
Supplementary Materials for Open-Reasoner-Zero

1 In this Supplementary Material, we provide more elaboration on the implementation details, experi-
2 ment results, and qualitative results. Specifically, we present more evaluation results in Section A,
3 thorough implementations of the model training in Section B, and additional analyses and experi-
4 ments in Section C and D. These materials offer deeper insights into our methodology, experimental
5 validation, and qualitative findings that support the conclusions presented in the main text.

6 A More Evaluation Results

7 In this section, we provide detailed results from evaluating ORZ models of varying parameter counts
8 (0.5B, 1.5B, 7B, and 32B) across multiple reasoning-oriented benchmarks. Specifically, we report
9 performance on AIME 2024, AIME 2025, MATH500, and GPQA Diamond. The results (see
10 Table 1) clearly demonstrate consistent improvements in reasoning ability with increased model
11 size, underscoring the strong scaling properties of our minimalist RL setup. We release these
12 comprehensive evaluation results as a reference to facilitate further research and reproducibility. We
also provide the training performance curves for ORZ- $\{0.5B, 1.5B\}$ in Figure 1.

Table 1: Reasoning-oriented benchmark performance across Open-Reasoner-Zero model sizes.

Model	AIME 2024	AIME 2025	MATH500	GPQA Dia.
ORZ-0.5B	1.0	0.2	31.0	12.1
ORZ-1.5B	3.5	1.0	58.0	16.8
ORZ-7B	17.9	15.6	81.4	36.6
ORZ-32B	48.1	36.0	92.2	55.5

13

14 B Detailed Setting for Training

15 We initialize both our policy and critic networks with Qwen-2.5 base models (7B and 32B variants),
16 where value head is randomly initialized from $\mathcal{U}(-\sqrt{5}, \sqrt{5})$ with no bias term. The policy and
17 critic do not share weights during training. For both policy and critic networks, we employ AdamW
18 optimizer with $\beta = [0.9, 0.95]$ without weight decay. The learning rates are set to 1×10^{-6} and
19 5×10^{-6} for the policy and critic networks, respectively. The learning rate schedulers are both
20 constant learning rate with linear warm-up of 50 optimizer steps. We employ sample packing during
21 training. Prompt for ORZ model training and evaluation are provided in Table 2.

A conversation between User and Assistant. The user asks a question, and the Assistant solves it.
The assistant first thinks about the reasoning process in the mind and then provides the user
with the answer. The reasoning process and answer are enclosed within `<think>` `</think>` and
`<answer>` `</answer>` tags, respectively, i.e., `<think>` reasoning process here `</think>`
`<answer>` answer here `</answer>`. User: You must put your answer inside `<answer>` `</answer>` tags, i.e.,
`<answer>` answer here `</answer>`. And your final answer will be extracted automatically by the `\boxed{ }` tag.
`{{prompt}}`
Assistant: `<think>`

Table 2: Template for Open-Reasoner-Zero. `prompt` will be replaced with the specific reasoning question during generation.

