RLLR: REINFORCEMENT LEARNING WITH LATENT REWARD

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning with verifiable rewards (RLVR) has achieved strong performance in tasks with reliably verifiable answers, such as mathematical reasoning and code generation. However, extending RLVR to general reasoning tasks (*e.g.,* healthcare, engineering, law, and business) remains challenging, since correct answers are diverse in format and wording, making it difficult to design task-specific verifiers. Recent verifier-free methods instead derive rewards from the policy model's probability of generating a reference answer conditioned on the prompt and generated rationale. Nevertheless, we observe that probability-based methods are susceptible to non-semantic noise tokens and suffer from reference answer length effects, leading to unreliable reward evaluation. To reduce sensitivity to text format and wording, we shift from the text space to the latent semantic space to evaluate response quality. We propose **Reinforcement Learning with Latent Reward (RLLR)**, a framework that extends RLVR to general reasoning tasks by defining rewards directly in the LLM's latent space without relying on external verifiers. Specifically, RLLR obtains the subspaces of the reference and generated answer's hidden states through Singular Value Decomposition (SVD) on answer hidden states, and measures their semantic consistency in subspaces. Experimental results demonstrate that RLLR improves the performance of Llama and Qwen models on five general-domain benchmarks and five mathematical benchmarks and outperforms rule-based, model-based, and probability-based reward methods.

1 INTRODUCTION

Reinforcement learning (RL) has emerged as an effective post-training method for enhancing the reasoning capabilities of large language models (LLMs). Recent works, such as DeepSeek-R1-Zero (Guo et al., 2025a), have shown that RL with verifiable rewards (RLVR) (Shao et al., 2024) can significantly boost the mathematical reasoning and code generation abilities of LLMs, where the model first generates the chain-of-thought (CoT) and a final answer, and a hand-crafted rule or executor then verifies the correctness of the final answer. The resulting binary signal is subsequently used as the reward for RL training.

The RLVR framework exhibits strong effectiveness when the correctness of the final answer can be reliably verified by rules or programs, such as in mathematical reasoning and code generation tasks. However, extending RLVR to general reasoning tasks, where answers are flexible and diverse in format and wording, is challenging, as RLVR relies heavily on task-specific verification rules. Some works (Ma et al., 2025; Su et al., 2025) attempt to address this limitation by fine-tuning an LLM as a learned verifier or a reward model. Nevertheless, the model-based approach requires substantial labeled data and incurs additional memory and inference overhead to host a separate verifier model. Additionally, the evaluation results of the model-based verifier are not sufficiently accurate. As shown in Figure 1, we rank responses for each prompt based on the corresponding rewards and compute the ROC-AUC using the annotations as the ground truth. A higher AUC indicates a more accurate reward ranking. The results show that the AUC of the model-based verifier is only 0.78.

Recently, several works have explored verifier-free reinforcement learning by deriving reward directly from the policy model itself, without relying on external rule- or model-based verifiers. In particular, VeriFree (Zhou et al., 2025) and RLPR (Yu et al., 2025b) evaluate the quality of generated responses using the probability assigned to the reference answer conditioned on the prompt and the generated

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

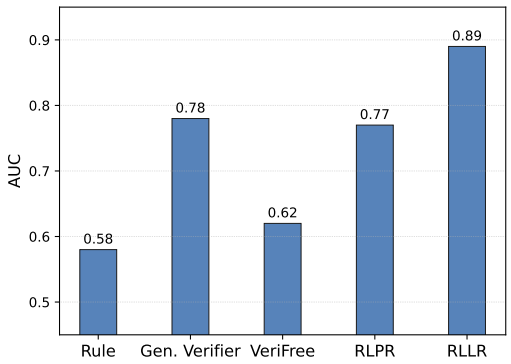


Figure 1: Reward quality comparison measured by ROC-AUC. We compute rewards by different methods for model-generated responses and use DeepSeek-R1 annotations as ground-truth labels to evaluate AUC. Higher is better.

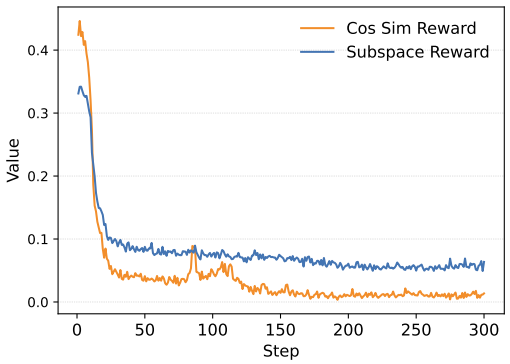


Figure 2: Training dynamics of reward standard deviation across different reward designs.

rationale. While promising, such probability-based rewards have limited ability to assess response quality as shown in Figure 1 and remain sensitive to non-semantic perturbations and reference answer length as shown in Table 3, limiting their reliability.

These issues with probability-based method arise because they evaluate responses in the *text space*, where the likelihood of the reference answer can be heavily influenced by non-semantic factors (*e.g.,* formatting artifacts and function words). Therefore, we instead focus on measuring *semantic consistency* and design rewards directly in the *latent space* based on the model's hidden states. A natural approach is to compute the cosine similarity between the hidden states of the generated answer and the reference answer as the reward. We find that the AUC of mean-pooled cosine similarity reaches 0.85, suggesting that hidden-state alignment can effectively reflect the quality of generated answers. However, in RL practice with cosine similarity as the reward, the mean reward quickly converges to 1 and the variance of reward collapses (Figure 2), making sustained training infeasible. This failure mode stems from the fact that mean pooling compresses the hidden state of a multi-token answer into a single direction, which lacks sufficient expressiveness and thus enables the policy model to easily hack the reward.

