

# Self-Aligned Reward: Towards Effective and Efficient Reasoners

Peixuan Han<sup>1\*</sup>, Adit Krishnan<sup>2</sup>, Gerald Friedland<sup>2</sup>, Jiaxuan You<sup>1</sup>, Chris Kong<sup>2</sup>

<sup>1</sup>University of Illinois Urbana-Champaign <sup>2</sup>Amazon Web Services

## Abstract

Reinforcement learning with verifiable rewards has significantly advanced reasoning with large language models (LLMs) in domains such as mathematics and logic. However, verifiable signals provide only coarse-grained or binary correctness feedback. This limitation results in inefficiencies like overly verbose or repetitive reasoning. Existing length-based solutions (e.g., length penalty) compromise accuracy. To address this deficiency, we introduce **self-aligned reward (SAR)**, a generic, universally applicable self-guided signal that complements verifiable rewards to enhance both reasoning accuracy and efficiency in RL. Specifically, SAR is defined as the relative perplexity difference between an answer conditioned on the query and the standalone answer, thereby favoring responses that are concise and query-specific. Quantitative analysis reveals that SAR reliably judges answer quality: concise, correct answers score higher than redundant ones, and partially correct answers score higher than entirely incorrect ones. Evaluation on 4 different models across 7 benchmarks shows that integrating SAR with prevalent RL algorithms like PPO and GRPO reduces answer length by 30%, while improving accuracy by 4%. Our analysis also shows that SAR generalizes well to out-of-domain tasks and achieves a Pareto-optimal frontier between correctness and efficiency compared to state-of-the-art baselines. We also show that SAR shortens unnecessary elaboration while preserving advanced reasoning behaviors. These results highlight the promise of self-aligned reward as a fine-grained complement to verifiable rewards, paving the way for efficient and effective LLM training.

## Introduction

Recently, reinforcement learning (RL) with verifiable rewards has attracted broad attention in LLM training, showing remarkable improvements in reasoning skills [14, 19]. However, such verifiable signals are inherently discrete and coarse: they only judge final answer correctness, but fail to capture finer distinctions among responses. For instance, an unnecessarily long solution receives no penalty as long as the final answer is correct, and an almost correct response is treated the same as a completely wrong one. This limitation often induces “overthinking”, where models generate unnecessary elaborations that increase latency and cost [42].

\*Work done during an internship at Amazon Web Services.  
Copyright © 2026, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Table 1: Comparison of different reward designs.

Reward Type	Fine Grained	Internal	Content Aware	Correct	Concise
Correctness	✗	✓	✓	✓	✗
Reward Model [32]	✓	✗	✓	✓	✗
Length Penalty [6]	✓	✓	✗	✗	✓
Entropy [2]	✓	✓	✓	✗	✗
<b>Self-Aligned (Ours)</b>	✓	✓	✓	✓	✓

To this end, researchers have proposed heuristic regulations such as length penalties or brevity-oriented objectives [28, 3]. While effective in reducing output verbosity, these methods often penalize both redundant and essential reasoning, thereby harming accuracy when necessary intermediate steps are suppressed. Consequently, this line of approaches struggles to balance efficiency with correctness. Using external signal sources, such as reward models, is also undesirable due to their vulnerability to reward hacking. This underscores the necessity of developing internally grounded reward mechanisms that provide precise and detailed guidance, discerning necessary reasoning from redundant elaboration. To close this gap, we introduce **Self-Aligned Reward (SAR)**, a self-guided proxy to judge answer quality based on *perplexity*, an informative metric modeling uncertainty [11, 2]. Specifically, SAR evaluates the perplexity of an answer both in isolation and when conditioned on the query, and then measures their relative difference between the two. Consequently, the reward promotes answers that are highly confident under the query context but unlikely to arise independently without the query, which typically corresponds to responses that are concise and strongly aligned with the query. Notably, SAR is the only fine-grained approach that promotes accuracy and efficiency at the same time, as shown in Table 1.

We first conduct a quantitative analysis of different types of answers to demonstrate that SAR provides an accurate fine-grained reward landscape over answers of different qualities. We then train LLMs by combining SAR and verifiable reward in PPO and GRPO, two prevalent reinforcement learning algorithms. **We find PPO and GRPO with SAR (denoted as SA-PPO and SA-GRPO) achieve notable gains over baselines across 4 models and 7 benchmarks, improving accuracy by 4% and efficiency by 30%. More-**

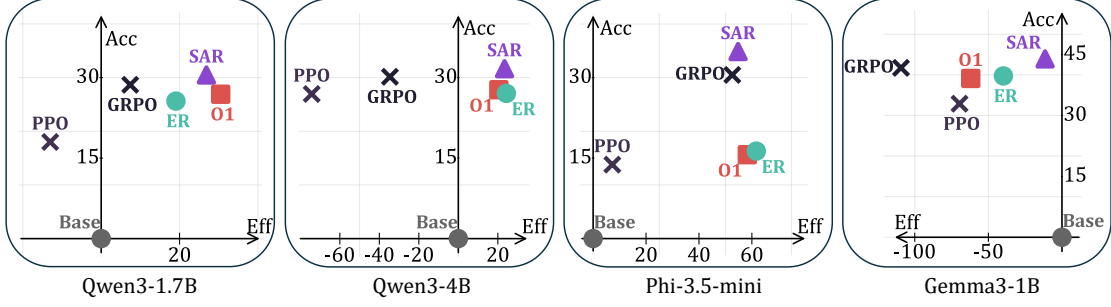


Figure 1: **Training with self-aligned reward enhances both efficiency and accuracy.** We present the relative gains in efficiency and accuracy compared to the respective base model in math reasoning benchmarks. Efficiency gain is measured as the drop in average response length.

over, SAR outperforms length-based rewards with a Pareto-optimal front in the accuracy-efficiency trade-off, as illustrated in Figure 1. In addition, we demonstrate the advantages of SAR over confidence-based methods and provide an analysis of its reasoning behaviors. Our findings suggest that combining verifiable rewards with intrinsic model self-judgment offers a new paradigm for RL training, enabling improvements in both reasoning capability and efficiency.

## Methodology

### Reinforcement Learning Formulation

Proximal Policy Optimization (PPO) [37] and Group Relative Policy Optimization (GRPO) [38] are two widely adopted RL algorithms. Given the current policy  $\pi_\theta$ , query  $q$ , and rollout  $a$ , PPO and GRPO can be formulated as:

$$\mathcal{J}_{\text{PPO}}(\pi_\theta) = \mathbb{E}_{q \sim D, a \sim \pi_\theta(q)} \left[ \frac{1}{|a|} \sum_{j=1}^{|a|} \min \left\{ r_j A_j, \text{clip}(r_j, 1 - \epsilon, 1 + \epsilon) A_j \right\} - \beta \text{KL}(\pi_\theta \| \pi_{\text{ref}}) \right] \quad (1)$$

$$\text{where } r_j = \frac{\pi_\theta(a_j | q, a_{1..j-1})}{\pi_{\text{old}}(a_j | q, a_{1..j-1})}, A_j = Q(q, a_{1..j}) - V(q, a_{1..j}). \quad (2)$$

$$\mathcal{J}_{\text{GRPO}}(\pi_\theta) = \mathbb{E}_{q \sim D, a_1 \dots a_N \sim \pi_\theta(q)} \left[ \frac{1}{N} \sum_{i=1}^N \frac{1}{L_{\max}} \sum_{j=1}^{|a_i|} \min \left\{ r_{i,j} A_i, \text{clip}(r_{i,j}, 1 - \epsilon, 1 + \epsilon) A_i \right\} - \beta \text{KL}(\pi_\theta \| \pi_{\text{ref}}) \right] \quad (3)$$

$$\text{where } r_{i,j} = \frac{\pi_\theta(a_{i,j} | q, a_{i,1..j-1})}{\pi_{\text{old}}(a_{i,j} | q, a_{i,1..j-1})}, A_i = R(q, a_i) - \text{mean}[R(q, a_{1..N})]. \quad (4)$$

In the above formulas,  $A$  is the advantage,  $D$  is the training dataset, and  $r$  refers to the importance sampling ratio.  $N$ ,  $\epsilon$ ,  $L_{\max}$  and  $\beta$  are hyperparameters. Specifically, we use Dr. GRPO [26], an unbiased variant of GRPO in this paper. For verifiable tasks, the reward for PPO and GRPO can be obtained by comparing the model answer with the expected ground truth ( $gt$ ):

$$R_{\text{PPO/GRPO}}(q, a, gt) = R_{\text{VR}}(q, a, gt) = \begin{cases} 1 & \text{if } gt \text{ in } a, \\ 0 & \text{otherwise.} \end{cases} \quad (5)$$

Although RL with verifiable reward has set new benchmarks in reasoning tasks, it induces undesired characteristics such as redundant output due to the coarse binary nature of the reward. To address this issue, two widely used methods introduce targeted penalties on output length: O1-pruner [28] and Efficient Reasoner [4] (referred to as **O1** and **ER**). Their detailed formulations are provided in the Appendix. Despite their effectiveness, both methods inevitably trade off accuracy, which constrains their broader applicability.