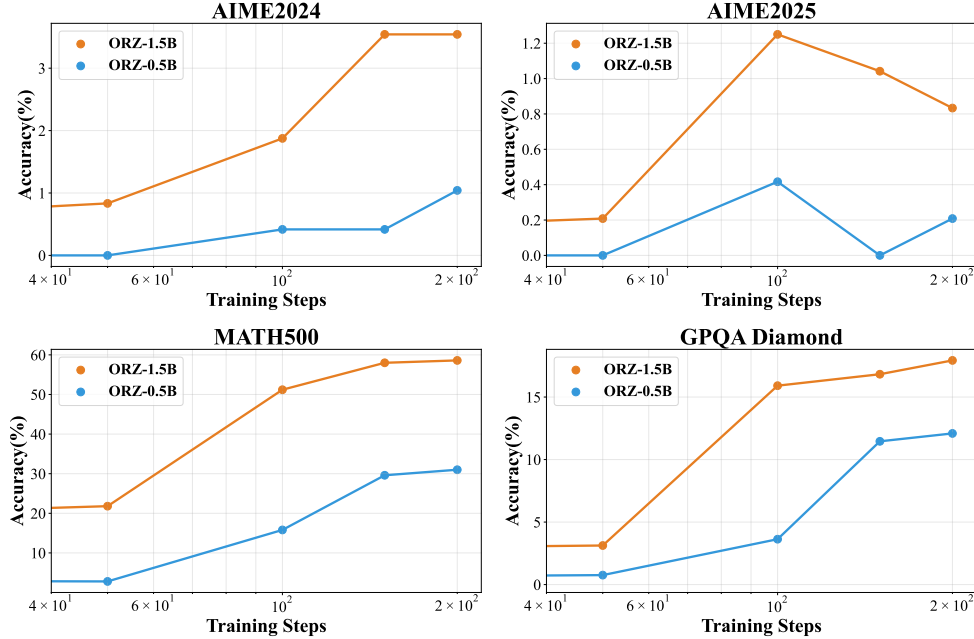


Figure 1: Evaluation performance of Open-Reasoner-Zero-{0.5B, 1.5B}. We report the average accuracy on the benchmark dataset for each question with 16 responses.

Each generation step contains 128 unique prompts sampled from the dataset, and policy generates 64 responses per prompt with temperature and top-p both set to 1.0. To maintain training stability, we implement strict on-policy optimization for the policy network, where each generation corresponds to exactly one policy optimization step. The critic network, being less sensitive to off-policy updates, processes the experiences in 12 mini-batches, effectively performing 12 optimization steps per iteration. We apply batch level advantage normalization in the training. Notably, our training process operates stably without any KL-related regularization terms or entropy bonuses, demonstrating that vanilla PPO can achieve stable training without these commonly used stabilization techniques.

For the 32B variant, we introduce an additional "annealing" training stage inspired by analogous practices in large language model pre-training [1]. Specifically, we leverage the training process of our 32B model itself to identify challenging and high-quality prompts for this annealing stage. We pinpoint 13k particularly difficult prompts, defined as those where the model achieves fewer than 4 correct answers out of a total of 64 attempts during the first 1100 steps of training. These identified prompts are then selectively used in a final training stage of 100 additional steps and apply a linear learning rate decay schedule, reducing to 3×10^{-7} . This targeted training phase is explicitly designed to enhance the model's capability on more complex reasoning tasks.

For the training of the ORZ-R1-Distill-Qwen-14B model, we initialized its weights from the DeepSeek-R1-Distill-Qwen-14B model. We utilize the mined 13k difficult prompts as training data. All other hyperparameters follow the basic configuration of the ORZ model family. The reported results correspond to the checkpoint at 300 training iterations.

C Additional Analyses and Experiments

C.1 More Analysis for Critic and Advantage Estimation

As noted in the main text, vanilla GRPO often suffers from significant training instability, a phenomenon also observed in many community implementations. This instability typically manifests as a deterioration in generation quality midway through training, with models tending to produce repetitive or incoherent text. Our large-scale experimental validation strongly corroborates these

observations and highlights the superior training stability of PPO compared to GRPO, as shown in Figure 2.

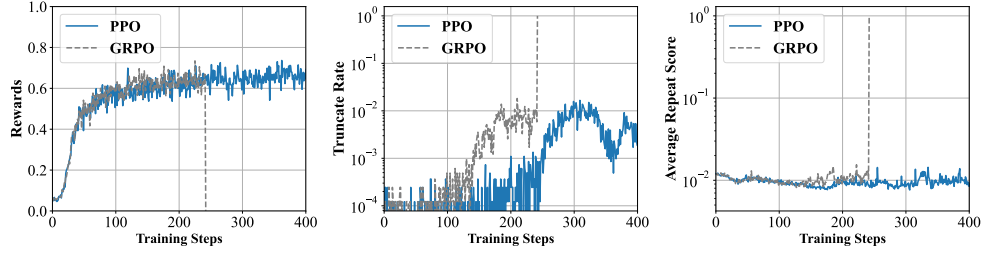


Figure 2: PPO vs. GRPO stability in the ORZ-7B setting. GRPO empirically demonstrates severe training instability around 240 training steps: its reward suddenly destabilizes, and responses degenerate as both Truncate Rate and Average Repeat Score surge to 1.0. In contrast, PPO maintains stable rewards and low Truncate/Repeat Scores throughout.

