

000 001 002 003 004 005 006 007 008 009 010 011 012 013 014 015 016 017 018 019 020 021 022 023 024 025 026 027 028 029 030 031 032 033 034 035 036 037 038 039 040 041 042 043 044 045 046 047 048 049 050 051 052 053 POLICY OPTIMIZATION WITH EXPERIENCE REPLAY: GUIDING REASONING MODELS TO COMPLETE THE REASONING PATH

Anonymous authors

Paper under double-blind review

ABSTRACT

To our knowledge, in the field of large language models, all existing reinforcement fine-tuning algorithms require generating a complete reasoning process starting from the question, which results in a substantial time overhead during the rollout phase of training. Challenging this conventional approach, we propose the assumption that during reinforcement fine-tuning, the model only needs to generate part of the reasoning process. We analyze the impact of different segments of the reasoning path on the correctness of the final result, and based on these insights, we introduce **Policy Optimization with Experience Replay (POER)**, a plug-and-play reinforcement fine-tuning algorithm. Unlike traditional reinforcement fine-tuning algorithms that generate full reasoning paths, POER trains the model by generating suffixes of the reasoning path using experience caching, thereby significantly reducing training time while improving training stability. From evaluations during the rollout phase of training, POER reduces token generation in this phase by approximately 95%, greatly lowering the theoretical time overhead. In practical training, compared with full-path reinforcement fine-tuning algorithms, POER reduces the training time of the 1.5B model by 90% and the 7B model by 72%, while maintaining performance comparable to typical algorithms such as GRPO and DAPO. We have open-sourced the code in an anonymous repository: <https://anonymous.4open.science/r/POER-4BF2>

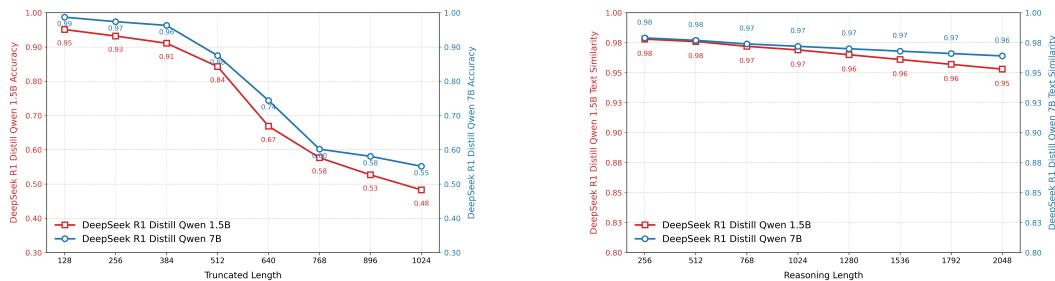


Figure 1: The left figure shows the **DeepSeekR1-Qwen-Distill-7b** and **DeepSeekR1-Qwen-Distill-1.5b** models. For each question, an initial answer is generated and then truncated; from the truncation point, 256 answers are subsequently generated, and the relationship between truncation length and the overall average accuracy is analyzed. The right figure shows 256 answers generated for each training question. Answers exceeding 2048 tokens are selected, and **BERT** is used to measure the similarity between equal-length prefix segments. The similarity metric is defined as: $\text{sim} = \frac{2}{n(n-1)} \sum_{i=1}^{n-1} \sum_{j=i+1}^n \frac{\text{BERT}(s_i) \cdot \text{BERT}(s_j)^\top}{\|\text{BERT}(s_i)\| \|\text{BERT}(s_j)\|}$

1 INTRODUCTION

In recent years, large language models (LLMs) (OpenAI et al., 2024b; Touvron et al., 2023; Zeng et al., 2023) have achieved remarkable breakthroughs in reasoning and generalization capabilities (Wang et al., 2025), particularly after the introduction of reinforcement learning (RL) during the post-training

stage (Ouyang et al., 2022). Pioneering works such as OpenAI’s O1 (OpenAI et al., 2024a) and DeepSeek-R1 (DeepSeek-AI et al., 2025) have demonstrated impressive reasoning-time efficiency, primarily due to the synergistic combination of reinforcement learning and chain-of-thought (CoT) reasoning (Wei et al., 2023). This paradigm shift highlights the transformative potential of RL-based post-training in pushing the boundaries of LLM performance.

Despite its promising prospects, applying reinforcement learning in post-training remains immature and highly challenging, with numerous obstacles hindering its widespread adoption. Regarding time overhead, RL fine-tuning typically generates many samples during the sampling stage. However, parameter updates cannot proceed until all samples are completed, leading to significant underutilization of computational resources. Furthermore, during RL fine-tuning of language models, rewards are computed only after generating the final token based on task-specific criteria. This paradigm, known as Reinforcement Learning with Verifiable Rewards (RLVR) (Lambert et al., 2025), lacks intermediate feedback and produces sparse rewards. Such sparsity hinders the model’s ability to learn optimal policies and contributes to training instability (Lightman et al., 2023).

Most current research efforts addressing these challenges focus on optimizing the policy gradient function (Sutton et al., 1999), which has achieved some success. However, these approaches often overlook the critical role of sampling strategies.

We observe that a significant underlying issue stems from the policy model’s need to identify a reasoning trajectory from the beginning of the problem to the correct answer. This approach—comparing entire reasoning paths using policy gradients—leads to excessive randomness during the sampling phase. Although it expands the search space, it often fails to find suitable reasoning paths, resulting in inefficient sampling and high variance.

An alternative perspective arises: since exploring a complete reasoning path from the beginning of the problem introduces various drawbacks, why not train the policy model to complete a reasoning path based on **partially correct reasoning process hint** instead? We found that this is feasible. Through our experiments, enabling the model to complete correct reasoning paths can still effectively teach it to generate whole reasoning trajectories from the initial problem statement. Based on this insight, we propose the *Policy Optimization with Experience Replay*(POER). Our method is grounded in a reasonable assumption: the early tokens of a reasoning path that leads to the correct answer are more likely to guide the model toward the correct reasoning trajectory. Furthermore, we investigate the relationship between the length of the truncated trailing tokens and the model’s generation accuracy. The results confirm that the initial tokens of correct answers play a crucial role in steering the model toward correct solutions, and that longer prefix lengths positively correlate with higher generation accuracy.

Specifically, we construct a cache pool for the GRPO to store previously generated reasoning paths and continuously update it during training. After we complete the sampling generation stage for each question, we add the reasoning path that leads to the correct answer into the cache. When we later reencounter the same question, we retrieve the first **n** tokens of the corresponding reasoning path from the cache, prepend them to the prompt, and then perform sampling. Experimental results show that this method is plug-and-play, improves training stability during the RL stage, significantly reduces the policy model’s sampling time cost, and achieves notable performance gains.

Contributions We propose POER, a novel framework for reinforcement fine-tuning of LLMs, introducing an experience replay mechanism in the sampling stage. Key advantages are: **plug-and-play**: easily integrates into other RL fine-tuning methods; **reduced resource consumption**: up to 92.6% faster training; **strong stability**: mitigates common RL instability in reasoning models.

We evaluate POER on Deepseek-R1-Distill-Qwen 1.5B and 7B across six datasets. Results show around 90% reduction in training time, a 2% performance improvement over GRPO and DAPO, and support for mini-batch, multi-step updates.

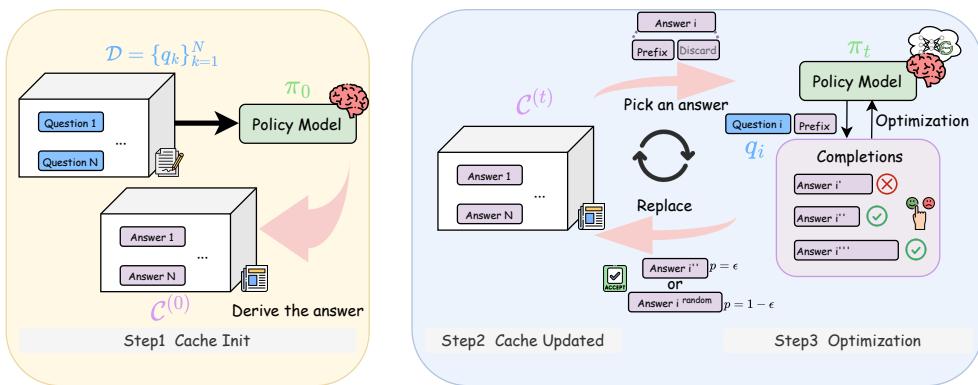
2 RELATED WORKS

Reinforcement Fine-Tuning Reinforcement Fine-Tuning (RFT) guides the model fine-tuning process through the reward mechanisms of reinforcement learning, greatly enhancing generalization and accuracy. Kimi v1.5 (Team et al., 2025) and ReFT (Luong et al., 2024) employ traditional

108 Proximal Policy Optimization (PPO) (Schulman et al., 2017) for RFT and have demonstrated excellent
 109 performance. DeepSeek-R1 (DeepSeek-AI et al., 2025) adopts GRPO and uses verifiable reward
 110 strategies to compute policy gradients directly. DAPO (Yu et al., 2025b) further optimizes GRPO to
 111 improve training stability. R1-V (Chen et al.), VLM-R1 (Shen et al., 2025a), and LMM-R1 (Peng
 112 et al., 2025) extend RFT into the multi-modal domain. While many reinforcement learning algorithms
 113 suffer from reward saturation as training steps increase, leading to reward hacking (Eisenstein et al.,
 114 2024). Satori (Shen et al., 2025b) uses SFT distillation to mitigate it, and O1-Prune (Luo et al., 2025)
 115 employs post-hoc length pruning to enhance stability.

116
Efficient Inference Test-time scaling (Chen et al., 2024) significantly increases training and
 117 inference costs, as models tend to generate lengthy reasoning chains. To reduce the time cost of RL
 118 training, UPFT (Ji et al., 2025) proposes fine-tuning the model using only the first n tokens. However,
 119 it is impossible to validate reasoning accuracy properly. ThinkPrune (Hou et al., 2025) sets a length
 120 constraint during RL training to limit the model’s thinking length and reduce inference costs. O1-
 121 Prune (Luo et al., 2025) enhances training stability through length pruning and reduces the high cost
 122 associated with long reasoning chains. Hao et al. (2024) optimizes inference by compressing lengthy
 123 reasoning chains into latent space, while Chen et al. (2025); Yu et al. (2025a) reduces inference costs
 124 by aggregating tokens.

126 3 METHOD



143 Figure 2: Overview of the POER framework. The entire training process is described as follows:
 144 Cached answer fragments are used by the model to generate new responses; either the best or a random
 145 response is selected based on the reward system for optimization; and the cache is continuously
 146 updated to improve training efficiency and stability.

