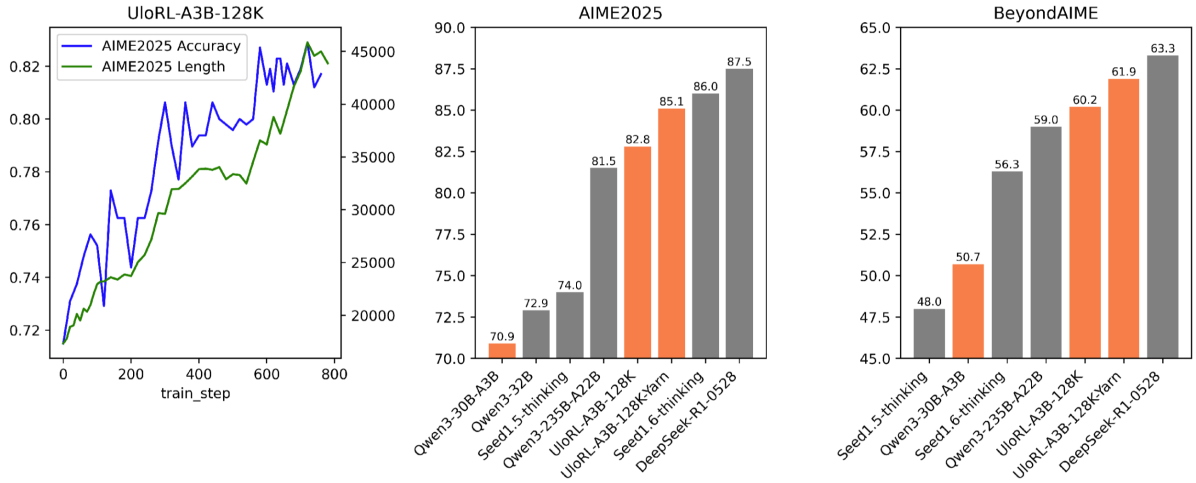
# UloRL:An Ultra-Long Output Reinforcement Learning Approach for Advancing Large Language Models' Reasoning Abilities

Dong Du*, Shulin Liu*, Tao Yang*, Shaohua Chen, Yang Li
Tencent Hunyuan Team
{dongdu,forestliu,rigorosyang,fafachen,youngyli}@tencent.com

*Contribute equally to this work.

## Abstract

Recent advances in large language models (LLMs) have highlighted the potential of reinforcement learning with verifiable rewards (RLVR) to enhance reasoning capabilities through extended output sequences. However, traditional RL frameworks face inefficiencies when handling ultra-long outputs due to long-tail sequence distributions and entropy collapse during training. To address these challenges, we propose an **U**ltra-**L**ong **O**utput **R**einforcement **L**earning (UloRL) approach for advancing large language models' reasoning abilities. Specifically, we divide ultra long output decoding into short segments, enabling efficient training by mitigating delays caused by long-tail samples. Additionally, we introduce dynamic masking of well-**M**astered **P**ositive **T**okens (MPTs) to prevent entropy collapse. Experimental results demonstrate the effectiveness of our approach. On the Qwen3-30B-A3B model, RL with segment rollout achieved 2.06x increase in training speed, while RL training with 128k-token outputs improves the model's performance on AIME2025 from 70.9% to 85.1% and on BeyondAIME from 50.7% to 61.9%, even surpassing Qwen3-235B-A22B with remarkable gains. These findings underscore the potential of our methods to advance the reasoning capabilities of LLMs with ultra-long sequence generation. We will release our code and model for further use by the community[1].




## 1 Introduction

Recent advances in large language models (LLMs) have significantly enhanced their reasoning capabilities across challenging domains such as mathematics and programming. This progress has been driven by state-of-the-art models like OpenAI o1 (Jaech et al., 2024), DeepSeek R1 (Guo et al., 2025), and other models that employ sophisticated test-time scaling strategies. A key breakthrough in this evolution is the adoption of reinforcement learning with verifiable rewards (RLVR). Unlike traditional reward shaping methods that focus on intermediate reasoning steps, this approach leverage rule-based verification systems to directly assess final answers, creating a powerful learning signal that guides the model toward generating correct and well-justified solutions through extended reasoning chains.

[1] https://github.com/liushulinle/ULORL

One of the key observations in recent advancements is that increasing the output length of models can significantly enhance their reasoning capabilities. However, traditional RL frameworks are not well-suited for such scenarios. In these frameworks, all samples in a batch must complete their decoding before training can proceed, leading to inefficiencies when dealing with long-tail distributions of sequence lengths. This inefficiency becomes particularly problematic when dealing with ultra-long outputs, such as outputs of up to 128k tokens, where a small fraction of long-tail samples can bottleneck the entire training process.

K1.5 (Team et al., 2025) proposed partial rollouts to address the aforementioned challenge. However, due to the lack of detailed descriptions of training strategies for segments from various models, as well as the absence of the setting of hyperparameters, it is challenging to reproduce their method. Similar with K1.5, we propose segment rollout which divides the decoding process into multiple stages. By decoding only a much shorter segment at each step, our method allows samples that have completed decoding to enter the experience pool for training immediately, while unfinished samples continue decoding in subsequent iterations. This approach not only accelerates training by avoiding unnecessary delays caused by long-tail samples but also ensures efficient utilization of computational resources. Furthermore, we introduce Segment-Aware Importance Sampling (SAIS) and Pesudo On-Policy Importance Sampling (POIS) to adapt the importance sampling mechanism to the segment rollout setting, ensuring accurate and stable training dynamics. We will release our code for further use by the community.

