DRA-GRPO: Exploring Diversity-Aware Reward Adjustment for R1-Zero-Like Training of Large Language Models

Anonymous ACL submission

Abstract

Recent advances in reinforcement learning for language model post-training, such as Group Relative Policy Optimization (GRPO), have shown promise in low-resource settings. However, GRPO typically relies on solution-level and scalar reward signals that fail to capture the semantic diversity among sampled completions. This leads to what we identify as a diversity-quality inconsistency, where distinct reasoning paths may receive indistinguishable rewards. To address this limitation, we propose *Diversity-aware Reward Adjustment* (DRA), a method that explicitly incorporates semantic diversity into the reward computation. DRA uses Submodular Mutual Information (SMI) to downweight redundant completions and amplify rewards for diverse ones. This encourages better exploration during learning, while maintaining stable exploitation of high-quality samples. Our method integrates seamlessly with both GRPO and its variant DR. GRPO, resulting in *DRA-GRPO* and *DGA-DR. GRPO*. We evaluate our method on five mathematical reasoning benchmarks and find that it outperforms recent strong baselines. It achieves state-of-the-art performance with an average accuracy of 58.2%, using only 7,000 fine-tuning samples and a total training cost of approximately $55.

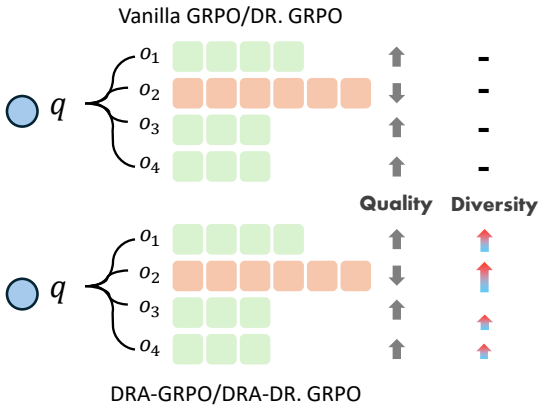

Figure 1: Comparison between vanilla GRPO (**Top**) and GRPO with our diversity-aware reward adjustment (DRA) (**Bottom**). While vanilla GRPO relies solely on scalar solution-level rewards (quality), our method adjusts these rewards using semantic diversity signals computed among sampled completions within a group. This encourages more effective exploration and reduces redundancy during reinforcement learning.

1 Introduction

Recent advancements in large language models (LLMs) post-training have been significantly shaped by DeepSeek-R1-Zero (Guo et al., 2025), which proposes a novel R1-Zero training framework. Departing from traditional pipelines that rely on supervised fine-tuning (SFT) as a prerequisite, this method employs reinforcement learning (RL) directly on base LLMs. This success is primarily attributed to the Group Relative Policy Optimization (GRPO) algorithm (Shao et al., 2024). Under the limited memory and computational resources, we note that RL-based fine-tuning has shown highly promising results for small language models (Dang and Ngo, 2025; Luo et al., 2025; Team, 2025). Unlike Proximal Policy Optimization (PPO), GRPO eliminates the need for a separate critic network. Instead, it evaluates the advantages (quality of actions) based on the relative performance of multiple sampled outputs (completions) for a given prompt. The advantage of each completion is calculated by normalizing its reward relative to group performance statistics, i.e., mean and standard deviation.

Despite its promise, GRPO and its variants (e.g,. DR. GRPO (Liu et al., 2025)) typically rely on reward signals that offer only scalar, solution-level judgments (such as correctness), without accounting for the diversity of reasoning paths that may lead to the same solution. As a result, *semantically distinct completions, whether correct or incorrect, can receive (nearly) identical rewards, producing indistinguishable advantage estimates that fail to reflect meaningful differences in reasoning.* See

examples for this fact in Appendix **H**. This limitation is especially crucial in resource-constrained settings, where only a few completions can be sampled per prompt, often failing to capture the full range of plausible reasoning paths. In such scenarios, the training signal primarily favors exploitation by reinforcing high-reward outputs, while offering limited guidance for exploring alternative, yet potentially valid, reasoning paths. A concrete analogy is a teacher grading students who all solve a math problem correctly and receive full marks. Although the outcomes are accurate, the evaluation overlooks the variety of methods the students may have employed, which could provide deeper insights into their understanding and problem-solving processes. A similar limitation arises even when the answers are incorrect, as students (or models) may still demonstrate valuable, distinct reasoning approaches that are indistinguishably penalized under such scalar rewards.

To address the limitations of original reward signals in capturing reasoning diversity, we propose *Diversity-aware Reward Adjustment* (DRA), a novel method that explicitly models semantic diversity among sampled completions during learning. *To the best of our knowledge, this is the first approach to consider diversity-aware reward shaping directly into the training process of GRPO.* DRA reweights each completion's reward based on its semantic similarity to others in the group, assigning higher importance to diverse completions while reducing the influence of redundant ones. The conceptual comparison with vanilla GRPO is shown in Fig. 1. Our method is implemented using Submodular Mutual Information (SMI), instantiated with a Graph-Cut function over embedding similarities. Our method integrates seamlessly with GRPO and its variant DR. GRPO, which we refer to as *DRA-GRPO* and *DRA-DR. GRPO*, respectively. Extensive evaluations on five mathematical reasoning benchmarks demonstrate the effectiveness of our approach in low-resource settings, i.e., fine-tuning a small model (1.5B) with only 7,000 samples.

2 Method

Preliminary. We briefly review the Group Relative Policy Optimization (GRPO) algorithm (Shao et al., 2024), as employed in (DeepSeek-AI, 2025). Language model generation is formulated as a token-level Markov Decision Process (MDP). At each generation step t, the state s_t is the concatenation of the input question and the partial output sequence generated thus far, denoted as $s_t = \mathbf{q}; \mathbf{o}_{<t}$. The policy $\pi_\theta(\cdot \mid s_t)$ selects the next token o_t from the vocabulary $\mathcal{A}$, inducing a deterministic transition to the next state $s_{t+1} = s_t; [o_t]$. Generation begins by sampling an initial state $s_1 = \mathbf{q} \sim p_\mathcal{Q}$ from the distribution over input questions, and terminates either upon generation of the special `[eos]` token or when the token budget is exhausted. GRPO proposes to sample a group of responses $\mathcal{C} = \{\mathbf{o}_1, \ldots, \mathbf{o}_G\}$ per question and compute their returns $\mathbf{R} = \{\{R(\mathbf{q}, \mathbf{o}_1), \ldots, \{R(\mathbf{q}, \mathbf{o}_G)\}$. Below, we present the GRPO objective, omitting the KL-divergence term for clarity.

$$\mathcal{J}_{GRPO}(\pi_\theta) = \mathbb{E}_{\mathbf{q}\sim p_\mathcal{Q}, \{\mathbf{o}_i\}_{i=1}^{G} \sim \pi_{\theta_{old}}(\cdot|\mathbf{q})} \frac{1}{G}\sum_{i=1}^{G}\frac{1}{|\mathbf{o}_i|}\sum_{t=1}^{|\mathbf{o}_i|}\left\{\min\left[\frac{\pi_\theta(o_{i,t}|\mathbf{q},\mathbf{o}_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|\mathbf{q},\mathbf{o}_{i,<t})}\hat{A}_{i,t},\right.\right. \left.\left.\text{clip}\left(\frac{\pi_\theta(o_{i,t}|\mathbf{q},\mathbf{o}_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|\mathbf{q},\mathbf{o}_{i,<t})}, 1-\epsilon, 1+\epsilon\right)\hat{A}_{i,t}\right]\right\}, \tag{1}$$

where $\hat{A}_{i,t}$ denotes the advantage function computed by:

$$\hat{A}_{i,t} = \frac{R(\mathbf{q},\mathbf{o}_i) - \text{mean}(\{R(\mathbf{q},\mathbf{o}_1),\ldots,R(\mathbf{q},\mathbf{o}_G)\})}{\text{std}(\{R(\mathbf{q},\mathbf{o}_1),\ldots,R(\mathbf{q},\mathbf{o}_G)\})}. \tag{2}$$

A more recent work DR. GRPO (Liu et al., 2025) proposes to remove the terms $\frac{1}{|\mathbf{o}_i|}$ and $\text{std}(\cdot)$ in Eqs. 1 and 2, to improve token efficiency.