Motivated by these observations, we propose the **RLLR** framework (**Reinforcement Learning with Latent Reward**), which extends RLVR to general reasoning tasks by defining rewards directly in the latent space without relying on external verifiers. Instead of compressing a multi-token answer into a single vector via mean pooling, RLLR leverages Singular Value Decomposition (SVD) to obtain the subspaces of the reference and a generated answer, and then calculates the ratio of their energies in the two subspaces as the reward. Empirically, this reward achieves higher discriminative AUC (Figure 1) and leads to consistent improvements in RL training (Figure 3). We summarize our contributions as follows:

- We find that probability-based reward assignment methods without external verifiers are susceptible to noise tokens and length variations of reference answers, leading to inaccurate reward signals.
- To eliminate the influence of answer format and wording, we propose designing rewards directly in the latent space (which captures the semantic information of answers), and thus introduce RLLR — a novel framework that assigns rewards by evaluating the quality of hidden states of generated answers in the latent space, without requiring any external verifiers.
- Extensive experiments on Qwen3 (0.6B, 1.7B, 4B, 8B) and Llama models across 10 reasoning benchmarks demonstrate that RLLR consistently outperforms rule-based RLVR, model-based RL methods, and probability-based RL methods.
- Experimental results further validate that our method is robust to noise tokens and length variations in reference answers, confirming the reliability of RLLR.

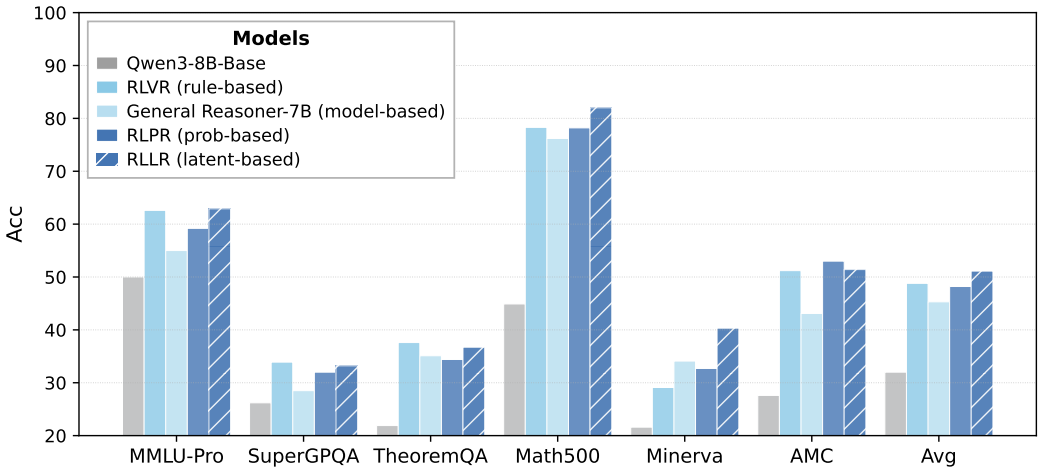


Figure 3: Accuracy comparison across general-domain and mathematical reasoning benchmarks. RLLR conducts reinforcement learning (RL) training using hidden states in the latent space to compute rewards without relying on external verifiers, consistently improving model performance on both general-domain and mathematical reasoning tasks. Notably, RLLR outperforms RLVR (heuristic rule-based verification), model-based verifier-driven RL methods, and RLPR (which leverages the intrinsic probability of LLMs generating reference answers as rewards). Avg: average accuracy across six benchmarks.

2 Related Works

Reinforcement learning with verifiable rewards. Recently, many works (Guo et al., 2025a; Yu et al., 2025a; Zheng et al., 2025; Ichihara et al., 2025; Team et al., 2024) have demonstrated that RLVR can significantly improve LLMs' reasoning performance in mathematics and code. Unlike RLHF (Jiang et al., 2024), which trains a reward model to capture subjective human preferences, RLVR relies on hand-crafted rules (Kydlíček, 2025), program execution (Cheng et al., 2024), or other custom checkers (Cui et al., 2025) to judge the correctness of model rollouts. However, such hand-designed verifiers are typically task-specific, and building a reliable verifier for a new domain is often time-consuming and labor-intensive. More importantly, in general reasoning domains (Ma et al., 2025), constructing accurate verifiers is nearly impossible because reference answers vary widely in format and wording. In contrast, our work aims to extend RLVR-style training from mathematics and code reasoning to the general reasoning domains where verifiable rewards are not available.

General domain reasoning. Compared with mathematical or code tasks, the core challenge in general reasoning lies in the difficulty of verifying answer correctness. Nemotron-CrossThink (Akter et al., 2025) reduces verification complexity by adopting structured templates (*e.g.,* multiple-choice) and filtering data samples with verifiable answers. Ma et al. (2025) and Su et al. (2025) fine-tune an LLM on annotated data to assess the correctness of LLMs' responses. VeriFree (Zhou et al., 2025) and RLPR (Yu et al., 2025b) further show that the probability that the policy model assigns to generating the reference answer conditioned on the prompt and generated rationale can serve as a proxy for answer quality without a verifier. However, probability-based methods are sensitive to noise tokens and response length. In contrast, our work compares the semantics of the reference and generated answers in latent space, which mitigates the influence of noise tokens and length in the text space.