Another critical challenge in RL training is the phenomenon of entropy collapse (Cheng et al., 2025; He et al., 2025; Yu et al., 2025; Zhu et al., 2025), where the model's diversity diminishes prematurely, leading to suboptimal performances. Existing research on addressing this issue can be broadly categorized into two approaches. The first approach involves directly incorporating an entropy loss term into the overall loss function, treating entropy as an additional optimization objective for the model (Guo et al., 2025; He et al., 2025; Wu et al., 2025; Cheng et al., 2025). However, since the goal of maintaining entropy is not fully aligned with the goal of improving reasoning abilities, this method may potentially hurt the model's performance ceiling. The second approach focuses on adjusting the samples or tokens involved in training (Yu et al., 2025; Zhu et al., 2025; Wang et al., 2025). For instance, DAPO (Yu et al., 2025) proposed increasing the clipping threshold to allow tokens with greater divergence from the current policy distribution to participate in training. However, this method is only effective in off-policy training, as on-policy training does not include a clipping mechanism. W-Reinforce (Zhu et al., 2025) proposed addressing the issue of entropy collapse by reducing the weight of positive samples duiring training. However, if the generation probabilities of certain important tokens within positive samples are inherently low, reducing the training weight in such cases may slow down the model's learning process and could even hurt the final performance.

In this work, we argue that the entropy collapse issue arises when the model overfits to well-**M**astered **P**ositive **T**okens (MPTs), i.e., tokens that the model already predicts with high confidence. To mitigate this, we introduce the **D**ynamic **M**asking of welll-**M**astered **P**ositive **T**okens (DMMPTs) strategy, which adaptively controls the training of such tokens based on the model's current entropy. Specifically, if the model's entropy falls below a predefined threshold, the MPTs are masked and excluded from the training process. Otherwise, all tokens are included in the training. The proposed DMMPTs neither introduces additional optimization objectives nor relies on importance sampling, thereby avoiding the limitations associated with the aforementioned approaches. Experiments on Qwen3-4B,Qwen3-8B and Qwen3-30B-A3B(Yang et al., 2025) illustrate that DMMPTs enables the model to maintain entropy stable during the training process.

Furthermore, we introduce a generative verifier model (Zhang et al., 2024) to enhance the accuracy of reward computation in RL training. Unlike traditional rule-based methods (Yu et al., 2025; Luo et al., 2025), which are prone to misjudgments in complex scenarios, our verifier model leverages generative capabilities to determine the equivalence of predicted and reference answers. Furthermore, to ensure the quality of the reward signal, we also emphasize the importance of data cleaning and transformation, including filtering noisy data, removing questions containing multiple sub-questions, standardizing problem formats and simplifying reference answer.

We conducted a series of experiments to validate the effectiveness of the proposed method. We performed RL training with an output length of 128k on the Qwen3-30B-A3B model. After training, the model's performance on AIME2025 improved from 70.9% to 85.1%, and on BeyondAIME(Seed et al., 2025), it improved from 50.7% to 61.9%, even surpassing the Qwen3-235B-A22B model (AIME2025: 81.5%, BeyondAIME: 59.0%) with remarkable gains.

## 2 Preliminary

### 2.1 PPO

PPO (Schulman et al., 2017) introduces a clipped surrogate objective for policy optimization. By constraining the policy updates within a proximal region of the previous policy using clip operations, PPO stabilizes training and improves sample efficiency. Specifically, PPO updates the policy parameters $\theta$ by maximizing the following objective:

$$\mathcal{J}_{\text{PPO}}(\theta) = \mathbb{E}_{(q,a)\sim\mathcal{D},o_{\leq t}\sim\pi_{\theta_{\text{old}}}(\cdot|q)}\left[\min\left(\frac{\pi_\theta(o_t \mid q,o_{<t})}{\pi_{\theta_{\text{old}}}(o_t \mid q,o_{<t})}\hat{A}_t,\ \text{clip}\Big(\frac{\pi_\theta(o_t \mid q,o_{<t})}{\pi_{\theta_{\text{old}}}(o_t \mid q,o_{<t})},1-\varepsilon,1+\varepsilon\Big)\hat{A}_t\right)\right] \tag{1}$$

where $(q,a)$ denotes a question-answer pair from the data distribution $\mathcal{D}$, $\varepsilon$ represents the clipping threshold that bounds policy updates, and $\hat{A}_t$ is the estimated advantage at step $t$. The advantage estimator $\hat{A}_t$ is computed using Generalized Advantage Estimation (GAE) (Schulman et al., 2015):

$$\hat{A}_t^{\text{GAE}(\gamma,\lambda)} = \sum_{l=0}^{\infty}(\gamma\lambda)^l\delta_{t+l} \tag{2}$$

with the temporal difference term $\delta_l$ given by:

$$\delta_l = R_l + \gamma V(s_{l+1}) - V(s_l), \quad 0 \leq \gamma,\lambda \leq 1 \tag{3}$$

where $R_l$ denotes the reward, $V$ represents the value function, $\gamma$ is the discount factor, and $\lambda$ controls bias-variance tradeoff in advantage estimation.

