R1 Experimental Report

Nguyen Manh Tien Anh

Vin Bigdata Instute, Hanoi, Vietnam
tienanh28122000@gmail.com

Abstract. This report explores the use of reinforcement learning methods to align reasoning large language models. The first section describes the reward formulations of each method used in the experiments. The second section outlines the experimental setup, including datasets, computational resources, training details and codebase. The third section presents the outcomes of the experiments and compares the models on a reasoning-focused test set. The final section concludes the report and summarizes the key findings.

1 Reward Formulations for Reasoning Alignment

1.1 Length-based Reward Functions

L1: Controlling how long a reasoning model thinks with reinforcement learning [1] This paper introduces a method to control the reasoning length of large language models (LLMs) using reinforcement learning (RL). The key idea is to prompt the model to reason for a target number of tokens and reward it based on both accuracy and how closely it adheres to the desired reasoning length.

Let the original dataset be:

$$D = \{(x_i, y_{\text{gold},i})\}_{i=1}^{N}$$

where x_i is the input (e.g., question), and $y_{\text{gold},i}$ is the corresponding ground-truth answer.

To guide reasoning length, each input is modified by appending an instruction:

$$x_i^{\text{new}} = \text{Concat}(x_i, \text{“Think for } n_{\text{gold},i} \text{ tokens”})$$

where $n_{\text{gold},i} \sim Z(n_{\min}, n_{\max})$ is the desired reasoning length sampled from a discrete distribution Z.

The modified dataset becomes:

$$D_{\text{new}} = \{(x_i^{\text{new}}, y_{\text{gold},i})\}_{i=1}^{N}$$

The reward function for training is:

$$r(y, y_{\text{gold}}, n_{\text{gold}}) = \mathbb{I}(y = y_{\text{gold}}) - \alpha \cdot |n_{\text{gold}} - n_y|$$

where:

- y: model's predicted answer.
- y_{gold}: ground-truth answer.
- $\mathbb{I}(y = y_{\text{gold}})$: indicator function, equals 1 if the answer is correct, 0 otherwise.
- n_y: number of tokens generated in the reasoning process.
- n_{gold}: desired number of reasoning tokens.
- α: a positive scalar controlling the penalty for deviating from the target length.

This reward encourages correctness while discouraging deviations from the target reasoning length. A higher α penalizes more strongly for mismatch in length. In the paper, this reward function is called *L1-Exact*.

The paper also proposes a variant called *L1-Max*, which uses a clipped cosine-shaped reward:

$$r(y, y_{\text{gold}}, n_{\text{gold}}) = \mathbb{I}(y = y_{\text{gold}}) \cdot \text{clip}(\alpha \cdot (n_{\text{gold}} - n_y) + \delta, 0, 1)$$

where:

- δ: a margin constant (e.g., 0.5) that softens the penalty.
- $\text{clip}(a, 0, 1)$: truncates the value a to lie within [0, 1].

This variant allows a smoother gradient and more flexibility in length control.

Demystifying Long Chain-of-Thought Reasoning in LLMs [11] This paper focuses on stabilizing the length-effectiveness trade-off in Chain-of-Thought (CoT) reasoning. It introduces a reward function that incorporates both answer correctness and the length of the generated reasoning steps.

The reward function is:

$$R(C, L_{\text{gen}}) = \begin{cases} \text{CosFn}(L_{\text{gen}}, L_{\text{max}}, r_c^0, r_c^L), & \text{if } C = 1 \text{ (correct)} \\ \text{CosFn}(L_{\text{gen}}, L_{\text{max}}, r_w^0, r_w^L), & \text{if } C = 0 \text{ (wrong)} \\ r_e, & \text{if } L_{\text{gen}} = L_{\text{max}} \end{cases}$$

Where:

$$\text{CosFn}(L_{\text{gen}}, L_{\text{max}}, r^0, r^L) = r^0 + \frac{1}{2}(r^L - r^0)\left(1 + \cos\left(\frac{L_{\text{gen}}\pi}{L_{\text{max}}}\right)\right)$$

Explanation of variables:

- $C \in \{0, 1\}$: correctness label (1 if correct, 0 if wrong).
- L_{gen}: number of tokens in the generated CoT reasoning path.
- L_{max}: maximum allowed number of tokens (length budget).
- $\text{CosFn}(\cdot)$: a cosine-shaped function that interpolates smoothly between reward values.
- r_c^0, r_c^L: reward range for correct responses from shortest to longest chain.
- r_w^0, r_w^L: reward range for incorrect responses.
- r_e: fixed penalty if the model hits the maximum length L_{max}.