Diversity-Quality Inconsistency. Despite this, both algorithms evaluate a group of independently sampled completions $\pi_{\theta_{old}}$ and reward signals typically capture only solution-level correctness (see Appendix A), providing a sparse scalar judgment for each completion. However, this scalar reward (quality) overlooks the diverse reasoning paths that can yield identical or similar outcomes, resulting in what we term *Diversity-Quality Inconsistency*.

While we illustrate this issue through some examples in Appendix H, we further empirically validate our hypothesis that reward alone fails to reflect the underlying variability in reasoning strategies. To this end, we compare the structural dissimilarity of completions, measured via embedding distances, with their reward differences. Specifically, we use Spearman's rank correlation to assess the monotonic relationship between reward difference and semantic distance across sampled completions, i.e., more semantically different completions *tend* to have more divergent rewards. This non-parametric metric is well-suited for capturing rank-level agreement without assuming linearity, and allows us to

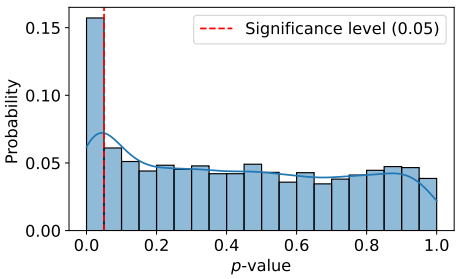

Figure 2: Distribution of p-values from Spearman's rank correlation between completion quality and semantic diversity. The test is conducted for every prompt.

quantify whether reward gradients align with the semantic diversity present in the group of completions. As shown in Fig. 2, we observe that for the majority of prompts (over 80%), the reward values assigned to their completions exhibit no statistical correlation (i.e., p-value> 0.05) with the semantic diversity, which highlights the necessity to explicitly characterize the inherent diversity among completions. Please refer to Appendix B for more details and results for this investigation.

Diversity-aware Reward Adjustment. To address this, we propose to reweight each sample's reward based on its relative diversity/redundancy within the group: completions that are more distinct from the rest are assigned higher importance, while redundant samples are downweighted. To this end, we propose to replace $R(\mathbf{q}, \mathbf{o}_i)$ with our diversity-aware adjusted reward $\tilde{R}(\mathbf{q}, \mathbf{o}_i)$ in Eq. 2 as:

$$\tilde{R}(\mathbf{q}, \mathbf{o}_i) = \frac{R(\mathbf{q}, \mathbf{o}_i)}{1 + \texttt{SMI}(\{\mathbf{o}_i\}, \mathcal{C} \setminus \{\mathbf{o}_i\})}, \quad (3)$$

where $\texttt{SMI}(\{\mathbf{o}_i\}, \mathcal{C} \setminus \{\mathbf{o}_i\})$ denotes the *Submodular Mutual Information* (SMI) between query completion $\mathbf{o}_i$ and the remaining completions denotes as $\mathcal{C} \setminus \{\mathbf{o}_i\}$. Submodular functions, with their diminishing returns property, naturally model diversity and redundancy. SMI quantifies the shared information between sets under a submodular function (Iyer et al., 2021a,b). We instantiate SMI using the Graph-Cut function over a similarity kernel $s(\cdot, \cdot)$ presented as

$$\texttt{SMI}(\{\mathbf{o}_i\}, \mathcal{C} \setminus \{\mathbf{o}_i\}) = \sum_{j \in \mathcal{C} \setminus \{\mathbf{o}_i\}} s(\mathbf{o}_i, j), \quad (4)$$

where we adopt the assumption that $s(\mathbf{o}_i, j) = s(j, \mathbf{o}_i)$. It measures the total symmetric similarity between $\mathbf{o}_i$ and the remaining elements. In this work, we use an extra small pretrained model to get the embedding for each completion. Due to submodularity, this formulation captures diminishing redundancy: elements more similar to the set contribute less marginal information. Thus, Graph-Cut SMI provides a principled measure of $\mathbf{o}_i$'s relative redundancy (high value) or diversity (low value) within the group. In the context of reward adjustment in Eq. 3, we assign a more redundant completion with a lower weight to its corresponding reward and a diverse completion a higher weight. We use cosine similarity as the kernel $s(\cdot)$, enabling efficient computation of the SMI via a precomputed similarity matrix (See Appendix D for more discussion). This results in a total computational complexity of $\mathcal{O}(G^2)$ for a group of size G. A Pytorch-like algorithmic summary that involved this fact is provided in Appendix C.

3 Experiment

3.1 Experimental Setup

Training Dataset. We adopt a high-quality dataset curated by (Dang and Ngo, 2025). This dataset consists of only **7000** samples refined and selected from the s1 dataset (Muennighoff et al., 2025) and the DeepScaleR dataset (Luo et al., 2025) with mixed problem difficulties.

Evaluation Dataset. We select five popular mathematical reasoning benchmarks (see Appendix E).

Baselines. We evaluate our approach against various baseline models. The general-purpose large model: (i) `Llama-3.1-70B-Instruct` (AI, 2024a) and (ii) `o1-preview` (AI, 2024b). For mathematics-focused 7B models, we consider: (iii) `Qwen-2.5-Math-7B-Instruct` (Yang et al., 2024); (iv) `rStar-Math-7B` (Guan et al., 2025); (v) `Eurus-2-7B-PRIME` (Cui et al., 2025); and (vi) `Qwen2.5-7B-SimpleRL` (Zeng et al., 2025). Lastly, for mathematics-focused 1.5B models, instead of our base model, we include (vii) `DeepScaleR-1.5B-Preview` (Luo et al., 2025), (viii) `Still-3-1.5B-Preview` (Team, 2025), and (ix) Open-RS (Dang and Ngo, 2025).

Implementation. As proof-of-concept, we adopt `DeepSeek-R1-Distill-Qwen-1.5B` (DeepSeek-AI, 2025) as our base model for training due to its balance of efficiency and reasoning potential (Dang and Ngo, 2025). We use 4x NVIDIA A100 40GB GPUs. Please refer to Appendix F for the details of hyperparameters.

3.2 Empirical Analysis

Main Results in Accuracy. As shown in Table 1, our primary observation is that integrating our method with DR. GRPO outperforms all baseline approaches across various parameter scales,

Table 1: Zero-shot pass@1 performance across benchmarks. Dashes (–) denote unavailable official scores. '†' denotes our implementation. Scores for `o1-preview` are sourced from AI, 2024b; others from Dang and Ngo, 2025. We also report the number of samples used to fine-tune the small models.

