbb{E}\left[\prod_{j=1}^{t} \frac{\binom{N-C_{i,j}}{k_j}}{\binom{N}{k_j}} \right] = \prod_{j=1}^{t} E\left[\frac{\binom{N-C_{i,j}}{k_j}}{\binom{N}{k_j}} \right] \tag{9}$$

$$= \prod_{j=1}^{t} (1 - r_{i,j})^{k_j} \tag{10}$$

Therefore:

$$\mathbb{E}[\hat{P}_i] = 1 - E\left[\prod_{j=1}^{t} \frac{\binom{N-C_{i,j}}{k_j}}{\binom{N}{k_j}} \right] \tag{11}$$

$$= 1 - \prod_{j=1}^{t} (1 - r_{i,j})^{k_j} = P_i \tag{12}$$

This proves that $\hat{P}_i$ is an unbiased estimator for Pass@$k|t$. □

C Experiment Setup

C.1 GRPO

We follow (Luo et al., 2025) and use the following hyper-parameters detailed in Table 6 for Zero RL training. We perform experiments on eight A100 GPUs. The model is trained using VERL (Sheng et al., 2024).

Table 6: This table shows the hyper-parameters for zero RL training.

Hyper-parameter	Value
Learning Rate	1×10^{-6}
Number of Epochs	9
Number of Devices	8
Rollout Batch Size	128
PPO Mini Batch Size	64
Max Prompt Length	1024
Max Response Length	3072 (QWEN2.5-MATH-7B), 8192 (OTHERS)
KL Coefficient	0.001
Rollout Engine	VLLM (V0.8.2)
Optimizer	Adamw
Learning Rate Scheduler	cosine
Warmup Ratio	0.1

C.2 Supervised Fine-tuning

Our model SFT is conducted using LLaMA-Factory (Zheng et al., 2024), on a server with four NVIDIA A100-SXM4-80GB GPUs. We follow (NovaSky, 2025) for the training parameters. Table 7 lists hyper-parameters for full parameter supervised fine-tuning.

D More Experiment results

D.1 Temporal Sampling for Best-of-N

Figure 8 demonstrates the effectiveness of Temporal Sampling when combined with Best-of-N (BoN) decoding on the AIME2024, AMC, and AIME2025 benchmarks. Using Qwen2.5-Math-PRM-72B (Zhang et al., 2025) as the process reward model, answers with the highest reward were selected as the final output. The results clearly show that Temporal Sampling with $t = 8$ checkpoints significantly outperforms the baseline ($t = 1$), achieving improvements of more than 7, 8, and 1 percentage points across the three benchmarks when sampling $k = 64$ responses. Figure 9 presents additional evidence for the effectiveness of Temporal Sampling with Best-of-N decoding when using the smaller Qwen2.5-Math-PRM-7B (Zhang et al., 2025) as the process reward model. This highlights the value of leveraging multiple training checkpoints for enhancing reward-based selection methods.

D.2 More Results of Temporal Forgetting

Table 8 provides a comprehensive list of the SOTA models evaluated in Table 1 along with their corresponding base models.

Figure 10 illustrates the performance comparison between base models and fine-tuned models using both Pass@1 and Pass@8 sampling on the OlympiadBench dataset. The figure shows that while fine-tuned models like DeepscaleR-1.5B and Still-3-1.5B achieve higher overall performance than their base models ($P_{\text{FT}} > P_{\text{Base}}$), they also exhibit the temporal forgetting phenomenon with substantial Lost Scores (P_{Lost}) for both Pass@1 sampling and Pass@8 sampling.

D.3 More Results of Forgetting Dynamics

We illustrate the correctness of answers to different OlympiadBench questions at various checkpoints during the RL training of Qwen2.5-7B. Figure 11 (a) demonstrates the phenomenon of **Forgetting Dynamics**: Questions exhibits alternating "Improve" and "Forget" events frequently during training, which means the model oscillates between correct and incorrect answers across checkpoints. In Figure 11 (b), we show the percentage of questions across different benchmarks that experienced the "Forget" event could achieve up to 32.3% in OlympiadBench and 52.5% in AMC.

Table 9 presents detailed performance metrics for different fine-tuned models evaluated specifically on AIME24 and AMC benchmarks. The table shows the base model performance (P_{Base}), fine-tuned model performance (P_{FT}), Ever Correct Score (P_{ECS}), and Temporal Forgetting Score (P_{TFS}) across various models with both GRPO and SFT training methods. Notably, models exhibit significant temporal forgetting, with P_{TFS} values ranging from 6.7% to 30%, which implies that many questions solved correctly at some point during training were ultimately answered incorrectly in the final checkpoint.