Intuitively:

- For correct responses, shorter responses receive higher rewards than longer responses, which incentivizes the model to use inference to compute efficiently
- For incorrect responses, shorter responses should receive higher penalties than longer responses, which encourages the model to extend its thinking time if it is less likely to get the correct answer
- The cosine shape ensures smooth reward and adjustable preferences.

This design allows flexible prioritization between short/efficient and long/thorough reasoning, and adapts the reward to different task difficulty levels.

Training Language Models to Reason Efficiently [2] This work proposes a framework to train LLMs that produce correct and efficient (i.e., short) reasoning traces. The model is encouraged to solve tasks using fewer tokens without sacrificing accuracy.

The expected reward for a generated answer y on input x is:

$$\mathbb{E}_y \left[\mathbb{I}(y = y^\star(x)) \cdot (1 - \alpha f(\mathrm{LEN}(y)))\right]$$

Explanation of variables:

- $y^\star(x)$: ground-truth answer for input x.
- $\mathbb{I}(y = y^\star(x))$: correctness indicator.
- $\mathrm{LEN}(y)$: number of tokens in the model's generated reasoning.
- $\alpha \in [0, 1)$: a scalar that controls how much length is penalized.
- $f(\mathrm{LEN}(y))$: a regularization function that maps output length to [0,1].

The regularization function $f(\cdot)$ is:

$$f(\mathrm{LEN}(y)) = \sigma\left(\frac{\mathrm{LEN}(y) - \mathrm{MEAN}(x)}{\mathrm{STD}(x)}\right)$$

where:

- $\sigma(\cdot)$: sigmoid function to smoothly bound the length penalty between 0 and 1.
- $\mathrm{MEAN}(x)$: expected length of correct reasoning outputs for input x:

$$\mathrm{MEAN}(x) = \mathbb{E}_{y\sim p(x),\, y=y^\star(x)}[\mathrm{LEN}(y)]$$

- $\mathrm{STD}(x)$: standard deviation of those lengths:

$$\mathrm{STD}(x) = \sqrt{\mathrm{Var}_{y\sim p(x),\, y=y^\star(x)}[\mathrm{LEN}(y)]}$$

This design penalizes excessively long answers relative to other correct answers on the same input, enabling adaptive length control. It ensures:

– Long reasoning for hard problems is not penalized unfairly.
– Short reasoning is encouraged when possible.
– Correctness is still prioritized due to the outer indicator.

Note: In [8], the authors summarize five recent studies that explore length-based reward functions. However, due to time and resource constraints, only three of these five proposals are implemented in this report. The remaining two proposals are more complex in terms of the GRPO training setup and require additional resources for experimentation, as demonstrated in [5].

1.2 Proposed Reward Functions

Three reward functions are proposed to guide large language models in generating outputs that are both accurate and diverse. Each reward function addresses a specific objective: majority consensus, diverse reasoning among correct outputs, and novelty in relation to prior completions.

Voting-based Reward Function Motivation. This reward function encourages model completions that align with the majority answer, under the assumption that the most frequent response among outputs reflects collective confidence and potential correctness. It is especially applicable in scenarios where explicit ground-truth labels are unavailable. This reward function is inspired by the approach proposed in [10].

Formulation. Given a question x, the language model generates N completions. Let the set of extracted answers be $P = \{\hat{y}_i\}_{i=1}^{N}$. The majority-voted answer y is determined by identifying the most frequent answer in P. The reward function is defined as:

$$R(\hat{y}_i, y) = \begin{cases} 1, & \text{if } \hat{y}_i = y \\ 0, & \text{otherwise} \end{cases}$$

Explanation of Variables.