149 3.1 CACHE POOL INITIALIZATION

151 First, we denote the dataset of samples as $\mathcal{D} = \{q_k\}_{k=1}^N$, where q_k represents the k -th question in the
 152 dataset. We denote the initial model parameters as θ_0 , and we represent the model’s answering policy
 153 by π_{θ_0} . Before training begins, we initialize the cache pool as $\mathcal{C}^{(0)}$ as follows:

$$155 \quad \mathcal{C}^{(0)} = \{(q_k, a_k) \mid a_k \sim \pi_{\theta_0}(\cdot | q_k), \forall q_k \in \mathcal{D}\} \quad (1)$$

157 This stage uses the initial model policy to sample the dataset \mathcal{D} . To retrieve the response a_k
 158 corresponding to question q_k from the cache pool, we define the retrieval operation as:
 159

$$160 \quad a_k := \{a \mid (q_k, a) \in \mathcal{C}\}. \quad (2)$$

161 Here, a_k denotes the answer associated with question q_k in the cache pool \mathcal{C} .

162 3.2 TRUNCATED ANSWER SAMPLING OPTIMIZATION
163164 **Sampling Generation** At each sampling stage, we use the POER strategy to retrieve the historical
165 response a_k for each question q_k from the cache pool \mathcal{C} . We then remove the last m tokens and
166 concatenate the remaining prefix with q_k to generate a new response o . We express this process as:
167

168
$$o = a_k^{[0:-m]} \| \pi_\theta(\cdot | q_k, a_k^{[0:-m]}), \quad \text{where} \quad \begin{cases} a_k := \{a \mid (q_k, a) \in \mathcal{C}\} \\ m \sim \mathcal{U}\{0, 1, \dots, L\} \end{cases} \quad (3)$$

169
170

171 Here, L is the maximum truncation length, $\mathcal{U}\{0, 1, \dots, L\}$ samples a truncation point uniformly from
172 $[0, L]$, $a_k^{[0:-m]}$ truncates the last m tokens of a_k , and $\pi_\theta(\cdot | q_k, a_k^{[0:-m]})$ generates a new continuation
173 based on the question and prefix. In this paper, L is either fixed or set dynamically based on the
174 shortest response in a sampling group G , denoted as ℓ , where:
175

176
$$\ell = \min\{\text{len}(o_1), \text{len}(o_2), \dots, \text{len}(o_G)\} \quad (4)$$

177
178

179 **Policy Optimization with Experience Replay** After completing the sampling generation, POER
180 adopts **Group Relative** estimation of advantage. For a given question-answer pair (q_k, a_k) , the
181 behavioral policy $\pi_{\theta_{t-1}}$ samples a group of G individual responses $\{o_i\}_{i=1}^G$ from the model. Then,
182 by normalizing the group rewards $\{R_i\}_{i=1}^G$, the advantage of each response is computed as:
183

184
$$\begin{aligned} \mathcal{J}_{\text{POER}}(\theta_t) &= \mathbb{E}_{(q, a) \sim \mathcal{C}^{(t-1)}(Q, A), \{o_i\}_{i=1}^G \sim a_k^{[0:-m]} \| \pi_\theta(\cdot | q_k, a_k^{[0:-m]})} \\ &\quad \frac{1}{G} \sum_{i=1}^G \frac{1}{|o_i|} \sum_{j=1}^{|o_i|} \left\{ \min \left[r_{i,j}(\theta_t) \hat{A}_{i,j}, \text{clip} \left(r_{i,j}(\theta_t), 1 - \epsilon, 1 + \epsilon \right) \hat{A}_{i,j} \right] - \beta D_{\text{KL}}(\pi_{\theta_t} \| \pi_{\text{ref}}) \right\} \end{aligned} \quad (5)$$

185
186
187
188
189

190 where:
191

192
$$r_{i,j}(\theta_t) = \frac{\pi_{\theta_t}(o_{i,j} | q, o_{i,<j})}{\pi_{\theta_{t-1}}(o_{i,j} | q, o_{i,<j})}, \quad \hat{A}_{i,j} = \frac{R_i - \text{mean}(\{R_i\}_{i=1}^G)}{\text{std}(\{R_i\}_{i=1}^G)} \quad (6)$$

193
194
195

196 Since POER is policy-agnostic, we propose a unified forward reinforcement learning paradigm based
197 on experience replay. We can then write the policy gradient function of POER in a more general form
198 as:
199

200
$$\nabla_\theta \mathcal{J}_{\text{POER}}(\theta) = \underbrace{\mathbb{E}_{(q, o) \sim \mathcal{C}}}_{\text{Data Source}} \left(\frac{1}{|o|} \sum_{j=1}^{|o|} \underbrace{\mathcal{G}(q, o, j, \pi_{\text{ref}})}_{\text{Gradient Coefficient}} \nabla_\theta \log \pi_\theta(o_j | q, o_{<j}) \right) \quad (7)$$

201
202
203

204 Equation 7 is derived from the standard policy gradient formulation. The above equations indicate that
205 only the sampling stage is affected by POER, while the policy gradient function remains unaltered. As
206 a result, POER exhibits a plug-and-play nature and can be easily integrated into other reinforcement
207 fine-tuning algorithms.208 Compared to the traditional GRPO strategy, a previously sampled historical response trajectory is
209 introduced by POER as a constraint into the subsequent sampling process. In this way, the policy
210 space π_θ explored during training is confined. Such a constraint regularizes the gradient descent
211 space during learning, which can be expressed as:
212

213
$$\text{Var}(\|\nabla_\theta \mathcal{J}_{\text{POER}}\|_2) \leq \text{Var}(\|\nabla_\theta \mathcal{J}_{\text{GRPO}}\|_2) \quad (8)$$

214

215 In theory, our method enables a more stable training process. We provide detailed mathematical
216 proofs in Appendix B.

216 3.3 CACHE POOL UPDATE
217

218 After each gradient update, we adopt the ε -greedy algorithm to update the experience cache by
219 selecting the highest-reward response from the current inference results. Specifically, when the
220 random variable $u \sim \mathcal{U}(0, 1)$ satisfies $u \leq \varepsilon$, we select the response with the highest group reward;
221 otherwise, we randomly select a suboptimal response. We formalize the update process as:

$$223 \quad \mathcal{C}^{(t)} = \begin{cases} \left(\mathcal{C}^{(t-1)} \cup \left\{ (q_k, o_{\arg\max\{R_i\}_{i=1}^G}) \right\} \right) \setminus \{(q_k, a_k)\}, & \text{if } u \leq \varepsilon, \\ 224 \quad \left(\mathcal{C}^{(t-1)} \cup \{(q_k, o_{g'})\} \right) \setminus \{(q_k, a_k)\}, & \text{otherwise.} \end{cases} \quad (9)$$

226 Here, we denote the highest-reward response in the group as $o_{\arg\max\{R_i\}_{i=1}^G}$, and we randomly select
227 another candidate response as $o_{g'}$.
228

229 3.4 LENGTH-AWARE REWARD SHAPING
230

231 However, since the same response prefix is shared during the sampling phase, the diversity of
232 responses within the group is reduced compared to the GRPO algorithm. This lower diversity results
233 in more similar reward signals, thereby diminishing the effectiveness of policy gradient estimation.
234 To ensure meaningful gradients, reasonable reward differences are maintained within the group, even
235 when all responses are correct.

236 A **Length-Aware Reward Shaping** method is proposed to address this issue. This method is based
237 on the assumption that: *For the same question, a reasoning path that reaches the correct answer
238 more concisely should be rewarded with a higher value.* Specifically, for each response o_i in the
239 group, its length-aware reward $R(s_i)$ is computed as:

$$241 \quad R(s_i) = \text{clip} \left(\frac{r(s_i)}{1 + e^{-\alpha(\ell_{\text{ref}} - \text{len}(o_i))}}, m, \mathcal{M} \right) \quad (10)$$

244 Here, $r(s_i)$ is the original reward, and ℓ_{ref} is the average length of group G , defined as $\ell_{\text{ref}} =$
245 $\frac{1}{|G|} \sum_{i=1}^{|G|} \text{len}(o_i)$. The parameter $\alpha > 0$ controls the sensitivity of the reward to length differences.
246 m and \mathcal{M} are the lower and upper bounds for reward clipping to avoid extremely large or small
247 values. $\text{clip}(\cdot, m, \mathcal{M})$ denotes restricting a value within the interval $[m, \mathcal{M}]$.

248 We then iterate the above steps in Sections 3.2 and 3.3 until a predefined stopping step T is reached.

250 Through mathematical derivation, we demonstrate that length-aware rewards are better suited for
251 the POER algorithm; the two can complement each other, and when the guiding path is within a
252 certain threshold, they can enable the model to achieve greater performance gains. In contrast, the
253 GRPO algorithm, lacking an initial fixed guiding path, results in high variance for length-aware
254 rewards, making it difficult to accurately estimate the true effective policy gradient, and is therefore
255 not suitable for using length-aware rewards. Detailed proof is provided in Appendix C.

256 4 EXPERIMENTS
257258 4.1 EXPERIMENTAL SETUP
259

260 **Base Models** To demonstrate the effectiveness and generality of POER, we evaluate it on two
261 open-source inference models with 1.5B and 7B parameters, namely **Deepseek-r1-qwen-distill-1.5b**
262 and **Deepseek-r1-qwen-distill-7b** (DeepSeek-AI et al., 2025; Bai et al., 2023). Notably, we skip
263 the supervised fine-tuning (SFT) phase, which is usually a prerequisite for reinforcement learning
264 to enhance performance (Chu et al., 2025), as the selected models have already undergone this
265 stage (DeepSeek-AI et al., 2025).

266 **Evaluation and Datasets** We evaluate the models on six standard reasoning evaluation datasets:
267 aime25(math ai, b), aime24(math ai, a), math500(Hendrycks et al., 2021), amc23(math ai, c),
268 minerva(Lewkowycz et al., 2022) and olympicbench(He et al., 2024). To ensure fairness, all
269 evaluations use the lighteval(Habib et al., 2023) toolkit.

270 **Implementation Details** During training, we use 7k samples from the `open-rs` dataset (Dang
 271 & Ngo, 2025) with a global batch size of 576 for 4 epochs. Experiments are run on a single H20
 272 machine with 8xH20 96G GPUs. We generate 6 samples per prompt, set the temperature to 0.7, and
 273 fix the maximum generation length at 4096. For the length-aware reward, we use $m = 0.5$, $M = 1$,
 274 and $\alpha = 0.01$. All models are fully fine-tuned. Due to time constraints, only zero-shot performance
 275 is averaged over three runs; all other ablation experiments are run once.

277 4.2 ZERO-SHOT PERFORMANCE

279 We set the maximum truncation length L for each group to half of the minimum response length ℓ .
 280 Then, we train the DeepSeek-R1-Qwen-1.5B and DeepSeek-R1-Qwen-7B models for four epochs
 281 using the original GRPO algorithm, the DAPO algorithm, as well as their POER variants, with a
 282 batch size of 576 and a maximum generation length of 4096 tokens. To ensure that the experimental
 283 results are not caused by randomness, we repeat the training three times for each experiment. We
 284 then compare their mean accuracy on the designated evaluation datasets.

285 Table 1: Performance of the POER algorithm on test datasets. Arrows indicate performance changes
 286 relative to the base model: \uparrow indicates improvement, \downarrow indicates decline. w/ R means length-aware
 287 reward is used, w/o R means length-aware reward is not used. +POER shows the effect of applying
 288 the POER algorithm on top of the above method.