Table 10 complements Table 2 by providing a more comprehensive view of base model (P_{Base}) and fine-tuned model (P_{FT}) performance across all five mathematical benchmark.

Table 7: This table shows the hyper-parameters for full parameter supervised fine-tuning.

Hyper-parameter	Value
Learning Rate	1×10^{-5}
Number of Epochs	3
Number of Devices	4
Per-device Batch Size	1
Optimizer	`Adamw`
Learning Rate Scheduler	`cosine`
Max Sequence Length	16384

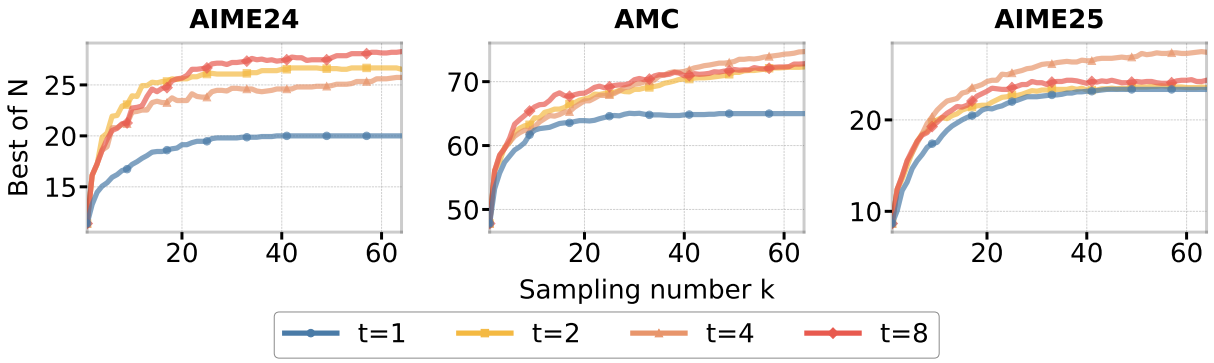

Figure 8: BoN (Best-of-N) decoding on the AIME2024, AMC, and AIME2025 benchmarks using Temporal Sampling. Qwen2.5-Math-PRM-72B is used as the process reward model. We choose the answer with the highest reward as the final answer. The case $t = 1$ represents the baseline of standard BoN on the final checkpoint. Our proposed Temporal Sampling with $t = 8$ checkpoints outperforms the baseline by more than 7, 8, and 1 percentage points on AIME2024, AMC, and AIME2025, respectively, when sampling 64 responses.

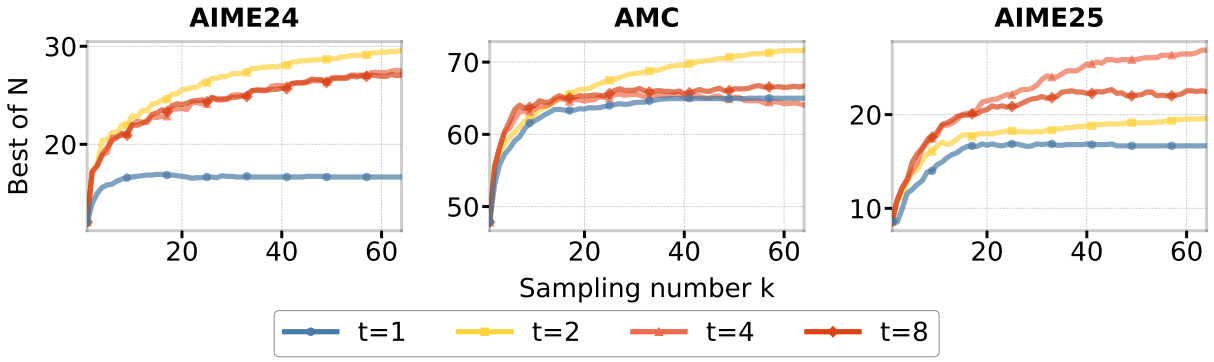

Figure 9: BoN (Best-of-N) decoding on the AIME2024, AMC, and AIME2025 benchmarks using Temporal Sampling. Qwen2.5-Math-PRM-7B is used as the process reward model. We choose the answer with the highest reward as the final answer. The case $t = 1$ represents the baseline of standard BoN on the final checkpoint. Our proposed Temporal Sampling with $t = 8$ checkpoints outperforms the baseline by more than 10, 2, and 5 percentage points on AIME2024, AMC, and AIME2025, respectively, when sampling 64 responses.