– $\hat{y}_i$: Predicted answer extracted from the i-th completion.
– y: Majority answer from the set P.
– $R(\hat{y}_i, y)$: Binary reward indicating agreement with the majority.

Novelty. This function converts majority voting into a weak supervision signal, offering a simple yet effective way to guide the model without reliance on human-annotated labels or correctness signal.

Group Intrinsic Reward Function The *group intrinsic reward function* is designed to not only reward correctness but also encourage diverse reasoning among correct completions for the same question. Instead of promoting uniformity in successful answers, this function favors semantic variation in the reasoning process, thereby fostering creative and robust model behavior.

Correct Completion Set. Given a batch of n completions $\{x_i\}_{i=1}^n$ for a single input question, let $\hat{y}_i$ denote the predicted answer and y_i the corresponding ground-truth answer. We define the set of indices corresponding to correct completions as:

$$\mathcal{C} = \{i \mid \hat{y}_i = y_i\}$$

This set $\mathcal{C}$ contains only those indices i for which the model produces a correct output.

Embedding Reasoning Spans. For each $i \in \mathcal{C}$, the model extracts the reasoning span r_i from the full completion x_i. This span reflects the part of the response that contains the model's rationale or explanation. A pretrained encoder (e.g., `OpenAI text-embedding-3-small` [7]) is used to embed the reasoning span into a dense vector:

$$\mathbf{e}_i \in \mathbb{R}^d$$

where d is the embedding dimensionality.

Group Centroid Excluding Self. To measure the semantic similarity of reasoning r_i with respect to other correct completions, a centroid of the embeddings—excluding the current index i—is computed:

$$\bar{\mathbf{e}}_{-i} = \frac{1}{|\mathcal{C}| - 1} \sum_{\substack{j \in \mathcal{C} \\ j \neq i}} \mathbf{e}_j$$

This vector $\bar{\mathbf{e}}_{-i}$ represents the average reasoning direction of all other correct responses, serving as a contextual reference for diversity.

Cosine Similarity to Centroid. Next, the semantic similarity between r_i and the group reasoning centroid is measured using cosine similarity:

$$s_i = \cos(\mathbf{e}_i, \bar{\mathbf{e}}_{-i}) = \frac{\mathbf{e}_i \cdot \bar{\mathbf{e}}_{-i}}{\|\mathbf{e}_i\| \, \|\bar{\mathbf{e}}_{-i}\|}$$

This value $s_i \in [-1, 1]$ quantifies how semantically aligned the reasoning of x_i is with the rest of the correct answers.

Dissimilarity-Based Reward Bonus. To encourage dissimilar reasoning (i.e., lower s_i), the reward includes a diversity bonus term proportional to $(1 - s_i)$. Since cosine similarity values outside $[0, 1]$ may lead to unstable gradients, the similarity score is clamped to this range:

$$\text{bonus} = C \cdot (1 - \text{clamp}(s_i, 0, 1))$$

where $C > 0$ is a hyperparameter controlling the strength of the diversity reward.

Final Group Intrinsic Reward. The full reward R_i assigned to each completion x_i is defined as:

$$R_i = \begin{cases} R_{\text{base}} + C \cdot (1 - s_i), & \text{if } i \in \mathcal{C} \text{ and } |\mathcal{C}| > 1 \\ R_{\text{base}}, & \text{if } i \in \mathcal{C} \text{ and } |\mathcal{C}| = 1 \\ 0, & \text{otherwise} \end{cases}$$

where:

- R_{base} is the fixed reward for generating a correct answer.
- C is the diversity scaling coefficient.

Summary. This reward design achieves two key goals:

1. It ensures that only correct completions are rewarded, filtering out irrelevant or wrong answers.
2. Among correct completions, it prioritizes those whose reasoning diverges from the common patterns, promoting creativity and discouraging redundant justifications.

As a result, the model is guided to not only optimize for accuracy but also to explore multiple valid reasoning paths, improving both robustness and interpretability.