Latent space of LLMs. The latent space of LLMs has been extensively explored across various fields, including hallucination detection (Burns et al., 2023; Ferrando et al., 2025), knowledge editing (Meng et al., 2022; Fang et al., 2025), safety alignment (Barez et al., 2025; Zhou et al., 2024), knowledge distillation (Dasgupta & Cohn, 2025), and latent space reasoning (Wang et al., 2025). These studies highlight the important role of latent space in shaping LLM behaviors and performance. Recent studies further investigate the relationship between hidden states and response quality. Specifically, Du et al. (2025a) analyze the structural differences in the latent space between

Algorithm 1 Latent Space Reward

Input: Policy π_θ, prompt x, generated CoT z, generated answer y, reference $y^\star$, target layer L, Top-k parameter k.
Output: Reward R.
 # Construct trajectories under the same context
$s_{\mathrm{g}} \leftarrow [x; z; y]$
$s_{\mathrm{r}} \leftarrow [x; z; y^\star]$
 # Extract answer-span hidden states from the L-th block
$H_g \leftarrow \mathrm{ExtractLayer}(\pi_\theta, s_{\mathrm{g}}, L)$
$H_r \leftarrow \mathrm{ExtractLayer}(\pi_\theta, s_{\mathrm{r}}, L)$
 # Construct subspaces using right singular vectors
$V_g \leftarrow \mathrm{SVD}(H_g)$
$V_r \leftarrow \mathrm{SVD}(H_r)$
 # Token-wise energies: $\mathbf{e}(A)_t = \|A_t\|_2^2$
$\mathrm{score}_r \leftarrow \dfrac{\mathrm{TopKMean}(\mathbf{e}(H_g V_r), k)}{\mathrm{TopKMean}(\mathbf{e}(H_r V_r), k)}$
$\mathrm{score}_g \leftarrow \dfrac{\mathrm{TopKMean}(\mathbf{e}(H_r V_g), k)}{\mathrm{TopKMean}(\mathbf{e}(H_g V_g), k)}$
return $R \leftarrow \dfrac{1}{2}\left(\mathrm{score}_r + \mathrm{score}_g\right)$

correct and incorrect responses. Du et al. (2025a); Guo et al. (2025b); Zhang et al. (2025) find that high-quality and low-quality responses are distinctly separable in the latent space and training a lightweight network (*e.g.,* MLP) on labeled data can effectively discriminate between high-quality and low-quality outputs, outperforming reward models with substantially more parameters in some cases. Building on this insight, we directly compute a metric in the latent space as a verifier-free reward, eliminating the need for neural networks and labeled data.

3 METHOD

In this section, we first introduce the preliminaries of RLVR and then present our latent space reward design.

3.1 PRELIMINARIES: REINFORCEMENT LEARNING WITH VERIFIABLE REWARDS

RLVR is a widely used post-training framework for LLMs, where task-specific verifiers assign a scalar reward score to each generated response. Formally, for a given input prompt x, the policy model π_θ generates a reasoning rationale z alongside a final answer y, *i.e.,* $o = (z, y) \sim \pi_\theta(\cdot \mid x)$. A verifier then evaluates the final answer against the ground-truth reference $y^\star$ and returns a reward $R_{\text{ver}}(x; o)$:

$$R_{\text{ver}}(x; o) = f_{\text{ver}}(y, y^\star)$$

where f_{ver} is a task-specific rule-based function , such as symbolic verifiers (Kydlíček, 2025) tailored for mathematical problems. However, a key limitation of this framework is that the design and implementation of such rule-based verifiers require substantial human effort and are highly task-specific. This limits the applicability of RLVR to broader general domain tasks.

3.2 LATENT SPACE REWARD

We denote the token sequence of a prompt as $x = (x_1, \ldots, x_{T_x})$, and let the policy model π_θ generate a response that consists of a chain-of-thought z followed by a final answer y, where $z = (z_1, \ldots, z_{T_z})$ and $y = (y_1, \ldots, y_{T_y})$. The reference final answer is denoted by $y^\star = (y^\star_1, \ldots, y^\star_{T_{y^\star}})$. We elaborate on the extraction of hidden states in Section 3.2.1, the construction of hidden state subspaces and energy calculation within subspaces in Section 3.2.2, and the formulation of the latent reward in Section 3.2.3. We present the algorithm for computing the latent reward in Algorithm 1.

3.2.1 EXTRACTING MODEL HIDDEN STATES IN LATENT SPACE

Given the prompt x, the policy model π_θ generates a CoT z and a final answer y, which together form the generated trajectory $s_{\mathrm{g}} = [x; z; y]$. To ensure the reference answer $y^\star$ is embedded in an identical contextual setting, we follow RLPR (Yu et al., 2025b) and VeriFree (Zhou et al., 2025) to construct the reference trajectory $s_{\mathrm{r}} = [x; z; y^\star]$ by concatenating the prompt, the generated CoT, and the reference answer. This design also avoids the introduction of golden CoT, which is both scarce and costly to acquire.

We encode both trajectories using the policy model π_θ (with gradients detached), and then extract hidden states corresponding to the spans of y and $y^\star$ (excluding the CoT) from the L-th transformer block. This yields hidden state matrices for the generated answer and reference answer: $H_{\mathrm{g}} \in \mathbb{R}^{T_g \times d}$, $H_{\mathrm{r}} \in \mathbb{R}^{T_r \times d}$, where T_g and T_r denote the number of tokens in the generated and reference answers, respectively, and d is the model's hidden dimension.

