Published as a workshop paper at the 1st Workshop on Scaling Post-training for LLMs (SPOT), ICLR 2026.

Beyond Scalar Critics: A Distributional Perspective on Reinforcement Learning with Verifiable Rewards for LLMs

Jinyi Liu[1], **Yiboyun Chen**[2], **Hongyao Tang**[1]*, **Yi Ma**[3], **Shuyue Hu**[4], **Yang Chen**[4], **Fei Ni**[5], **Qiaosheng Zhang**[4], **Lei Bai**[4], **Yan Zheng**[2]*, **Jianye Hao**[1]*
[1]School of Computer Software, Tianjin University
[2]School of New Media and Communication, Tianjin University
[3]School of Computer and Information Technology, Shanxi University
[4]Shanghai Artificial Intelligence Laboratory
[5]Imperial College London

Abstract

Reinforcement learning with verifiable rewards (RLVR) has become prevalent for LLM post-training, yet its reward signal is often terminal and near-binary, yielding prompt-conditional return distributions that are frequently long-tailed. Standard scalar critics usually adopted in RLVR obscure the distributional return structures and attenuate tail information, leading to less informative advantages and reduced optimization stability. To this end, we focus on modeling the return distribution for LLM RL fine-tuning and propose DistRLVR, a unified distributional RLVR framework that learns a critic with both categorical and quantile distributions. To stabilize distribution learning under long-horizon and terminal-sparse rewards, we introduce dual Sample-Replacement targets to diversify supervision. Building on the learned return distributions, we develop tail-aware advantage shaping that selectively amplifies informative tails. Across a range of mathematical reasoning benchmarks, DistRLVR delivers consistent gains in sample efficiency, Pass@k and average performance, achieving a 24.1% overall improvement over PPO. These results suggest that exploiting distributional structure is a practical and promising direction for more reliable RLVR post-training.

1 Introduction

Large Language Models (LLMs) are increasingly post-trained with Reinforcement Learning (RL) to improve reasoning, tool use, and instruction following abilities Feng et al. (2025); Guo et al. (2025); Shao et al. (2024). In RL with verifiable rewards (RLVR), supervision is provided by objective verifiers, often terminal, sparse, and delayed. For a fixed prompt, stochastic decoding produces diverse rollouts, while verification typically yields near-binary outcomes, causing prompt-conditional return distributions that are frequently bimodal and long-tailed. These rare tail events are precisely where learning signal concentrates. Accurately modeling tail cases improves value estimation fidelity, providing a more informative and stable baseline for policy updates, and selectively upweighting high-return tails may accelerate the discovery and consolidation of effective reasoning. Despite this, prevailing methods such as PPO Schulman et al. (2017) summarize future outcomes with a scalar value function to estimate only the mean return. As a result, scalar critic inevitably compresses dispersion and tail structure, weakening advantage informativeness and slowing learning.

The limitations of reducing a random return to its expectation have long been studied in RL literature Bellemare et al. (2017); Hessel et al. (2018). To address this issue, Distributional RL learns a full return distribution rather than a point estimate, commonly via categorical Bellemare et al. (2017) or quantile-based representations Dabney et al. (2018b;a). Beyond improved prediction, distributional critics also support risk-sensitive control by operating on tail statistics of the learned

*Corresponding authors: jianye.hao@tju.edu.cn, yanzheng@tju.edu.cn, tanghongyao@tju.edu.cn.

distribution Liu et al. (2024); Ma et al. (2025). Motivated by these insights, distributional perspectives have begun to appear in LLM alignment, for example through quantile reward models in RLHF Dorka (2024) and distributional value learning under KL-regularized formulations with restrictive assumptions Zhou et al. (2025). However, RLVR constitutes a distinct and underexplored regime. Existing risk-seeking variants such as RS-GRPO Jiang et al. (2025) inject tail preference by reweighting advantages computed from groups of sampled completions at the outcome level, but they do not learn a token-conditional critic. As a consequence, tail information is not generalized beyond the sampled completions, and long-horizon credit assignment remains largely driven by sparse trajectory-level outcomes.