### Self-Aligned Reward

We propose **Self-Aligned Reward (SAR)** (denoted as  $R_{\text{SA}}$ ), an unsupervised holistic signal that combines generation quality, conciseness, and query-answer relevance. The self-aligned reward is estimated entirely by the model policy without any external feedback. We then combine our reward with the binary verifiable reward to address the shortcomings of previous methods, aiming to improve both accuracy and generation efficiency. The reward for self-aligned PPO and self-aligned GRPO (**SA-PPO** and **SA-GRPO**) can be formulated as follows<sup>1</sup>:

$$R_{\text{SA-PPO/GRPO}}(q, a_i, gt) = R_{\text{VR}} + \alpha R_{\text{SA}},$$

$$R_{\text{SA}} = \text{clip} \left( \frac{\text{ppl}(a_i) - \text{ppl}(a_i|q)}{\text{ppl}(a_i)}, -1, 1 \right) \quad (6)$$

$$\text{where } \text{ppl}(a) = e^{-\frac{1}{|a|} \sum_{j=1}^{|a|} \log P(a_j | a_{1..j-1})},$$

$$\text{ppl}(a|q) = e^{-\frac{1}{|a|} \sum_{j=1}^{|a|} \log P(a_j | q, a_{1..j-1})}. \quad (7)$$

The self-aligned reward captures the relative difference between  $\text{ppl}(a)$  and  $\text{ppl}(a|q)$ , or the **conditioned perplexity drop**, based on the current model policy. Perplexity reflects the likelihood that the model will produce the given

<sup>1</sup>Since  $\text{ppl} > 0$ , we always have  $R_{\text{SA}} < 1$ . In practice,  $R_{\text{SA}}$  lies in the (0, 1) range in most cases.

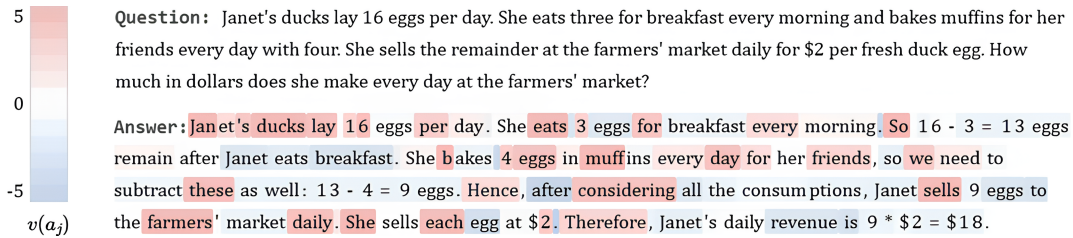


Figure 2: An illustration of token-level importance scores (i.e.  $v(a_j)$ ). Red color means the token is considered informative for our self-aligned reward, and blue indicates a negative  $v(a_j)$ . **Tokens extracting new information from the query get high scores.**

response, with lower values indicating higher confidence. Thus,  $R_{SA}$  can be interpreted as: “How much less likely does the answer become if the query is not present?”

When an answer is tightly tailored to the query, the conditioned perplexity  $\text{ppl}(a|q)$  will be significantly lower than the standalone perplexity  $\text{ppl}(a)$ , leading to a higher  $R_{SA}$ . In contrast, if parts of the answer are irrelevant to the query or include noisy, verbose text, the two perplexities will be similar, leading to a smaller  $R_{SA}$ . Therefore, a larger value of  $R_{SA}$  indicates the answer’s stronger dependency and better alignment to the query.

The following section provides an in-depth analysis of this reward formulation, where we demonstrate that self-aligned reward encourages concise reasoning and effective use of query information, thereby improving both accuracy and efficiency, and penalizing noisy verbose answers. In addition, the computation of  $R_{SA}$  integrates seamlessly into the RL pipeline with negligible computational overhead.

### Why Self-aligned Reward Works

This section analyzes the self-aligned reward formulation through detailed case studies, providing insights on how SAR enhances the reasoning effectiveness and efficiency.

#### SAR Provides Accurate and Fine-grained Signals

An ideal reward design should be able to rate different types of answers based on their qualities. To assess different reward functions, we analyze 6 different types of answers to 200 math questions sampled from five common math benchmarks in Table 2. Types (1) to (4) are obtained by sampling Qwen3-1.7B rollouts at temperature = 1, and using GPT-4o to annotate the responses (see the Appendix for details). Types (5) and (6) are artificially synthesized to simulate memorization, where the LLM directly extracts answers from its knowledge without reasoning.

We present the advantage values (Equation (4)) for each type of response using the following rewards: verifiable GRPO rewards ( $A_{GRPO}$ ), length-based rewards from the O1 and ER methods ( $A_{GRPO-O1}$ ,  $A_{GRPO-ER}$ ) and our self-aligned reward  $A_{SA-GRPO}$ . From Table 2, we can observe that SAR provides rich and accurate signals to different answers, exhibiting the following features:

- SAR favors concise and correct answers; it gives a lower reward to long and redundant answers, promoting efficiency. Our next experiment explains this phenomenon in detail.

Table 2: We calculate advantage values over the 6 types of answers to the same question ( $\alpha = 1$ ), and report the average over 200 questions. **The advantage values of SA-GRPO accurately rate answers of different qualities.**

Answer Type	Length	$A_{GRPO}$	$A_{GRPO-O1}$	$A_{GRPO-ER}$	$A_{SA-GRPO}$ (Ours)
(1): Correct and concise	143.5	0.5	1.04	0.81	1.15
(2): Correct but redundant	236.0	0.5	0.35	-0.11	1.00
(3): Partly correct	457.1	-0.5	-1.28	-0.65	-0.01
(4): Completely wrong	405.8	-0.5	-1.19	-0.65	-1.44
(5): Correct but no thought	5.0	0.5	1.04	1.25	-0.04
(6): Incorrect and no thought	5.0	-0.5	0.04	-0.65	-0.68

- SAR provides partial credit to partly correct answers and penalizes completely irrelevant ones, helping the model learn basic reasoning patterns in the initial stage of training.

- SAR penalizes the synthesized “no thought” answers, even if they’re correct and short. This indicates that the reasoning process plays a critical role in SAR, and memorization is discouraged.

On the other hand, the verifiable reward (GRPO) fails to discern answers that are both correct and incorrect, and O1 or ER focus solely on response length, making signals biased and not suitable for maximizing accuracy. For instance, ER isn’t applicable to wrong answers, and O1 favors irrelevant answers over partly correct ones, simply because completely wrong answers are typically shorter.

#### SAR Promotes Exploiting Query Information

In this experiment, we take a deeper look at SAR at the token level, aiming to reveal which tokens contribute more to the overall score. Specifically, we decompose the self-aligned reward (Equations (6) and (7)) formulation to calculate the score for individual tokens<sup>2</sup>:

$$R_{SA} = 1 - \frac{\text{ppl}(a|q)}{\text{ppl}(a)} = 1 - e^{-\frac{1}{|a|} \sum_{j=1}^{|a|} \log \frac{P(a_j|q, a_1 \dots a_{j-1})}{P(a_j|a_1 \dots a_{j-1})}}. \quad (8)$$

We then define  $v(a_j) = \log \frac{P(a_j|q, a_1 \dots a_{j-1})}{P(a_j|a_1 \dots a_{j-1})}$  to measure the importance of each token  $a_j$  on the whole metric  $R_{SA}$ . A token with a higher  $v(a_j)$  is considered valuable, while a token with a low or even negative  $v(a_j)$  indicates it’s less informative and independent of the query.

From Figure 2, we observe that tokens drawing on information from the question for the first time, like “Janet”,

<sup>2</sup>We omit the clipping for simplicity.

Table 3: Evaluation on math benchmarks. **Self-aligned reward significantly reduces token usage while achieving the best reasoning accuracy.** GRPO-O1 and GRPO-ER results in the table use  $\alpha = 0.05$ , and SA-GRPO uses  $\alpha = 0.2$ . Best results among each model are bolded.