D.4 Ablation on Checkpoint Selection of Temporal Sampling

We find that later checkpoints are more stable near convergence thus we choose the recent checkpoints for temporal sampling. We compared two different ordering strategies:

Reverse (most recent first): ckpts 8, 7, 6, ..., 1

Forward (least recent first): ckpts 8, 1, 2, ..., 7

We observed that the reverse order consistently outperforms forward across most benchmarks in the Table 11.

D.5 Ablation on Non-Qwen-based models

We trained an RL model using a non-Qwen-based model, DeepSeek-Math-7B, and applied Temporal Sampling. Results are shown in Table 12. The results confirm that Temporal Sampling provides consistent gains on non-Qwen models.

Table 8: Full list of SOTA models evaluated in Table 1 and their corresponding base models.

Model	Based on	Training
DeepScaleR-1.5B	Distill-R1-1.5B	RL
Still-1.5B	Distill-R1-1.5B	RL
S1.1-1.5B	Qwen2.5-1.5B-Instruct	SFT
II-thought-1.5B-preview	Distill-R1-1.5B	RL
S1.1-3B	Qwen2.5-3B-Instruct	SFT
SmallThinker-3B	Qwen2.5-3B-Instruct	SFT
S1.1-7B	Qwen2.5-7B-Instruct	SFT
OpenR1-Qwen-7B	Qwen2.5-Math-7B-Instruct	SFT
OpenThinker-7B	Qwen2.5-7B-Instruct	SFT
s1-32B	Qwen2.5-32B-Instruct	SFT
Sky-T1-32B-Preview	Qwen2.5-32B-Instruct	SFT
Bespoke-Stratos-32B	Qwen2.5-32B-Instruct	SFT
OpenThinker-32B	Qwen2.5-32B-Instruct	SFT

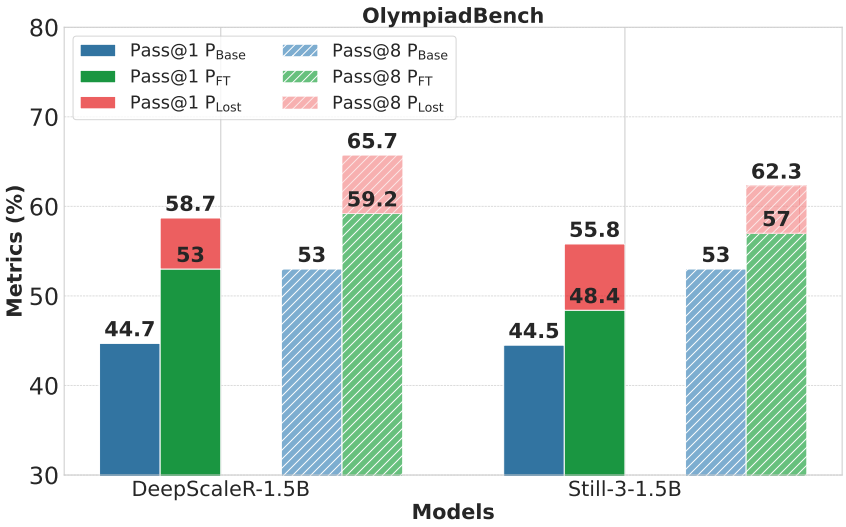

Figure 10: Performance of the base model (P_{Base} ↑), the fine-tuned model (P_{FT} ↑) and the Lost Score (P_{Lost} ↓) for Pass@1 sampling and Pass@8 sampling. Fine-tuned models like DeepscaleR-1.5B (Luo et al., 2025) and Still-3-1.5B (Face, 2025) outperform the base model overall but also forget many questions the base model answered correctly.

E Practical Deployment Costs of Temporal Sampling

A natural concern with Temporal Sampling is that serving multiple checkpoints could inflate memory, storage, and latency costs. In this section, we show that Temporal Sampling can be better deployed in practice to decompose these costs.

Sequential deployment keeps peak VRAM constant. Temporal Sampling does *not* require all checkpoints to reside in VRAM simultaneously. Given a total sampling budget of k responses distributed across t checkpoints, a simple sequential schedule is:

1. Load checkpoint i into VRAM.
2. Batch-generate k/t samples for all problems.
3. Unload checkpoint i and load checkpoint $i+1$.

Under this schedule, peak VRAM usage is $\mathcal{O}(1)$ in the number of checkpoints—identical to running a single model. For example, generating k=64 samples from t=8 checkpoints requires only enough VRAM to hold one checkpoint at a time.