In this paper, we propose **DistRLVR**, a unified framework for RLVR that models token-level return distributions and exploits them for stable, tail-aware policy optimization. We first adapt standard distributional critics to the token-level episodic MDP used in RLVR, covering categorical atoms Bellemare et al. (2017), quantile regression Dabney et al. (2018b), and implicit quantile representations Dabney et al. (2018a) within a common training pipeline. Second, to further improve the efficiency of the distribution modelling, we introduce *dual Sample-Replacement* (dSR) multi-step distributional targets to alleviate the strong temporal correlation caused by one-step bootstrapping under terminal-sparse verification. Moreover, to explicitly exploit the tail behavior, we develop *tail-aware advantage shaping*, which maps tail statistics into PPO-compatible advantages, enabling explicit control over rare successes and enhancing policy updates.

In the experiments, we evaluate DistRLVR on a range of mathematical-reasoning RLVR benchmarks under both i.i.d. and distribution shift settings. Our results demonstrate that distributional critics consistently outperform scalar-critic PPO in both Pass@k and average accuracy, achieving a 24.1% relative improvement over PPO with superior sample efficiency in the early stages. Our dSR targets are crucial for stabilizing distributional value learning, while exploiting tail information also yields additional gains. Crucially, we observe that our methods enable the policy to reduce entropy and commit to high-quality solutions significantly faster than PPO, achieving robust convergence with lower variance while avoiding brittle mode collapse.

Our contributions are summarized as follows. (i) We propose DistRLVR, a unified distributional framework for LLM post-training that learns token-level return distributions to enable stable and even tail-aware optimization. (ii) We introduce an efficient dual Sample-Replacement (dSR) targets, to mitigate bootstrapping bias under terminal-sparse verification. dSR significantly stabilizes critic learning, delivering consistent improvements in sample efficiency and final Pass@k. (iii) We empirically demonstrate that distributional value supervision facilitates more decisive yet robust policy optimization. Our results show that DistRLVR accelerates commitment to high-quality solutions while avoiding the brittle mode collapse typical of scalar-based methods.

2 Related Work

In the realm of RLHF, human preference data often suffers from noise, subjectivity, or distributional shifts Kaufmann et al. (2025). Motivated by the limitations of mean-based training signals, several recent works incorporate distributional perspectives to improve robustness and stability. DVPO proposes an asymmetric risk-control formulation to mitigate the impact of noisy supervision and stabilize policy optimization Zhu et al. (2025a). Quantile Reward Models (QRMs) model preference feedback via quantile regression, learning a reward distribution rather than a point estimate Dorka (2024). Q# further leverages distributional value learning to characterize the return distribution of a reference policy and derives an optimal KL-regularized solution under specific assumptions Zhou et al. (2025). These methods demonstrate that distributional modeling can be beneficial in preference-based alignment. However, these works primarily focus on preference distributions or distributional analyses under preference supervision only.

Recent RLVR methods have focused on improving rollout efficiency and data reuse Yan et al. (2025); Liang et al. (2025), stabilizing optimization via refined objectives Yu et al. (2025), and encouraging exploration Anonymous (2026); Peng et al. (2025) to enhance reasoning performance Yue et al. (2025). Besides, more recent progress has emphasized objectives that utilize multiple rollouts or risk-sensitive shaping to exploit rare high-reward outcomes. FlowRL incorporates the concept of flow balance, inspired by GFlowNets, aiming to align the sample distribution generated by the policy with the reward distribution Zhu et al. (2025b). Risk-Sensitive GRPO extends GRPO by using

group rollouts to approximate outcome-level return statistics and pursue high-reward tails Jiang et al. (2025). However, these approaches typically rely on outcome-level heuristics or rollout statistics and do not learn a token-level distributional critic that can generalize and provide fine-grained credit assignment in long-horizon sequence generation. Moreover, while risk-seeking shaping has been studied in RLVR, the use of a learned distributional critic to stabilize bootstrapped value learning under terminal-sparse verification and provide principled tail-aware supervision remains underexplored.