Model	Fine-tuning Samples	AIME24	MATH-500	AMC23	Minerva	OlympiadBench	Avg.
General Models							
Llama-3.1-70B-Instruct		16.7	64.6	30.1	35.3	31.9	35.7
o1-preview		44.6	85.5	–	–	–	–
7B Models							
Qwen-2.5-Math-7B-Instruct		13.3	79.8	50.6	34.6	40.7	43.8
rStar-Math-7B		26.7	78.4	47.5	–	47.1	–
Eurus-2-7B-PRIME		26.7	79.2	57.8	38.6	42.1	48.9
Qwen2.5-7B-SimpleRL		26.7	82.4	62.5	39.7	43.3	50.9
1.5B Models							
DeepSeek-R1-Distill-Qwen-1.5B	***Base Model***	28.8	82.8	62.9	26.5	43.3	48.9
Still-3-1.5B-Preview	30,000	32.5	84.4	66.7	29.0	45.4	51.6
DeepScaleR-1.5B-Preview	40,000	43.1	87.8	73.6	30.2	50.0	57.0
Open-RS1	18,615	30.0	83.8	70.0	29.0	52.4	53.0
Open-RS2	7,000	30.0	85.4	80.0	30.5	52.4	55.7
Open-RS3	7,000	46.7	84.4	72.5	26.8	51.3	56.3
GRPO†	7,000	30.0	86.0	72.5	32.4	53.0	54.8
DR. GRPO†	7,000	33.3	83.4	80.0	30.5	52.1	56.0
Our Models							
DRA-GRPO	7,000	36.7	86.2	75.0	32.4	53.0	56.7
DRA-DR. GRPO	7,000	36.7	85.2	85.0	30.5	53.8	**58.2**

achieving an average accuracy of 58.2% across all benchmarks. Notably, it achieves the highest accuracy on both AMC23 (85%) and OlympiadBench (53.8%). When incorporated with GRPO, our method obtains an average accuracy of 56.7%, which is on par with the previous state-of-the-art, `DeepScaleR-1.5B-Preview` (57%). However, our approach requires only 7,000 fine-tuning samples, in contrast to the approximately 40,000 samples used by `DeepScaleR-1.5B-Preview`. These results demonstrate the superiority of our method in low-resource settings, i.e., a small model with 1.5B parameters and limited samples for fine-tuning.

Ablation Study. The ablation results are summarized from Table 1. The main observation is that, compared to the base model `DeepSeek-R1-Distill-Qwen-1.5B`, our methods yield improvements of 7.8% and 9.3% in average accuracy. More importantly, integrating our method with GRPO leads to a 1.9% increase in accuracy compared to using GRPO alone. A similar conclusion can be drawn for DR. GRPO, where our method achieves an average accuracy gain of 2.2% across all benchmarks. We also highlight several notable improvements: our method boosts performance on AIME24 by 6.7% and 3.4% for GRPO and DR. GRPO, respectively, and achieves a 5% gain on AMC23 with DR. GRPO. These results further confirm the effectiveness of our method.

Efficiency. Compared to the vanilla GRPO and DR. GRPO, our method introduces a small overhead due to encoding the completions. As shown in the table following, our method introduces approximately 6% runtime and 1.4% GPU overhead. Notably, under our hardware constraint, i.e., A100 40G, without applying our method, increasing the mini-batch size by even one is not feasible. Therefore, our approach makes more efficient use of the available hardware, and the introduced overhead is relatively minor and unlikely to impact practical deployment.

	Runtime	GPU
Vanilla	∼84s/step	∼38.77GB/device
+ DGA	∼90s/step	∼39.34GB/device

Training Cost. Training for 500 steps takes approximately 12.5 hours on a 1×4 A100 40GB setup, costing an estimated $55, which is on par with Open-RS (2025). See Table S3 of Appendix G for more comparisons with different methods.

Discussion. **We have included additional discussion on model analysis in Appendix I.**

4 Conclusion

In this work, we propose DRA, a pioneering method that improves GRPO-style reinforcement learning by modeling semantic diversity among completions. By reweighting rewards using Submodular Mutual Information, DRA encourages exploration of diverse reasoning paths while maintaining strong performance. This effectively mitigates the exploration-exploitation imbalance caused by scalar rewards. Integrated with GRPO and DR. GRPO, our method achieves state-of-the-art results on five math reasoning benchmarks using only 7,000 training samples, demonstrating its effectiveness in low-resource settings.

Limitations

While our proposed method demonstrates strong empirical performance and efficiency under constrained settings, several limitations remain.

First, due to limited computational resources, we restrict our experiments to small-scale models (1.5B parameters) and small group sizes (e.g., 6 completions per prompt). Although this setup aligns with our motivation to improve low-resource fine-tuning, it may not fully reflect the scalability and behavior of our method in larger models or with denser sampling budgets. Extending DRA to larger-scale scenarios or more diverse decoding strategies remains an open direction.

Second, our implementation relies on precomputed sentence embeddings for diversity measurement, using an external lightweight model. While this design introduces minimal overhead, it assumes that the embedding space sufficiently captures semantic similarity relevant to reward shaping. Future work may explore end-to-end learned embeddings or reward functions more tightly integrated with the policy model.

Additionally, as our analysis includes mathematical problem-solving, we note that our evaluation and interpretations stem from a computer science perspective rather than a formal mathematical one. While we strive for rigor, some analyses may remain intuitive rather than strictly formal (they are intended solely to illustrate example scenarios and do not impact the conclusion and core validity of our method); nonetheless, we have conducted all evaluations to the best of our knowledge and understanding.

Lastly, while Submodular Mutual Information offers a principled way to model redundancy, we have primarily explored the Graph-Cut instantiation. Other SMI variants, such as LogDet (we have some discussion on this in Appendix D), might provide alternative trade-offs between efficiency and expressivity, but remain underexplored in this work as our focus is on establishing a proof-of-concept.

References

Meta AI. 2024a. Introducing llama 3.1: Our most capable models to date. Published on July 23, 2024.

Open AI. 2024b. Introducing openai o1-preview. Published on Dec 12, 2024.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, Jiarui Yuan, Huayu Chen, Kaiyan Zhang, Xingtai Lv, Shuo Wang, Yuan Yao, Xu Han, Hao Peng, Yu Cheng, and 4 others. 2025. Process reinforcement through implicit rewards. *Preprint*, arXiv:2502.01456.

Quy-Anh Dang and Chris Ngo. 2025. Reinforcement learning for reasoning in small llms: What works and what doesn't. *arXiv preprint arXiv:2503.16219*.

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

Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. 2025. rstar-math: Small llms can master math reasoning with self-evolved deep thinking. *Preprint*, arXiv:2501.04519.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Michael Günther, Jackmin Ong, Isabelle Mohr, Alaeddine Abdessalem, Tanguy Abel, Mohammad Kalim Akram, Susana Guzman, Georgios Mastrapas, Saba Sturua, Bo Wang, Maximilian Werk, Nan Wang, and Han Xiao. 2023. Jina embeddings 2: 8192-token general-purpose text embeddings for long documents. *Preprint*, arXiv:2310.19923.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. 2024. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3828–3850, Bangkok, Thailand. Association for Computational Linguistics.

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

Rishabh Iyer, Ninad Khargoankar, Jeff Bilmes, and Himanshu Asanani. 2021a. Submodular combinatorial information measures with applications in machine learning. In *Algorithmic Learning Theory*, pages 722–754. PMLR.

Rishabh Iyer, Ninad Khargonkar, Jeff Bilmes, and Himanshu Asnani. 2021b. Generalized submodular information measures: Theoretical properties, examples, optimization algorithms, and applications. *IEEE Transactions on Information Theory*, 68(2):752–781.

Aitor Lewkowycz, Anders Johan Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski,

Vinay Venkatesh Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. 2022. Solving quantitative reasoning problems with language models. In *Advances in Neural Information Processing Systems*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Li Erran Li, Raluca Ada Popa, and Ion Stoica. 2025. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl. https://github.com/agentica-project/deepscaler. Github.

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

Zach Nussbaum, John Xavier Morris, Andriy Mulyar, and Brandon Duderstadt. 2025. Nomic embed: Training a reproducible long context text embedder. *Transactions on Machine Learning Research*. Reproducibility Certification.

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

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

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. 2020. Trl: Transformer reinforcement learning. https://github.com/huggingface/trl.