### 2.2 GRPO

GRPO (Shao et al., 2024) presents a group-relative advantage estimation alternative to PPO that eliminates dependency on value functions. For any question-answer pair $(q,a)$, the behavioral policy $\pi_{\theta_{\text{old}}}$ generates a group of $G$ distinct responses $\{o_i\}_{i=1}^G$. The advantage for the $i$-th response $\hat{\mathcal{A}}_{i,t}$ is derived through group-level normalization:

$$\hat{\mathcal{A}}_{i,t} = \frac{r_i - \text{mean}(\{\mathcal{R}_i\}_{i=1}^G)}{\text{std}(\{\mathcal{R}_i\}_{i=1}^G)} \tag{4}$$

GRPO also adopts a clipped surrogate function, together with explicit KL regularization against a reference policy:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{(q,a)\sim\mathcal{D},\{o_i\}_{i=1}^G\sim\pi_{\theta_{\text{old}}}(\cdot|q)} \left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\left(\min\Big(r_{i,t}(\theta)\hat{\mathcal{A}}_{i,t},\text{clip}\big(r_{i,t}(\theta),1-\varepsilon,1+\varepsilon\big)\hat{\mathcal{A}}_{i,t}\Big) - \beta D_{\text{KL}}\left(\pi_\theta \parallel \pi_{\text{ref}}\right)\right)\right] \tag{5}$$

where the importance ratio $r_{i,t}(\theta)$ measures policy update magnitude:

$$r_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t} \mid q,o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t} \mid q,o_{i,<t})} \tag{6}$$

### 2.3 DAPO

DAPO (Yu et al., 2025) proposed a series of effective modifications based on GRPO for large scale RL training, including dynamic sampling, token-level gradient loss, clip higher, overlong reward shaping and removing KL divergence. The final objective is as follows:

$$\begin{aligned}\mathcal{J}_{\text{DAPO}}(\theta) = \mathbb{E}_{(q,a)\sim\mathcal{D},\{o_i\}_{i=1}^G\sim\pi_{\theta_{\text{old}}}(\cdot|q)} &\left[\frac{1}{\sum_{i=1}^G|o_i|}\sum_{i=1}^{G}\sum_{t=1}^{|o_i|}\left(\min\Big(r_{i,t}(\theta)\hat{\mathcal{A}}_{i,t},\text{clip}\big(r_{i,t}(\theta),1-\varepsilon_{low},1+\varepsilon_{high}\big)\hat{\mathcal{A}}_{i,t}\Big)\right)\right] \\ &\text{s.t.} \quad 0 < \left|\{o_i \mid \text{is\_equivalent}(a,o_i)\}\right| < G\end{aligned} \tag{7}$$

Following DAPO, we adopt dynamic sampling, token-level gradient loss and removing KL divergence in our approach.

## 3 UloRL

We propose an **U**ltra-**L**ong **O**utput **R**einforcement **L**earning (UloRL) approach for advancing large language models' reasoning abilities. In this section, we will introduce the key techniques associated with UloRL. Our implementation is built on the verl framework[2] (Sheng et al., 2024).

### 3.1 Segment Rollouts with Pseudo On-policy Importance Sampling

Increasing the output length of models can enhance their reasoning capabilities (Team et al., 2025). However, in scenarios involving ultra-long outputs, such as sequences with a length of 128k, the long-tail effect becomes a significant bottleneck. For example, within a batch, 80% of the samples may have lengths within 64k, but all samples must wait for the longest 128k output to complete before next decoding iteration. This greatly reduces training efficiency and resource utilization.

To address this challenge, we divide the decoding of an ultra-long output into multiple stages. In each stage, only a segment of the sequence is decoded. Samples that complete decoding are immediately added to the experience replay buffer for training, while incomplete samples are carried over to the next iteration, where the results from the previous stage are concatenated and decoding continues.

**Algorithm 1** RL Training with Segment Rollouts

```
1:  Initialize:
2:      unfinished_pool ← {}                                   ▷ Samples that unfinished decoding
3:      experience_pool ← {}                                   ▷ Samples ready for training
4:      global_max_seq_len ← 128K          ▷ Assume the global maximum decoding length is 128K
5:      max_segment_count ← 8              ▷ Assume the sequence is divided into 8 segments
6:      each_segment_length ← global_max_seq_len / max_segment_count
7:  for step ∈ {1, 2, ..., total_steps} do
8:          batch ← rollout({unfinished_pool, prompts}, max_len = each_segment_length)   ▷ Rollout
9:          unfinished_pool ← update_unfinished_pool(batch, unfinished_pool)
10:         experience_pool ← update_experience_pool(batch, experience_pool)
11:         update_model(experience_pool)                                                ▷ Update model
12: end for
```

#### 3.1.1 Segment Rollouts

Algorithm 1 illustrates the RL training process with segment rollouts. As illustrated in line 8, the input for each decoding step comes from two sources: (1) unfinished samples from the previous rollout step, and (2) new prompts from the RL dataset. The decoding process terminates under one of the following three conditions:

- **End-of-sequence (EOS) token is encountered** In this case, the sample is considered complete and is added to the experience pool for training.
- **Segment reaches the maximum segment length but not the global maximum length** This indicates that the sample is not yet fully decoded and will be added to the unfinished pool for continuation in the next step.
- **Global maximum length is reached** In this case, the sequence is truncated and added to the experience pool for training.