290 Model	291	AIME25	AIME24	MATH500	AMC23	Minerva	OlyB	Avg
1.5B Models								
292 DeepSeek-R1-Qwen-1.5B	293	16.7	28.8	82.2	62.9	26.5	43.3	43.4
+ GRPO(w/o R)		24.4 \uparrow	31.1 \uparrow	85.7 \uparrow	72.5 \uparrow	29.8 \uparrow	51.3 \uparrow	49.1 \uparrow
+ POER		24.4 \uparrow	25.6 \downarrow	84.3 \uparrow	69.2 \uparrow	29.5 \uparrow	51.7 \uparrow	47.5 \uparrow
+ GRPO(w/ R)		22.2 \uparrow	32.2 \uparrow	83.8 \uparrow	70.8 \uparrow	27.5 \uparrow	50.5 \uparrow	47.8 \uparrow
+ POER		24.4 \uparrow	35.6 \uparrow	85.3 \uparrow	83.3 \uparrow	29.8 \uparrow	51.8 \uparrow	51.7 \uparrow
+ DAPO(w/o R)		30.0 \uparrow	24.4 \downarrow	86.2 \uparrow	84.2 \uparrow	29.7 \uparrow	52.7 \uparrow	51.2 \uparrow
+ POER		28.9 \uparrow	24.4 \downarrow	86.0 \uparrow	84.5 \uparrow	29.3 \uparrow	52.1 \uparrow	50.9 \uparrow
+ DAPO(w/ R)		26.7 \uparrow	30.0 \uparrow	85.0 \uparrow	84.1 \uparrow	29.7 \uparrow	51.1 \uparrow	50.2 \uparrow
+ POER		32.2 \uparrow	30.0 \uparrow	86.2 \uparrow	86.1 \uparrow	29.1 \uparrow	52.3 \uparrow	52.7 \uparrow
7B Models								
300 DeepSeek-R1-Qwen-7B	301	43.3	55.5	92.8	90.0	44.5	67.4	65.6
+ GRPO(w/o R)		43.3	53.3 \downarrow	95.0 \uparrow	90.0	44.5	67.2 \downarrow	65.6
+ POER		43.3	46.6 \downarrow	92.5 \downarrow	89.2 \uparrow	42.3 \downarrow	67.7 \uparrow	63.6 \downarrow
+ GRPO(w/ R)		40.0 \downarrow	48.9 \downarrow	95.0 \uparrow	88.3 \uparrow	43.5 \downarrow	66.0 \downarrow	63.6 \downarrow
+ POER		50.0 \uparrow	61.1 \uparrow	94.2 \uparrow	90.8 \uparrow	43.7 \downarrow	67.3 \downarrow	67.8 \uparrow
+ DAPO(w/o R)		43.3	53.3 \downarrow	94.6 \uparrow	90.2 \uparrow	45.1 \uparrow	67.7 \uparrow	65.7 \uparrow
+ POER		46.7 \uparrow	52.2 \downarrow	94.2 \uparrow	91.2 \uparrow	42.7 \downarrow	64.9 \downarrow	65.3 \downarrow
+ DAPO(w/ R)		42.2 \downarrow	56.7 \uparrow	93.2 \uparrow	91.8 \uparrow	44.6 \uparrow	64.5 \downarrow	65.5 \downarrow
+ POER		46.7 \uparrow	54.5 \downarrow	94.8 \uparrow	95.2 \uparrow	43.1 \downarrow	64.5 \downarrow	66.5 \uparrow

309 As shown in Table 1, as mentioned in the Method section, length-aware rewards complement the
 310 POER algorithm. Incorporating group-wise length-aware rewards enables POER to achieve higher
 311 accuracy on test benchmarks than GRPO and DAPO for both the 1.5B and 7B model sizes. Without
 312 group-wise length-aware rewards, POER may experience some performance degradation; therefore,
 313 when using POER for accelerated training, it is recommended to include group-wise length-aware
 314 rewards to enhance performance.

315 4.3 TRAINING TIME OVERHEAD

317 To investigate the training time overhead of GRPO and POER, each experiment is conducted on
 318 a machine with 8 H20 GPUs, using only a single GPU for sampling during the training phase. It
 319 should be noted that POER introduces additional inference overhead during the cache initialization
 320 phase, where parallel inference is performed across all GPUs using the `v11m` framework. When the
 321 dataset size is 7k and the parallel batch size is 256, this phase takes approximately 20 minutes. Our
 322 experiments reveal that the primary factors affecting the relative training speed between POER and
 323 GRPO are the number of group samples G and the maximum truncation length L , while the impact
 of batch size is relatively minor.

324
325
326
327 Table 2: The average number of tokens generated per sample with the POER method
328
329

L	1.5B Model	7B Model
300	145.88	147.06
500	158.41	168.17
800	382.20	397.89
GRPO	2689.51	2457.91

330
331
332
333
334
335
336
337
338
339 Table 3: Training time of POER and GRPO
340 under 4 epochs with $L = 800$, h represents
341 hours
342
343

Method	1.5B Model	7B Model
GRPO	77.28 h	84.53 h
+POER	8.37 h	23.50 h

342
343
344
345
346
347
348
349 We study training speed for both 1.5B and 7B models. With maximum truncation length fixed at
350 $L = 300$, we set per-GPU batch sizes of 2 (1.5B) and 1 (7B), and evaluate group sizes $G = 6, 8, 16$.
351 As shown in Figure 3, smaller G yields greater acceleration for POER, reducing training time to 7.4%
352 of GRPO for 1.5B and 21.1% for 7B. We also study the effect of L with $G = 6$. Figure 3 shows that
353 larger L increases POER’s relative training time, with a smaller rise for the 7B model than for the
354 1.5B model.

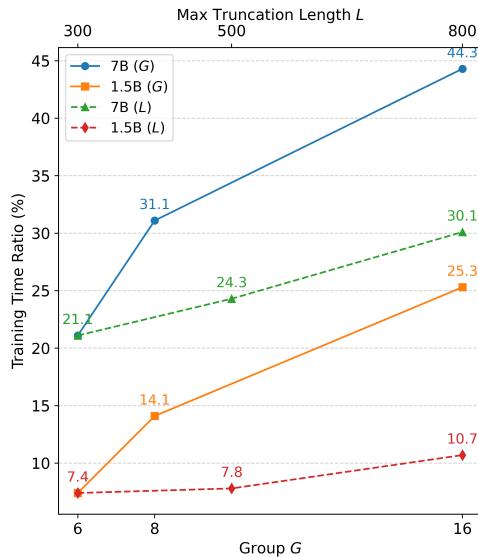
355 The actual training speed is affected by many factors, so we propose a fairer comparison: using
356 the average tokens generated per sample. Since prefill is much faster than decoding, more tokens
357 in prefill lead to shorter decoding time. As shown in Table 2, under the original GRPO algorithm,
358 each sample requires an average of 2689.51 tokens and 2457.91 tokens for the 1.5B and 7B models,
359 respectively. In contrast, with the POER algorithm, the number can be reduced to as low as 145.88
360 tokens and 147.06 tokens. From the perspective of the decode stage, the time overhead of POER is
361 only about 5% of that of GRPO. Table 3 presents the detailed training time overhead of the original
362 GRPO and POER algorithms over 4 epochs.

363 It is worth noting that the average number of tokens generated by the GRPO algorithm for the 1.5B
364 model is higher than that for the 7B model. However, when constrained by POER, the number of
365 tokens generated is lower. This is because the POER algorithm preserves shorter correct answers,
366 and the exploration capability of the 1.5B model, once guided, is weaker compared to that of the 7B
367 model, leading to this phenomenon.

368 369 4.4 STABILITY ANALYSIS 370

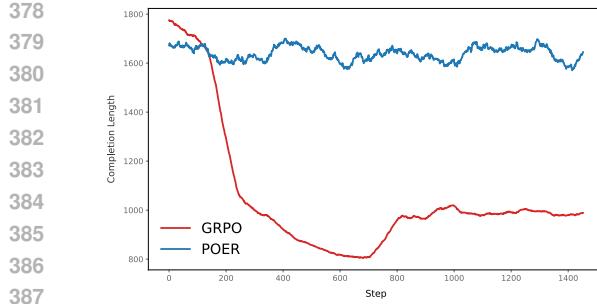
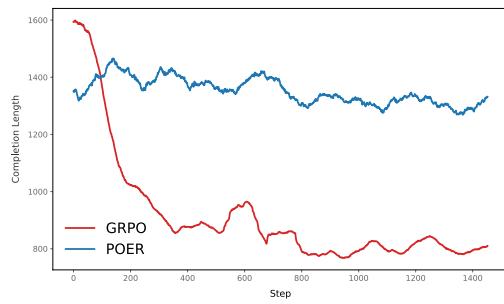
371 Traditional RL methods like GRPO and PPO are unstable in multi-step training: performance often
372 degrades with more iterations, and response length tends to shorten. Consequently, GRPO fine-tuning
373 usually limits iteration numbers to prevent deterioration, with accuracy and response length used to
374 measure model degradation (DeepSeek-AI et al., 2025). POER addresses this by using a cache pool
375 mechanism, and we conduct comparative experiments to quantify its improved training stability over
376 GRPO.

377 During the training process, we use a batch size of 18 to train the 7B and 1.5B models for four epochs,
378 and monitor changes in response length and model performance, as shown in Figure 4 and 5. In this



379
380
381
382
383
384
385
386
387
388
389 Figure 3: The impact of maximum truncation length L and group size G on the acceleration
390 ratio of POER under the 1.5B and 7B model settings. Due to the memory limitation of a single
391 machine, the maximum generation length for 7B (G) is 2048, while for all other cases is 4096.