Setting	GSM8k		GSM-sym		MATH		NuminaMath		AIME		Average		
	acc	len	acc	len	acc	len	acc	len	acc	len	acc	len	AES
<b>Qwen3-1.7B</b>	69.22	281	46.76	365	56.89	700	24.58	1242	9.97	1539	41.48	825.4	0.000
+PPO	82.71	332	62.88	460	63.04	784	27.02	1358	9.11	1718	48.95	930.4	0.773
<b>+SA-PPO</b>	<b>85.51</b>	281	<b>68.02</b>	369	66.27	587	32.05	866	10.61	1015	52.49	623.6	1.572
+GRPO	84.53	335	67.66	413	67.31	697	33.44	1091	13.93	1278	53.37	762.8	1.509
+GRPO-O1	84.15	285	65.78	363	66.51	528	33.15	775	13.61	910	52.64	572.2	1.652
+GRPO-ER	82.71	244	64.74	320	66.54	557	34.35	980	12.22	1232	52.11	666.6	1.474
<b>+SA-GRPO</b>	<b>85.51</b>	267	67.66	346	<b>67.96</b>	564	<b>36.03</b>	841	<b>13.50</b>	992	<b>54.13</b>	602.0	<b>1.795</b>
<b>Qwen3-4B</b>	76.19	315	66.22	435	63.79	652	38.80	1142	20.15	1281	53.03	765.0	0.000
+PPO	91.36	373	85.42	485	82.15	1132	50.80	1995	26.90	2680	67.33	1333.0	0.606
<b>+SA-PPO</b>	92.12	266	83.52	345	78.67	652	49.09	1133	28.19	1397	66.32	758.6	1.260
+GRPO	92.62	320	86.30	414	82.51	821	53.79	1485	30.11	2113	69.07	1030.6	1.165
+GRPO-O1	91.89	262	86.76	350	80.41	549	51.06	866	28.62	1016	67.75	608.6	1.592
+GRPO-ER	92.80	219	85.24	297	79.48	503	50.47	854	28.94	1022	67.38	579.0	<b>1.596</b>
<b>+SA-GRPO</b>	<b>93.40</b>	239	<b>87.64</b>	323	<b>82.63</b>	762	<b>57.70</b>	1358	<b>35.69</b>	1788	<b>71.41</b>	894.0	1.564
<b>Phi-3.5-mini</b>	71.19	287	56.86	398	33.83	1132	11.56	1492	2.35	943	35.15	850.4	0.000
+PPO	82.49	209	64.88	322	38.56	831	11.45	1707	<b>3.85</b>	926	40.24	799.0	0.784
<b>+SA-PPO</b>	87.04	238	73.52	344	46.34	300	<b>17.43</b>	586	2.79	501	45.42	393.8	1.997
+GRPO	87.56	235	<b>73.76</b>	337	48.69	415	16.37	582	3.21	672	45.91	448.2	2.003
+GRPO-O1	79.83	203	62.24	279	44.41	390	14.08	509	2.57	583	40.62	392.8	1.316
+GRPO-ER	82.03	154	63.54	240	43.61	363	12.55	511	2.47	606	40.84	374.8	1.368
<b>+SA-GRPO</b>	<b>87.95</b>	207	72.72	289	<b>50.99</b>	356	16.08	455	3.22	535	<b>46.19</b>	368.4	<b>2.137</b>
<b>Gemma3-1B</b>	42.15	325	19.58	427	36.40	1042	12.40	1414	2.25	2279	22.56	1097	0.000
+PPO	56.86	942	30.28	1187	44.17	1408	15.57	1904	1.60	2422	29.69	1572.6	1.146
<b>+SA-PPO</b>	55.80	683	31.00	936	42.94	1025	14.22	1523	1.50	1683	29.10	1170.0	1.383
+GRPO	59.97	1208	<b>34.70</b>	1552	45.40	1693	16.59	2264	2.25	2613	31.78	1866.0	1.343
+GRPO-O1	60.80	873	32.06	1157	44.76	1298	16.16	1871	<b>2.35</b>	2449	31.23	1529.6	1.528
+GRPO-ER	59.44	748	32.36	1063	46.23	1131	16.41	1545	2.04	1696	31.29	1236.6	1.808
<b>+SA-GRPO</b>	<b>61.26</b>	552	34.52	772	<b>46.60</b>	952	<b>16.70</b>	1302	2.14	1509	<b>32.24</b>	1017.4	<b>2.218</b>

“duck”, “16”, tend to receive high scores. This is because such information is present in the query but absent from previous answer tokens, making  $P(a_j|q, a_{1..j-1})$  relatively high while  $P(a_j|a_{1..j-1})$  remains low. In contrast, repeating information already generated, like mentioning “Janet” for the second time, results in low scores as both probabilities become similarly high.

Generally, tokens in the earlier part of an answer typically achieve higher  $v(a_j)$  values, since extracting new content from the query is easier at the beginning. Later tokens, by comparison, struggle to introduce novel information once much of the query has already been incorporated. This explains why SAR favors short, concise answers and promotes more efficient generation.

## Experiments

### Experimental Settings

**Datasets.** We utilize five math reasoning benchmarks: GSM8k [8], MATH [18], NuminaMath 1.5 [25], GSM-symbolic [31], and AIME 1983–2024 [45], covering a difficulty range from primary school to competition level. For training, we combine the training splits of the first three datasets, while the remaining two are not present during training to evaluate generalization.

**Models.** We utilize four base models: Qwen3-1.7B

(Base), Qwen3-4B (Base) [49], Phi-3.5-mini (Instruct, 3.8B) [1], and Gemma3-1B (Instruct) [43], covering different sizes and model families.

**Settings.** We train models with the following methods:

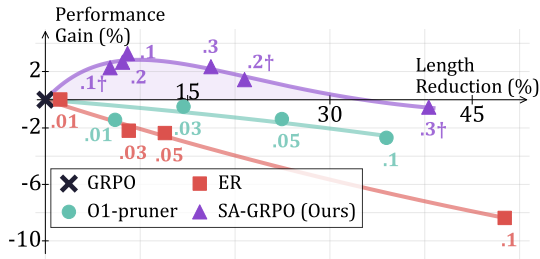
- PPO, GRPO: two traditional RL algorithms, using only the verifiable reward.
- GRPO-O1, GRPO-ER: algorithms with length penalties, aimed to enhance efficiency. See Appendix for details.
- SA-PPO, SA-GRPO: algorithms using **self-aligned reward (SAR)**. See Method for details.

Training details and settings are shown in the Appendix.

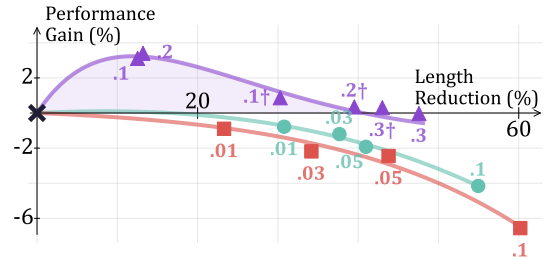
**Metrics.** In this work, we focus on accuracy and efficiency (measured by average answer length) of LLM reasoning. We report these two metrics as well as an Accuracy-Efficiency trade-off Score (AES). For a trained policy  $\pi_\theta$  and its base model  $\pi_{\text{ref}}$ , we define  $\Delta \text{len} = \frac{\text{len}(\pi_{\text{ref}}) - \text{len}(\pi_\theta)}{\text{len}(\pi_{\text{ref}})}$  and  $\Delta \text{acc} = \frac{\text{acc}(\pi_\theta) - \text{acc}(\pi_{\text{ref}})}{\text{acc}(\pi_{\text{ref}})}$ . Then,  $\text{AES}(\pi_\theta) = \Delta \text{len} + \gamma \Delta \text{acc}$  measures the trade-off.

### Main Results

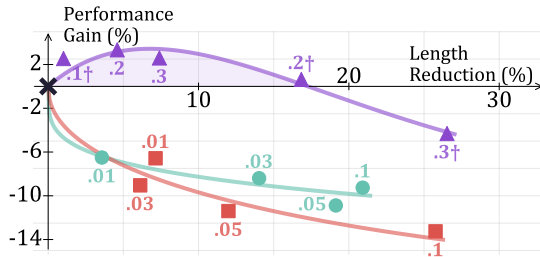
From Table 3, we observe that **baseline approaches struggle to balance accuracy and efficiency**. PPO and GRPO significantly improve accuracy at the cost of longer and possibly redundant answers. While GRPO-O1 and GRPO-ER effectively reduce computation overhead, they compro-



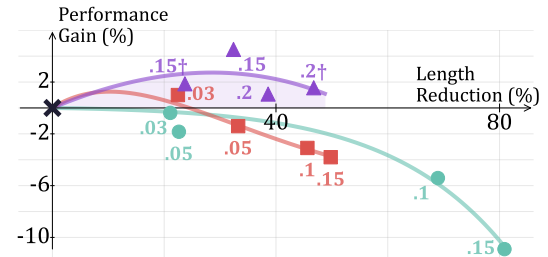
(a) Qwen3-1.7B.



(b) Qwen3-4B.



(c) Phi-3.5-mini.



(d) Gemma3-1B.

Figure 3: Accuracy-efficiency balance of different algorithms. **SA-GRPO reaches a Pareto-optimal curve and achieves notable gains on both axes.** Numbers around each point indicate the  $\alpha$  values, and † indicates using 20% of the total training data with the same total training steps.

mise reasoning capability. In contrast, **SAR delivers substantial improvements in both accuracy and efficiency.** Across four base models, SA-GRPO consistently achieves the highest reasoning accuracy while maintaining highly efficient reasoning, with at least 4% improvement in accuracy and 30% reduction in length compared to GRPO. Notably, SA-GRPO produces answers of comparable or even shorter length than GRPO-O1 and GRPO-ER, which are explicitly designed for efficient reasoning. A similar phenomenon is observed for SA-PPO, indicating that self-aligned reward applies to diverse RL algorithms. We show two examples comparing GRPO and SA-GRPO in the Appendix.

These findings highlight the superiority of the self-aligned reward mechanism. With a more fine-grained and intelligent signal, SAR makes the model preserve the necessary reasoning which are closely related to query information, and reduces unnecessary content that introduces token burdens, leading to more effective and efficient reasoning.

### Self-aligned Reward Achieves a Pareto-Optimal in Dynamic Balance

The hyperparameter  $\alpha$  in Equation (6) controls the weights of verifiable reward and self-aligned reward, offering flexibility to focus more on efficiency or accuracy. Therefore, we train SA-GRPO with different  $\alpha$  values and compare it with GRPO-O1 and GRPO-ER on 2-dimensional plots.

From Figure 3, we can observe that the curves for SA-GRPO are consistently on the top-right side over curves for GRPO-O1 and GRPO-ER, indicating SA-GRPO balances accuracy and efficiency better than length-based methods.