C.2 Ablation on Data Curation

Based on our analysis of data quality issues, we conduct comprehensive ablation studies to evaluate how different data curation strategies affect model training stability and performance. Motivated by OpenR1’s finding [2] that SFT performance degradation on Chinese subsets was due to simpler question patterns, we experiment with two data curation approaches: using English-only data versus using both English and Chinese data. As shown in Figure 3, the English-only dataset yields superior training stability and final model performance.

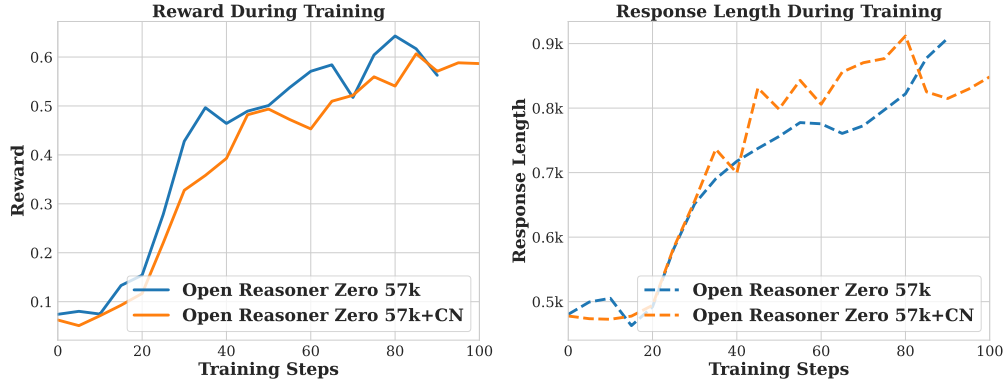


Figure 3: Data Curation Ablation. CN represents Chinese data and EN represents English data. English-only dataset yields superior training stability and final model performance.

57 **D Derivation and Code for PPO with GAE($\gamma = 1, \lambda = 1$)**

58 This section provides a detailed derivation for the GAE in the specific case where $\gamma = 1$ and $\lambda = 1$.

59 We substitute $\gamma = 1$ and $\lambda = 1$ into the general GAE formulation (defined in the main text).
 60 Recall that $\delta_{t+k} = r_{t+k} + \gamma V_\phi(s_{t+k+1}) - V_\phi(s_{t+k})$. With $\gamma = 1$, this becomes $\delta_{t+k} = r_{t+k} +$
 61 $V_\phi(s_{t+k+1}) - V_\phi(s_{t+k})$. Thus:

$$\begin{aligned}
 \hat{A}_t^{GAE(\gamma=1, \lambda=1)} &= \sum_{k=0}^{T-t-1} (1 \cdot 1)^k \delta_{t+k} \\
 &= \sum_{k=0}^{T-t-1} (r_{t+k} + V_\phi(s_{t+k+1}) - V_\phi(s_{t+k})) \\
 &= \sum_{k=0}^{T-t-1} r_{t+k} + \sum_{k=0}^{T-t-1} (V_\phi(s_{t+k+1}) - V_\phi(s_{t+k})) \quad (1) \\
 &= R - V_\phi(s_t) \quad (2)
 \end{aligned}$$

62 The step from (1) to (2) follows because: (i) the sum of rewards $\sum_{k=0}^{T-t-1} r_{t+k}$ equals the single
 63 terminal reward R for the trajectory, as intermediate rewards are zero; and (ii) the second sum
 64 $\sum_{k=0}^{T-t-1} (V_\phi(s_{t+k+1}) - V_\phi(s_{t+k}))$ is a telescoping series that evaluates to $V_\phi(s_T) - V_\phi(s_t)$, where
 65 s_T is the terminal state, and $V_\phi(s_T) = 0$.