392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
10010
10011
10012
10013
10014
10015
10016
10017
10018
10019
10020
10021
10022
10023
10024
10025
10026
10027
10028
10029
10030
10031
10032
10033
10034
10035
10036
10037
10038
10039
10040
10041
10042
10043
10044
10045
10046
10047
10048
10049
10050
10051
10052
10053
10054
10055
10056
10057
10058
10059
10060
10061
10062
10063
10064
10065
10066
10067
10068
10069
10070
10071
10072
10073
10074
10075
10076
10077
10078
10079
10080
10081
10082
10083
10084
10085
10086
10087
10088
10089
10090
10091
10092
10093
10094
10095
10096
10097
10098
10099
100100
100101
100102
100103
100104
100105
100106
100107
100108
100109
100110
100111
100112
100113
100114
100115
100116
100117
100118
100119
100120
100121
100122
100123
100124
100125
100126
100127
100128
100129
100130
100131
100132
100133
100134
100135
100136
100137
100138
100139
100140
100141
100142
100143
100144
100145
100146
100147
100148
100149
100150
100151
100152
100153
100154
100155
100156
100157
100158
100159
100160
100161
100162
100163
100164
100165
100166
100167
100168
100169
100170
100171
100172
100173
100174
100175
100176
100177
100178
100179
100180
100181
100182
100183
100184
100185
100186
100187
100188
100189
100190
100191
100192
100193
100194
100195
100196
100197
100198
100199
100200
100201
100202
100203
100204
100205
100206
100207
100208
100209
100210
100211
100212
100213
100214
100215
100216
100217
100218
100219
100220
100221
100222
100223
100224
100225
100226
100227
100228
100229
100230
100231
100232
100233
100234
100235
100236
100237
100238
100239
100240
100241
100242
100243
100244
100245
100246
100247
100248
100249
100250
100251
100252
100253
100254
100255
100256
100257
100258
100259
100260
100261
100262
100263
100264
100265
100266
100267
100268
100269
100270
100271
100272
100273
100274
100275
100276
100277
100278
100279
100280
100281
100282
100283
100284
100285
100286
100287
100288
100289
100290
100291
100292
100293
100294
100295
100296
100297
100298
100299
100300
100301
100302
100303
100304
100305
100306
100307
100308
100309
100310
100311
100312
100313
100314
100315
100316
100317
100318
100319
100320
100321
100322
100323
100324
100325
100326
100327
100328
100329
100330
100331
100332
100333
100334
100335
100336
100337
100338
100339
100340
100341
100342
100343
100344
100345
100346
100347
100348
100349
100350
100351
100352
100353
100354
100355
100356
100357
100358
100359
100360
100361
100362
100363
100364
100365
100366
100367
100368
100369
100370
100371
100372
100373
100374
100375
100376
100377
100378
100379
100380
100381
100382
100383
100384
100385
100386
100387
100388
100389
100390
100391
100392
100393
100394
100395
100396
100397
100398
100399
100400
100401
100402
100403
100404
100405
100406
100407
100408
100409
100410
100411
100412
100413
100414
100415
100416
100417
100418
100419
100420
100421
100422
100423
100424
100425
100426
100427
100428
100429
100430
100431
100432
100433
100434
100435
100436
100437
100438
100439
100440
100441
100442
100443
100444
100445
100446
100447
100448
100449
100450
100451
100452
100453
100454
100455
100456
100457
100458
100459
100460
100461
100462
100463
100464
100465
100466
100467
100468
100469
100470
100471
100472
100473
100474
100475
100476
100477
100478
100479
100480
100481
100482
100483
100484
100485
100486
100487
100488
100489
100490
100491
100492
100493
100494
100495
100496
100497
100498
100499
100500
100501
100502
100503
100504
100505
100506
100507
100508
100509
100510
100511
100512
100513
100514
100515
100516
100517
100518
100519
100520
100521
100522
100523
100524
100525
100526
100527
100528
100529
100530
100531
100532
100533
100534
100535
100536
100537
100538
100539
100540
100541
100542
100543
100544
100545
100546
100547
100548
100549
100550
100551
100552
100553
100554
100555
100556
100557
100558
100559
100560
100561
100562
100563
100564
100565
100566
100567
100568
100569
100570
100571
100572
100573
100574
100575
100576
100577
100578
100579
100580
100581
100582
100583
100584
100585
100586
100587
100588
100589
100590
100591
100592
100593
100594
100595
100596
100597
100598
100599
100600
100601
100602
100603
100604
100605
100606
100607
100608
100609
100610
100611
100612
100613
100614
100615
100616
100617
100618
100619
100620
100621
100622
100623
100624
100625
100626
100627
100628
100629
100630
100631
100632
100633
100634
100635
100636
100637
100638
100639
100640
100641
100642
100643
100644
100645
100646
100647
100648
100649
100650
100651
100652
100653
100654
100655
100656
100657
100658
100659
100660
100661
100662
100663
100664
100665
100666
100667
100668
100669
100670
100671
100672
100673
100674
100675
100676
100677
100678
100679
100680
100681
100682
100683
100684
100685
100686
100687
100688
100689
100690
100691
100692
100693
100694
100695
100696
100697
100698
100699
100700
100701
100702
100703
100704
100705
100706
100707
100708
100709
100710
100711
100712
100713
100714
100715<br

378
379
380
381
382
383
384
385
386
387
Figure 4: Response Length of GRPO and POER
389 on a 1.5B Model as a Function of Training Step
390388
389
390
Figure 5: Response Length of GRPO and POER
391 on a 7B Model as a Function of Training Step
392

393 multi-step iterative training setup, GRPO experiences a collapse in response length around the 200th
394 iteration, while POER maintains stable response lengths throughout the process. On the other hand,
395 as shown in Table 4, the model performance after training with GRPO deteriorated, especially for the
396 1.5B model, where accuracy dropped by 8.6%. In contrast, POER results in a 5.4% improvement in
397 accuracy.

398
399 Table 4: Performance of GRPO and POER on Evaluation Datasets in Multi-Step Iteration Scenarios
400

Model	AIME25	AIME24	MATH500	AMC23	Minerva	OlyB	Avg
DeepSeek-R1-Qwen-7B	43.3	55.5	92.8	90.0	44.5	67.4	65.6
+ GRPO(w/o R)	40.0 \downarrow	50.0 \downarrow	94.2 \uparrow	90.0	41.2 \downarrow	66.7 \downarrow	63.7 \downarrow
+ POER	46.6 \uparrow	56.7 \uparrow	92.8	90.0	41.4 \downarrow	66.1 \downarrow	65.6
DeepSeek-R1-Qwen-1.5B	16.7	28.8	82.2	62.9	26.5	43.3	43.4
+ GRPO(w/o R)	10.0 \downarrow	10.0 \downarrow	67.0 \downarrow	45.0 \downarrow	20.6 \downarrow	31.4 \downarrow	34.8 \downarrow
+ POER	20.0 \uparrow	36.7 \uparrow	82.8 \uparrow	72.5 \uparrow	29.4 \uparrow	51.5 \uparrow	48.8 \uparrow

401 Beyond the instability from reward sparsity, GRPO suffers from strong locality due to its inter-group
402 comparison strategy, limiting performance improvements. POER mitigates this by introducing an
403 experience cache, using an external cached policy π_c to approximate the main policy π_θ during
404 updates. This provides a global context, enhances training stability, and allows POER to maintain
405 consistent performance over long iterations.

406
407
408
409
410
411 Table 5: Impact of α and L on validation accuracy (%) of DeepSeek-R1-Qwen-1.5B
412

L	α	AIME25	AIME24	MATH500	AMC23	Minerva	OlyB	Avg
300	0	26.7	33.3	82.8	70.0	27.9	52.4	48.9
	0.01	26.7	33.3	85.2	77.5	29.4	52.0	50.7
	0.1	23.3	36.7	85.8	75.0	32.4	51.0	50.7
	1	23.3	36.7	83.2	72.5	31.6	53.2	50.1
0.5ℓ	0	30.0	33.3	84.4	70.0	29.0	50.5	49.5
	0.01	23.3	16.7	86.4	67.5	32.0	55.3	46.9
	0.1	36.7	33.3	85.2	75.0	27.9	52.7	51.8
	1	30.0	30.0	82.6	60.0	31.6	51.6	47.6
ℓ	0	33.3	26.7	84.6	75.0	27.8	53.3	50.1
	0.01	36.7	30.0	84.0	70.0	31.6	52.3	50.7
	0.1	30.0	36.7	84.4	70.0	28.3	53.5	50.4
	1	26.7	26.7	85.4	65.0	30.5	52.3	47.8

426
427
428
429
430
431
5 MORE ANALYSIS

432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1

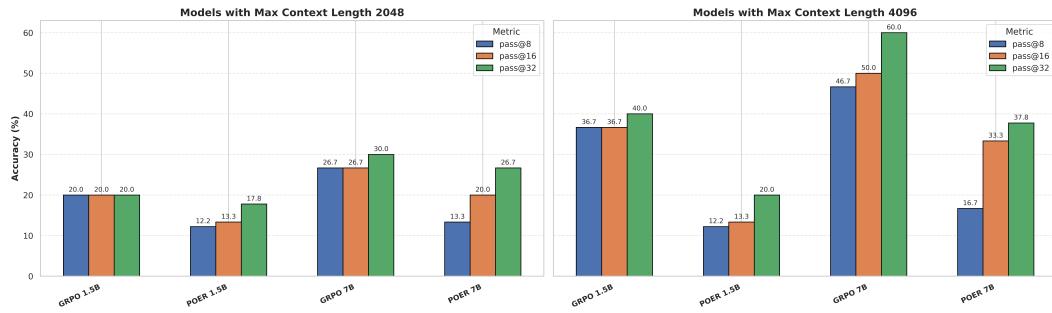


Figure 6: Pass@N performance of the GRPO and POER algorithms on the AIME24 dataset with the maximum truncation length set to 300. The left figure shows the case with a maximum generation length of 2048, and the right figure shows the case with a maximum generation length of 4096

The DeepSeek-R1-Qwen-1.5B model is trained for two epochs with a batch size of 336 to amplify differences in training outcomes for easier observation. The evaluation results are shown in Table 5: under a fixed L , performance first improves as α increases and then declines.

Effect of max_length on Exploration Capability To investigate the impact of `max_length` settings on the model’s initial exploration ability, we set the maximum truncate length of POER to 300 and examined the performance of the 1.5B and 7B models on the AIME24 dataset under two settings: `max_length` = 2048 and `max_length` = 4096. As shown in Figure 6, POER demonstrates lower exploration ability compared to GRPO, and the gap between the two methods gradually widens as `max_length` increases. This result also indicates that POER exhibits a certain disadvantage in exploration ability during the early iterations.

Impact of Cache Pool Update Strategy on Model’s Pass@N Performance To study the effect of training epochs on exploration, we evaluate the 1.5B and 7B models on AIME24 with `max_length` set to 2048 and 4096 under `epoch` = 1, 2. As shown in Figure 7, updating the cache pool over epochs enables the model to explore more diverse and higher-quality solution paths, steadily improving performance to match or surpass GRPO. Overall, while POER reduces raw exploration, the experience cache and epsilon-greedy strategy guide the model toward higher-quality paths.

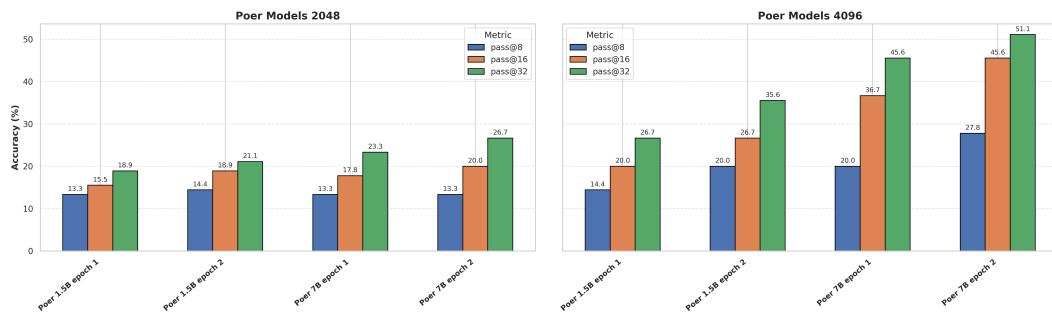


Figure 7: Pass@N performance of the GRPO and POER algorithms on the AIME24 dataset with the maximum truncation length set to 300 and epochs set to 1 and 2. The left figure shows the case with a maximum generation length of 2048, and the right figure shows the case with a maximum generation length of 4096

6 CONCLUSION

In this paper, we present POER, a plug-and-play algorithm designed to optimize the reinforcement fine-tuning of large models. POER aims to enhance the fine-tuning phase of large language models by introducing an experience replay mechanism. This mechanism allows the model to learn from previously collected high-quality responses during generation. POER significantly reduces model training time while improving the fine-tuned model’s performance and enhancing stability during the reinforcement fine-tuning phase.