Moreover, length-based methods are always under the x-axis in most cases, showing that these methods sacrifice accuracy for reduced tokens, while SA-GRPO is the only method consistently above the x-axis (illustrated as the light purple area in the figure), showing better accuracy and better efficiency compared to vanilla GRPO. The Pareto-optimal balance between accuracy and efficiency of SA-GRPO further demonstrates the effectiveness of SAR.

### Self-aligned reward Generalizes to Logical Reasoning

In this section, we examine the generalization ability of self-aligned reward (SAR) in a different domain—logical reasoning. We evaluate the models on two benchmark datasets: LogicBench [33] and ProntoQA [36]. Although none of the models are fine-tuned on the logical reasoning domain, logical reasoning shares certain similarities with mathematical reasoning. Therefore, a model with strong generalization ability should be able to transfer its reasoning skills.

From Table 4, we can find that SA-GRPO enhances accuracy compared to GRPO on 5 out of 8 columns, and outperforms length-based methods (GRPO-O1 and ER) on all cases. Similarly, SA-PPO outperforms the PPO baseline on 6 columns. In addition, SAR also maintains an efficiency benefit in the logical domain, not only reducing answer length compared to GRPO, but it’s even more efficient than length-based methods in most cases. These results show that SAR generalizes well to logical reasoning benchmarks, which are out-of-domain tasks, benefiting from the reward design that ensures answers are tailored to the input ques-

Table 4: SA-PPO and SA-GRPO maintains robustness and efficiency in out-of-domain tasks.

(a) LogicBench.									(b) ProntoQA.								
Model	Qwen3 -1.7B		Qwen3 -4B		Phi-3.5 -mini		Gemma3 -1B		Model	Qwen3 -1.7B		Qwen3 -4B		Phi-3.5 -mini		Gemma3 -1B	
	acc	len	acc	len	acc	len	acc	len		acc	len	acc	len	acc	len	acc	len
Base	56.0	182	75.4	336	66.0	309	50.6	303	Base	69.0	616	89.6	988	94.8	339	56.2	712
PPO	60.2	340	80.0	455	60.8	229	48.8	264	PPO	71.8	920	97.8	561	96.4	301	56.4	904
SA-PPO	64.6	368	77.6	342	64.2	205	<b>55.6</b>	241	SA-PPO	65.6	735	99.0	405	<b>98.2</b>	304	57.6	690
GRPO	64.2	372	78.4	416	<b>67.6</b>	243	53.0	277	GRPO	74.0	756	<b>100.0</b>	546	<b>98.2</b>	254	57.4	963
GRPO-O1	60.6	264	76.0	350	51.0	212	52.6	223	GRPO-O1	73.0	505	98.8	469	87.2	227	59.0	938
GRPO-ER	61.4	206	77.6	263	49.8	143	50.0	250	GRPO-ER	73.6	424	99.0	309	86.6	205	58.6	919
SA-GRPO	<b>65.0</b>	278	<b>80.4</b>	208	57.4	173	55.2	222	SA-GRPO	<b>77.8</b>	482	99.8	301	90.8	216	<b>60.6</b>	733

Table 5: GRPO training results on Qwen3-4B with different internal reward signals ( $\alpha = 0.2$ ). **The SA-GRPO formulation obtains optimal performance among baselines.**

Setting	GSM8k		GSM-sym		MATH		NuminaMath		AIME		Average	
	acc	len	acc	len	acc	len	acc	len	acc	len	acc	len
Base	76.19	315	66.22	435	63.79	652	38.80	1142	20.15	1281	53.03	765.0
$R_{VR}$	92.62	320	86.30	414	82.51	821	53.79	1485	30.11	2113	69.07	1030.6
$R_{EM}$	79.15	319	67.94	414	68.89	970	39.86	1885	21.44	2556	55.46	1228.8
$R_{SA}$	39.65	87	19.54	102	24.72	74	16.48	80	4.39	79	20.96	84.4
$R_{VR} + \alpha R_{EM}$	92.25	351	87.36	414	81.83	779	54.78	1371	33.01	1767	69.85	936.4
$R_{VR} + \alpha R_{SA}$	<b>93.40</b>	239	<b>87.64</b>	323	<b>82.63</b>	762	<b>57.70</b>	1358	<b>35.69</b>	1788	<b>71.41</b>	894.0

tions and contain dense information.

## Analysis

### Ablation Study

In this section, we investigate two critical components of our algorithm that make internal signals effective during training. Firstly, the self-aligned reward (Equation (6)) measures **conditioned perplexity drop**, which is the relative difference between  $ppl(a|q)$  and  $ppl(a)$ , as a proxy for query-answer relevancy. However, existing methods focus mainly on  $ppl(a|q)$  alone, which measures entropy or uncertainty [54, 2]. Secondly, our approach combines **verifiable reward** with self-aligned reward. We ablate these components and derive the following rewards to compare against SA-GRPO ( $R = R_{VR} + \alpha R_{SA}$ ):

- $R = R_{SA}$  removes the **verifiable reward**, using only the self-aligned reward.
- $R = R_{VR} + \alpha R_{EM} = R_{VR} - \alpha \log ppl(a|q)$  uses “entropy minimization”, a metric measuring self-confidence, as the internal reward, instead of **conditioned perplexity drop**;
- $R = R_{EM}$  is the entropy minimization reward [2], ablating both components.

From Table 5, we can observe that both verifiable signal and conditioned drop measurement are critical for optimal performance. Specifically,  $R_{EM}$  shows limited accuracy gain but a large efficiency drop, and  $R_{SA}$  converges to shallow reasoning with fewer tokens and poor accuracy. Failure of these methods indicates that the ground-truth signals are still critical for models to develop reasoning skills and ensure training stability, similar to what’s found in [53]

$R_{VR} + \alpha R_{EM}$  also underperforms SA-GRPO in both accuracy and efficiency, indicating that minimizing entropy is less effective than our approach, which is maximizing conditioned perplexity drop. This is because conditioned perplexity drop provides a more accurate measure of answer quality, as demonstrated in previous sections. Moreover, it avoids issues such as overconfidence or entropy collapse, which can hinder exploration [53, 10]. Overall, the results suggest “verifiable signal” and “conditioned perplexity drop” are two crucial components in SAR, without which SA-GRPO won’t be able to reach its optimal performance.

### Reasoning Behaviors of SA-GRPO

Previous work [12, 52] has shown that certain reasoning behaviors are critical to effective and deep reasoning. In this section, we investigate four typical behaviors: backtracking, verification, subgoal setting, and enumeration. Specifically, we use GPT-4o to annotate the reasoning behaviors for models trained with different algorithms (refer to the appendix for details about these behaviors).

From Table 6, we can observe that GRPO-O1/ER exhibit fewer reasoning behaviors compared to GRPO, as reasoning behaviors require additional tokens which are penalized by length-based reward functions. However, **SA-GRPO maintains almost the same frequency of reasoning behaviors with GRPO**, notably with 30% fewer tokens than GRPO. This stems from the self-aligned reward’s content-aware feature, showcasing its ability to accurately distinguish useful reasoning behaviors from unnecessary content, guiding model behaviors in a more accurate and unbiased manner.

Table 6: Frequency of reasoning behaviors. **B, V, S, E** refer to **B**acktracking, **V**erification, **S**ubgoal setting and **E**numeration, respectively. Scores are in percentages. The base model is Qwen3-1.7B. Unlike length-based methods, **SA-GRPO maintains a high usage of reasoning behaviors.**

Dataset Behavior	MATH				NuminaMath				AIME				Average
	<b>B</b>	<b>V</b>	<b>S</b>	<b>E</b>	<b>B</b>	<b>V</b>	<b>S</b>	<b>E</b>	<b>B</b>	<b>V</b>	<b>S</b>	<b>E</b>	
Base	0.8	24.2	90.8	12.2	5.4	27.0	93.4	34.8	4.4	36.2	93.2	37.4	38.0
GRPO	1.0	29.2	95.4	14.6	4.6	38.8	97.8	44.2	5.0	42.2	98.2	49.0	43.4
GRPO-O1	1.2	26.6	90.4	14.6	4.8	37.0	93.2	39.6	4.2	43.4	96.0	42.8	40.6
GRPO-ER	1.4	26.0	89.8	13.4	6.2	37.0	90.4	40.4	5.8	44.2	96.2	47.6	41.2
<b>SA-GRPO</b>	0.8	29.2	93.4	13.8	7.0	36.0	93.8	43.8	8.8	47.0	97.0	49.6	43.0

### Training Cost of self-aligned reward

We report the training cost for SA-GRPO in Table 7, from which we can find that SA-GRPO doesn’t introduce burdens in “Update” phase compared to vanilla GRPO. The only additional computation in SAR is  $ppl(a)$  ( $ppl(a|q)$  is already calculated in GRPO, for KL penalty and importance sampling), which only requires a forward pass, making it highly efficient. Additionally, SA-GRPO even takes less time in the “Rollout” phase due to reduced answer lengths. This suggests that enhancing reasoning efficiency is not only valuable during inference – it can also accelerate RL training and maximize the gain of reasoning capability under limited computation.

Table 7: Comparison of training time. We report GPU hours (Training time  $\times$  GPU count) of training Qwen3-4B for the first 200 steps.