In this case, storing t checkpoints on disk increases storage consumption linearly in t, but disk is orders of magnitude cheaper than VRAM per byte, and these checkpoints are typically saved anyway as a byproduct of standard training workflows. A model load/unload also occurs only t times in total (not per sample), so the resulting I/O latency is dwarfed by the compute cost of generating long

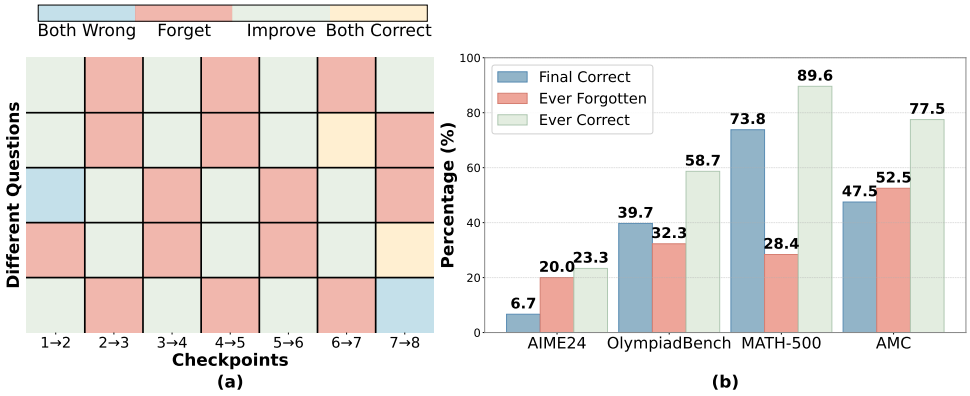

Figure 11: Forgetting dynamics of Qwen2.5-7B during RL training. (a) Answer correctness trajectories for OlympiadBench questions across training checkpoints, illustrating solutions oscillate between correct and incorrect states. "Forget" implies that an answer was correct at the previous checkpoint but incorrect at the current one. Conversely, "Improve" implies that an answer that was incorrect at the previous checkpoint but correct at the current one. (b) Percentage of questions per benchmark that are ever forgotten or ever correct at some checkpoint during GRPO.

Table 9: Performance of fine-tuned models ($P_{FT} \uparrow$), the Ever Correct Score ($P_{ECS} \uparrow$), and the Temporal Forgetting Score ($P_{TFS} \downarrow$) of different fine-tuned models evaluated on AIME24 and AMC. We observed both high P_{ECS} and P_{TFS} in spite of the improving overall performance, which implies a high percentage of questions (from 6.7% to 30%) are answered correctly at some checkpoint during training but are ultimately incorrect in the final checkpoint.

Model	AMC				AIME24			
	P_{Base}	P_{FT}	P_{ECS}	P_{TFS}	P_{Base}	P_{FT}	P_{ECS}	P_{TFS}
Qwen2.5-7B (GRPO)	32.5	47.5	77.5	30.0	6.7	6.7	23.4	16.7
Qwen2.5-7B (SFT)	32.5	52.5	75.0	22.5	6.7	10.0	20.0	10.0
Qwen2.5-1.5B (GRPO)	0.0	30.0	45.0	15.0	0.0	3.3	10.0	6.7
Qwen2.5-1.5B (SFT)	0.0	15.0	35.0	20.0	0.0	0.0	6.7	6.7
Qwen2.5-Math-7B (GRPO)	32.5	72.5	82.5	10.0	13.3	16.7	40.0	23.3
Qwen2.5-Math-7B (SFT)	32.5	50.0	75.0	25.0	13.3	20.0	40.0	20.0

reasoning chains.

Flexible parallelism. Users with abundant resources can alternatively load multiple checkpoints in parallel to further reduce wall-clock time, while users with limited VRAM can fall back to the sequential schedule. Both schedules produce identical samples and therefore identical performance gains, making the choice purely a compute/latency trade-off. We conclude that Temporal Sampling is practical for real-world deployment with only modest additional disk storage.

F Why Does Temporal Forgetting Occur? A Capability-Shift View

We hypothesize that Temporal Forgetting stems from a **capability shift** during fine-tuning, e.g., RL training enhances specific reasoning-oriented skills (e.g., logical inference, arithmetic calculation), but concurrently *degrades* non-reasoning capabilities (e.g., question comprehension, factual knowledge) that were previously sufficient to solve certain problems. Under this hypothesis, "forgotten" problems are those whose correctness depends disproportionately on the capabilities that RL erodes.