486 THE USAGE OF LLM
487488 In this work, we use LLMs to polish the paper, generate materials for framework diagrams, and
489 retrieve related work.
490491 ETHICS STATEMENT
492493 This study does not involve any personal data, sensitive information, or high-risk application sce-
494 narios. No ethically controversial datasets or models were used. All experimental data are standard
495 benchmark datasets that are publicly available, and the sole purpose of this research is to advance the
496 development of reinforcement fine-tuning algorithm. Therefore, we believe this work does not pose
497 any significant ethical risks.
498499 REPRODUCIBILITY STATEMENT
500501 To ensure the reproducibility of our experiments, we have provided the complete implementation
502 code in the supplementary materials. All technical details, including the evaluation benchmarks,
503 baseline methods, and training hyperparameter settings used in this work, can be found in Section
504 4.1.
505506 REFERENCES
507508 Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge,
509 Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu,
510 Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanchi Tan,
511 Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin
512 Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng
513 Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou,
514 Xiaohuan Zhou, and Tianhang Zhu. Qwen technical report. *arXiv preprint arXiv:2309.16609*,
515 2023.516 Liang Chen, Lei Li, Haozhe Zhao, and Yifan Song. Vinci r1-v: Reinforcing super generalization
517 ability in vision-language models with less than 3.
518519 Yi Chen, Jian Xu, Xu-Yao Zhang, Wen-Zhuo Liu, Yang-Yang Liu, and Cheng-Lin Liu. Recoverable
520 compression: A multimodal vision token recovery mechanism guided by text information. In
521 *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 2293–2301, 2025.522 Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong
523 Ye, Hao Tian, Zhaoyang Liu, et al. Expanding performance boundaries of open-source multimodal
524 models with model, data, and test-time scaling. *arXiv preprint arXiv:2412.05271*, 2024.525 Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V.
526 Le, Sergey Levine, and Yi Ma. Sft memorizes, rl generalizes: A comparative study of foundation
527 model post-training, 2025. URL <https://arxiv.org/abs/2501.17161>.528 Quy-Anh Dang and Chris Ngo. Reinforcement learning for reasoning in small llms: What works and
529 what doesn't, 2025. URL <https://arxiv.org/abs/2503.16219>.531 DeepSeek-AI, Daya Guo, Dejian Yang, and et al. Deepseek-r1: Incentivizing reasoning capability in
532 llms via reinforcement learning, 2025. URL <https://arxiv.org/abs/2501.12948>.533 Jacob Eisenstein, Chirag Nagpal, Alekh Agarwal, Ahmad Beirami, Alexander Nicholas D'Amour,
534 Krishnamurthy Dj Dvijotham, Adam Fisch, Katherine A Heller, Stephen Robert Pfohl, Deepak
535 Ramachandran, et al. Helping or herding? reward model ensembles mitigate but do not eliminate
536 reward hacking. In *First Conference on Language Modeling*, 2024.538 Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang.
539 rstar-math: Small llms can master math reasoning with self-evolved deep thinking, 2025. URL
<https://arxiv.org/abs/2501.04519>.

540 Nathan Habib, Clémentine Fourrier, Hynek Kydlíček, Thomas Wolf, and Lewis Tunstall. Light-
 541 eval: A lightweight framework for llm evaluation, 2023. URL <https://github.com/huggingface/lighteval>.
 542

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

546 Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu,
 547 Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. Olympiad-
 548 bench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal
 549 scientific problems, 2024. URL <https://arxiv.org/abs/2402.14008>.
 550

551 Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song,
 552 and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset, 2021. URL
 553 <https://arxiv.org/abs/2103.03874>.
 554

555 Bairu Hou, Yang Zhang, Jiabao Ji, Yujian Liu, Kaizhi Qian, Jacob Andreas, and Shiyu Chang.
 556 Thinkprune: Pruning long chain-of-thought of llms via reinforcement learning. *arXiv preprint*
 557 *arXiv:2504.01296*, 2025.

558 Ke Ji, Jiahao Xu, Tian Liang, Qiuzhi Liu, Zhiwei He, Xingyu Chen, Xiaoyuan Liu, Zhiping Wang,
 559 Junying Chen, Benyou Wang, et al. The first few tokens are all you need: An efficient and effective
 560 unsupervised prefix fine-tuning method for reasoning models. *arXiv preprint arXiv:2503.02875*,
 561 2025.
 562

563 Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman,
 564 Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria
 565 Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca
 566 Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3:
 567 Pushing frontiers in open language model post-training, 2025. URL <https://arxiv.org/abs/2411.15124>.
 568

569 Aitor Lewkowycz, Anders Johan Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski,
 570 Vinay Venkatesh Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo,
 571 Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative rea-
 572 soning problems with language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave,
 573 and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL
 574 <https://openreview.net/forum?id=IFXTZERXdM7>.
 575

576 Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan
 577 Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023. URL
 578 <https://arxiv.org/abs/2305.20050>.

579 Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao,
 580 and Dacheng Tao. O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning.
 581 *arXiv preprint arXiv:2501.12570*, 2025.

582 Trung Quoc Luong, Xinbo Zhang, Zhanming Jie, Peng Sun, Xiaoran Jin, and Hang Li. Reft:
 583 Reasoning with reinforced fine-tuning. *arXiv preprint arXiv:2401.08967*, 3, 2024.

584 math ai. AIME24, year = 2024, url = <https://huggingface.co/datasets/math-ai/aime24>, a.

585 math ai. AIME25, year = 2025, url = <https://huggingface.co/datasets/math-ai/aime25>, b.

586 math ai. AMC23, year = 2023, url = <https://huggingface.co/datasets/math-ai/amc23>, c.

587 OpenAI, , Aaron Jaech, and Adam Kalai et al. Openai o1 system card, 2024a. URL <https://arxiv.org/abs/2412.16720>.

588 OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, and et al. Gpt-4 technical report, 2024b.
 589 URL <https://arxiv.org/abs/2303.08774>.

594 Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong
 595 Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton,
 596 Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and
 597 Ryan Lowe. Training language models to follow instructions with human feedback, 2022. URL
 598 <https://arxiv.org/abs/2203.02155>.

599 Yingzhe Peng, Gongrui Zhang, Miaosen Zhang, Zhiyuan You, Jie Liu, Qipeng Zhu, Kai Yang,
 600 Xingzhong Xu, Xin Geng, and Xu Yang. Lmm-r1: Empowering 3b llms with strong reasoning
 601 abilities through two-stage rule-based rl. *arXiv preprint arXiv:2503.07536*, 2025.

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

604 Haozhan Shen, Peng Liu, Jingcheng Li, Chunxin Fang, Yibo Ma, Jiajia Liao, Qiaoli Shen, Zilun
 605 Zhang, Kangjia Zhao, Qianqian Zhang, et al. Vlm-r1: A stable and generalizable r1-style large
 606 vision-language model. *arXiv preprint arXiv:2504.07615*, 2025a.

607 Maohao Shen, Guangtao Zeng, Zhenting Qi, Zhang-Wei Hong, Zhenfang Chen, Wei Lu, Gregory
 608 Wornell, Subhro Das, David Cox, and Chuang Gan. Satori: Reinforcement learning with chain-of-
 609 action-thought enhances llm reasoning via autoregressive search. *arXiv preprint arXiv:2502.02508*,
 610 2025b.

611 Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient
 612 methods for reinforcement learning with function approximation. In S. Solla, T. Leen,
 613 and K. Müller (eds.), *Advances in Neural Information Processing Systems*, volume 12.
 614 MIT Press, 1999. URL https://proceedings.neurips.cc/paper_files/paper/1999/file/464d828b85b0bed98e80ade0a5c43b0f-Paper.pdf.

615 Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun
 616 Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1.5: Scaling reinforcement learning with
 617 llms. *arXiv preprint arXiv:2501.12599*, 2025.

618 Hugo Touvron, Thibaut Lavrille, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée
 619 Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand
 620 Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language
 621 models, 2023. URL <https://arxiv.org/abs/2302.13971>.

622 Xinning Wang, Jian Xu, Aslan H Feng, Yi Chen, Haiyang Guo, Fei Zhu, Yuanqi Shao, Minsi Ren,
 623 Hongzhu Yi, Sheng Lian, et al. The hitchhiker’s guide to autonomous research: A survey of
 624 scientific agents. *Authorea Preprints*, 2025.

625 Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le,
 626 and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023.
 627 URL <https://arxiv.org/abs/2201.11903>.

628 Gaotong Yu, Yi Chen, and Jian Xu. Sparsity meets similarity: Leveraging long-tail distribution
 629 for dynamic optimized token representation in multimodal large language models, 2025a. URL
 630 <https://arxiv.org/abs/2409.01162>.

631 Qiyi Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong
 632 Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi
 633 Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi
 634 Wang, Hongli Yu, Weinan Dai, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying
 635 Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source
 636 llm reinforcement learning system at scale, 2025b. URL <https://arxiv.org/abs/2503.14476>.

637 Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu,
 638 Wendi Zheng, Xiao Xia, Weng Lam Tam, Zixuan Ma, Yufei Xue, Jidong Zhai, Wenguang Chen,
 639 Peng Zhang, Yuxiao Dong, and Jie Tang. Glm-130b: An open bilingual pre-trained model, 2023.
 640 URL <https://arxiv.org/abs/2210.02414>.

641 Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keping He, Zejun Ma, and Junxian He. Simplerl-
 642 zoo: Investigating and taming zero reinforcement learning for open base models in the wild, 2025.
 643 URL <https://arxiv.org/abs/2503.18892>.

648 A PSEUDO CODE FOR POER TRAINING PROCESS
649650 The pseudo code of *Policy Optimization with Experience Replay*(POER) in the training process is as
651 follows.
652653 **Algorithm 1:** Compact POER Training654 **Input:** Dataset \mathcal{D} , model π_{θ_0} , cache $\mathcal{C}^{(0)}$, params η, β, ϵ, G 655 **Output:** Model π_{θ_T} , cache $\mathcal{C}^{(T)}$ 656 **Initialize Cache:**657 $\mathcal{C}^{(0)} = \emptyset$ 658 **for** $q_k \in \mathcal{D}$ **do**659 $\quad \mathcal{C}^{(0)} \leftarrow \mathcal{C}^{(0)} \cup \{(q_k, \pi_{\theta_0}(\cdot | q_k))\}$ 660 **for** $t \leftarrow 1$ **to** T **do**661 **Rollout:**662 **for** $q_k \in \mathcal{D}$ **do**663 $a_k \leftarrow \{a \mid (q_k, a) \in \mathcal{C}^{(t-1)}\}$ 664 **for** $g \leftarrow 1$ **to** G **do**665 $\tilde{a}_i^{(g)} \leftarrow \text{Concat}(a_k^{[0:-m]}, \pi_{\theta_{t-1}}(\cdot | q_k, a_k^{[0:-m]}))$
666 $R_i^{(g)} \leftarrow r(q_k, \tilde{a}_i^{(g)})$ 667 **Optimize:**668 $\theta_t \leftarrow \theta_{t-1} + \eta \nabla_{\theta} J_{\text{GRPO-Cache}}$ 669 **Update Cache:**670 **for** $q_k \in \mathcal{D}$ **do**671 **if** $u \leq \epsilon$ **then**672 $\mathcal{C}^{(t-1)} \cup \{(q_k, o_{\text{argmax}\{R_i\}_{i=1}^G})\} \setminus \{(q_k, a_k)\}$ 673 **else**674 $\mathcal{C}^{(t)} \leftarrow \mathcal{C}^{(t-1)} \cup \{(q_k, o_{g'})\} \setminus (q_k, a_k); \quad // g' \sim \mathcal{U}(1, G)$

675 B PROOF OF POER GRADIENT STABILITY

676 **Policy gradient estimation.** The reasoning process of traditional GRPO and *Policy Optimization*
677 with *Experience Replay*(POER) can be expressed as $\pi(\cdot | q_k)$ and $\pi(\cdot | (q_k, a_k))$, where $q_k \in C_{\text{raw}}$, is
678 a sample in raw training dataset C_{raw} and corresponding $(q_k, a_k) \in C$, is one example in training
679 replay buffer C , which updates during the training process.680 For any sample q_k , it holds that $(q_k) \subset (q_k, a_k)$. Hence, for the response space of an arbitrary policy
681 model, the total variance can give