Assuming the global maximum length is set to 128k, and the segment count is set to 8. Then the model only needs to decode 128k/8=16k at a time to perform an update. This avoids the inefficiency caused by waiting for a few ultra long samples to complete decoding, significantly improving training efficiency. To evaluate the impact of segment rollout on training efficiency, we conducted experiments on Qwen3-30B-A3B with 64k output and the results are illustrated in Table 1. From the table we observe that training with two segments and four segments can improve the training speed by 1.6x and 2.06x, respectively.

#### 3.1.2 Training

In the original GRPO, each sample in the experience pool is generated by a single model. However, under the segment rollout setting, as illustrated in Figure 1(a), a single sample may consist of segments generated by multiple models. Consequently, the term $\pi_{\theta_{\text{old}}}$ in Equation 6 needs to be adjusted. To

[2] https://github.com/volcengine/verl

| segment count | time cost per step | speed |
|---|---|---|
| 1 | 1240s | 1.0x |
| 2 | 774s | 1.6x |
| 4 | 601s | 2.06x |

Table 1: The impact of segment count on training speed.

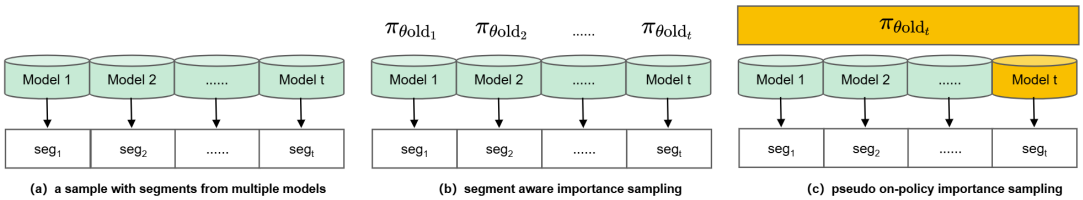



Figure 1: Illustration of a sample with segments from multiple models.

address this, we propose two methods for computing importance sampling value under segment rollout setting.

**Segment Aware Importance Sampling (SAIS)** As illustrated in Figure 1 (b), different segments are generated by different models, and therefore their corresponding $\pi_{\theta_{\text{old}}}$ vary. We denote the segment generated by the model at time $t$ as $\text{seg}_t$, then a sample $s$ can be represented as $s = [\text{seg}_1; \text{seg}_2; \ldots; \text{seg}_t]$. For the $i$-th token in $s$, the importance sampling value can be computed using Equation 8, where $f(i)$ is the function to map the $i$-th token to its segment id.

$$r_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t} \mid q, o_{i,<t})}{\pi_{\theta_{\text{old}_{f(i)}}}(o_{i,t} \mid q, o_{i,<|s_i|})} \tag{8}$$

**Pseudo On-policy Importance Sampling (POIS)** Recent work (He et al., 2025; Hao et al., 2025) demonstrated that on-policy training exhibits more stable entropy and better performance than off-policy training. This is primarily because, in off-policy training, tokens that deviate significantly from the current policy are clipped by the clipping operation in Equation 7, which reduces the diversity of the model. In contrast, in on-policy training, all tokens are generated by the current model, therefore $\pi_\theta = \pi_{\theta_{\text{old}}}$. As a result, the importance sampling weight for all tokens is equal to 1, ensuring that no tokens are clipped. This allows the model to observe more diverse data during training. To leverage this advantage of on-policy training, we modify the importance sampling item to enable on-policy training.

As shown in Figure 1 (c), at time step $t$, the last segment $\text{seg}_t$ is on-policy data, while segments generated from time steps 1 to $t-1$ are off-policy data. To achieve on-policy training, we simply replace the $\pi_{\theta_{\text{old}}}$ of all time steps with the $\pi_{\theta_{\text{old}_t}}$. Under this modification, the importance sampling weight for all tokens becomes 1. In this approach, samples with only one segment are true on-policy samples. For samples with more than one segment, the last segment is true on-policy, while other segments are pseudo on-policy.

**Experimental Results** To evaluate the effectiveness of the aforementioned methods, we conducted experiments on the Qwen3-30B-A3B model. The experimental settings are as follows:

- **TOIS** True On-Policy Importance Sampling, with segment count $= 1$
- **SAIS** Segment-Aware Importance Sampling, with segment count $= 4$
- **POIS** Pseudo On-Policy Importance Sampling, with segment count $= 4$

Figure 2 illustrates the dynamics of entropy and accuracy for output lengths of 4k, 32k and 64k. Under the 4k output setting, it is surprising to observe that the entropy and evaluation curves of POIS and TOIS nearly overlap, and both outperform SAIS. Furthermore, the POIS also outperform SAIS with both 32k and 64k output. The effectiveness of POIS can potentially be attributed to the fact that the last segment of each sample is true on-policy data, which may mitigate the negative impact of training on pseudo on-policy data to some extent. Moreover, we also applied the Clip-higher strategy with $\epsilon_{high} = 0.28$ as suggested in Yu et al. (2025) on SAIS. However, we observed entropy explosion, a phenomenon consistent with the findings of He et al. (2025). **Based on these observations, we adopt POIS for subsequent experiments**.