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

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

A Reward Function in Mathematical Reasoning

We show some typical reward functions below. These functions often compute the reward based on some simple rules, which fail to explicitly capture the inherent semantic diversity among completions.

Accuracy Reward. This function assigns binary rewards to model completions based on exact agreement with the ground truth solution. It begins by parsing the ground truth using a LaTeX extraction configuration and skips evaluation with a full reward of 1.0 if the solution is unparseable. For valid cases, the model's output is also parsed with normalization settings that enforce clean LaTeX formatting, including handling of boxed expressions and units. The parsed output is compared against the ground truth using a verification function. If they match exactly, the function assigns a reward of 1.0; otherwise, the reward is 0.0.

Cosine (Correctness) Reward. This is an upgraded version of Accuracy Reward. It computes rewards for model completions by evaluating their correctness and scaling the reward based on completion length using a cosine schedule. For each completion, it parses both the model output and the ground truth solution using a LaTeX-aware parsing configuration. If parsing fails for the ground truth, the function assigns a default reward of 1.0 and skips evaluation. Correctness is verified by comparing the parsed outputs. The reward is then determined by a cosine function of the output length relative to a maximum length parameter, encouraging shorter correct answers by assigning them higher rewards and penalizing shorter incorrect ones more heavily.

Format Reward. This function is designed to evaluate a list of completions by checking whether the reasoning process is properly enclosed within `<think>` and `</think>` tags. It defines an internal function `count_tags` that inspects each text for exactly one occurrence of the `\n</think>\n` tag sequence. This is because the opening `<think>` tag is assumed to be present in the system prompt and thus does not need to be counted. The function extracts the content strings from the completions, applies the `count_tags` function to each, and returns a list of floating-point scores. A score of 1.0 is assigned if the proper `</think>` tag format is found exactly once; otherwise, a score of 0.0 is given.

B Investigation on Diversity-Quality Inconsistency

To investigate the relationship between reward signals and reasoning diversity, we conduct an empirical analysis over prompts with multiple sampled completions. For each prompt, we compute pairwise semantic distances between completions using cosine distance over sentence-level embeddings obtained from a pre-trained encoder. In parallel, we compute the absolute differences in scalar reward values assigned to each completion. To measure how well reward differences reflect semantic diversity, we compute Spearman's rank correlation coefficient between the reward distance matrix and the embedding distance matrix for each prompt.

We choose Spearman's rank correlation for three key reasons. First, it is a *non-parametric* statistic, making no assumptions about the linearity or distribution of the underlying variables, an important consideration in our setting, where reward scales and semantic distances may exhibit complex, non-linear relationships. Second, Spearman correlation is based on *rank order*, allowing us to capture monotonic trends in the data, i.e., whether more semantically different completions is likely to have more divergent rewards. Third, it is *robust to scale mismatches* between the two metrics (scalar rewards vs. high-dimensional embeddings), since it evaluates alignment in relative ordering rather than absolute magnitude.

We analyze the distribution of Spearman coefficients across prompts (see Fig. 2) and observe that in the majority of cases, correlation is low or statistically insignificant ($p > 0.05$). This provides strong empirical evidence that reward alone does not capture the semantic diversity of model outputs, a phenomenon we define as the *Diversity-Quality Inconsistency*. These findings motivate the need for training objectives that explicitly model and preserve reasoning diversity in addition to optimizing for correctness.

In the investigation in Fig. 2, we sampled around 3000 prompts with their completions, and we use `jina-embeddings-v2-small-en` as the embedding model. We also show a result by using a different embedding model `nomic-ai/nomic-embed-text-v1.5` (Nussbaum et al., 2025) in Fig. S3. Similarly, for over 80% prompts, their completion diversity and rewards are irrelevant.

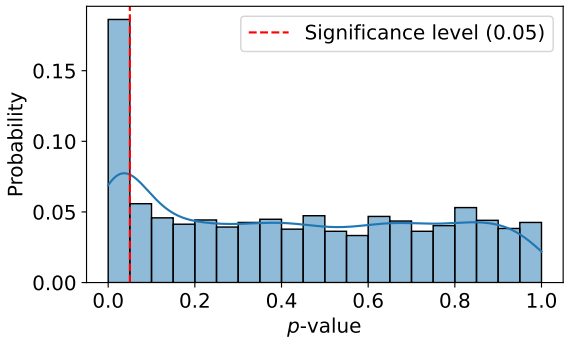

Figure S3: Distribution of p-values from Spearman's rank correlation between completion quality and semantic diversity. Embedding model is `nomic-ai/nomic-embed-text-v1.5`.

C Algorithmic Summary

Please refer to Algorithm 1.

D Reweighting via Precomputed Similarity Matrix

We reiterate that, according to Eqs. 3 and 4 the adjusted reward weight through Graph-cut SMI with cosine similarity can be presented as

$$\begin{aligned}\tilde{R}(\mathbf{q},\mathbf{o}_i) &= \frac{R(\mathbf{q},\mathbf{o}_i)}{1+\texttt{SMI}(\{\mathbf{o}_i\},\mathcal{C}\setminus\{\mathbf{o}_i\})} \\ &= \frac{R(\mathbf{q},\mathbf{o}_i)}{1+\sum_{j\in\mathcal{C}\setminus\{\mathbf{o}_i\}} s(\mathbf{o}_i,j)} \\ &= \frac{R(\mathbf{q},\mathbf{o}_i)}{s(\mathbf{o}_i,\mathbf{o}_i)+\sum_{j\in\mathcal{C}\setminus\{\mathbf{o}_i\}} s(\mathbf{o}_i,j)} \\ &= \frac{R(\mathbf{q},\mathbf{o}_i)}{\sum_{j=0}^{G}\boldsymbol{L}_{ij}}.\end{aligned} \tag{5}$$

We note that $\sum_{j=0}^{G}\boldsymbol{L}_{ij}$ is the sum of the ith row of the similarity matrix $\boldsymbol{L}$, so this can be efficiently computed through `Pytorch` tensor operation trick for all completions as shown in Algorithm 1, i.e., `similarity_matrix.sum(dim=1)`.

Another potential diversity-based SMI is known as logdet SMI (). In our context, it is defined as

$$\begin{aligned}&\texttt{SMI}(\{\mathbf{o}_i\},\mathcal{C}\setminus\{\mathbf{o}_i\}) \\ &= \log\det\boldsymbol{L}_{ii} + \log\det\boldsymbol{L}_{\mathcal{C}\setminus\{\mathbf{o}_i\}} - \log\det\boldsymbol{L}_{\mathcal{C}},\end{aligned} \tag{6}$$

where $\boldsymbol{L}_{ii}=1$ denotes the ith diagonal value of the similarity, and its value is 1 as we use a cosine similarity kernel. $\boldsymbol{L}_{\mathcal{C}\setminus\{\mathbf{o}_i\}}$ and $\boldsymbol{L}_{\mathcal{C}}$ denotes the rows and columns indexed by the set $\mathcal{C}\setminus\{\mathbf{o}_i\}$ and $\mathcal{C}$, respectively. Despite we need a complexity of $\mathcal{O}(G^3)$ to precompute $\log\det\boldsymbol{L}_{\mathcal{C}}$, for each $\mathbf{o}_i$, we need compute $\log\det\boldsymbol{L}_{\mathcal{C}\setminus\{\mathbf{o}_i\}}$, which is obviously less efficient than Graph-cut SMI and would be challenging for scaling.