682
$$\text{Var}(\cdot | q_k) = \mathbb{E}_{(q_k, a_k) | q_k} [\text{Var}(\cdot | (q_k, a_k))] + \text{Var}_{(q_k, a_k) | q_k} (\mathbb{E}[\cdot | (q_k, a_k)]). \quad (11)$$

683 Because the second term on the right-hand side is non-negative, i.e. $\text{Var}_{(q_k, a_k) | q_k} (\mathbb{E}[\cdot | (q_k, a_k)]) \geq 0$,
684 we obtain

685
$$\text{Var}(\cdot | q_k) \geq \mathbb{E}_{(q_k, a_k) | q_k} [\text{Var}(\cdot | (q_k, a_k))]. \quad (12)$$

686 Treating (q_k, a_k) as an augmentation of q_k allows this inequality to simplify to

687
$$\text{Var}(\cdot | q_k) \geq \text{Var}(\cdot | (q_k, a_k)). \quad (13)$$

688 In the policy space, equation 13 becomes

689
$$\text{Var}(\pi_{\theta}(a_g | q_k)) \geq \text{Var}(\pi_{\theta}(a_g | (q_k, a_k))). \quad (14)$$

690 Assume there exists a parameter vector θ_0 such that the policy can be locally approximated by the
691 first-order expansion

692
$$\pi_{\theta} = \pi_{\theta_0} + \nabla \pi^{\top}(\theta_0)(\theta - \theta_0). \quad (15)$$

693 The variance of π_{θ} in a neighbourhood of θ_0 can then be estimated as

694
$$\sigma^2(\pi_{\theta}) \approx \nabla \pi_{\theta_0}^{\top} \Sigma_{\theta} \nabla \pi_{\theta_0}, \quad (16)$$

702 where Σ_θ denotes the covariance matrix of the parameter estimates.
 703

704 Throughout training, the realizations of π_θ can be treated as i.i.d. random variables. As the sample
 705 size $n \rightarrow \infty$, the empirical mean and variance converge to

$$706 \quad \mu(\pi_\theta) = \mathbb{E}[\pi_\theta] = \frac{1}{n} \sum_{i=1}^n (\pi_\theta)_i, \quad \text{Var}(\pi_\theta) = \mathbb{E}[\pi_\theta^2] - \mu(\pi_\theta)^2 \approx \frac{n}{n-1} \nabla \pi_{\theta_0}^\top \Sigma_\theta \nabla \pi_{\theta_0}.$$

709 On account of $\mathbb{E}[\pi_\theta^2] = \mu(\pi_\theta)^2$, while $\Sigma_\theta = I$ also holds, then
 710

$$711 \quad \text{Var}(\pi_\theta) = \mathbb{E}[\pi_\theta^2] - \mu(\pi_\theta)^2 = \nabla \pi_{\theta_0}^\top \nabla \pi_{\theta_0} = \|\nabla \pi_{\theta_0}\|_2^2. \quad (17)$$

712 Combining the above with equation 13 yields the policy space gradient estimation.
 713

$$714 \quad \|\nabla \pi_\theta(a_g | q_k)\|_2 \geq \|\nabla \pi_\theta(a_g | (q_k, a_k))\|_2. \quad (18)$$

715 This establishes that conditioning on the augmented information (q_k, a_k) strictly reduces—or at worst
 716 preserves—the magnitude of the policy-gradient variance.
 717

718 Let's review the GRPO update policy. At training step t , the optimisation target of *Generative*
 719 *Reinforcement Policy Optimisation* (GRPO) can be written as
 720

$$721 \quad \mathcal{J}_{\text{GRPO}}(\theta_t) = \mathbb{E}_{(q, a) \sim \mathcal{C}^{(t-1)}(Q, A), \{o_i\}_{i=1}^G \sim \pi_{\theta_{t-1}}(\cdot | q)} \left[\frac{1}{G} \sum_{i=1}^G \frac{1}{|o_i|} \sum_{j=1}^{|o_i|} \min \left(r_{i,j}(\theta_t) \hat{A}_{i,j}, \text{clip} \left(r_{i,j}(\theta_t), 1 - \epsilon, 1 + \epsilon \right) \hat{A}_{i,j} - \beta D_{\text{KL}}(\pi_{\theta_t} \| \pi_{\text{ref}}) \right) \right] \quad (19)$$

722 Here

$$723 \quad r_{i,j}(\theta_t) = \frac{\pi_{\theta_t}(o_{i,j} | q)}{\pi_{\theta_{t-1}}(o_{i,j} | q)}, \quad \hat{A}_{i,j} = \frac{R_i - \text{mean}(\{R_i\}_{i=1}^G)}{\text{std}(\{R_i\}_{i=1}^G)}.$$

724 Relative to GRPO, POER only changes the policy ratio by conditioning on the prefix $o_{i,<j}$:
 725

$$726 \quad r_{i,j}^{\text{POER}}(\theta_t) = \frac{\pi_{\theta_t}(o_{i,j} | q, o_{i,<j})}{\pi_{\theta_{t-1}}(o_{i,j} | q, o_{i,<j})}. \quad (20)$$

727 Because the `clip` operation truncates high-error updates, both algorithms behave identically whenever
 728 clipping is activated.
 729

730 **Gradient of the optimization policy.** For the plain GRPO ratio one obtains
 731

$$732 \quad \nabla_\theta r_{i,j}(\theta_t) = \frac{\nabla_\theta \pi_{\theta_t}(o_{i,j} | q)}{\pi_{\theta_{t-1}}(o_{i,j} | q)} = \frac{\pi_{\theta_t}(o_{i,j} | q)}{\pi_{\theta_{t-1}}(o_{i,j} | q)} \nabla_\theta \log \pi_{\theta_t}(o_{i,j} | q). \quad (21)$$

733 The Kullback–Leibler divergence with respect to a frozen reference policy π_{ref} satisfies
 734

$$735 \quad \nabla_\theta D_{\text{KL}}(\pi_{\theta_t} \| \pi_{\text{ref}}) = \nabla_\theta \mathbb{E}_{\pi_{\theta_t}} [\log \pi_{\theta_t} - \log \pi_{\text{ref}}] \\ 736 \quad = \mathbb{E}_{\pi_{\theta_t}} [\nabla_\theta \log \pi_{\theta_t} + (\log \pi_{\theta_t} - \log \pi_{\text{ref}}) \nabla_\theta \log \pi_{\theta_t}] \\ 737 \quad = \mathbb{E}_{\pi_{\theta_t}} [\nabla_\theta \log \pi_{\theta_t} (\log \frac{\pi_{\theta_t}}{\pi_{\text{ref}}} + 1)] \quad (22)$$

738 As $\mathbb{E}_{\pi_{\theta_t}} [\nabla_\theta \log \pi_{\theta_t}] = 0$ by normalisation, the expression simplifies to
 739

$$740 \quad \nabla_\theta D_{\text{KL}} = \mathbb{E}_{\pi_{\theta_t}} [\nabla_\theta \log \pi_{\theta_t}(o | s) \cdot \log \pi_{\theta_t}(o | s)] \\ 741 \quad = \sum_{\pi_\theta} \pi_\theta \nabla_\theta \log \pi_\theta(o | s) \cdot \log \pi_\theta(o | s) \\ 742 \quad = \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \pi_\theta \nabla_\theta \log \pi_\theta(o_i | s) \cdot \log \pi_\theta(o_i | s) \quad (23)$$

756 **Resulting policy gradient.** Aggregating the intra-group updates yields the estimator
 757

$$\begin{aligned}
 758 \quad \nabla_{\theta} \mathcal{J}_{\text{GRPO}} &= \mathbb{E}_{q, \{o_i\}} \left[\frac{1}{G} \sum_{i=1}^G \frac{1}{|o_i|} \sum_{j=1}^{|o_i|} \left(\frac{\hat{A}_{i,j}}{\pi_{\theta_{t-1}}(o|q)} - \beta \log \pi_{\theta_t}(o|q) \right) \pi_{\theta_t}(o|q) \nabla_{\theta} \log \pi_{\theta_t}(o|q) \right] \\
 759 \\
 760 \quad &= \mathbb{E}_{q, \{o_i\}} \left[\frac{1}{G} \sum_{i=1}^G \frac{1}{|o_i|} \sum_{j=1}^{|o_i|} \left(\frac{\hat{A}_{i,j}}{\pi_{\theta_{t-1}}(o|q)} - \beta \log \pi_{\theta_t}(o|q) \right) \nabla_{\theta} \pi_{\theta_t}(o|q) \right]. \\
 761 \\
 762 \\
 763 \\
 764 \end{aligned} \tag{24}$$

765 The second line follows by noting that $\pi_{\theta_t} \nabla_{\theta} \log \pi_{\theta_t} = \nabla_{\theta} \pi_{\theta_t}$. Equation above provides the final
 766 form of the GRPO gradient used for parameter updates at step t .
 767
 768

769 **Without consideration of KL divergence.** If the KL–divergence term is temporarily ignored, the
 770 GRPO gradient estimator reduces to
 771

$$\nabla_{\theta} \mathcal{J}_{\text{GRPO}} = \mathbb{E}_{q, \{o_i\}} \left[\frac{1}{G} \sum_{i=1}^G \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \left(\frac{\hat{A}_{i,t}}{\pi_{\theta_{t-1}}} \right) \nabla \pi_{\theta}(a_g | q_k) \right]. \tag{25}$$

775 Because of equation equation 18,
 776

$$\|\nabla_{\theta} \mathcal{J}_{\text{GRPO}}\|_2 \geq \|\nabla_{\theta} \mathcal{J}_{\text{POER}}\|_2. \tag{26}$$

778 **Including the KL divergence.** At the initial step ($t = 0$) both algorithms share the same reference
 779 policy, hence
 780

$$\nabla_{\theta} \mathcal{J}_{\text{GRPO}} = \nabla_{\theta} \mathcal{J}_{\text{POER}}. \tag{27}$$

782 For the first update ($t = 1$) equation 18 implies
 783

$$\|\nabla \pi_{\theta_1}(a_g | q_k)\|_2 \geq \|\nabla \pi_{\theta_1}(a_g | q_k, a_k)\|_2. \tag{28}$$

785 Here we record $\nabla \pi_{\theta_1}(a_g | q_k, a_k)$ as $\nabla \pi'_{\theta_1}$. Consequently, the difference of the two policy gradients
 786 becomes
 787

$$\begin{aligned}
 \Delta_1 &= \|\nabla_{\theta} \mathcal{J}_{\text{GRPO}}\|_2 - \|\nabla_{\theta} \mathcal{J}_{\text{POER}}\|_2 \\
 &= \mathbb{E}_{q, \{o_i\}} \left[\frac{1}{G} \sum_{i=1}^G \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \left(\left\| \left(\frac{\hat{A}_{i,1}}{\pi_{\theta_0}} - \beta \log \pi_{\theta_1} \right) \nabla \pi_{\theta_1} \right\|_2 - \left\| \left(\frac{\hat{A}'_{i,1}}{\pi_{\theta_0}} - \beta \log \pi'_{\theta_1} \right) \nabla \pi'_{\theta_1} \right\|_2 \right) \right]. \\
 788 \\
 789 \\
 790 \\
 791 \\
 792 \end{aligned} \tag{29}$$