Method	Rollout	Update	Total
GRPO	32.95	15.13	48.08
GRPO-O1	30.36	15.92	46.28
GRPO-ER	30.92	15.24	46.16
<b>SA-GRPO</b>	31.44	15.20	46.64

### Related Work

**Reinforcement Learning for LLMs.** Reinforcement learning (RL) has emerged as a powerful paradigm for fine-tuning large language models (LLMs) to enhance their performance in reasoning tasks [32, 14]. Proximal Policy Optimization (PPO) [37] and Group Relative Policy Optimization (GRPO) [38] are typical algorithms that are widely adopted in diverse scenarios and domains [22, 7, 15, 27]. Improvements to these algorithms have also been proposed, namely Dr.GRPO [26], GSPO [55], and Clip-Cov [10].

**Efficient Reasoning.** Reasoning models often suffer from overthinking [41, 9, 42], leading to unnecessary computation burdens. Prompt engineering [17, 29, 34] and instruction tuning [51, 21, 48, 17, 30] have been widely used to mitigate this drawback. Recently, researchers have also explored using RL to elicit efficient reasoning abilities through reward signals that penalize overly lengthy answers [3, 44] or relatively longer answers [28, 4]. In addition, Yeo et al. [50] explores the length penalty’s relationship with reasoning behaviors, and Chen et al. [6] proposes a difficulty-sensitive

method for token compression. However, these methods all sacrifice accuracy to gain efficiency, which is the key issue this paper aims to address.

**Self-judging of LLMs.** Self-judging is a fundamental capability of LLMs [35], playing a key role in scalable training and self-evolution. The concept of self-judging is widely adopted. For instance, confidence estimation is widely used to reduce hallucination [13, 47, 20], probing internal states provides a lens to analyze model behavior [16, 5], and majority voting [46] is critical in inference-time scaling techniques [40]. Self-judging has also been used in RL to train scalable reasoning models, with a primary focus on entropy-based metrics [2, 24, 54]. Zhang et al. [53] discussed the theoretical foundation and limitations of self-judged signals. Our work also focuses on this direction, and we novelly propose the self-aligned reward measuring question-answer relevancy instead of self-confidence, and combine it with verifiable signals.

### Conclusion

In this work, we propose **Self-Aligned Reward (SAR)**, an internal perplexity-based signal evaluating the answer’s relevancy with the query, enabling fine-grained supervision beyond binary correctness. Through comprehensive experiments on 4 base models and 7 benchmarks, we demonstrated that SAR enables reinforcement learning to achieve consistent gains of up to 4% in accuracy while reducing response length and computational cost by 30%. Moreover, SAR exhibits a favorable accuracy–efficiency balance compared with length-based baselines, offering a fine-grained and content-aware reward signal that complements verifiable correctness. Our analysis further shows that SAR generalizes robustly to out-of-domain tasks and preserves advanced reasoning behaviors, underscoring its broad applicability. These findings highlight the significance of incorporating intrinsic model self-assessment into the RL framework, establishing a new paradigm that advances both the effectiveness and efficiency in training next-generation reasoning models.

## APPENDIX

### Datasets

We evaluate our approach on five mathematical reasoning datasets with varying difficulty levels, and two logical reasoning datasets. We list the datasets and provide details below.

#### Mathematical Reasoning Benchmarks

1. **GSM8k** [8]: A dataset of grade school math word problems.
2. **MATH** [18]: A challenging dataset of mathematics problems covering various topics.
3. **NuminaMath 1.5** [25]: A comprehensive dataset containing 860k pairs of competition math problems and solutions. We select a subset from the Open-r1 project<sup>3</sup>.
4. **GSM-symbolic** [31]: A dataset of GSM8k-style problems with different numbers. This dataset is free from data contamination, making it suitable to evaluate generalization ability.
5. **AIME 1983-2024** [45]<sup>4</sup>: A collection of problems from the American Invitational Mathematics Examination spanning over four decades. Requires complex reasoning.

To ensure parsing correctness, we only select questions where the answer is a single integer or fraction number, and remove questions involving geometric plots or hyperlinks. During training, we combined the training splits of GSM8k, MATH, and NuminaMath. GSM-symbolic and AIME were used exclusively for evaluation to test out-of-distribution generalization.

#### Logical Reasoning Benchmarks

To evaluate the generalization capabilities of our approach beyond mathematical reasoning, we used the following logical reasoning benchmarks:

1. **ProntoQA** [36]: A dataset evaluating multi-step syllogistic reasoning.
2. **LogicBench** [33]: A comprehensive benchmark for evaluating logical reasoning capabilities of language models across various logical relationships.

These datasets were used only for evaluation purposes and were not part of the training process. Both datasets are in the form of multiple-choice questions, where ProntoQA has 2 choices and LogicBench has 4 for each question.

Table 8: Sizes of datasets.

Dataset	GSM8k	MATH	NuminaMath	GSM-sym	AIME	ProntoQA	LogicBench
# Train	7473	5654	10000	-	-	-	-
# Eval	1319	3742	2742	5000	933	500	500

Table 8 shows the statistics of all datasets used.

<sup>3</sup><https://huggingface.co/datasets/open-r1/OpenR1-Math-220k>

<sup>4</sup><https://www.kaggle.com/datasets/hemishveeraboina/aime-problem-set-1983-2024>

### Evaluation Details

We construct a unified prompt format for all questions:

**System Prompt:** You are a reasoning expert assistant. Given a question, you will use your reasoning skills to solve the problem.

**User Prompt: [Question]**

Please explain your reasoning process before providing an answer.

During evaluation, we parse the last integer or fraction number in the model’s output and compare it with the ground truth using `math_verify` package. We don’t introduce a strict answer format, as previous work [52] suggest it may hinder exploration.

### Implementation Details

#### Implementation of SAR

We implemented all baselines, as well as our approach building on the VERL open-source framework [39]. The VERL framework provides standard implementations of PPO and GRPO. To implement our approach, we need to calculate the perplexity values for the rollouts given the query ( $\text{ppl}(a|q)$ ) as well as the perplexity of standalone rollouts ( $\text{ppl}(a)$ ).

In most RL algorithms, calculating log probabilities for rollouts are necessary to apply KL penalty. These log-probs can be directly applied to calculate  $\text{ppl}(a|q)$ :

```
1 def compute_ppl(log_probs, mask):
2     """
3     log_probs: Tensor[batch_size, seq_len]
4     mask: Boolean Tensor[batch_size, seq_len],
5           positions of the answer part are 1;
6           positions of query and padding tokens are 0.
7     """
8     sum_log_probs = (log_probs * mask).sum(dim=1)
9     lengths = mask.sum(dim=1)
10    avg_log_probs = sum_log_probs / lengths
11    return torch.exp(-avg_log_probs)
```

The calculation of  $\text{ppl}(a)$  consists of two steps. The first step is to construct the sequence without the user prompt (we still keep the system prompt):

```
1 def construct_empty_question_data(data, tokenizer,
2     q_max_len):
3     """
4     data: DataProto, a standard protocol for data
5           exchange in VERL.
6     data contains "input_ids", "attention_mask" and "
7           position_ids", which are Tensors[batch_size,
8           q_max_len+a_max_len]. Query are left-padded
9           and answers are right-padded, which means
10          the answer always begin at index q_max_len.
11     """
12    empty_q = f"System:{sys_prompt}\nUser:\nAssistant
13            : "
14    empty_q_tokens = tokenizer(empty_q, padding="left
15                               ", max_length=q_max_len)
16
17    batch_size = data["input_ids"].shape[0]
```

```

10 for key in ["input_ids", "attention_mask", "
11     position_ids"]:
12     data[key][:, :q_max_len] = empty_q_tokens[key
13         ].repeat(batch_size, 1)
14
15 last_query_pos = data["position_ids"][:,
16     q_max_len - 1]
17 first_answer_pos = data["position_ids"][:,
18     q_max_len] # the first answer token
19 gap = first_answer_pos - last_query_pos - 1 #
20     this gap should be zero
21 data["position_ids"][:, q_max_len:] -= gap.
22     unsqueeze(-1) # Shift the answer positions
23     backward by this gap so they are continuous
24
25 return data

```

The second step is to calculate log probabilities for the “new” sequences. Combining them, we implement the self-aligned reward:

```

1 def R_SA(data, tokenizer, q_max_len, policy):
2     ppl_qa = compute_ppl(data["log_probs"], data["
3         response_mask"])
4
5     empty_q_data = construct_empty_question_data(data
6         , tokenizer, q_max_len)
7     empty_q_log_probs = policy.calc_log_probs(
8         empty_q_data)
9     ppl_a = compute_ppl(empty_q_log_probs, data["
10         response_mask"])
11
12     return max((ppl_a - ppl_qa) / ppl_a, -1)

```

In conclusion, the self-aligned reward calculation is fully compatible with VERL’s existing optimizations, making it highly efficient for training large models.