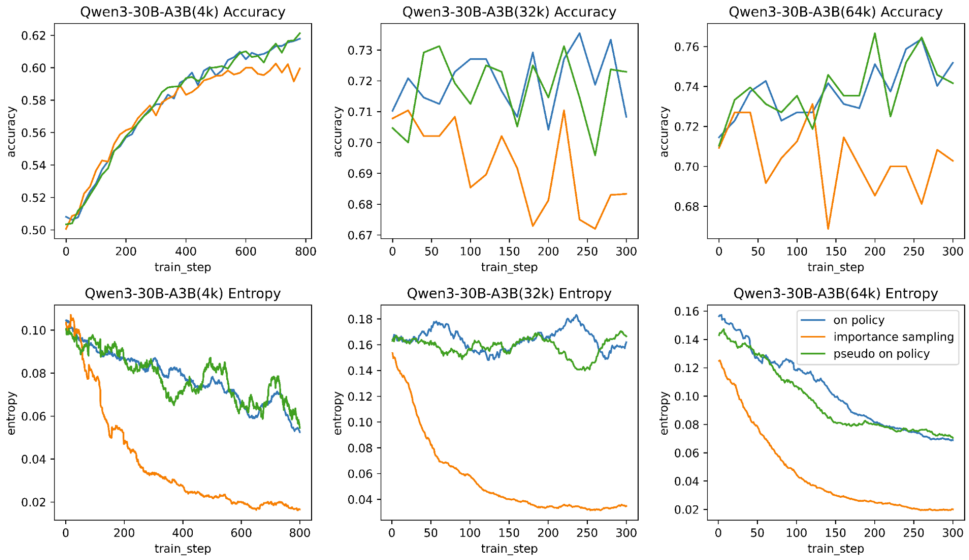



Figure 2: Training Dynamics of Different Importance Sampling Approaches.

### 3.2 Avoiding Entropy Collapse: Do Not Train Well-mastered Positive Tokens

#### 3.2.1 Well-mastered Positive Tokens Result in Entropy Collapse

Zhu et al. (2025) pointed out that training on positive samples is the primary cause of entropy reduction. We argue that the true reason is the overtraining of tokens that the model has already mastered within positive samples. Here, positive samples refer to those with a reward of 1, and "already mastered tokens" are defined as tokens for which the model's predicted probability exceeds a high threshold $\tau$. We refer to such tokens as well-**M**astered **P**ositive **T**okens (MPTs).

$$\text{MPTs} = \bigcup_{k=1}^{N} \bigcup_{i=1}^{L_k} \{t_i \in s_k\}, \text{where } \; p(t_i) \geq \tau, r(s_k) = 1 \tag{9}$$

As shown in the left part of Figure 3, updating the MPTs further increases its predicted probability. This, in turn, sharpens the distribution, making it more concentrated around the chosen token. As a result, the entropy of the model decreases. In contrast, as illustrated in the right part of Figure 3, updating non-MPTs does not necessarily lead to a decrease in entropy.

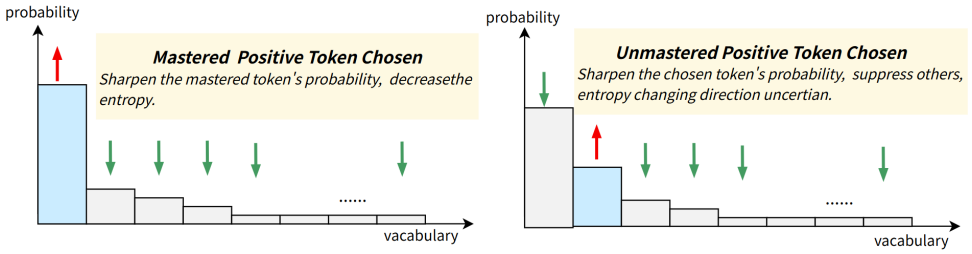



Figure 3: Entropy changing direction of updating MPTs (left) and non-MPTs (right), where blue block denotes the chosen token.

We conducted experiments to validate the above hypothesis. The experimental setup is as follows:

- **Baseline** All tokens are included in the training process.
- **Masking MPTs**: Only tokens excludes MPTs are included in the training process, where the threshold $\tau$ in Equation 9 is set to 0.99.

The experiments were performed on the Qwen3-4B, Qwen3-8B, and Qwen3-30B-A3B. The output length is set to 128k, which is divided into 8 segments. Figure 4 illustrates the entropy dynamics. As shown in the results, for all three models, the entropy of the baseline gradually decreases as training progresses. However, when MPTs are excluded from the training process, the entropy of the model increases over

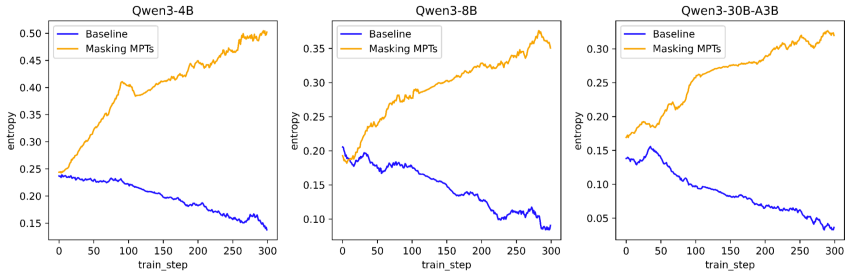



Figure 4: Training dynamics of RL with masking MPTs.