793 Let $\delta := \nabla_{\theta} \pi_{\theta_1}(i) - \nabla_{\theta} \pi'_{\theta_1}(i) \geq 0$. For one random dimension the mean-value theorem yields
 794

$$\pi_{\theta_1}(i) \log \pi_{\theta_1}(i) - \pi'_{\theta_1}(i) \log \pi'_{\theta_1}(i) = \frac{\partial(\pi \log \pi)}{\partial \pi} \Big|_{\pi=\zeta} (\pi_{\theta_1}(i) - \pi'_{\theta_1}(i)), \quad \zeta \in [\pi'_{\theta_1}, \pi_{\theta_1}] \subset [0, 1]. \tag{30}$$

797 Taking the directional derivative with respect to θ gives
 798

$$\begin{aligned}
 799 \quad (1 + \log \pi_{\theta_1}(i)) \nabla \pi_{\theta_1}(i) - (1 + \log \pi'_{\theta_1}(i)) \nabla \pi'_{\theta_1}(i) &= \frac{\partial \pi_{\theta} \log \pi_{\theta}}{\partial \pi_{\theta}} \Big|_{\pi_{\theta}=\zeta} (\nabla \pi_{\theta_1}(i) - \nabla \pi'_{\theta_1}(i)) \\
 800 \\
 801 &= \frac{\partial \pi_{\theta} \log \pi_{\theta}}{\partial \pi_{\theta}} \Big|_{\pi_{\theta}=\zeta} \delta \\
 802 \\
 803 \end{aligned} \tag{31}$$

804 Hence
 805

$$\log \pi_{\theta_1}(i) \nabla \pi_{\theta_1}(i) - \log \pi'_{\theta_1}(i) \nabla \pi'_{\theta_1}(i) = \left(\frac{\partial(\pi \log \pi)}{\partial \pi} \Big|_{\pi=\zeta} - 1 \right) \delta = (\log \zeta) \delta \leq 0, \tag{32}$$

806 because $\log \zeta < 0$.
 807

808 Extending this argument component-wise to the full parameter vector shows
 809

$$\log \pi_{\theta_1} \nabla \pi_{\theta_1} \preceq \log \pi'_{\theta_1} \nabla \pi'_{\theta_1}, \tag{33}$$

810 and therefore

$$811 \quad \| -\beta \log \pi_{\theta_1} \nabla \pi_{\theta_1} \|_2 \geq \| -\beta \log \pi'_{\theta_1} \nabla \pi'_{\theta_1} \|_2, \quad (34)$$

812 We have thus established

$$814 \quad \| \nabla_{\theta_1} \mathcal{J}_{\text{GRPO}} \|_2 \geq \| \nabla_{\theta_1} \mathcal{J}_{\text{POER}} \|_2. \quad (35)$$

816 Let the generic update rule be

$$817 \quad \theta_i = \theta_{i-1} + \eta \nabla_{\theta} \mathcal{J}. \quad (36)$$

819 Then

$$820 \quad \frac{\nabla \pi_{\theta_i}}{\pi_{\theta_{i-1}}} = \frac{\nabla (\pi_{\theta_{i-1}} + \eta \nabla \pi_{\theta_{i-1}} \nabla_{\theta_{i-1}} \mathcal{J})}{\pi_{\theta_{i-1}}} \geq \frac{\nabla \pi_{\theta_{i-1}}}{\pi_{\theta_{i-1}}} = \nabla \log \pi_{\theta_{i-1}}, \quad (37)$$

823 i.e. each step re-enters the original policy-gradient (PG) regime. Using equation 35 one obtains for
824 every $i \geq 1$

$$825 \quad \frac{\nabla \pi_{\theta_i}}{\pi_{\theta_{i-1}}} - \frac{\nabla \pi'_{\theta_i}}{\pi'_{\theta_{i-1}}} \geq \frac{\nabla \pi_{\theta_{i-1}}}{\pi_{\theta_{i-1}}} - \frac{\nabla \pi'_{\theta_{i-1}}}{\pi'_{\theta_{i-1}}} \geq \nabla \log \frac{\pi_{\theta_{i-1}}}{\pi'_{\theta_{i-1}}}. \quad (38)$$

827 By induction this yields the general relation

$$829 \quad \| \nabla_{\theta} \mathcal{J}_{\text{GRPO}} \|_2 \geq \| \nabla_{\theta} \mathcal{J}_{\text{POER}} \|_2 \quad \text{for all optimisation steps.} \quad (39)$$

831 Equation equation 39 completes the proof that, under identical hyper-parameters, GRPO provides
832 gradient updates at least as large as those of POER, both without and with the KL divergence.
833 Meanwhile, since both of them follow a normal distribution with zero mean, it follows that:

$$834 \quad \text{Var}(\| \nabla_{\theta} \mathcal{J}_{\text{POER}} \|_2) \leq \text{Var}(\| \nabla_{\theta} \mathcal{J}_{\text{GRPO}} \|_2) \quad (40)$$

836 C THEOREM

839 **Preliminary.** Let q denote the prompt, o a sampled response with length $\ell = \text{len}(o)$, and let the
840 group-wise reference length be $\ell_{\text{ref}} = \frac{1}{G} \sum_{i=1}^G \text{len}(o_i)$. Write $\Delta \ell = \ell - \ell_{\text{ref}}$ and fix a window
841 $|\Delta \ell| \leq \tau$. For a sensitivity parameter $\alpha > 0$ define the length weight $s_{\alpha}(\ell) = \sigma(\alpha(\ell_{\text{ref}} - \ell)) =$
842 $(1 + e^{-\alpha(\ell_{\text{ref}} - \ell)})^{-1} \in (0, 1)$ and the shaped reward $R_{\alpha} = \text{clip}(s_{\alpha}(\ell) r, m, \mathcal{M})$ with clipping
843 bounds $m < \mathcal{M}$, where r is the original per-sample reward.

844 Consider POER with replay distribution μ and current policy π_{θ} , truncated importance ratio
845 $\rho = \min(c, \frac{\pi_{\theta}(o|q)}{\mu(o|q)})$ for a constant $c \geq 1$, token-averaged score function $\nabla_{\theta} \log \pi_{\theta}(o | q) =$
846 $\frac{1}{|o|} \sum_{j=1}^{|o|} \nabla_{\theta} \log \pi_{\theta}(o_j | o_{<j}, q)$, and a centered advantage $A'_{\alpha} = R_{\alpha} - b_{\alpha}$ with group baseline
847 $b_{\alpha} = \mathbb{E}[R_{\alpha} | q, \text{group}]$. The single-sample gradient contribution is

$$850 \quad g_{\alpha} = \rho \left(A'_{\alpha} - \beta \log \pi_{\theta}(o | q) \right) \nabla_{\theta} \log \pi_{\theta}(o | q). \quad (41)$$

852 Assume $\| \nabla_{\theta} \log \pi_{\theta}(o | q) \| \leq L, \mathbb{E}[r^2] < \infty$, and that

- 854 1. the conditional variance $\sigma_A^2(\ell) := \text{Var}(A | \ell)$ of the unshaped advantage A is nondecreasing
855 in ℓ ,
- 856 2. the tail probability $\mathbb{P}(\frac{\pi_{\theta}}{\mu} > c | \ell)$ is nondecreasing in ℓ .

858 If $\alpha\tau \leq 1$, then there exists $\alpha^* > 0$ such that for all $0 < \alpha \leq \alpha^*$ the mean-squared error
859 $\text{MSE}_{\alpha} := \text{Var}(g_{\alpha}) + \|\mathbb{E}[g_{\alpha}] - \nabla_{\theta} J\|^2$ of the POER gradient estimator with length-aware shaping
860 satisfies

$$861 \quad \text{MSE}_{\alpha} < \min \left\{ \text{MSE}_0^{\text{POER}}, \text{MSE}_0^{\text{GRPO}}, \inf_{\tilde{\alpha} > 0} \text{MSE}_{\tilde{\alpha}}^{\text{GRPO}} \right\}, \quad (42)$$

863 that is, it strictly improves upon both the unshaped POER baseline and the GRPO baselines in a
nontrivial neighborhood of $\alpha = 0$.

864 **Proof.** The proof makes explicit the first-order behavior in α of both the variance and the bias terms.
 865 Throughout the window $|\Delta\ell| \leq \tau$ the sigmoid admits the uniform Taylor expansion
 866

$$867 \quad s_\alpha(\ell) = \frac{1}{2} - \frac{\alpha}{4} \Delta\ell + R_2(\alpha, \ell), \quad |R_2(\alpha, \ell)| \leq C_2 \alpha^2 \tau^2, \quad (43)$$

869 for some constant C_2 independent of α and ℓ . Writing $R_\alpha = s_\alpha(\ell) r$ on the non-clipping region and
 870 absorbing the clipping into the moment bounds later, the centered advantage becomes
 871

$$872 \quad A'_\alpha = \left(\frac{1}{2} r - \mathbb{E}\left[\frac{1}{2} r\right] \right) - \frac{\alpha}{4} \left(\Delta\ell r - \mathbb{E}[\Delta\ell r] \right) + \underbrace{R_2(\alpha, \ell) r - \mathbb{E}[R_2(\alpha, \ell) r]}_{=:E_2(\alpha)}. \quad (44)$$

874 Substituting equation 44 into equation 41 and taking expectations yields
 875

$$876 \quad \mathbb{E}[g_\alpha] - \mathbb{E}[g_0] = -\frac{\alpha}{4} \mathbb{E}\left[\rho (\Delta\ell r - \mathbb{E}[\Delta\ell r]) \nabla_\theta \log \pi_\theta(o | q)\right] + \mathbb{E}\left[\rho E_2(\alpha) \nabla_\theta \log \pi_\theta(o | q)\right]. \quad (45)$$

877 By Cauchy–Schwarz and the bounds on ρ and the score function, the norm of the first term on the
 878 right-hand side satisfies
 879

$$880 \quad \left\| \mathbb{E}\left[\rho (\Delta\ell r - \mathbb{E}[\Delta\ell r]) \nabla_\theta \log \pi_\theta\right] \right\| \leq c L \left(\mathbb{E}[(\Delta\ell r - \mathbb{E}[\Delta\ell r])^2] \right)^{1/2} \leq c L \tau (\mathbb{E}[r^2])^{1/2}, \quad (46)$$

881 hence $\|\mathbb{E}[g_\alpha] - \mathbb{E}[g_0]\| \leq \frac{\alpha}{4} c L \tau (\mathbb{E}[r^2])^{1/2} + c L \mathbb{E}[|E_2(\alpha)|]$. Using $|E_2(\alpha)| \leq 2C_2 \alpha^2 \tau^2 |r|$ and
 882 $\mathbb{E}[r^2] < \infty$ gives the bias bound
 883

$$884 \quad \|\mathbb{E}[g_\alpha] - \mathbb{E}[g_0]\| \leq C_b \alpha \tau + C'_b \alpha^2 \tau^2, \quad (47)$$

885 for constants C_b, C'_b depending only on $(c, L, \mathbb{E}[r^2], C_2)$. Consequently the squared-bias contribution
 886 to MSE_α is $O(\alpha^2 \tau^2)$.
 887