Intuitively, if the generated final answer y is semantically consistent with the reference final answer $y^\star$, then the answer-span representations H_{g} and H_{r} should be similar in a suitable representation space under the same $[x, z]$ context. So in the next section, we introduce a subspace-based similarity measure that compares H_{g} and H_{r} through their projections onto their subspaces.

3.2.2 COMPUTING ANSWER ENERGY IN SUBSPACES

Directly comparing H_{g} and H_{r} in the full latent space is challenging because the answers may have different lengths and can contain non-semantic or noisy token hidden states. A straightforward baseline is to compute the cosine similarity between mean-pooled hidden states; however, we find that this reward quickly saturates during RL training, with the mean reward approaching 1 and the reward standard deviation collapsing as shown in Figure 2. This is because mean pooling compresses a multi-token answer into a single direction, which is not expressive enough to capture the rich structure of answer spans and makes the reward easier to hack. To obtain a more robust and expressive latent reward, we represent each answer span as a low-dimensional subspace and measure semantic alignment via projection energies within these subspaces.

Motivated by these analysis, after extracting the answer-span hidden states H_{r} and H_{g}, we construct the reference and generated subspaces via SVD. Taking the reference answer as an example, the SVD of H_{r} is given by:

$$H_{\mathrm{r}} = U_{\mathrm{r}} \Sigma_{\mathrm{r}} V_{\mathrm{r}}^\top,$$

where $U_{\mathrm{r}} \in \mathbb{R}^{T_r \times m}$, $\Sigma_{\mathrm{r}} \in \mathbb{R}^{m \times m}$, and $V_{\mathrm{r}} \in \mathbb{R}^{d \times m}$ with $m = \min(T_r, d)$. We adopt the right singular vectors V_{r} as an orthonormal basis of the reference subspace $\mathrm{span}(V_r) \subset \mathbb{R}^d$, since they span directions in the shared feature space $\mathbb{R}^d$ and thus define a representation-level subspace that is comparable across answers of varying lengths. In contrast, the left singular vectors U_{r} reside in the token-index space $\mathbb{R}^{T_r}$ and are inherently tied to length of the reference sequence, making them unsuitable for comparing answers with different lengths.

We first project the hidden states of both the reference and generated answers onto the reference subspace spanned by V_r, with the projection operations formally defined as:

$$P_{\mathrm{r}\to\mathrm{r}} = H_{\mathrm{r}} V_r \in \mathbb{R}^{T_r \times m}, \qquad P_{\mathrm{g}\to\mathrm{r}} = H_{\mathrm{g}} V_r \in \mathbb{R}^{T_g \times m},$$

where $P_{\mathrm{r}\to\mathrm{r}}$ denotes the projection of reference answer tokens onto the reference subspace, and $P_{\mathrm{g}\to\mathrm{r}}$ denotes the projection of the generated answer tokens onto the same reference subspace. We then compute the token-wise projection energy as the squared ℓ_2-norm of each projected token, a metric that quantifies the alignment degree between each token and the reference subspace. The token-wise energy for the projections onto the reference subspace are:

$$(\mathbf{E}_{\mathrm{r}\to\mathrm{r}})_t = (P_{\mathrm{r}\to\mathrm{r}})_t (P_{\mathrm{r}\to\mathrm{r}})_t^\top, \qquad t = 1, \ldots, T_r$$

$$(\mathbf{E}_{\mathrm{g}\to\mathrm{r}})_t = (P_{\mathrm{g}\to\mathrm{r}})_t (P_{\mathrm{g}\to\mathrm{r}})_t^\top, \qquad t = 1, \ldots, T_g$$

where $\mathbf{E}_{\mathrm{r}\to\mathrm{r}} \in \mathbb{R}^{T_r}$ and $\mathbf{E}_{\mathrm{g}\to\mathrm{r}} \in \mathbb{R}^{T_g}$ denote the token-wise energy vectors of the reference and generated answers in the reference subspace, respectively. Similarly, we apply the same procedure to the generated answer subspace spanned by V_{g} (obtained from the SVD of H_{g}), resulting in $\mathbf{E}_{\mathrm{r}\to\mathrm{g}}$ and $\mathbf{E}_{\mathrm{g}\to\mathrm{g}}$.

3.2.3 COMPUTING LATENT REWARD

To derive a scalar reward, we aggregate the token-wise energy vector $\mathbf{E}$ into a single value. Specifically, we adopt a top-k aggregation strategy instead of directly computing the mean to avoid the influence of noisy tokens in the answer span. Intuitively, if the generated answer is semantically consistent with the reference answer, it should exhibit high projection energy in the reference subspace relative to the reference answer. We thus define the score within the reference subspace as:

$$\text{score}_{\text{r}} = \frac{\text{TopKMean}(\mathbf{E}_{\text{g}\to\text{r}})}{\text{TopKMean}(\mathbf{E}_{\text{r}\to\text{r}})}.$$

Here $\text{TopKMean}(\mathbf{E})$ denotes the mean of the k largest elements in $\mathbf{E}$, *e.g.,*

$$\text{TopKMean}(\mathbf{E}) = \frac{1}{k} \sum_{e \in \text{TopK}(\mathbf{E})} e. \tag{1}$$

Similarly, we compute the analogous energy ratio within the generated subspace, denoted as score_{g}:

$$\text{score}_{\text{g}} = \frac{\text{TopKMean}(\mathbf{E}_{\text{r}\to\text{g}})}{\text{TopKMean}(\mathbf{E}_{\text{g}\to\text{g}})}.$$

The final symmetric latent reward is defined as the average of these two scores:

$$R(x, z, y, y^\star) = \frac{\text{score}_{\text{r}} + \text{score}_{\text{g}}}{2}.$$

where x denotes the input prompt context, z denotes the model-generated CoT, y denotes the generated final answer, and $y^\star$ denotes the ground-truth reference answer.