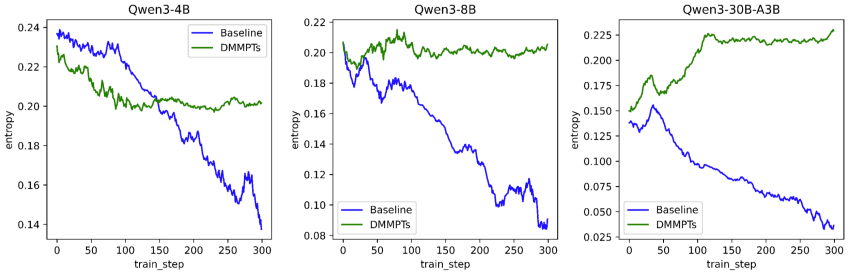



Figure 5: The entropy dynamics of DMMPTs.

time. This observation supports our hypothesis that the overtraining of MPTs is a key factor contributing to the reduction in entropy. By excluding MPTs, the model maintains a more diverse output distribution.

### 3.2.2 Dynamic Mask Well-mastered Positive Tokens

As shown in Figure 4, simply excluding MPTs from training leads to a continuous increase in entropy, which is also detrimental to the stability of training. Ideally, the model's entropy during training should be maintained around an appropriate entropy to ensure stable and effective learning. To achieve this, we propose a method called **D**ynamic **M**asking of MPTs (DMMPTs). Specifically, we introduce a new hyperparameter $\sigma$ to represent the target entropy. During training, MPTs are masked only when the current entropy falls below the target entropy $\sigma$:

$$
\begin{aligned}
\mathcal{J}(\theta) =& \mathbb{E}_{(q,a)\sim\mathcal{D},\, \{o_i\}_{i=1}^{G}\sim\pi_{\text{old}}(\cdot \mid q)} \\
& \left[ \frac{1}{\sum_{i=1}^{G}|o_i|} \sum_{i=1}^{G}\sum_{t=1}^{|o_i|} [1 - \mathbb{I}_{\text{msk}}] \min\left(r_{i,t}(\theta)\hat{A}_{i,t}, \text{clip}\left(r_{i,t}(\theta), 1-\varepsilon, 1+\varepsilon\right)\hat{A}_{i,t}\right) \right] \\
& \mathbb{I}_{\text{msk}}^{(i,t)} = \begin{cases} 1 & \bar{\mathrm{H}}_i < \sigma \text{ and } o_i^t \in \text{MPTs} \\ 0 & \text{otherwise} \end{cases} \\
& \bar{\mathrm{H}}_i = \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \sum_{j=1}^{|\mathcal{V}|} p_t^j \log p_t^j
\end{aligned}
\tag{10}
$$

This dynamic adjustment ensures that the model maintains a balanced entropy level, avoiding both excessive sharpness and excessive randomness in the output distribution. Note that, since MPTs have already been well-mastered by the model, it is expected that our approach will not have any negative impact on the model's performance.

To verify whether the proposed method achieves the desired objectives, we conducted experiments on the Qwen3-4B, Qwen3-8B, and Qwen3-30B-A3B. The experimental results are presented in Figure 5. From the results, it can be observed that after incorporating the DMMPTs, the entropy of all three models, regardless of their size, remains stable around the predefined target range. This demonstrates the effectiveness of the proposed method in maintaining a balanced entropy level during training, thereby ensuring stable and robust learning.

### 3.3 Generative Verifier Model

Reward models based on outcomes have been proven to be highly effective for reinforcement learning (RL) in reasoning tasks (Guo et al., 2025; Yu et al., 2025). Following DAPO, we directly use the final accuracy of a verifiable task as the outcome reward. The reward is computed using the following rule:

$$R(\widehat{y}, y) = \begin{cases} 1, & \text{is\_equivalent}(\widehat{y}, y) \\ 0, & \text{otherwise.} \end{cases}$$

Unlike DAPO, our reward values are designed to be $\{0, 1\}$ instead of $\{-1, 1\}$. The advantage of this design is that the average reward across the dataset directly corresponds to its accuracy. Furthermore, under the GRPO framework, when both positive and negative samples exist within a group, the advantage for samples with a reward of 1 is always greater than 0, resulting in a positive gradient direction. Conversely, the advantage for samples with a reward of 0 is always less than 0, leading to a negative gradient direction. This behavior aligns well with the optimization objective.

Additionally, determining whether two answers $(\hat{y}, y)$ are equivalent is not a trivial task, as rule-based methods are prone to misjudgments. For example, pairs such as $(27\text{cm}, 0.27\text{m})$ or $(1/2, \text{one half})$ can easily be misclassified as non-equivalent. To address this issue, we trained a generative model to evaluate whether two given answers are semantically equivalent. This approach ensures a more robust and accurate equivalence judgment.