Intrinsic Completion Reward Function The *intrinsic completion reward function* is designed to incentivize language model completions that are not only correct but also diverse in reasoning. This function focuses solely on rewarding novelty among correct responses, based on semantic similarity to past successful examples. This reward function is inspired by the approach proposed in [4].

Correctness Evaluation. For a model-generated completion c_i, the predicted answer $\hat{a}_i$ is extracted from the output using structured tags (e.g., `<answer>`). Let a_i denote the gold answer. Correctness is determined using a binary indicator:

$$\text{Correct}(c_i) = \begin{cases} 1, & \text{if } \hat{a}_i = a_i \\ 0, & \text{otherwise} \end{cases}$$

This correctness check serves as the gateway for any reward assignment—only correct completions receive further scoring.

Diversity-Based Reward. To encourage novelty, the reward mechanism compares the semantic representation of the reasoning part (extracted from `<reasoning>` tags) of c_i against an archive of previous correct completions stored in a LanceDB table. The steps are as follows:

1. Compute the embedding vector $\mathbf{e}_i$ of the reasoning using a language model (e.g., OpenAI `text-embedding-3-small` [7]).
2. Retrieve the top-k most similar embeddings from the success archive.
3. Compute the centroid $\mathbf{c}_k$ of these k nearest neighbors.
4. Calculate the cosine similarity $s(c_i)$ between $\mathbf{e}_i$ and the centroid $\mathbf{c}_k$.

Reward Function Definition. The final intrinsic reward $R(c_i)$ is assigned as:

$$R(c_i) = \begin{cases} R_{\text{base}} + \lambda \cdot (1 - s(c_i)), & \text{if Correct}(c_i) = 1 \\ 0, & \text{otherwise} \end{cases}$$

where:

– R_{base} is the fixed reward for correctness (e.g., 1.0).
– λ is a scaling factor for diversity (e.g., 0.5).
– $s(c_i)$ is the cosine similarity between the reasoning vector and the centroid of similar correct examples. A lower $s(c_i)$ indicates higher novelty.

Key Properties. This completion intrinsic reward function provides the following benefits:

– **Simplicity.** Avoids the complexity of moving averages, thresholds, and failure tracking.
– **Focused Incentives.** Rewards diverse reasoning only among correct completions.
– **Efficient Similarity Search.** Uses LanceDB for scalable nearest neighbor queries.

Table 1. Training Hyperparameters for GSM8K

Hyperparameter	**GSM8K**
Learning Rate	5×10^{-6}
Adam β_1	0.9
Adam β_2	0.99
Weight Decay	0.1
Warmup Ratio	0.1
Learning Rate Scheduler	Cosine
Batch Size	4
Gradient Accumulation Steps	4
Number of GRPO Generations	16
Maximum Prompt Length	256
Maximum Completion Length	512
Training Epochs	1
Maximum Gradient Norm	0.1
Mixed Precision	BF16

Conclusion. The completion intrinsic reward mechanism remains effective in promoting creative yet correct completions by leveraging semantic diversity. By focusing exclusively on successful cases in the past, it balances implementation simplicity with meaningful exploration incentives.

2 Experimental Setup and Implementation Details

2.1 Training

The Qwen2.5-1.5B-Instruct model [9] is fine-tuned using 7 NVIDIA A100 80GB GPUs. Specifically, 1 GPU is allocated for deploying vLLM to accelerate generation during training, while the remaining 6 GPUs are used for updating the model weights after each step. Training is conducted on the GSM8K dataset [6], utilizing the training set of 7,743 samples. Due to time constraints, each method described in the previous sections is trained only once to obtain the final checkpoint (excluding any attempts or failures for each method). The implementation and execution of the training process are publicly available at `https://github.com/tienanh28122000/R1`.

Table 1 provides the detailed hyperparameters used during training on GSM8K.

2.2 RL Algorithm

All experiments use the same training dataset and reinforcement learning (RL) algorithm, specifically the GRPO algorithm. The only variation across experiments is the reward method used.