4 EXPERIMENTS

We evaluate RLLR across multiple reasoning benchmarks and base models. Our experiments are designed to answer the following questions: (i) Does RLLR improve final reasoning performance compared to rule-based RL, model-based RL, and probability-based RL? (ii) Does RLLR remain effective for small-scale models, where hidden states may be less reliable? (iii) How do design choices (*e.g.,* layer selection and the Top-k prameter k) affect performance? (iv) Is RLLR robust to non-semantic perturbations and variations in reference answer length?

4.1 EXPERIMENTAL SETUP

Models. We conduct experiments with open-source small and medium-scale LLMs, including Qwen3 base models (0.6B, 1.7B, 4B and 8B) (Yang et al., 2025) and Llama-3.1-8B-Instruct (Grattafiori et al., 2024).

Training Data. We adopt the WebInstruct training dataset used in RLPR (Yu et al., 2025b). It contains 77K high-quality reasoning questions spanning 14 challenging domains (*e.g.,* biology, business, chemistry, and law).

Baselines. Evaluation Benchmarks. We evaluate the reasoning capabilities using multiple general reasoning and mathematical reasoning benchmarks. For mathematical reasoning tasks, we evaluate Math500 (Hendrycks et al., 2021), AMC, Minerva, AIME2024 and AIME2025. For general domains, we adopt the following benchmarks:

- MMLU-Pro (Wang et al., 2024) includes reasoning-intensive questions across diverse domains, containing 12K questions.
- GPQA Diamond contains 198 graduate-level questions covering biology, physics, and chemistry.
- SuperGPQA (Du et al., 2025b) is designed to evaluate the knowledge and reasoning abilities of LLMs across 285 graduate-level disciplines with 26K questions.
- TheoremQA (Chen et al., 2023) contains 800 questions that cover 350 theorems.

Table 1: Accuracy comparison on general-domain reasoning benchmarks.

Method	Verifier	MMLU-Pro	SuperGPQA	GPQA*	TheoremQA	WebInstruct-Val	Avg
Llama models							
Llama3.1-8B-Inst	-	46.1	21.3	25.8	**19.6**	50.8	32.7
RLVR	Rule	45.9	**25.1**	**32.1**	17.8	57.9	35.7
RLLR (ours)	✗	**48.3**	25	32	16.3	**61**	**36.5**
Qwen models							
Qwen3-8B-Base	-	50	26.2	34.1	21.9	59.1	38.3
RLVR	Rule	62.6	**33.9**	41.3	36.6	73	49.3
General Reasoner	Model	55	28.5	31.4	35.1	73	44.5
RLPR	✗	59.2	32	**42.8**	34.4	63.9	46.5
RLLR (ours)	✗	**62.9**	33.2	41.5	**36.7**	**73.2**	**49.5**

Table 2: Accuracy cbomparison on mathematical benchmarks. AIME is the average accuracy on AIME2024 and AIME2025.

Method	Math500	Minerva	AMC	AIME	Avg
Qwen3-8B-Base	44.9	21.6	27.6	0.1	23.5
RLVR	78.3	29.1	51.2	**17.1**	43.9
General Reasoner	76.2	34.1	43.1	7.1	40.1
RLPR	78.2	32.7	**53**	15.8	44.9
RLLR (ours)	**82**	**40.3**	51.4	**17.1**	**47.7**

Table 3: Robustness evaluation (ROC-AUC) of different reward methods against non-semantic perturbations and reference-length effects.

Data	RLVR	Gen. Verifier	RLPR	RLLR
Random Data	0.57	0.78	0.77	0.89
Perturbation	0.52	0.8	0.73	0.88
Long answer Data	0.52	0.78	0.61	0.89

- WebInstruct-Val, is designed to evaluate the multidisciplinary reasoning abilities of LLMs.

We compare our RLLR against three categories of RL baselines, as follows:

- Rule-based RL (RLVR). For RLVR, the reward is given by hand-crafted.
- Model-based RL (General-Reasoner-7B). General-Reasoner-7B (Ma et al., 2025) uses an verifier model, to evaluate general-domain responses.
- Probability-based RL (RLPR). RLPR (Yu et al., 2025b) adopts the sum of the probabilities of the reference answer tokens.