66 Now we derive the simplified form for the value target V_t^{target} . As defined in the main text, we have:

$$\begin{aligned}
 V_t^{\text{target}} &= \hat{A}_t^{GAE(1,1)} + V_\phi(s_t) \\
 &= (R - V_\phi(s_t)) + V_\phi(s_t) \\
 &= R \quad (3)
 \end{aligned}$$

67 Substituting this simplified target $V_t^{\text{target}} = R$ into the general value loss formulation (defined in the
 68 main text):

$$\mathcal{J}_{\text{value}}(\phi) = \frac{1}{2} \mathbb{E}_{T \sim \pi_{\theta_{\text{old}}}} \left[\sum_{t=0}^{T-1} (V_\phi(s_t) - R)^2 \right] \quad (4)$$

69 We also provide a detailed algorithm implementation in 1.

Algorithm 1 PPO with GAE($\gamma = 1, \lambda = 1$)

Require: Initial policy parameters θ_0 , initial value parameters ϕ_0 , prompt dataset \mathcal{D} .

Require: Hyperparameters: clip range ϵ , trajectories per prompt n , minibatch size M .

```
1: Initialize policy  $\pi_\theta \leftarrow \pi_{\theta_0}$ , value function  $V_\phi \leftarrow V_{\phi_0}$ .
2: Initialize  $\theta_{\text{old}} \leftarrow \theta_0, \phi_{\text{old}} \leftarrow \phi_0$ .
3: for iteration = 1, 2, ... do
4:   Initialize experience buffer  $\mathcal{B} \leftarrow \emptyset$ .
5:   ▷ — Rollout Phase —
6:   Sample batch of prompts  $\{q_j\}$  from  $\mathcal{D}$ .
7:   for all prompts  $q_j$  in the batch do
8:     Generate trajectory  $\tau = (s_0, a_0, \dots, s_{T-1}, a_{T-1})$  using policy  $\pi_{\theta_{\text{old}}}$ .
9:     Compute terminal reward  $R \in \{0, 1\}$  for  $\tau$ .
10:    Compute value estimate  $V_t^{\text{old}} = V_{\phi_{\text{old}}}(s_t)$ .
11:    Compute advantage  $\hat{A}_t = R - V_t^{\text{old}}$ . ▷ Using  $\gamma = 1, \lambda = 1$ 
12:    Store tuple  $(\tau, \log \pi_{\theta_{\text{old}}}(a_t|s_t), R, \hat{A}_t)$  in buffer  $\mathcal{B}$ .
13:  end for
14:  ▷ — Update Phase —
15:  ▷ Update critic model
16:  for all minibatches  $(\tau, R)$  from  $\mathcal{B}$  do
17:    Compute current critic value  $V_\phi(s_t)$ .
18:     $L_{\text{VF}}(\phi) = \frac{1}{2}(V_\phi(s) - R)^2$ .
19:    Backward and update  $\phi$ 
20:  end for
21:  ▷ Update policy model
22:  for all minibatches  $(\tau, \log \pi_{\text{old}}(a|s), R, \hat{A})$  from  $\mathcal{B}$  do
23:    Compute current policy log-probability  $\log \pi_\theta(a|s)$ .
24:    Calculate probability ratio  $\rho(\theta) = \exp(\log \pi_\theta(a|s) - \log \pi_{\text{old}}(a|s))$ .
25:     $L_{\text{CLIP}}(\theta) = \min(\rho(\theta)\hat{A}, \text{clip}(\rho(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A})$ .
26:    Backward and update  $\theta$ 
27:  end for
28:  Update old parameters:  $\theta_{\text{old}} \leftarrow \theta, \phi_{\text{old}} \leftarrow \phi$ .
29: end for
Ensure: Final policy parameters  $\theta$ .
```

References

- [1] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.
- [2] Loubna Ben Allal, Lewis Tunstall, Anton Lozhkov, Elie Bakouch, Guilherme Penedo, and Gabriel Martín Blázquez Hynek Kydlíček. Open r1: Evaluating llms on uncontaminated math competitions, February 2025.