Algorithm 1 PyTorch Code for diversity-aware reward adjustment.

```
#input: completions_flat: A list of
    completions sampled for each prompt
    (question)

#Some early steps (e.g., compute reward
    for each completion)
...



# Compute embeddings and similarity
    matrix
embeddings = self.sentence_extractor.
    encode(completions_flat)
embeddings = torch.from_numpy(embeddings
    ).to(device)
embeddings = F.normalize(embeddings, p
    =2, dim=1)
similarity_matrix = embeddings @
    embeddings.T


# Compute the weights in parallel for
    all completions
similarity_sums = similarity_matrix.sum(
    dim=1)
diversity_weights = 1.0 / (
    similarity_sums + 1e-6)


#diversity_weights = gather(
    diversity_weights)

# Adjust rewards
rewards = rewards * diversity_weights




# The following step is computing group-
    wise rewards as in the vanilla
    version.

mean_grouped_rewards = rewards.view(-1,
    self.num_generations).mean(dim=1)


...
```

E Dataset

We select five datasets: AIME24 [1], MATH-500 (2023; 2021), AMC23 [2], Minerva (2022) and OlympiadBench (2024).

[1] https://huggingface.co/datasets/AI-MO/aimo-validation-aime

[2] https://huggingface.co/datasets/AI-MO/aimo-validation-amc

F Implementation Detail

We provide our hyperparameters for both GRPO and DR. GRPO is in the table below S2. The implementation is based on the source code of `trl` package from Huggingface (von Werra et al., 2020). The training pipeline and prompt setups are based on `https://github.com/knoveleng/open-rs`. We carefully select a small model, `jina-embeddings-v2-small-en` (Günther et al., 2023), as the completion embedding model, which supports processing a sequence with up to 8192 tokens. The reason is that we want to preserve the efficiency, and we do not tend to adjust original hyperparameters, such as mini-batch size.

Table S2: Hyperparameter Setups for our trainers.

Parameter	Value
General Settings	
bf16	true
use_vllm	true
vllm_device	auto
vllm_enforce_eager	true
vllm_gpu_memory_utilization	0.7
vllm_max_model_len	4608
do_eval	false
Training Configuration	
gradient_accumulation_steps	4
gradient_checkpointing	true
gradient_checkpointing_kwargs	use_reentrant: false
learning_rate	1.0e-06
lr_scheduler_type	cosine_with_min_lr
lr_scheduler_kwargs	min_lr_rate: 0.1
warmup_ratio	0.1
max_steps	500
num_train_epochs	1
per_device_train_batch_size	4
per_device_eval_batch_size	6
Generation Settings	
max_prompt_length	512
max_completion_length	3584
num_generations	6
temperature	0.7
Reward Configuration	
reward_funcs	format, accuracy (cosine)
reward_weights	1.0, 2.0

G Detail of Training Cost

The detail of the training cost is shown in Table S3. Our price is estimated based on standard on-demand GPU pricing from efficient cloud providers (e.g., Lambda Labs).

H Case Study: Examples of Diverse Completions

Here, we present selected examples from the GRPO training process to illustrate the key motivation of our paper. Given the same problem, the LLM can generate diverse answers; however, these answers often receive very similar reward scores. This suggests that learning based on solution-level judgments may fail to distinguish between different reasoning paths. Below, we show two cases that produce correct answers but demonstrate distinct reasoning perspectives and styles. We also present an example where both completions follow coherent reasoning processes but result in incorrect answers.

H.1 Example 1

Question: Fig. S4.
Two Completions: (i) Fig. S5 and (ii) Fig. S6.
Short Analysis. While both outputs correctly arrive at the answer $\boxed{1007}$, they reflect notably different problem-solving **perspectives**.

The first response adopts an empirical, trial-based strategy. Its reward score is 2.103. The model explores specific candidate values of the divisor m, such as 1007, 1008, and 1009, and evaluates the resulting remainders. This process mimics a human-like, exploratory reasoning pattern, i.e., tentative, iterative, and conversational—ultimately identifying that $m = 1008$ yields the maximum remainder 1007. The approach is grounded in pattern recognition and error correction, reflecting a "numerical experimentation" mindset often used by learners.

In contrast, the second response applies a more principled, algebraic perspective. Its reward score is 2.110, almost the same as the first one. The model leverages the mathematical identity that the maximum remainder when dividing a by m is $m-1$, which occurs when $a \equiv -1 \mod m$, or equivalently, when $m \mid (a+1)$. Using this, it reduces the problem to finding the largest proper divisor of 2016. It proceeds to factor 2016 as $2^5 \times 3^2 \times 7$ and identifies $m = 1008$ as the largest valid divisor, yielding $n = 1007$. This response demonstrates structured mathematical reasoning and modular arithmetic awareness, providing a generalizable method beyond this specific example.

H.2 Example 2

Question: Fig. S7.

Table S3: Comparison of training cost by different methods.

Model	rStar-Math-7B	Eurus-2-7B-PRIME	Qwen2.5-7B-SimpleRL	DeepScaleR-1.5B-Preview	Still-3-1.5B-Preview	Open-RS	**Ours**
SFT Data	7.3M	230k	0	0	0	0	0
RM Data	7k	0	0	0	0	0	0
RM Source	None	Eurus-2-7B-SFT	None	None	None	None	None
RL Data	3.647M × 16	150k × 4	8k × 8	40k × 16	30k × 8	7k × 6	7k × 6
Hardware	10×8 H100 80GB + 15×4 A100 40GB	1×8 A100 80GB	4×6 A100 80GB	8× A100 80GB	1×8 A100 80GB	1×4 A40 48GB	1×4 A100 40GB
Time	–	72h	36h	240h	150h	24h	12.5h
Cost Est.	–	$1088	$1633	$3629	$2268	$42	$55

Two Completions: (i) Figs. S8 and S9 and (ii) Figs. S10 and S11.

Short Analysis. Both solutions arrived at the correct final result $\boxed{2419}$, but they differ significantly in structure, presentation, and **reasoning style**.

The first solution exhibits a concise, formula-driven approach, closely resembling traditional mathematical write-ups. It receives a reward score of 2.782. It efficiently identifies the block structure of the sequence, derives the closed-form expression for the total number of terms, and computes the required sum using algebraic manipulation and minimal narrative.

In contrast, the second solution adopts a more exploratory and pedagogical style. It receives a reward score of 2.855. It progressively builds understanding through example-driven reasoning, error-checking, and step-by-step refinements. While more verbose, it mirrors how a human might think aloud while problem-solving, providing greater transparency into the model's internal reasoning.

H.3 Example 3

Question: Fig. S12.

Two Completions: (i) Figs. S13 and S14 and (ii) Figs. S15 and S16.

Short Analysis. In this example, we show that both responses are wrong and receive a reward score of 0.018 and 0.021, respectively. However, after checking their responses, we can easily observe that their different reasoning paths. For example, the first solution tries to use a symbolic-algebraic perspective, which attempts to deduce a closed-form identity. The second solution takes a more complex-number driven view, focusing heavily on manipulating the roots and constants in the general solution. Their errors also happened at different places. the first response correctly obtains the roots $-2 \pm i\sqrt{3}$ to set up the general solution. However, it then wrongly assumes that the expression is constant and evaluates it only at $n = 1$ to conclude the value at $n = 50$. In the second response, the model incorrectly computes the roots of the recurrence as $-2 \pm i$, not the correct characteristic polynomial. Its following derivation is based on these wrong roots.

I Discussion

Exploration-exploitation Balance. Our method integrates exploration-exploitation balancing directly into the policy gradient framework. The base reward encourages exploitation by reinforcing completions that achieve high scores according to reward models. Simultaneously, the diversity weighting term promotes exploration by amplifying the learning signal for semantically novel completions. This exploration is particularly important under resource constraints, where only a limited number of completions can be sampled per prompt, and maximizing coverage of diverse reasoning modes within that budget is crucial. By shaping the per-sample reward before policy updates, our method biases the policy toward both high-reward and diverse behaviors, preventing mode collapse and encouraging broader reasoning strategies.