Implementation Details. Experiments are conducted using the GRPO (Shao et al., 2024) algorithm implemented with the verl framework (Sheng et al., 2025). For Qwen3-8B-Base and Llama3.1-8B-Instruct, we train for 300 steps with a batch size of 512 prompts. For each prompt, we sample 8 responses (*e.g.,* group size$=8$) with a sampling temperature of 1.0 in the training process. We do not apply KL regularization or KL penalties following prior works (Liu et al., 2025; Hu et al., 2025). The maximum generation length for training and evaluation is 3072 tokens. For smaller-scale models (Qwen3-4B-Base, Qwen3-1.7B-Base, Qwen3-0.6B-Base), we train for 300 steps with a batch size of 256 prompts and group size $= 6$. The maximum generation length for training and evaluation is 2048 tokens. Specifically, we extract hidden states from layer 27 for Qwen3-8B-Base and Qwen3-4B-Base, layer 21 for Qwen3-1.7B-Base and Qwen3-0.6B-Base, and layer 24 for Llama-3.1-8B-Instruct. During evaluation, the sampling temperature is set to 0.6 and top-p to 0.7. Due to the large size of MMLU-Pro and SuperGPQA, we sample one response per question for these benchmarks. For all other benchmarks, we sample 8 responses per question and report Avg@8 results (the average accuracy over eight independent runs). Please see the detailed settings in the Appendix A.

4.2 Main Results

RLLR improves general reasoning capabilities. Table 1 compares the accuracy of different methods across five general reasoning benchmarks. Overall, RLLR consistently improves models' general reasoning performance without relying on any external verifiers. In particular, RLLR improves accuracy by 11.2% over Qwen3-8B-Base and by 3.8% over Llama3.1-8B-Instruct, demonstrating the effectiveness of leveraging the model's hidden states for response quality assessment. RLLR also outperforms rule-based RLVR, and model-based General Reasoner. Moveover, RLLR surpasses

Table 4: Results of small models on general and mathematical benchmarks.

Method	MMLU-Pro	SuperGPQA	GPQA*	WebInstruct-Val	Math500	Minerva	AMC	Avg
Qwen3-0.6B-Base								
Base	16.9	10.8	16.9	25.3	31	9.6	12.2	17.6
RLVR	6.2	5.7	6.4	5.7	0	0	0	3.4
RLPR	9.8	7.6	19.2	26.3	0.1	0	0	9
RLLR (ours)	**28.9**	**16.6**	**27.4**	**50.1**	**47.1**	**12.3**	**15.7**	**28.3**
Qwen3-1.7B-Base								
Base	31.4	16.4	24.4	43.1	48	17.6	20.6	28.8
RLVR	33.8	18.9	26.8	55.3	55.4	21.3	26.3	33.9
RLPR	37.8	20.7	27.5	62.8	**64.2**	**28.5**	26.4	38.3
RLLR (ours)	**41.4**	**22.6**	**31.4**	**64.6**	61.4	20.8	**28.9**	**38.8**
Qwen3-4B-Base								
Base	43.4	22.7	27.7	48.1	46.1	23.6	31.3	34.7
RLVR	53.9	27.9	36.6	74.1	77.1	31.1	43.1	49.1
RLPR	56.3	28.5	38.8	**74.6**	77.7	32.3	**47.7**	50.8
RLLR (ours)	**60.1**	**31.5**	**42.2**	74.1	**78.7**	**33.5**	47.6	**52.5**

RLPR by 3% on Qwen3-8B-Base as well—another method that does not use external verifiers and instead uses the probability of the policy model generating reference answers as rewards. This result further highlights the advantages of computing rewards in the latent space rather than in the text space.

RLLR improves mathematical reasoning abilities. Table 2 presents the experimental results on mathematical reasoning tasks. The results are similar to those in Table 1: our RLLR method consistently outperforms the vanilla models (without GRPO training) as well as RLVR, General-Reasoner, and RLPR in terms of average performance, and achieves the best accuracy on Math500, Minerva, and AIME. This indicates that RLLR can learn strong transferable reasoning abilities.

RLLR is also effective with small-scale models. The quality of hidden states extracted from small-scale models may be inferior to those of larger-scale models. To evaluate whether RLLR remains effective on small models, we conduct experiments on Qwen3-0.6B-Base, Qwen3-1.7B-Base and Qwen3-4B-Base. As shown in Table 4, RLLR consistently improves the average accuracy and outperforms the base model as well as RLVR and RLPR on all three models. Notably, on Qwen3-0.6B-Base, RLVR and RLPR lead to substantial performance degradation, dropping by 14.2% and 8.6% compared to the base model, respectively. This indicates that the rule-based and the probability-based methods suffer from training collapse on extremely small models. In contrast, RLLR remains stable and achieves an average accuracy of 28.3%, 10% higher than the base model. These results demonstrate that RLLR can provide reliable rewards and enable stable RL training even for small-scale models with weaker hidden states.

4.3 Ablation Study

The ablation experiments are conducted on Qwen3-4B-Base with 200 training steps to ensure efficiency.

Layer Selection. To investigate which layer is optimal for hidden state extraction, we conduct experiments by comparing different layers of the Qwen3-4B-Base model. The Qwen3-4B-Base has $L = 36$ transformer layers, and we extract an early layer ($L/4$), a middle layer ($L/2$), a late layer ($3L/4$), and the last layer (L) for the experiments. Figure 4 shows that the middle-to-late layers (*e.g.,* $L/2$ and $3L/4$) yield the strongest performance, which aligns with their role in encoding higher-level semantic information (Skean et al., 2024).