Experimental setup. We performed an error-type analysis on the incorrect responses of Qwen-7B before and after RL fine-tuning, using samples drawn from MATH-500, Olympiad, and GPQA-Diamond. We used `o4-mini` as an automated judge to classify each incorrect response into one of the following categories:

- **Reasoning errors**: errors in logical inference (*Logic*) or numerical computation (*Calculation*).
- **Non-reasoning errors**: errors in question comprehension (*Understanding*) or factual recall (*Knowledge*).

Table 10: Detailed performance score of base models (P_{Base}) and fine-tuned models (P_{FT}) across five mathematical benchmarks, served as complementary of Table 2.

Model	Olympiad		MATH-500		GPQA		AMC		AIME	
	P_{Base}	P_{FT}	P_{Base}	P_{FT}	P_{Base}	P_{FT}	P_{Base}	P_{FT}	P_{Base}	P_{FT}
Qwen2.5-7B (GRPO)	22.1	39.7	53.2	73.8	29.8	33.8	32.5	47.5	6.7	6.7
Qwen2.5-7B (SFT)	22.1	40.1	53.2	69.8	29.8	25.3	32.5	52.5	6.7	10.0
Qwen2.5-1.5B (GRPO)	0.6	18.8	0.6	55.6	3.0	26.8	0.0	30.0	0.0	3.3
Qwen2.5-1.5B (SFT)	0.6	11.0	0.6	36.2	3.0	13.1	0.0	15.0	0.0	0.0
Qwen2.5-Math-7B (GRPO)	19.3	41.0	60.2	79.8	30.3	32.8	32.5	72.5	13.3	16.7
Qwen2.5-Math-7B (SFT)	19.3	43.9	60.2	76.4	30.3	30.8	32.5	50.0	13.3	20.0

Table 11: Ablation on Checkpoint Selection of Temporal Sampling

Task	**Order**	**Pass@k**				**Average**
		pass@8	**pass@16**	**pass@32**	**pass@64**	
AIME24	Reverse	26.1	30.8	35.1	38.1	32.53
	Forward	25.5	29.4	32.6	34.8	30.58
AIME25	Reverse	24.1	29.6	34.2	37.4	31.33
	Forward	22.1	27.7	32.9	37.0	29.93
AMC	Reverse	77.6	83.7	88.3	90.9	85.13
	Forward	77.8	84.0	88.9	91.8	85.63

- **Others**: errors that did not fit the above categories (e.g., formatting issues).

Results. Table 13 reports the distribution of error types before and after RL training: (1) Reasoning errors decrease from 50.2% to 46.1% after RL, driven primarily by a reduction in calculation errors ($17.9\% \rightarrow 14.0\%$). This is the expected benefit of RL fine-tuning. (2) Non-reasoning errors increase from 42.6% to 45.4%, driven by a rise in understanding errors ($26.6\% \rightarrow 28.8\%$). That is, RL makes the model *more* likely to misinterpret questions or hallucinate facts, even on problems it previously answered correctly.

This pattern supports the capability-shift view: the problems that the final model "forgets", i.e., that it answered correctly earlier in training, are disproportionately those that rely on robust non-reasoning capabilities, which degrade as the model specializes in reasoning-heavy skills. Intermediate checkpoints retain a more balanced capability profile, which explains why sampling across checkpoints (Temporal Sampling) recovers many of these forgotten problems.

Table 12: Performance of Temporal Sampling on DeepSeek-Math-7B model after RL. We show that Temporal Sampling provides consistent gains on this model.

Method	Task	Pass@k						
		@1	**@2**	**@4**	**@8**	**@16**	**@32**	**@64**
Final Checkpoint	AIME25	0.4	0.8	1.6	2.8	4.6	6.3	6.7
Temporal Sampling (t=8)	AIME25	0.4	1.0	1.7	3.3	5.9	9.9	14.8
Final Checkpoint	AIME24	1.1	2.2	4.0	6.6	9.9	13.3	16.7
Temporal Sampling (t=8)	AIME24	1.1	2.1	3.9	6.7	10.0	13.7	18.6
Final Checkpoint	AMC	17.1	25.4	34.9	45.3	55.7	63.9	70.0
Temporal Sampling (t=8)	AMC	17.1	26.3	36.9	48.6	59.9	70.4	79.8

Model	Reasoning (Logic+Calc)	Non-reasoning (Under.+Knowl.)	Others
Base	50.2% (32.3 + 17.9)	42.6% (26.6 + 16.0)	7.1%
After RL	46.1% (32.1 + 14.0)	45.4% (28.8 + 16.6)	8.5%

Table 13: Error-type distribution on incorrect responses, aggregated over MATH-500, Olympiad, and GPQA-D, before and after RL fine-tuning. RL reduces reasoning errors but concurrently increases non-reasoning errors.