### 3.4 Data Cleaning and Transformation

To ensure the accuracy of the rewards generated by the Verifier Model, we applied a series of preprocessing steps to the RL training data, addressing both the question and reference answer dimensions:

**Question Dimension**

- **Deleting Problems of Multiple Sub-questions** We removed instances of multiple sub-questions within a single problem to avoid pseudo-negative reward caused by incomplete summaries of answers to sub-questions.
- **Converting Special Questions to Short-answer Format** We convert multiple-choice, proof-based and true/false questions to short-answer format to prevents the model from simply guessing the correct answer without understanding the problem.
- **Deleting Overly Simple Questions** To enhance the efficiency of reinforcement learning (RL) training, we utilized the Qwen3-30B-A3B model to perform inference on all data 8 times. Questions that were answered correctly in all 8 attempts were deemed overly simple and subsequently removed from the training dataset.

**Answer Dimension**

- **Extracting Short Answers For Reference Answers** This reduces the complexity of the verifier model's judgment task, improving its accuracy.
- **Deleting Questions With Excessively Long Reference Answers** For example, problems with matrix-based answers were excluded to avoid unnecessary complexity for the verifier model.
- **Deleting Questions With Incorrect Reference Answers** To identify such cases, we used multiple SOTA models to predict the same question. If the outputs of multiple SOTA models were consistent but differed from the reference answer, the reference answer was deemed incorrect, and the corresponding data was removed.

### 3.5 Overlong Punishment

For the truncated samples, DAPO proposes two handling strategies: overlong filtering and soft overlong punishment. However, our experiments reveal that overlong filtering leads to a rapid increase in the output length, which is undesirable. Additionally, we observed that the performance of soft overlong punishment is comparable to directly treating overlong samples as incorrect answers. Therefore, in this work we simply treat overlong samples as incorrect answers and assigning them a reward of 0.

## 4 Experiments

### 4.1 Training Details

In this section, we conducted experiments on Qwen3-30B-A3B to verify the effectiveness of UloRL. Experimental settings are represented as follows.

| Model | AIME-2025 | BeyondAIME | AVG |
|---|---|---|---|
| DeepSeek-R1-0528 | 87.5 | 63.3* | 75.4 |
| Seed-1.6-thinking | 86 | 56.3 | 71.2 |
| Qwen3-235B-A22B | 81.5 | 59.0* | 70.3 |
| Qwen3-30B-A3B | 70.9 | 50.7* | 60.8 |
| UloRL-A3B-128k | 82.8 | 60.2 | 71.5 |
| UloRL-A3B-w/o-DMMPTs | 78.6 | 57.1 | 67.9 |
| UloRL-A3B-128k-Yarn | 85.1 | 61.9 | 73.5 |

Table 2: The overall results of the proposed UloRL trained on Qwen3-30B-A3B. Metrics marked with an * are results from our evaluation, while the others are from official reports.

**Hyperparameter Settings**

For optimization, we utilize the AdamW optimizer (Zhang et al., 2018) with a constant learning rate of $1 \times 10^{-6}$. During rollout, the prompt batch size is set to 128, and we sample 8 responses for each prompt. The sampling temperature is set to 0.85, with top_p $= 1.0$ and top_k $= -1$. The maximum response length is set to 128k tokens, divided into a maximum of 8 segments, with each segment containing 16k tokens.

For training, the mini-batch size is set to 1024, meaning one gradient update is performed for each rollout step. The probability threshold for MPTs, $\tau$, is set to 0.99, and the target entropy, $\sigma$, is set to 0.2.

**Evaluation Setup**

For evaluation, we use the AIME-2025 and BeyondAIME(Yu et al., 2025) datasets as benchmarks. Each evaluation set is repeated 32 times, and we report the average score (avg@32) to ensure result stability. The inference hyperparameters are set to a sampling temperature of 0.85, topp of 0.95 and topk of 20.

### 4.2 Overall Results

Table 2 presents the evaluation results. The first group includes the performance metrics of SOTA models. The second group consists of three models tuned using different RL algorithms based on the Qwen3-30B-A3B model:

- **UloRL-A3B-128k** This model is trained using the full UloRL algorithm, with training hyperparameters detailed in Section 4.1.
- **UloRL-A3B-w/o-DMMPTs**: This is a variant of UloRL excluding the DMMPTs component. The training hyperparameters are identical to those used for the full UloRL method.
- **UloRL-A3B-Yarn-140k** Following An et al. (2025), we employ Yarn (Peng et al., 2023) to further extend the output length to 140k.

From Table 2, we can make the following observations. (1) **UloRL-A3B-128k** outperforms **Qwen3-30B-A3B** with significant gains, even surpasses that of **Qwen3-235B-A22B**. These results confirm the effectiveness of the proposed **UloRL** algorithm , highlighting its ability to achieve state-of-the-art performance with a more efficient and scalable approach. (2) A comparison between **UloRL-A3B-w/o-DMMPTs** and **UloRL-A3B-128k** reveals that removing the DMMPTs strategy results in a significant degradation in model performance. This validates the efficacy of the proposed DMMPTs method. (3) By extending the output length to 140k using Yarn, the model achieved further improvements. This indicates that continuously expanding the length can further enhance the model's reasoning ability.