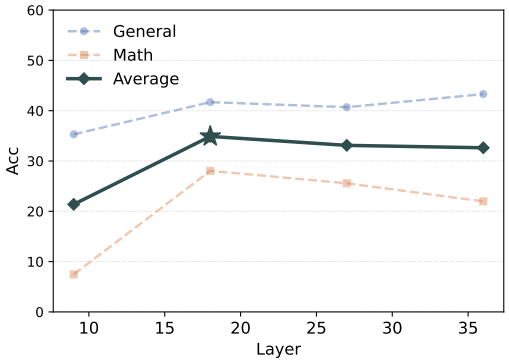


Figure 4: Performance comparison of RLLR with hidden states extracted from different transformer layers. Math: average accuracy on Math500, Minerva, and AMC benchmarks; General: average accuracy on MMLU-Pro, SuperGPQA, and GPQA diamond benchmarks; Average: average accuracy across all six benchmarks.

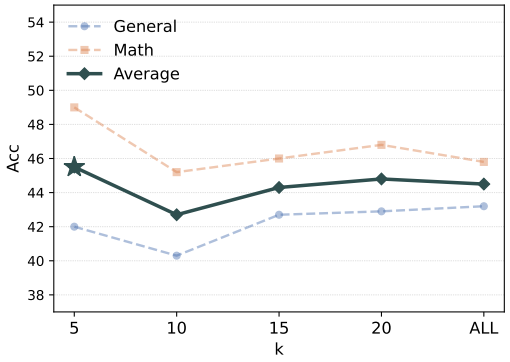


Figure 5: Performance comparison of RLLR with different values of the Top-k parameter k. Math: average accuracy on Math500, Minerva, and AMC benchmarks; General: average accuracy on MMLU-Pro, SuperGPQA, and GPQA diamond benchmarks; Average: average accuracy across all six benchmarks.

Top-k Selection. We analyze how the choice of k in the TopKMean aggregation (Eq. 1) affects model performance. We conduct experiments by varying $k \in \{5, 10, 15, 20, \text{All}\}$, where "All" represents the case where we compute the mean over all tokens. As shown in Figure 5, the optimal performance of RLLR is achieved at $k = 5$, after which the performance plateaus or slightly decreases. This suggests that while larger values of k allow capturing more tokens' contributions, they are also more likely to introduce the influence of noisy tokens. Thus, $k = 5$ achieves an optimal trade-off between acquiring sufficient information and avoiding noisy information.

4.4 Robustness Analysis

We evaluate the robustness of our latent-based reward to (i) non-semantic perturbations and (ii) reference answer length. We randomly sample 200 prompts from WebInstruct and generate 50 responses per prompt using Qwen3-4B-Base. Responses are ranked by each reward, and we compute ROC-AUC against DeepSeek-R1 (Guo et al., 2025a) annotations, where a higher AUC indicates better discriminative ability.

As shown in Table 3, on the random subset, RLLR achieves the best AUC (0.89), outperforming the model-based verifier (0.78), RLPR (0.77), and rule-based RLVR (0.57). To evaluate robustness to non-semantic perturbations, we introduce a simple perturbation by adding two leading and trailing whitespaces to the reference answers, which does not change their semantics. This reduces the AUC of RLVR and RLPR by 0.05 and 0.04, respectively, while RLLR drops by only 0.01 as shown in Table 3. Table 3 also shows that for long reference answers (> 20 tokens), RLLR maintains its AUC, whereas RLPR and RLVR degrade by 0.16 and 0.05. Overall, these results demonstrate that latent space rewards are more robust than text space rewards to superficial variations in reference answers.

5 Conclusion

In this paper, we explore evaluating the quality of model responses in the model's latent space without relying on an external verifier, thereby extending RLVR from verifiable tasks to general reasoning tasks that are difficult to verify. Extensive experiments on different models and benchmarks, along with ablation studies, show the effectiveness of our RLLR framework. Compared with rule-based, model-based, and probability-based methods that evaluate model responses quality in the text space, we find that evaluating in the latent space is more effective and less susceptible to noisy tokens and variations in the length of reference answers. This reflects that the model's latent space encodes more intrinsic and robust semantic information related to response correctness. We hope our work offers a new perspective for solving problems using the latent space of LLMs.

AI USE STATEMENT

In this work, we used generative AI tools only for polishing of the manuscript.

REFERENCES

Syeda Nahida Akter, Shrimai Prabhumoye, Matvei Novikov, Seungju Han, Ying Lin, Evelina Bakhturina, Eric Nyberg, Yejin Choi, Mostofa Patwary, Mohammad Shoeybi, et al. Nemotron-crossthink: Scaling self-learning beyond math reasoning. *arXiv preprint arXiv:2504.13941*, 2025.

Fazl Barez, Tingchen Fu, Ameya Prabhu, Stephen Casper, Amartya Sanyal, Adel Bibi, Aidan O'Gara, Robert Kirk, Ben Bucknall, Tim Fist, et al. Open problems in machine unlearning for ai safety. *arXiv preprint arXiv:2501.04952*, 2025.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering latent knowledge in language models without supervision. In *ICLR*, 2023.

Wenhu Chen, Ming Yin, Max Ku, Pan Lu, Yixin Wan, Xueguang Ma, Jianyu Xu, Xinyi Wang, and Tony Xia. Theoremqa: A theorem-driven question answering dataset. In *EMNLP*, 2023.

Yao Cheng, Jianfeng Chen, Jie Chen, Li Chen, Liyu Chen, Wentao Chen, Zhengyu Chen, Shijie Geng, Aoyan Li, Bo Li, et al. Fullstack bench: Evaluating llms as full stack coders. *arXiv preprint arXiv:2412.00535*, 2024.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Jiacheng Chen, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, et al. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025.