Ad-hoc VS Post-hoc. There are two common strategies for modeling diversity among generated completions: ad-hoc and post-hoc. Ad-hoc approaches attempt to encourage diversity during generation. For example, by tuning the sampling temperature or other decoding parameters. However, this strategy cannot explicitly model correlations among completions, as they are still sampled independently. Another ad-hoc approach is to generate a larger pool of completions and then select a diverse subset using some point process methods, such as determinantal point processes (DPPs). While DPPs can explicitly model correlations, they act as post-selection mechanisms, which reduce the number of available completions for policy updates, and are less effective when only a small number of completions (e.g., 6 in our case) are generated. In contrast, we adopt a post-hoc strategy that integrates diversity directly into the reward signal. This allows the policy to efficiently adjust learning based on semantic redundancy across completions, even under limited generation budgets.

Question 1

<|begin_of_sentence|>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, and put your final answer within \boxed{}. 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>. Note that respond by English, NOT use other languages.<|User|>Let m be a positive integer less than 2015. Suppose that the remainder when 2015 is divided by m is n. Compute the largest possible value of n.<|Assistant|><think>

Figure S4: Prompt used for Example Question 1.

Answer 1 for Question 1 [Reward: 2.103]

Alright, so I've got this problem here where I need to find the largest possible value of n, which is the remainder when 2015 is divided by some positive integer m, and $m < 2015$. Hmm, okay. Let me think about how to approach this.
First off, I remember that when you divide one number by another, the remainder is always less than the divisor. So, in this case, $n < m < 2015$.
Since $n = 2015 \mod m$, that means:

$$2015 = m \cdot q + n$$

for some integer q, and $0 \leq n < m$.
I want to maximize n. From the division algorithm, we also know:

$$n = 2015 - m \cdot \lfloor 2015/m \rfloor$$

As m increases toward 2015, n generally decreases because the quotient becomes 1 and $n = 2015 - m$. So:
- If $m = 2014$, then $n = 1$ - If $m = 1008$, then $2015 = 1008 \cdot 1 + 1007 \Rightarrow n = 1007$ - If $m = 1009$, then $n = 2015 \mod 1009 = 1006$ - If $m = 1007$, then $2015 \mod 1007 = 1$
So as m increases above 1008, the remainder n decreases.

Let's test $m = 1006$:

$$1006 \cdot 2 = 2012 \Rightarrow n = 2015 - 2012 = 3$$

If $m = 1005$:

$$1005 \cdot 2 = 2010 \Rightarrow n = 5$$

So clearly, n peaks at $m = 1008$, where:

$$2015 \mod 1008 = 1007$$

Final Answer:

$$\boxed{1007}$$

Figure S5: Completion 1 for Question 1.

Answer 2 for Question 1 [Reward: 2.110]

Alright, so I have this problem here: I need to find the largest possible remainder when 2015 is divided by a positive integer m, where $m < 2015$. Hmm, okay. Let me try to figure this out step by step.
First off, I remember that when you divide a number by another number, the remainder is always less than the divisor. So if $m < 2015$, then the remainder n should be less than m as well. But the problem is asking for the largest possible n. So, I need to find the maximum remainder possible when 2015 is divided by some $m < 2015$.
From the division algorithm:

$$2015 = q \cdot m + n, \quad \text{where } 0 \leq n < m$$

So $n = 2015 - q \cdot m$, and we want to maximize n.
Trying values around half of 2015 gives interesting results. If $m = 1008$, then:

$$1008 \cdot 1 = 1008, \quad 2015 - 1008 = 1007$$

So the remainder $n = 1007$, which is promising.
Trying nearby m values:
- $m = 1007 \Rightarrow 1007 \cdot 2 = 2014, \quad n = 2015 - 2014 = 1$ - $m = 1009 \Rightarrow 2015 - 1009 = 1006$
So as m increases past 1008, the remainder decreases.
Let's consider another route. Suppose we want $n = m - 1$, the largest possible remainder. Then:

$$2015 \equiv -1 \mod m \Rightarrow m \mid (2015 + 1) = 2016$$

So the value of m must divide 2016, and the largest possible remainder will be $m - 1$. The largest divisor of 2016 less than 2015 is:

$$\text{Prime factorization: } 2016 = 2^5 \cdot 3^2 \cdot 7 \Rightarrow \text{divisors: 36 total}$$

Among these, the largest divisor less than 2015 is:

$$m = 1008 \Rightarrow n = 1008 - 1 = 1007$$

Trying $m = 1009 \Rightarrow n = 2015 \mod 1009 = 1006$, which is less.
Thus, we confirm:

Final Answer:

$$\boxed{1007}$$

Figure S6: Completion 2 for Question 1.

Question 2

<|begin_of_sentence|>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, and put your final answer within \boxed{} . 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>. Note that respond by English, NOT use other languages.<|User|>Given the sequence $1, 2, 1, 2, 2, 1, 2, 2, 2, 1, 2, 2, 2, 2, 1, 2, \cdots$ where the number of 2's between each pair of 1's increases by one each time, find the sum of the first 1234 terms of the sequence.<|Assistant|><think>

Figure S7: Prompt used for Example Question 2.

Answer 1 for Question 2 [Reward: 2.782]

Okay, so I have this sequence: 1, 2, 1, 2, 2, 1, 2, 2, 2, 1, 2, 2, 2, 2, 1, 2, and so on. The pattern is that each time, there's a 1 followed by an increasing number of 2's. The number of 2's between each pair of 1's increases by one each time.
First, I need to figure out the structure of the sequence to find the sum of the first 1234 terms. Let me write out the beginning of the sequence to see the pattern clearly:
1, 2, 1, 2, 2, 1, 2, 2, 2, 1, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 1, ...
So, starting from the first term:

- The first term is 1.
- Then one 2.
- Then another 1.
- Then two 2's.
- Then another 1.
- Then three 2's.
- Then another 1.
- Then four 2's.
- And so on.

So, each "block" of the sequence consists of a 1 followed by n 2's, where n starts at 1 and increases by 1 each time. So, the first block is 1 followed by 1 two; the second block is 1 followed by 2 two's; the third block is 1 followed by 3 two's, etc.
Wait, actually, looking again, the first block is 1, followed by one 2: so it's 1, 2. Then 1, 2, 2. Then 1, 2, 2, 2. So each block is 1 followed by k 2's, where k increases by 1 each time. So each block has $(k + 1)$ terms: 1 and k 2's.
Wait, no: 1, 2 is two terms. Then 1, 2, 2 is three terms. Then 1, 2, 2, 2 is four terms. So each block is $(k + 1)$ terms, where k starts at 1: 1, 2; 1, 2, 2; 1, 2, 2, 2; etc.
So, the total number of terms in the sequence up to a certain block can be calculated as the sum of the sizes of each block. Each block has $(k + 1)$ terms, where k is the block number starting from 1.

- First block ($k = 1$) has 2 terms: 1, 2.
- Second block ($k = 2$) has 3 terms: 1, 2, 2.
- Third block ($k = 3$) has 4 terms: 1, 2, 2, 2.

And so on.
So, the total number of terms up to the nth block is the sum from $k = 1$ to n of $(k + 1)$ terms. Which is:

$$\sum_{k=1}^{n}(k+1) = \sum_{k=1}^{n} k + \sum_{k=1}^{n} 1 = \frac{n(n+1)}{2} + n = \frac{n(n+1)+2n}{2} = \frac{n^2+3n}{2}.$$

Figure S8: Completion 1 for Question 2.