### Implementation of GRPO-O1 and GRPO-ER

Signals besides verifiable reward have also been widely studied, where enhancing efficiency is a prevalent direction. In this paper, we consider O1-pruner [28] and Efficient Reasoner (ER) [4], two typical length-based methods as baselines. Since the original works used offline RL algorithms, we made slight changes to the formula in order to adapt to GRPO:

$$R_{\text{GRPO-O1}}(q, a_i, gt) = R_{\text{VR}} + \alpha R_{\text{O1}},$$

$$R_{\text{O1}} = \text{clip} \left( \frac{\text{mean}[\text{len}(a_{1..N})] - \text{len}(a_i)}{\text{len}(a_i)}, -1, 1 \right) \quad (9)$$

$$R_{\text{GRPO-ER}}(q, a_i, gt) = R_{\text{VR}} + \alpha R_{\text{ER}},$$

$$R_{\text{ER}} = \begin{cases} 2\sigma \left( \frac{\text{mean}[\text{len}(\mathbf{A}_{\text{cor}})] - \text{len}(a_i)}{\text{std}[\text{len}(\mathbf{A}_{\text{cor}})] + \epsilon} \right) - 1, & \text{if } R_{\text{VR}} = 1, \\ 0, & \text{otherwise.} \end{cases} \quad (10)$$

where  $\sigma(x) = \frac{1}{1+e^{-x}}$  is the Sigmoid function, and  $\mathbf{A}_{\text{cor}} = \{a | a \in a_{1..N} \wedge R_{\text{VR}}(q, a, gt) = 1\}$  refers to the set of all correct answers.

Table 9: Training configs for PPO and GRPO.

Hyperparameter	PPO	GRPO
Actor learning rate	$1e-6$	
Critic learning rate	$2e-6$	-
train_batch_size <sup>5</sup>	128	
mini_batch_size	64	
micro_batch_size	16	
Training step	500	
Max response length	4096	
Num of rollouts	-	8
Rollout temp ( $\tau$ )	1.0	
KL penalty ( $\beta$ )	$1e-3$	
Advantage clip ( $\epsilon$ )	0.2	

We also provide the pseudocode for calculating  $R_{\text{O1}}$  and  $R_{\text{ER}}$  for reference.

```

1
2 def R_O1(length: List[int], correctness: List[bool]):
3     avg = statistics.mean(length)
4     ol_scores = []
5     for len in length:
6         x = (avg - len) / len
7         x = max(min(x, 1), -1)
8         ol_scores.append(x)
9     return ol_scores
10
11 def R_ER(length: List[int], correctness: List[bool]):
12     if correctness.count(True) <= 1:
13         return [0] * len(length)
14     cor_lens = [len for len, correct in zip(length,
15         correctness) if correct]
16     avg = statistics.mean(cor_lens)
17     std = statistics.stdev(cor_lens)
18     er_scores = []
19     for len, cor in zip(length, correctness):
20         if cor:
21             x = (avg - len) / std
22             x = 2 / (1 + math.exp(-x)) - 1
23             er_scores.append(x)
24         else:
25             er_scores.append(0)
26     return er_scores

```

### Training Settings

Models are trained on 4 NVIDIA H100 80GB GPUs. For PPO and GRPO, we list all training hyperparameters in Table 9. PPO and SA-PPO use the same config; GRPO, GRPO-O1, GRPO-ER and SA-GRPO also use the same config; the only exception is  $\alpha$  in the reward function. The default  $\alpha$  for GRPO-O1 and GRPO-ER are 0.05, and 0.2 for SA-GRPO. Using a different  $\alpha$  will make these algorithms focus more on accuracy or efficiency, as shown in Figure 3. We use vLLM [23] framework for inference. Models are evaluated

<sup>5</sup>The three batch sizes in the table are hyperparameters in the VERL framework. train\_batch\_size is the batch size for sampling rollouts, mini\_batch\_size is the batch size to perform policy updates, and micro\_batch\_size is the batch size for rollouts and back-propagation. Theoretically, micro\_batch\_size won’t affect the training result, so one can set a smaller or larger value depending on the compute resources.

using 1 NVIDIA G100 80GB GPU. By default, the maximum response length is 4096, and greedy decoding is used ( $\tau = 0$ ).

## LLM Annotation Prompts

Below is the prompt to categorize responses into the 4 types in the “Why Self-aligned Reward Works” section:

**System Prompt:** You are an expert reasoner and LLM judge. Given a reasoning problem and an answer, you need to categorize the answer into one of the following categories:

- 1: Correct and concise answer. Small mistakes are acceptable.
- 2: Correct answer, but a bit lengthy, or contains unnecessary steps. Small mistakes are acceptable.
- 3: Partly correct answers, which makes some mistake and fails to reach the final ground truth.
- 4: Completely wrong or irrelevant answers, indicating the model doesn't understand the problem.
- 0: The answer is correct but contains extra content after the answer, like random characters or talking about an irrelevant topic.

**Output Format:**

[thought] Provide your thought process on how you identify the reasoning behaviors. [/thought]  
[answer] One single number, indicating the type of the answer. [/answer]

### The one-shot example:

Query: Making a cake requires 1 cup of flour, 1 cup of sugar, and 2 eggs. Suppose you have 2 cups of flour, 3 cups of sugar, and 6 eggs, how many cakes can you make?  
Ground truth answer: 2

Answer: I shall first determine how many cakes each ingredient can support. Flour allows 2 cakes ( $2/1=2$ ), sugar allows 3 ( $3/1=3$ ), and eggs allow 3 ( $6/2=3$ ). Therefore, I can make at most 3 cakes. Wait a second, the maximum number actually depends on the low-resource ingredient. In this case, it's flour, which only supports 2 cakes. The final answer is 2.

[thought] The answer correctly understands the problem and provides a concise solution. It identifies the limiting ingredient (flour) and calculates the maximum number of cakes that can be made based on that. The reasoning is clear and follows a logical sequence. [/thought]  
[answer] 1 [/answer]

Below is the prompt to identify reasoning behaviors in Analysis Section:

**System Prompt:** You are an expert reasoner and LLM judge. Given a reasoning problem and an answer, you need to identify the reasoning behaviors exhibited in the solution. There are four reasoning behaviors that requires identification:

1. **Verification:** This behavior involves systematically checking intermediate results or assumptions to ensure they are correct.

- Example: "Let's verify this result by checking if the two expressions are always equal."

2. **Backtracking:** This behavior occurs when the model explicitly revises its approach after detecting an error or realizing that the current path won't lead to the solution.

- Example: "The assumption that  $a > 0$  doesn't work, we need to try something else."

3. **Subgoal Setting:** This behavior involves breaking down a complex problem into smaller, more manageable steps.

- Example: "The first step is to find the range of a and b respectively."

4. **Enumeration:** Solving problems by exhaustively considering multiple cases or possibilities.

- Example: "After investigating 7 days of a week, only Wednesday and Friday satisfies the condition."

**Guidelines for Identification:**

- It is possible for a single solution to exhibit multiple behaviors or none of them.

- Your annotation should be based on the reasoning process, not just the final answer.

- You should only count a behavior if it concretely contributes to the reasoning process.

**Output Format:**

[thought] Provide your thought process on how you identify the reasoning behaviors. [/thought]  
[answer] Behavior(s) separated by commas, or "None" if no behavior is identified. [/answer]

## Additional Experiments

### Training Trajectories

In this section, we show the training trajectories of GRPO, SA-GRPO and SA-GRPO without verifiable reward (a baseline setting discussed in Analysis Section) in Figure 4. We report the verifiable reward, self-aligned reward<sup>6</sup>, and average response length at each step.

<sup>6</sup>While  $R_{SA}$  isn't used in training the GRPO model, we still calculate and record the values for comparison.

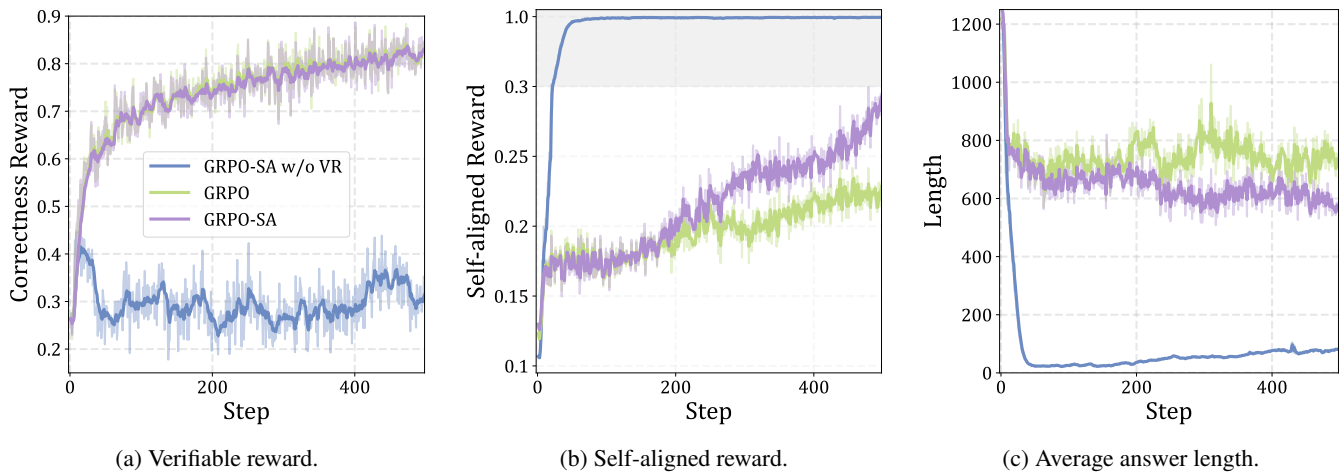


Figure 4: Training plots for Qwen3-4B.