Sayantan Dasgupta and Trevor Cohn. Improving language model distillation through hidden state matching. In *ICLR*, 2025.

Hanwen Du, Yuxin Dong, and Xia Ning. Latent thinking optimization: Your latent reasoning language model secretly encodes reward signals in its latent thoughts. *arXiv preprint arXiv:2509.26314*, 2025a.

Xinrun Du, Yifan Yao, Kaijing Ma, Bingli Wang, Tianyu Zheng, King Zhu, Minghao Liu, Yiming Liang, Xiaolong Jin, Zhenlin Wei, et al. Supergpqa: Scaling llm evaluation across 285 graduate disciplines. *arXiv preprint arXiv:2502.14739*, 2025b.

Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Jie Shi, Xiang Wang, Xiangnan He, and Tat-Seng Chua. Alphaedit: Null-space constrained model editing for language models. In *ICLR*, 2025.

Javier Ferrando, Oscar Balcells Obeso, Senthooran Rajamanoharan, and Neel Nanda. Do i know this entity? knowledge awareness and hallucinations in language models. In *ICLR*, 2025.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

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

Jizhou Guo, Zhaomin Wu, and Philip S Yu. Reward inside the model: A lightweight hidden-state reward model for llm's best-of-n sampling. *arXiv preprint arXiv:2505.12225*, 2025b.

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

Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model. *arXiv preprint arXiv:2503.24290*, 2025.

Yuki Ichihara, Yuu Jinnai, Tetsuro Morimura, Mitsuki Sakamoto, Ryota Mitsuhashi, and Eiji Uchibe. Mo-grpo: Mitigating reward hacking of group relative policy optimization on multi-objective problems. *arXiv preprint arXiv:2509.22047*, 2025.

Ruili Jiang, Kehai Chen, Xuefeng Bai, Zhixuan He, Juntao Li, Muyun Yang, Tiejun Zhao, Liqiang Nie, and Min Zhang. A survey on human preference learning for large language models. *arXiv preprint arXiv:2406.11191*, 2024.

Hynek Kydlíček. Math-Verify: Math Verification Library, 2025. URL `https://github.com/huggingface/math-verify`.

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

Xueguang Ma, Qian Liu, Dongfu Jiang, Ge Zhang, Zejun Ma, and Wenhu Chen. General-reasoner: Advancing llm reasoning across all domains. *arXiv preprint arXiv:2505.14652*, 2025.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *NeurIPS*, 35:17359–17372, 2022.

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

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *ECCS*, 2025.

Oscar Skean, Md Rifat Arefin, Yann LeCun, and Ravid Shwartz-Ziv. Does representation matter? exploring intermediate layers in large language models. *arXiv preprint arXiv:2412.09563*, 2024.

Yi Su, Dian Yu, Linfeng Song, Juntao Li, Haitao Mi, Zhaopeng Tu, Min Zhang, and Dong Yu. Crossing the reward bridge: Expanding rl with verifiable rewards across diverse domains. *arXiv preprint arXiv:2503.23829*, 2025.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

Yiming Wang, Pei Zhang, Baosong Yang, Derek F Wong, and Rui Wang. Latent space chain-of-embedding enables output-free llm self-evaluation. In *ICLR*, 2025.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *NeurIPS*, 2024.

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

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

Tianyu Yu, Bo Ji, Shouli Wang, Shu Yao, Zefan Wang, Ganqu Cui, Lifan Yuan, Ning Ding, Yuan Yao, Zhiyuan Liu, et al. Rlpr: Extrapolating rlvr to general domains without verifiers. *arXiv preprint arXiv:2506.18254*, 2025b.

Chenheng Zhang, Tianqi Du, Jizhe Zhang, Mingqing Xiao, Yifei Wang, Yisen Wang, and Zhouchen Lin. Language ranker: A lightweight ranking framework for llm decoding. In *NeurIPS*, 2025.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, et al. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.

Xiangxin Zhou, Zichen Liu, Anya Sims, Haonan Wang, Tianyu Pang, Chongxuan Li, Liang Wang, Min Lin, and Chao Du. Reinforcing general reasoning without verifiers. *arXiv preprint arXiv:2505.21493*, 2025.

Zhenhong Zhou, Haiyang Yu, Xinghua Zhang, Rongwu Xu, Fei Huang, and Yongbin Li. How alignment and jailbreak work: Explain llm safety through intermediate hidden states. In *EMNLP*, 2024.

A EXPERIMENTAL DETAILS

In our implementation of RLLR, we extract hidden states from the $3L/4$-th transformer block and set the Top-k parameter to $k = 5$. Specifically, we use layer 27 for Qwen3-8B-Base and Qwen3-4B-Base, layer 21 for Qwen3-1.7B-Base and Qwen3-0.6B-Base, and layer 24 for LLaMA-3.1-8B-Instruct. Table 5 shows more experiments details.

Table 5: Implementation details of Experiments.

Experiment Table/Figure	Batch Size	Mini Batch Size	Training Steps	Clip Threshold	Lr	Scheduler
Table1, Table2	512	256	300	(0.2,0.27)	1e-6	constant
Table3	256	256	300	(0.2,0.27)	1e-6	constant
Table4, Table5	256	256	200	(0.2,0.27)	1e-6	constant

B PROMPT

Prompt

system:
A conversation between User and Assistant. The user asks a question, and the Assistant should solve it step by step and present the final answer in the following format: `\boxed{X}`, where X is the final answer.
user:
{Question}