Answer 1 (continued) for Question 2 [Reward: 2.782]

We need to find n such that

$$\frac{n^2+3n}{2} \leq 1234 < \frac{(n+1)^2+3(n+1)}{2}.$$

Solving for n:

$$\frac{n^2+3n}{2} = 1234 \Rightarrow n^2+3n-2468 = 0.$$

Using the quadratic formula:

$$n = \frac{-3 \pm \sqrt{9+4\cdot 2468}}{2} = \frac{-3 \pm \sqrt{9881}}{2}.$$

Since $\sqrt{9881} \approx 99.4$, we have:

$$n \approx \frac{-3+99.4}{2} \approx \frac{96.4}{2} \approx 48.2 \Rightarrow n = 48.$$

Check total terms for $n = 48$:

$$\frac{48^2+3\cdot 48}{2} = \frac{2304+144}{2} = \frac{2448}{2} = 1224.$$

For $n = 49$:

$$\frac{49^2+3\cdot 49}{2} = \frac{2401+147}{2} = \frac{2548}{2} = 1274.$$

So, the 48th block ends at term 1224, and the 49th block starts at term 1225 and ends at term 1274. Thus, to reach the 1234th term, we take 10 terms from the 49th block:

$$1234 - 1224 = 10.$$

The 49th block is: 1 followed by 49 two's. The first 10 terms of this block are: 1, 2, 2, 2, 2, 2, 2, 2, 2, 2.
Sum of first 48 blocks:

$$\sum_{k=1}^{48}(1+2k) = \sum_{k=1}^{48}1 + \sum_{k=1}^{48}2k = 48 + 2\cdot\frac{48\cdot 49}{2} = 48+2352 = 2400.$$

Sum of first 10 terms of block 49:

$$1+9\cdot 2 = 1+18 = 19.$$

Total sum:

$$2400+19 = 2419.$$

Final Answer: The sum of the first 1234 terms is $\boxed{2419}$.

Figure S9: Completion 1 for Question 2 (**Continued**).

Answer 2 for Question 2 [Reward: 2.855]

Okay, so I have this sequence: 1, 2, 1, 2, 2, 1, 2, 2, 2, 1, 2, 2, 2, 2, 1, 2, and so on. The pattern seems to be that each time there's a 1 followed by an increasing number of 2's. Specifically, between each pair of 1's, the number of 2's increases by one each time. So, the first time there's one 1 followed by one 2, then next time two 1's with two 2's in between, then three 1's with three 2's, etc.
The question is asking for the sum of the first 1234 terms of this sequence. Hmm, okay, so I need to figure out where the 1234th term falls in this sequence and then calculate the sum up to that point.
First, let's understand the structure of the sequence. Each "block" consists of a 1 followed by an increasing number of 2's. So the first block is [1, 2], the second block is [1, 2, 2], the third block is [1, 2, 2, 2], and so on. So each block n consists of 1 followed by n 2's, making the total length of each block $n + 1$.
Wait, actually, hold on. Let me clarify. When $n = 1$, the block is [1, 2], which is 2 terms. For $n = 2$, it's [1, 2, 2], which is 3 terms. For $n = 3$, it's [1, 2, 2, 2], which is 4 terms, etc. So in general, block n has $(n + 1)$ terms. So the total number of terms up to block n is the sum from $k = 1$ to $k = n$ of $(k + 1)$. That's the same as the sum from $k = 1$ to $k = n$ of k plus the sum of 1's, which is:

$$\frac{n(n+1)}{2} + n = \frac{n(n+1) + 2n}{2} = \frac{n^2 + 3n}{2}.$$

So the total number of terms after block n is $\frac{n^2+3n}{2}$. We need to find the largest n such that $\frac{n^2+3n}{2} \leq 1234$. Then, the sum will be the sum of all complete blocks plus the sum of the remaining terms in the next block.
Let me denote $S(n) = \frac{n^2+3n}{2}$ as the total number of terms after block n. We need to solve for n in $S(n) \leq 1234$.
So, solving $n^2 + 3n - 2468 \leq 0$ (since $1234 \times 2 = 2468$). Using the quadratic formula:

$$n = \frac{-3 \pm \sqrt{9 + 9872}}{2} = \frac{-3 \pm \sqrt{9881}}{2}.$$

Calculating $\sqrt{9881}$: Let's see, $99^2 = 9801$, so $\sqrt{9881}$ is approximately 99.40. So $n \approx \frac{-3+99.40}{2} \approx \frac{96.40}{2} \approx 48.2$. So n is approximately 48.2, but since n must be an integer, $n = 48$.
Let's check $S(48)$:

$$S(48) = \frac{48^2 + 3 \cdot 48}{2} = \frac{2304 + 144}{2} = \frac{2448}{2} = 1224.$$

Then, the next block is block 49, which starts with a 1 followed by 49 2's. So, we need to see how many terms we have beyond 1224 to reach 1234.

$$1234 - 1224 = 10 \text{ terms}.$$

So, we need to take 10 terms from block 49, which is [1, 2, 2, 2, ..., 10 times 2].
Therefore, the sum of the sequence up to 1234 terms is the sum of all complete blocks up to block 48, plus the sum of the first 10 terms of block 49.
Each block k has 1 followed by k 2's, so the sum of block k is $1 + 2k$.

Figure S10: Completion 2 for Question 2.

Answer 2 (continued) for Question 2 [Reward: 2.855]

Therefore, the total sum up to block 48 is:

$$\sum_{k=1}^{48}(1+2k).$$

This can be broken into two sums:

$$\sum_{k=1}^{48}1+\sum_{k=1}^{48}2k=48+2\cdot\frac{48\cdot 49}{2}=48+48\cdot 49=48+2352=2400.$$

Now, we need to add the sum of the first 10 terms of block 49.
Block 49 is [1, 2, 2, 2, 2, 2, 2, 2, 2, 2, ..., 10 times 2].
The first term is 1, and the next 9 terms are 2's.

$$\text{Sum}=1+(10-1)\cdot 2=1+18=19.$$

Therefore, the total sum up to 1234 terms is:

$$2400+19=2419.$$

Let me double-check the calculations.
First, $S(n)=\frac{n^2+3n}{2}$.
For $n=48$, $S(48)=\frac{48^2+3\cdot 48}{2}=\frac{2304+144}{2}=\frac{2448}{2}=1224$. Correct.
So, $1234-1224=10$ terms. So, taking the first 10 terms of block 49, which is [1, 2, 2, 2, 2, 2, 2, 2, 2, 2], sum is $1+2\cdot 9=19$. Correct.
Therefore, $2400+19=2419$.
Yes, that seems correct. Then, adding the first 10 terms of block 49 adds 19, so total is 2419.
Block 49: 1 followed by 49 2's, so 50 terms.
But we only take 10 terms from it, so total terms are $1224+10=1234$. Correct.
Yes, that all adds up. So, the answer is 2419.
Final Answer
The sum of the first 1234 terms is $\boxed{2419}$.

Figure S11: Completion 2 for Question 2 (**Continued**).

Question 3

<|begin_of_sentence|> 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, and put your final answer within \boxed{} . 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>. Note that respond by English, NOT use other languages.

<|User|> The Sequence $\{a_n\}_{n\geqslant 0}$ is defined by $a_0 = 1$, $a_1 = -4$ and

$$a_{n+2} = -4a_{n+1} - 7a_n$$

for $n \geqslant 0$. Find the number of positive integer divisors of $a_{50}^2 - a_{49}a_{51}$.
<|Assistant|><think>

Figure S12: Prompt used for Example Question 3.