888 For the variance term, expand the second moment as
 889

$$890 \quad \mathbb{E}[\|g_\alpha\|^2] \leq c^2 \mathbb{E}\left[(A'_\alpha - \beta \log \pi_\theta)^2 \|\nabla_\theta \log \pi_\theta\|^2\right] \leq c^2 L^2 \mathbb{E}\left[(A'_\alpha - \beta \log \pi_\theta)^2\right]. \quad (48)$$

891 The cross terms between A'_α and $\beta \log \pi_\theta$ are uniformly bounded in α by Jensen and the finite second
 892 moments of r and $\log \pi_\theta$. The α -dependent leading component arises from $\mathbb{E}[A'_\alpha]^2$. Within the
 893 non-clipping region and after centering, the contribution that depends on length is proportional to
 894

$$895 \quad \mathbb{E}[s_\alpha(\ell)^2 \sigma_A^2(\ell)] = \mathbb{E}[s_\alpha(\ell)^2] \mathbb{E}[\sigma_A^2(\ell)] - \text{Cov}(s_\alpha(\ell)^2, \sigma_A^2(\ell)). \quad (49)$$

896 Since $s_\alpha(\ell)$ is nonincreasing in ℓ while $\sigma_A^2(\ell)$ is nondecreasing in ℓ by assumption, the reverse
 897 Chebyshev inequality ensures that the covariance in equation 49 is nonpositive and is strictly negative
 898 unless $s_\alpha(\ell)^2$ and $\sigma_A^2(\ell)$ are almost surely constant. Differentiating $\mathbb{E}[s_\alpha(\ell)^2]$ at $\alpha = 0$ and using
 899 equation 43 yields $\mathbb{E}[s_\alpha(\ell)^2] = \frac{1}{4} + O(\alpha^2 \tau^2)$, while differentiating the covariance at $\alpha = 0$ gives a
 900 strictly negative slope whenever the variance $\sigma_A^2(\ell)$ is not degenerate. Therefore there exists $\eta > 0$
 901 such that
 902

$$903 \quad \text{Var}(g_\alpha) \leq c^2 L^2 \left(\frac{1}{4} \overline{\sigma_A^2} - \eta \alpha + O(\alpha^2 \tau^2) \right) + C_\beta, \quad (50)$$

904 where $\overline{\sigma_A^2} = \mathbb{E}[\sigma_A^2(\ell)]$ and C_β collects the β -dependent but α -independent finite terms.
 905

906 The POER-specific truncation bias can be written as the deviation between the untruncated importance-
 907 weight estimator and the truncated one. Let $w = \frac{\pi_\theta}{\mu}$ and $X_\alpha = (A'_\alpha - \beta \log \pi_\theta) \nabla_\theta \log \pi_\theta$. The bias
 908 vector equals
 909

$$910 \quad b_{\text{clip}}(\alpha) = \mathbb{E}[(w - \rho) X_\alpha] = \mathbb{E}[(w - c)^+ X_\alpha], \quad (51)$$

911 so that $\|b_{\text{clip}}(\alpha)\| \leq \mathbb{E}[(w - c)^+ \|X_\alpha\|] \leq \mathbb{E}[(w - c)^+ (|A'_\alpha| + |\beta| \|\log \pi_\theta\|) L]$.
 912

913 Assumption 2 implies that the event $\{w > c\}$ is more likely at larger ℓ , whereas $|A'_\alpha|$ is reduced at
 914 larger ℓ because $s_\alpha(\ell)$ decreases with ℓ and the clipping of R_α further upper-bounds its magnitude.
 915

916 Consequently the mapping $\alpha \mapsto \|b_{\text{clip}}(\alpha)\|$ is nonincreasing for small α , and in particular
 917 $\|b_{\text{clip}}(\alpha)\| \leq \|b_{\text{clip}}(0)\|$. Since MSE_α contains $\|b_{\text{clip}}(\alpha)\|^2$, this term does not increase with α
 918 near zero.

918 Combining equation 47 and equation 50 and adding the nonincreasing truncation-bias square gives
 919

$$920 \text{MSE}_\alpha = \text{Var}(g_\alpha) + \|\mathbb{E}[g_\alpha] - \nabla_\theta J\|^2 \leq c^2 L^2 \left(\frac{1}{4} \overline{\sigma_A^2} - \eta \alpha + O(\alpha^2 \tau^2) \right) + \|b_{\text{clip}}(\alpha)\|^2 + O(\alpha^2 \tau^2). \quad (52)$$

922 Choosing $\alpha^* > 0$ sufficiently small so that the linear decrease $-\eta \alpha$ dominates the aggregated
 923 $O(\alpha^2 \tau^2)$ remainders ensures that $\text{MSE}_\alpha < \text{MSE}_0^{\text{POER}}$ for all $0 < \alpha \leq \alpha^*$ with $\alpha \tau \leq 1$.
 924

925 Since GRPO coincides with the on-policy case without any truncation channel, its α -
 926 dependence shares the same variance reduction mechanism but lacks the nonincreasing
 927 truncation-bias term $\|b_{\text{clip}}(\alpha)\|^2$; therefore the same choice of α also yields $\text{MSE}_\alpha^{\text{POER}} <$
 928 $\min\{\text{MSE}_0^{\text{GRPO}}, \inf_{\tilde{\alpha} > 0} \text{MSE}_{\tilde{\alpha}}^{\text{GRPO}}\}$ whenever $\|b_{\text{clip}}(0)\| > 0$, which holds generically under as-
 929 sumption (ii). This proves the stated improvement.
 930

931 **Remark.** The token-wise averaging in GRPO, $\frac{1}{|o|} \sum_{j=1}^{|o|}$, multiplies the effective per-sample weight
 932 by $|o|^{-1}$ and thus accentuates the negative covariance in equation 49, because $|o|^{-1}$ is also nonin-
 933 creasing in ℓ . The group baseline b_α used to define A'_α guarantees that the constant component of
 934 $s_\alpha(\ell)$ is removed, while the window condition $\alpha \tau \leq 1$ keeps $s_\alpha(\ell)$ within the near-linear regime
 935 where equation 43 is valid and the remainder terms are uniformly controlled.
 936

937 D USING A LARGE MODEL'S CACHE POOL TO GUIDE SMALL MODEL 938 TRAINING

941 We design the following experiment to explore whether introducing a more powerful model for
 942 question sampling during the cache pool initialization phase can influence the resulting cache policy,
 943 thereby allowing the original model to indirectly benefit from the distillation of the stronger model's
 944 reasoning capabilities.

945 Specifically, we use the cache pool initialized by deepseek-r1-qwen-7b as the initial cache pool for
 946 deepseek-r1-qwen-1.5b. Then, following the original experimental setup, we train for two epochs and
 947 evaluate the final performance. As shown in Table 6, when trained using the cache pool generated by
 948 the 7B model, the 1.5B model did not significantly improve performance.
 949

950 Table 6: The Performance of a 7B Model's Cache Pool on a 1.5B Model
 951

952 Model	AIME24	MATH500	AMC23	Minerva	OlyB	Avg
954 DeepSeek-R1-Qwen-1.5B	23.3	84.8	75.0	28.7	53.5	53.1

955 956 957 E COST OVERHEAD

958 In this section, we present the cost overhead of several additional open-source models with the same
 959 parameters, as well as that of the series of models based on our POER algorithm.
 960

963 Table 7: Comparison of data usage and computational costs with 1.5B models.
 964

	DeepScaleR-1.5B-Preview	Still-3-1.5B-Preview	POER
967 Base Model	DeepSeek-R1-Distill-Qwen-1.5B		
968 Hardware	8 × A100 80GB	1 × 8 A100 80GB	1 × 8 A100 80GB
969 Time	240h	150h	3h
970 Cost Est.	\$3629	\$2268	\$24

Table 8: Comparison of data usage and computational costs with 7B models.

	rStar-Math-7B(Guan et al., 2025)	Eurus-2-7B-PRIME
Base Model	Qwen2.5-Math-7B	
Hardware	10×8 H100 80GB, 15×4 A100 40GB	1×8 A100 80GB
Time	–	72h
Cost Est.	–	\$1088
	Qwen2.5-7B-SimpleRL(Zeng et al., 2025)	POER
Base Model	Qwen2.5-Math-7B	DeepSeek-R1-Distill-Qwen-1.5B
Hardware	4×6 A100 80GB	1×8 A100 80GB
Time	36h	7h
Cost Est.	\$1633	\$56

F MORE ANALYSIS

The impact of cache pool update strategies To investigate the impact of different cache pool update strategies on model performance, we set ϵ to 0, 0.1, 0.5, and 1 during training. In addition, we also evaluate the model with cache pool updates completely disabled. As shown in Table 9, the performance of the 1.5B model exhibits a trend of first improving and then declining as ϵ increases. The best performance is achieved when $\epsilon = 0.1$, with an average accuracy of 52.6%.

Table 9: The impact of ϵ on model zero-shot performance. In the table, no update denotes the case where the cache pool is not updated, which serves as a baseline for comparison.

ϵ	AIME25	AIME24	MATH500	AMC23	Minerva	OlyB	Avg
0	23.3	30.0	84.8	75.0	28.3	52.4	50.0
0.1	23.3	36.7	85.4	87.5	29.4	53.0	52.6
0.5	30.0	26.7	83.8	72.5	28.3	52.4	49.0
0.9	26.7	36.7	84.4	70.0	29.8	53.0	50.1
no update	26.7	30.0	82.8	75.0	29.4	51.2	49.2

G TIME OVERHEAD FOR CACHE POOL INITIALIZATION

This section reports whether POER can still achieve significant training acceleration and performance improvement under extreme conditions, such as when the number of epochs is only 1.

Table 10: Cache pool initialization time (minutes) for 1.5B and 7B models under different GPU types, dataset sizes, and GPU counts

1.5B Model					7B Model				
Dataset	GPU	1	4	8	Dataset	GPU	1	4	8
7k	H20	34.78	24.61	15.10	7k	H20	58.57	26.44	17.95
	A100	32.11	21.45	13.98		A100	55.43	22.56	16.19
70k	H20	347.91	249.14	160.87	70k	H20	582.95	261.28	179.13
	A100	327.19	214.78	135.89		A100	566.49	238.91	167.57

Table 10 shows the model initialization time for the 1.5B and 7B models under different GPU count configurations. Table 11 shows the training time for one epoch on an 8-card H20 machine and an 8-card A100 machine, including the computational overhead of cache initialization. As seen from the

1026 table, even in extreme cases with only a single epoch of training, POER can still provide significant
 1027 acceleration.
 1028

1029 Table 11: Training time comparison (in hours) of DeepSeek-R1-Qwen models on H20 and A100
 1030 GPUs.

Model	H20 (hours)	A100 (hours)
DeepSeek-R1-Qwen-1.5B		
POER	14.45	12.41
GRPO	40.98	37.35
DeepSeek-R1-Qwen-7B		
POER	39.64	37.38
GRPO	114.50	105.70

1031
 1032
 1033
 1034
 1035
 1036
 1037
 1038
 1039
 1040
 1041
 1042
 1043
 1044
 1045
 1046
 1047
 1048
 1049
 1050
 1051
 1052
 1053
 1054
 1055
 1056
 1057
 1058
 1059
 1060
 1061
 1062
 1063
 1064
 1065
 1066
 1067
 1068
 1069
 1070
 1071
 1072
 1073
 1074
 1075
 1076
 1077
 1078
 1079