## Cases: Comparison of GRPO and SA-GRPO

### Case 1: Improved Conciseness

**Question:** Let  $r, s,$  and  $t$  be solutions to the equation

$$2x^3 - 3x^2 + 4x - 1 = 0.$$

Find  $r^2 + s^2 + t^2$ .

**Ground truth:**  $\boxed{-\frac{7}{4}}$

#### Qwen3-1.7B + GRPO Answer:

To find  $r^2 + s^2 + t^2$  where  $r, s,$  and  $t$  are the solutions to the equation  $2x^3 - 3x^2 + 4x - 1 = 0$ , we can use the relationships between the roots and the coefficients of the polynomial. Specifically, we can use the identity:

$$r^2 + s^2 + t^2 = (r + s + t)^2 - 2(rs + rt + st)$$

First, we need to find the values of  $r + s + t, rs + rt + st,$  and  $rst$  using the coefficients of the polynomial.

The general form of a cubic polynomial is  $ax^3 + bx^2 + cx + d = 0$ . For our polynomial  $2x^3 - 3x^2 + 4x - 1 = 0$ , we have:

$$a = 2, \quad b = -3, \quad c = 4, \quad d = -1$$

Using Vieta's formulas, we know:

$$r + s + t = -\frac{b}{a} = -\frac{-3}{2} = \frac{3}{2},$$

$$rs + rt + st = \frac{c}{a} = \frac{4}{2} = 2,$$

$$rst = -\frac{d}{a} = -\frac{-1}{2} = \frac{1}{2}.$$

Now, we can substitute these values into the identity for  $r^2 + s^2 + t^2$ :

$$r^2 + s^2 + t^2 = (r + s + t)^2 - 2(rs + rt + st)$$

Substituting the values we found:

$$r^2 + s^2 + t^2 = \left(\frac{3}{2}\right)^2 - 2^2$$

Calculating each term:

$$\left(\frac{3}{2}\right)^2 = \frac{9}{4}, \quad 2^2 = 4$$

So,

$$r^2 + s^2 + t^2 = \frac{9}{4} - 4$$

To subtract these, we need a common denominator:

$$4 = \frac{16}{4}$$

Thus,

$$r^2 + s^2 + t^2 = \frac{9}{4} - \frac{16}{4} = \frac{9 - 16}{4} = \frac{-7}{4}$$

Therefore, the answer is:

$$\boxed{-\frac{7}{4}}$$

#### Qwen3-1.7B + SA-GRPO Answer:

To solve this problem, we can use Vieta's formulas, which relate the coefficients of the polynomial to sums and products of its roots. For the polynomial

$$2x^3 - 3x^2 + 4x - 1 = 0,$$

the roots are  $r, s,$  and  $t$ .

Vieta's formulas tell us:

- The sum of the roots  $r + s + t = -\frac{-3}{2} = \frac{3}{2}$ .
- The sum of the product of the roots taken two at a time  $rs + rt + st = \frac{4}{2} = 2$ .
- The product of the roots  $rst = -\frac{-1}{2} = \frac{1}{2}$ .

We need to find  $r^2 + s^2 + t^2$ . We can use the identity:

$$r^2 + s^2 + t^2 = (r + s + t)^2 - 2(rs + rt + st).$$

Substituting the values from Vieta's formulas:

$$r^2 + s^2 + t^2 = \left(\frac{3}{2}\right)^2 - 2(2).$$

$$r^2 + s^2 + t^2 = \frac{9}{4} - 4.$$

$$r^2 + s^2 + t^2 = \frac{9}{4} - \frac{16}{4}.$$

$$r^2 + s^2 + t^2 = \frac{9 - 16}{4}.$$

$$r^2 + s^2 + t^2 = \frac{-7}{4}.$$

Therefore, the answer is  $\boxed{\frac{-7}{4}}$ .

### Case 2: Improved Conciseness and Accuracy

**Question:** How many integers  $x$  satisfy

$$|x| + 5 < 7 \quad \text{and} \quad |x - 3| > 2?$$

**Ground truth:**  $\boxed{2}$

#### Qwen3-1.7B + GRPO Answer:

To solve the problem, we need to find the integers  $x$  that satisfy both inequalities:

$$1. |x| + 5 < 7 \quad 2. |x - 3| > 2$$

Let's solve each inequality step by step.

Step 1: Solve the first inequality  $|x| + 5 < 7$

Subtract 5 from both sides:

$$|x| < 2$$

This absolute value inequality can be rewritten as:

$$-2 < x < 2$$

So, the integers  $x$  that satisfy this inequality are:

$$x = -1, 0, 1$$

Step 2: Solve the second inequality  $|x - 3| > 2$

This absolute value inequality can be split into two separate inequalities:

$$x - 3 > 2 \quad \text{or} \quad x - 3 < -2$$

Solving these inequalities:

$$x > 5 \quad \text{or} \quad x < 1$$

So, the integers  $x$  that satisfy this inequality are:

$$x = \dots, -2, -1, 0, 1, 2, 3, 4, 5, \dots$$

Step 3: Find the intersection of the solutions from both inequalities

We need to find the integers that satisfy both  $-2 < x < 2$  and  $x > 5$  or  $x < 1$ .

From the first inequality, the integers are:

$$x = -1, 0, 1$$

From the second inequality, the integers are:

$$x = \dots, -2, -1, 0, 1, 2, 3, 4, 5, \dots$$

The intersection of these two sets of integers is:

$$x = -1, 0, 1$$

Step 4: Count the number of integers in the intersection

The integers that satisfy both inequalities are:

$$x = -1, 0, 1$$

There are 3 integers in this set.

Therefore, the number of integers  $x$  that satisfy both inequalities is:

$$\boxed{3}$$

#### Qwen3-1.7B + SA-GRPO Answer:

To solve the problem, we need to analyze the two inequalities separately and then find the intersection of the solutions.

Step 1: Solve the first inequality  $|x| + 5 < 7$

Subtract 5 from both sides:

$$|x| < 2$$

This means that  $x$  must be within 2 units of 0:

$$-2 < x < 2$$

Step 2: Solve the second inequality  $|x - 3| > 2$

This inequality can be broken into two separate inequalities:

$$x - 3 > 2 \quad \text{or} \quad x - 3 < -2$$

Solving these, we get:

$$x > 5 \quad \text{or} \quad x < 1$$

Step 3: Find the intersection of the solutions

We need to find the values of  $x$  that satisfy both inequalities:

$$-2 < x < 2$$

$$x > 5 \quad \text{or} \quad x < 1$$

The intersection of these intervals is:

$$-2 < x < 1$$

Step 4: Identify the integers in the interval  $-2 < x < 1$

The integers in this interval are:

$$-1, 0$$

Step 5: Count the integers

There are 2 integers that satisfy both inequalities.

Therefore, the number of integers  $x$  that satisfy the given conditions is:

$$\boxed{2}$$

## References

- [1] Abdin, M.; Aneja, J.; Awadalla, H.; Awadallah, A.; Awan, A. A.; Bach, N.; Bahree, A.; Bakhtiari, A.; Bao, J.; Behl, H.; et al. 2024. Phi-3 technical report: A highly capable language model locally on your phone, 2024. URL <https://arxiv.org/abs/2404.14219>, 2: 6.
- [2] Agarwal, S.; Zhang, Z.; Yuan, L.; Han, J.; and Peng, H. 2025. The unreasonable effectiveness of entropy minimization in llm reasoning. *arXiv preprint arXiv:2505.15134*.
- [3] Aggarwal, P.; and Welleck, S. 2025. L1: Controlling how long a reasoning model thinks with reinforcement learning. *arXiv preprint arXiv:2503.04697*.
- [4] Arora, D.; and Zanette, A. 2025. Training language models to reason efficiently. *arXiv preprint arXiv:2502.04463*.
- [5] Chen, C.; Liu, K.; Chen, Z.; Gu, Y.; Wu, Y.; Tao, M.; Fu, Z.; and Ye, J. 2024. INSIDE: LLMs' internal states retain the power of hallucination detection. *arXiv preprint arXiv:2402.03744*.
- [6] Chen, W.; Yuan, J.; Jin, T.; Ding, N.; Chen, H.; Liu, Z.; and Sun, M. 2025. The Overthinker's DIET: Cutting Token Calories with Difficulty-Aware Training. *arXiv preprint arXiv:2505.19217*.
- [7] Chen, X.; Li, G.; Wang, Z.; Jin, B.; Qian, C.; Wang, Y.; Wang, H.; Zhang, Y.; Zhang, D.; Zhang, T.; et al. 2025. Rm-r1: Reward modeling as reasoning. *arXiv preprint arXiv:2505.02387*.
- [8] Cobbe, K.; Kosaraju, V.; Bavarian, M.; Chen, M.; Jun, H.; Kaiser, L.; Plappert, M.; Tworek, J.; Hilton, J.; Nakano, R.; et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- [9] Cuadron, A.; Li, D.; Ma, W.; Wang, X.; Wang, Y.; Zhuang, S.; Liu, S.; Schroeder, L. G.; Xia, T.; Mao, H.; et al. 2025. The danger of overthinking: Examining the reasoning-action dilemma in agentic tasks. *arXiv preprint arXiv:2502.08235*.
- [10] Cui, G.; Zhang, Y.; Chen, J.; Yuan, L.; Wang, Z.; Zuo, Y.; Li, H.; Fan, Y.; Chen, H.; Chen, W.; et al. 2025. The entropy mechanism of reinforcement learning for reasoning language models. *arXiv preprint arXiv:2505.22617*.
- [11] Friedland, G.; Huang, X.; Cui, Y.; Kapoor, V.; Khetan, A.; and Das, S. 2024. PPLqa: An Unsupervised Information-Theoretic Quality Metric for Comparing Generative Large Language Models. *arXiv preprint arXiv:2411.15320*.
- [12] Gandhi, K.; Chakravarthy, A.; Singh, A.; Lile, N.; and Goodman, N. D. 2025. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars. *arXiv preprint arXiv:2503.01307*.
- [13] Geng, J.; Cai, F.; Wang, Y.; Koepl, H.; Nakov, P.; and Gurevych, I. 2023. A survey of confidence estimation and calibration in large language models. *arXiv preprint arXiv:2311.08298*.
- [14] Guo, D.; Yang, D.; Zhang, H.; Song, J.; Zhang, R.; Xu, R.; Zhu, Q.; Ma, S.; Wang, P.; Bi, X.; et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.
- [15] Han, P.; Liu, Z.; and You, J. 2025. ToMAP: Training Opponent-Aware LLM Persuaders with Theory of Mind. *arXiv preprint arXiv:2505.22961*.
- [16] Han, P.; Qian, C.; Chen, X.; Zhang, Y.; Zhang, D.; and Ji, H. 2025. SafeSwitch: Steering Unsafe LLM Behavior via Internal Activation Signals. *arXiv preprint arXiv:2502.01042*.
- [17] Han, T.; Wang, Z.; Fang, C.; Zhao, S.; Ma, S.; and Chen, Z. 2024. Token-budget-aware llm reasoning. *arXiv preprint arXiv:2412.18547*.
- [18] Hendrycks, D.; Burns, C.; Kadavath, S.; Arora, A.; Basart, S.; Tang, E.; Song, D.; and Steinhardt, J. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.
- [19] Jaeck, A.; Kalai, A.; Lerer, A.; Richardson, A.; El-Kishky, A.; Low, A.; Helyar, A.; Madry, A.; Beutel, A.; Carney, A.; et al. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.
- [20] Ji, Z.; Yu, T.; Xu, Y.; Lee, N.; Ishii, E.; and Fung, P. 2023. Towards mitigating LLM hallucination via self reflection. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, 1827–1843.
- [21] Kang, Y.; Sun, X.; Chen, L.; and Zou, W. 2025. C3ot: Generating shorter chain-of-thought without compromising effectiveness. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 24312–24320.
- [22] Kulkarni, M.; Tangarajan, P.; Kim, K.; and Trivedi, A. 2024. Reinforcement learning for optimizing rag for domain chatbots. *arXiv preprint arXiv:2401.06800*.
- [23] Kwon, W.; Li, Z.; Zhuang, S.; Sheng, Y.; Zheng, L.; Yu, C. H.; Gonzalez, J. E.; Zhang, H.; and Stoica, I. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.
- [24] Lei, S.; Cheng, Z.; Jia, K.; and Tao, D. 2025. Revisiting LLM Reasoning via Information Bottleneck. *arXiv preprint arXiv:2507.18391*.
- [25] Li, J.; Beeching, E.; Tunstall, L.; Lipkin, B.; Soletskyi, R.; Huang, S.; Rasul, K.; Yu, L.; Jiang, A. Q.; Shen, Z.; et al. 2024. NuminaMath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. *Hugging Face repository*, 13(9): 9.
- [26] Liu, Z.; Chen, C.; Li, W.; Qi, P.; Pang, T.; Du, C.; Lee, W. S.; and Lin, M. 2025. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*.
- [27] Liu, Z.; Han, P.; Yu, H.; Li, H.; and You, J. 2025. TimeR1: Towards Comprehensive Temporal Reasoning in LLMs. *arXiv preprint arXiv:2505.13508*.

- [28] Luo, H.; Shen, L.; He, H.; Wang, Y.; Liu, S.; Li, W.; Tan, N.; Cao, X.; and Tao, D. 2025. O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning. *arXiv preprint arXiv:2501.12570*.
- [29] Ma, W.; He, J.; Snell, C.; Griggs, T.; Min, S.; and Zaharia, M. 2025. Reasoning models can be effective without thinking. *arXiv preprint arXiv:2504.09858*.
- [30] Ma, X.; Wan, G.; Yu, R.; Fang, G.; and Wang, X. 2025. Cot-valve: Length-compressible chain-of-thought tuning. *arXiv preprint arXiv:2502.09601*.
- [31] Mirzadeh, I.; Alizadeh, K.; Shahrokhi, H.; Tuzel, O.; Bengio, S.; and Farajtabar, M. 2024. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. *arXiv preprint arXiv:2410.05229*.
- [32] Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744.
- [33] Parmar, M.; Patel, N.; Varshney, N.; Nakamura, M.; Luo, M.; Mashetty, S.; Mitra, A.; and Baral, C. 2024. Logicbench: Towards systematic evaluation of logical reasoning ability of large language models. *arXiv preprint arXiv:2404.15522*.
- [34] Renze, M.; and Guven, E. 2024. The benefits of a concise chain of thought on problem-solving in large language models. In *2024 2nd International Conference on Foundation and Large Language Models (FLLM)*, 476–483. IEEE.
- [35] Renze, M.; and Guven, E. 2024. Self-reflection in llm agents: Effects on problem-solving performance. *arXiv preprint arXiv:2405.06682*.
- [36] Saparov, A.; and He, H. 2022. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. *arXiv preprint arXiv:2210.01240*.
- [37] Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.
- [38] Shao, Z.; Wang, P.; Zhu, Q.; Xu, R.; Song, J.; Bi, X.; Zhang, H.; Zhang, M.; Li, Y.; Wu, Y.; et al. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.
- [39] Sheng, G.; Zhang, C.; Ye, Z.; Wu, X.; Zhang, W.; Zhang, R.; Peng, Y.; Lin, H.; and Wu, C. 2025. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, 1279–1297.
- [40] Snell, C.; Lee, J.; Xu, K.; and Kumar, A. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*.
- [41] Su, J.; Healey, J.; Nakov, P.; and Cardie, C. 2025. Between underthinking and overthinking: An empirical study of reasoning length and correctness in llms. *arXiv preprint arXiv:2505.00127*.
- [42] Sui, Y.; Chuang, Y.-N.; Wang, G.; Zhang, J.; Zhang, T.; Yuan, J.; Liu, H.; Wen, A.; Zhong, S.; Chen, H.; et al. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*.
- [43] Team, G.; Kamath, A.; Ferret, J.; Pathak, S.; Vieillard, N.; Merhej, R.; Perrin, S.; Matejovicova, T.; Ramé, A.; Rivière, M.; et al. 2025. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.
- [44] Team, K.; Du, A.; Gao, B.; Xing, B.; Jiang, C.; Chen, C.; Li, C.; Xiao, C.; Du, C.; Liao, C.; et al. 2025. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*.
- [45] Veeraboina, H. 2024. AIME Problem Set (1983–2024). <https://www.kaggle.com/datasets/hemishveeraboina/aime-problem-set-1983-2024>.
- [46] Wang, X.; Wei, J.; Schuurmans, D.; Le, Q.; Chi, E.; Narang, S.; Chowdhery, A.; and Zhou, D. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.
- [47] Wen, B.; Xu, C.; Wolfe, R.; Wang, L. L.; Howe, B.; et al. 2024. Mitigating overconfidence in large language models: A behavioral lens on confidence estimation and calibration. In *NeurIPS 2024 Workshop on Behavioral Machine Learning*.
- [48] Xia, H.; Leong, C. T.; Wang, W.; Li, Y.; and Li, W. 2025. Tokenskip: Controllable chain-of-thought compression in llms. *arXiv preprint arXiv:2502.12067*.
- [49] Yang, A.; Li, A.; Yang, B.; Zhang, B.; Hui, B.; Zheng, B.; Yu, B.; Gao, C.; Huang, C.; Lv, C.; et al. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.
- [50] Yeo, E.; Tong, Y.; Niu, M.; Neubig, G.; and Yue, X. 2025. Demystifying long chain-of-thought reasoning in llms. *arXiv preprint arXiv:2502.03373*.
- [51] Yu, P.; Xu, J.; Weston, J.; and Kulikov, I. 2024. Distilling system 2 into system 1. *arXiv preprint arXiv:2407.06023*.
- [52] Zeng, W.; Huang, Y.; Liu, Q.; Liu, W.; He, K.; Ma, Z.; and He, J. 2025. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. *arXiv preprint arXiv:2503.18892*.
- [53] Zhang, Y.; Zhang, Z.; Guan, H.; Cheng, Y.; Duan, Y.; Wang, C.; Wang, Y.; Zheng, S.; and He, J. 2025. No Free Lunch: Rethinking Internal Feedback for LLM Reasoning. *arXiv preprint arXiv:2506.17219*.
- [54] Zhao, X.; Kang, Z.; Feng, A.; Levine, S.; and Song, D. 2025. Learning to reason without external rewards. *arXiv preprint arXiv:2505.19590*.
- [55] Zheng, C.; Liu, S.; Li, M.; Chen, X.-H.; Yu, B.; Gao, C.; Dang, K.; Liu, Y.; Men, R.; Yang, A.; Zhou, J.; and Lin, J. 2025. Group Sequence Policy Optimization. *arXiv:2507.18071*.