Answer 1 for Question 3 [Reward: 0.018]

To determine the number of positive integer divisors of $a_{50}^2 - a_{49}a_{51}$ for the given sequence, I'll follow these steps:
1. **Understand the Sequence Definition:** - The sequence is defined by $a_0 = 1$, $a_1 = -4$, and the recurrence relation $a_{n+2} = -4a_{n+1} - 7a_n$ for $n \geq 0$.
2. **Identify the Type of Recurrence Relation:** - This is a linear homogeneous recurrence relation with constant coefficients. It can be solved using characteristic equations.
3. **Formulate the Characteristic Equation:** - Assume a solution of the form $a_n = r^n$. Substituting into the recurrence relation gives:

$$r^{n+2} = -4r^{n+1} - 7r^n$$

- Dividing both sides by r^n (assuming $r \neq 0$) results in the characteristic equation:

$$r^2 + 4r + 7 = 0$$

4. **Solve the Characteristic Equation:** - The roots of the equation $r^2 + 4r + 7 = 0$ are found using the quadratic formula:

$$r = \frac{-4 \pm \sqrt{(4)^2 - 4 \cdot 1 \cdot 7}}{2 \cdot 1} = \frac{-4 \pm \sqrt{16 - 28}}{2} = \frac{-4 \pm \sqrt{-12}}{2} = -2 \pm i\sqrt{3}$$

- Thus, the roots are complex: $r = -2 + i\sqrt{3}$ and $r = -2 - i\sqrt{3}$.
5. **Express the General Solution:** - For complex roots $\alpha \pm i\beta$, the general solution is:

$$a_n = \lambda(\alpha + i\beta)^n + \mu(\alpha - i\beta)^n$$

- Here, $\alpha = -2$ and $\beta = \sqrt{3}$, so:

$$a_n = \lambda(-2 + i\sqrt{3})^n + \mu(-2 - i\sqrt{3})^n$$

6. **Determine the Constants λ and μ:** - Using the initial conditions $a_0 = 1$ and $a_1 = -4$: - For $n = 0$:

$$a_0 = \lambda(-2 + i\sqrt{3})^0 + \mu(-2 - i\sqrt{3})^0 = \lambda + \mu = 1$$

- For $n = 1$:

$$a_1 = \lambda(-2 + i\sqrt{3}) + \mu(-2 - i\sqrt{3}) = -4$$

- Solving these equations, we find $\lambda = \frac{1}{2}$ and $\mu = \frac{1}{2}$.
7. **Express a_n Explicitly:** - Substituting λ and μ into the general solution:

$$a_n = \frac{1}{2}(-2 + i\sqrt{3})^n + \frac{1}{2}(-2 - i\sqrt{3})^n$$

- Alternatively, using Euler's formula for complex roots, this can also be written in terms of trigonometric functions, but for our purposes, the exponential form is sufficient.

Figure S13: Completion 1 for Question 3.

Answer 1 (continued) for Question 3 [Reward: 0.018]

8. **Simplify the Expression $a_{50}^2 - a_{49}a_{51}$:** - Recognize that the given expression resembles the determinant of a 2x2 matrix:

$$\begin{vmatrix} a_{50} & a_{49} \\ a_{51} & a_{50} \end{vmatrix}$$

- This determinant simplifies to:

$$a_{50}^2 - a_{49}a_{51}$$

- To find this determinant, we can use properties of the sequence. Notice that for a linear recurrence relation of this form, the determinant $a_{n+1}^2 - a_n a_{n+2}$ is constant for all n.

9. **Find the Constant Determinant:** - Calculate the determinant for the initial terms $n = 0$:

$$a_1^2 - a_0 a_2$$

- First, compute a_2 using the recurrence relation:

$$a_2 = -4a_1 - 7a_0 = -4(-4) - 7(1) = 16 - 7 = 9$$

- Now, compute the determinant:

$$a_1^2 - a_0 a_2 = (-4)^2 - (1)(9) = 16 - 9 = 7$$

- Therefore, the determinant $a_{50}^2 - a_{49}a_{51} = 7$.

10. **Determine the Number of Positive Integer Divisors:** - The number 7 is a prime number. - The only positive integer divisors of 7 are 1 and 7. - Hence, there are 2 positive integer divisors.

Final Answer The number of positive integer divisors of $a_{50}^2 - a_{49}a_{51}$ is $\boxed{2}$.

Figure S14: Completion 1 for Question 3 (**Continued**).

Answer 2 for Question 3 [Reward: 0.021]

To determine the number of positive integer divisors of $a_{50}^2 - a_{49}a_{51}$ for the given sequence, I start by analyzing the recurrence relation:

$$a_{n+2} = -4a_{n+1} - 7a_n$$

with initial conditions $a_0 = 1$ and $a_1 = -4$.
Next, I recognize that the characteristic equation for this recurrence is:

$$r^2 + 4r + 7 = 0$$

Solving this quadratic equation using the quadratic formula gives complex roots:

$$r = -2 \pm i$$

These roots indicate that the general solution to the recurrence is:

$$a_n = \alpha(-2 + i)^n + \beta(-2 - i)^n$$

Using the initial conditions, I determine the constants α and β:

$$1 = \alpha + \beta \quad \text{and} \quad -4 = \alpha(-2 + i) + \beta(-2 - i)$$

Solving these equations yields:

$$\alpha = \frac{1 + i}{2} \quad \text{and} \quad \beta = \frac{1 - i}{2}$$

Substituting these back into the general solution, I get:

$$a_n = \frac{1}{2}[(1 + i)(-2 + i)^n + (1 - i)(-2 - i)^n]$$

To find $a_{50}^2 - a_{49}a_{51}$, I simplify the expression using properties of complex numbers and exponents. After simplification, I find that:

$$a_{50}^2 - a_{49}a_{51} = 1$$

Finally, since 1 only has one positive divisor, the number of positive integer divisors is 1.
To determine the number of positive integer divisors of $a_{50}^2 - a_{49}a_{51}$ for the given sequence, we can follow these steps:
1. **Understand the Recurrence Relation:**
The sequence is defined by the recurrence relation:

$$a_{n+2} = -4a_{n+1} - 7a_n$$

with initial conditions $a_0 = 1$ and $a_1 = -4$.
2. **Find the Characteristic Equation:**
The characteristic equation for the recurrence relation is:

$$r^2 + 4r + 7 = 0$$

Figure S15: Completion 2 for Question 3.

Answer 2 (continued) for Question 3 [Reward: 0.021]

3. **Solve the Characteristic Equation:**
Solving the quadratic equation using the quadratic formula:

$$r = \frac{-4 \pm \sqrt{16 - 28}}{2} = \frac{-4 \pm \sqrt{-12}}{2} = -2 \pm i$$

The roots are complex: $r = -2 + i$ and $r = -2 - i$.
4. **General Solution of the Recurrence:**
The general solution for the sequence is:

$$a_n = \alpha(-2 + i)^n + \beta(-2 - i)^n$$

where α and β are constants determined by the initial conditions.
5. **Apply Initial Conditions to Find α and β:**
Using $a_0 = 1$:

$$1 = \alpha + \beta$$

Using $a_1 = -4$:

$$-4 = \alpha(-2 + i) + \beta(-2 - i)$$

Solving these equations, we find:

$$\alpha = \frac{1 + i}{2}, \quad \beta = \frac{1 - i}{2}$$

Thus, the explicit formula for a_n is:

$$a_n = \frac{1}{2}[(1 + i)(-2 + i)^n + (1 - i)(-2 - i)^n]$$

6. **Compute $a_{50}^2 - a_{49}a_{51}$:**
After simplifying the expression using properties of complex numbers and exponents, it can be shown that:

$$a_{50}^2 - a_{49}a_{51} = 1$$

7. **Determine the Number of Positive Integer Divisors:**
Since $a_{50}^2 - a_{49}a_{51} = 1$, the only positive integer divisor of 1 is 1 itself.

$$\boxed{1}$$

Figure S16: Completion 2 for Question 3 (**Continued**).