### 4.3 Effect of Output Length on Model Performance

In this subsection, we investigate the effect of output length on model performance. We compared output lengths of 32k, 64k, 96k, and 128k. Except for the length and segment parameters, all other hyperparameters were kept consistent with those described in Section 4.1. For the 32k experiment, the segment count was set to 1. For the 64k experiment, the output was divided into 4 segments, while for 96k and 128k, the output was divided into 8 segments.

The experimental results are shown in Table 3. From the table, it can be observed that the performance improvement achieved with 32k reinforcement learning is minimal. This is primarily because Qwen3-30B-A3B is already a highly strong 32k-output model, and without significant changes to the output

| Model | AIME-2025 | BeyondAIME | AVG |
|---|---|---|---|
| Qwen3-30B-A3B | 70.9 | 50.7 | 60.8 |
| UloRL-A3B-32k | 73.5 | 52.3 | 62.9 |
| UloRL-A3B-64k | 79.9 | 58.5 | 69.2 |
| UloRL-A3B-96k | 81.6 | 59.4 | 70.5 |
| **UloRL-A3B-128k** | **82.8** | **60.2** | **71.5** |

Table 3: The performances of models training with different output length.

length, it is challenging to further enhance its reasoning capabilities. However, when the output length is extended to 64k, the model's reasoning ability improves significantly.

Overall, the results show a clear trend: the longer the output length, the better the model's reasoning performance. This demonstrates that extending the output length is an effective approach to improving the reasoning capabilities of large language models.

## 5 Conclusions

In this work, we proposed UloRL, an ultra-long output reinforcement learning algorithm for advancing Large Language Models' reasoning abilities. We first introduce the segment rollout to mitigate the inefficiencies caused by long-tail sequence distributions, enabling faster and more resource-efficient RL training. By incorporating Segment-Aware Importance Sampling (SAIS) and Pseudo On-Policy Importance Sampling (POIS), we ensure stable and accurate training dynamics in the segmented rollout setting. Furthermore, to tackle the issue of entropy collapse, we proposed the Dynamic Masking of well-Mastered Positive Tokens (DMMPTs) strategy, which adaptively balances exploration and exploitation without introducing additional optimization objectives or relying on importance sampling. Experimental results demonstrate the effectiveness of our methods.

## References


Chenxin An, Zhihui Xie, Xiaonan Li, Lei Li, Jun Zhang, Shansan Gong, Ming Zhong, Jingjing Xu, Xipeng Qiu, Mingxuan Wang, and Lingpeng Kong. Polaris: A post-training recipe for scaling reinforcement learning on advanced reasoning models, 2025. URL https://hkunlp.github.io/blog/2025/Polaris.

Daixuan Cheng, Shaohan Huang, Xuekai Zhu, Bo Dai, Wayne Xin Zhao, Zhenliang Zhang, and Furu Wei. Reasoning with exploration: An entropy perspective. *arXiv preprint arXiv:2506.14758*, 2025.

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

Yaru Hao, Li Dong, Xun Wu, Shaohan Huang, Zewen Chi, and Furu Wei. On-policy rl with optimal reward baseline. *arXiv preprint arXiv:2505.23585*, 2025.

Jujie He, Jiacai Liu, Chris Yuhao Liu, Rui Yan, Chaojie Wang, Peng Cheng, Xiaoyu Zhang, Fuxiang Zhang, Jiacheng Xu, Wei Shen, et al. Skywork open reasoner 1 technical report. *arXiv preprint arXiv:2505.22312*, 2025.

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

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Erran Li, Raluca Ada Popa, and Ion Stoica. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl. https://pretty-radio-b75.notion.site/DeepScaleR-Surpassing-O1-Preview-with-a-1-5B-Model-by-Scaling-RL-19681902c1468005bed8ca303013a4e2, 2025. Notion Blog.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. Yarn: Efficient context window extension of large language models. *arXiv preprint arXiv:2309.00071*, 2023.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

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

ByteDance Seed, Jiaze Chen, Tiantian Fan, Xin Liu, Lingjun Liu, Zhiqi Lin, Mingxuan Wang, Chengyi Wang, Xiangpeng Wei, Wenyuan Xu, et al. Seed1. 5-thinking: Advancing superb reasoning models with reinforcement learning. *arXiv preprint arXiv:2504.13914*, 2025.

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

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

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

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, et al. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *arXiv preprint arXiv:2506.01939*, 2025.

Lixin Wu, Na Cai, Qiao Cheng, Jiachen Wang, and Yitao Duan. Confucius3-math: A lightweight high-performance reasoning llm for chinese k-12 mathematics learning. *arXiv preprint arXiv:2506.18330*, 2025.

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

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. International conference on learning representations. In *International Conference on Learning Representations*, volume 2, 2018.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction. *arXiv preprint arXiv:2408.15240*, 2024.

Xinyu Zhu, Mengzhou Xia, Zhepei Wei, Wei-Lin Chen, Danqi Chen, and Yu Meng. The surprising effectiveness of negative reinforcement in llm reasoning. *arXiv preprint arXiv:2506.01347*, 2025.