Category	**Method**	**Extractive Match (%)**
Base		65.20
R1		71.98
Length-based	[1]	72.43
	[2]	73.31
	[11]	73.19
Proposed	Voting-based	73.62
	Group Intrinsic	73.81
	Completion Intrinsic	73.83

Table 2. Comparison of different methods on Extractive Match accuracy, based on Qwen2.5-1.5B-Instruct.

2.3 Evaluation

Evaluation is performed on the GSM8K test set, with all methods following the zero-shot setting. The evaluation process is based on Lighteval, employing its extractive match metric. This metric applies regex-based conditions to accurately extract and parse the generated answers.

3 Evaluation Results and Comparative Analysis

Table 2 presents the test accuracy of different reward function strategies using the extractive match metric, compared against a baseline. Overall, fine-tuning with reinforcement learning (RL) consistently improves model performance. The baseline model, trained with supervised fine-tuning (SFT) using instruction-only data, achieves a test accuracy of just 65.20

The original R1 method proposed in [3] achieves 71.98%, showing a noticeable improvement over the baseline. Incorporating length-based reward functions yields an additional absolute improvement of approximately 1% compared to the original R1. Among these, the best performing variant achieves 73.19%.

Moreover, three proposed reward functions further improve upon the original R1. Specifically, the voting-based method reaches 73.62%, while the group intrinsic and completion intrinsic methods achieve 73.81% and 73.83%, respectively. These results demonstrate an absolute improvement of around 2% over the R1 baseline.

It is important to note that, due to time constraints, it was unable to fully optimize or re-evaluate the proposed methods to their full potential. With additional tuning and analysis, these approaches could likely achieve even stronger performance.

For training performance details of each method, please refer to the repository at `https://github.com/tienanh28122000/R1`.

4 Conclusion

This report investigated the application of reinforcement learning techniques to align LLMs for reasoning tasks. Despite limitations in time and computational resources that may have led to certain mistakes, the work has provided a solid foundation for understanding how to train reasoning-capable LLMs using RL-based methods. The insights gained through this exploration not only enable further development in this area but also reaffirm the potential and excitement of this research direction.

Last but not least, Happy Vietnam's Reunification Day!

References

1. Aggarwal, P., Welleck, S.: L1: Controlling how long a reasoning model thinks with reinforcement learning. arXiv preprint arXiv:2503.04697 (2025)
2. Arora, D., Zanette, A.: Training language models to reason efficiently. arXiv preprint arXiv:2502.04463 (2025)
3. Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al.: Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. arXiv preprint arXiv:2501.12948 (2025)
4. Le, H., Do, D., Nguyen, D., Venkatesh, S.: Reasoning under 1 billion: Memory-augmented reinforcement learning for large language models. arXiv preprint arXiv:2504.02273 (2025)

5. Luo, H., Shen, L., He, H., Wang, Y., Liu, S., Li, W., Tan, N., Cao, X., Tao, D.: O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning. arXiv preprint arXiv:2501.12570 (2025)
6. OpenAI: Gsm8k: Grade school math 8k dataset. `https://huggingface.co/datasets/openai/gsm8k` (2021), accessed: 2025-05-01
7. OpenAI: Openai platform documentation: Embeddings. `https://platform.openai.com/docs/guides/embeddings` (2024), accessed: 2025-05-01
8. Sui, Y., Chuang, Y.N., Wang, G., Zhang, J., Zhang, T., Yuan, J., Liu, H., Wen, A., Chen, H., Hu, X., et al.: Stop overthinking: A survey on efficient reasoning for large language models. arXiv preprint arXiv:2503.16419 (2025)
9. Team, Q.: Qwen2.5-1.5b-instruct. `https://huggingface.co/Qwen/Qwen2.5-1.5B-Instruct` (2024), accessed: 2025-05-01
10. Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., Zhou, D.: Self-consistency improves chain of thought reasoning in language models. arXiv preprint arXiv:2203.11171 (2022)
11. Yeo, E., Tong, Y., Niu, M., Neubig, G., Yue, X.: Demystifying long chain-of-thought reasoning in llms. arXiv preprint arXiv:2502.03373 (2025)