3 METHOD

RLVR commonly relies on terminal verification rewards that must be credit-assigned over long token sequences, while realized returns remain highly stochastic due to completion diversity. Mean-only value estimation used in existing RLVR algorithms obscures this return-distribution structure, limiting critic fidelity and attenuating informative tail events. To address these issues, we propose DISTRLVR, a distributional RLVR framework comprising: (i) a *token-level distributional critic* $Z_\phi(s_t)$ that models the prompt-conditional return distribution beyond its mean; (ii) *Sample-Replacement* multi-step distributional targets that stabilize bootstrapped critic learning; and (iii) *tail-aware advantage shaping* that extracts PPO-compatible learning signals from distributional functionals, enabling controllable tail sensitivity in policy updates. Note that due to page constraints, the preliminary discussions on MDPs, value distributions, and related concepts have been moved to the appendix A.

3.1 DISTRIBUTIONAL CRITIC IN TOKEN-LEVEL MDPS

Building on the token-level MDP in Sec. A.1, we model the return random variable $Z^\pi(s_t)$ and learn a critic $Z_\phi(s) \approx Z^\pi(s)$. We start from the standard one-step distributional TD target with a target network $\bar{\phi}$:

$$Z_t^{\text{tgt}} \triangleq r_t + \gamma Z_{\bar{\phi}}(s_{t+1}). \tag{1}$$

Eq. equation 1 is a standard one-step bootstrap target. However, in terminal-sparse RLVR, we argue that such a standard TD(0) target becomes sample-inefficient for learning $Z^\pi(s_t)$. In Sec. 3.2, we introduce a more informative target that better utilizes each rollout to supervise prefix distributions.

Given the target values, we train the distributional critic by regressing its predicted return distribution $Z_\phi(s_t)$ toward a target return distribution Z_t^{tgt}, by minimizing a standard distributional regression objective,

$$\mathcal{L}_{\text{critic}}(\phi) = \mathbb{E}_t\Big[\ell_{\text{dist}}\big(Z_\phi(s_t),\, Z_t^{\text{tgt}}\big)\Big], \tag{2}$$

where ℓ_{dist} is the canonical loss in distributional RL (defined in Appendix D). We next describe the distributional parameterizations of Z_ϕ.

To demonstrate the flexibility of DISTRLVR, we instantiate it using three widely-adopted parameterizations that represent typical approaches to distribution modeling:

Categorical (C51) Bellemare et al. (2017): We represent $Z_\phi(s)$ as a categorical distribution over a fixed support $\{v_\ell\}_{\ell=1}^L$, defined as $Z_\phi(s) = \sum_{\ell=1}^L p_\phi^\ell(s)\,\delta_{v_\ell}$, where $p_\phi^\ell(s) \geq 0$ and $\sum_\ell p_\phi^\ell(s) = 1$. The target distribution is projected onto this support using the projection operator Π, and the critic is optimized via cross-entropy loss.

Quantile Regression (QR) Dabney et al. (2018b): We approximate the distribution by predicting $N_q \in \mathbb{N}$ fixed quantiles $\hat{z}_\phi(s;\tau_i)$ at cumulative probabilities $\tau_i = \frac{i-\frac{1}{2}}{N_q} \in (0,1)$. The model is trained to minimize the pairwise quantile Huber regression loss against the target quantile set $\{\hat{z}_{\bar{\phi}}(s_{t+1};\tau_j)\}_{j=1}^{N_q}$.

Implicit Quantiles (IQN) Dabney et al. (2018a): Instead of fixed quantiles, we learn a continuous mapping by sampling $\tau \sim \mathcal{U}(0,1)$ to predict $\hat{z}_\phi(s;\tau)$. This parameterization is trained using the same pairwise quantile regression loss, allowing for arbitrary resolution of the return distribution.

3.2 DISTRIBUTIONAL MULTI-STEP TARGETS VIA SAMPLE REPLACEMENT

In token-level RLVR, verification rewards are typically terminal and sparse, so $r_t \approx 0$ for most prefix steps. Consequently, the standard one-step distributional TD target $Z_t^{\text{tgt}} = r_t + \gamma Z_{\bar{\phi}}(s_{t+1})$ is often

Table 1: Performance comparison. We report mean over three seeds. Best results are **underlined and bold**, second best are **bold**.

Method	AIME24		AIME25		AMC23		Olympiad		Avg.	
	avg.32	p@128	avg.32	p@128	avg.32	p@128	avg.32	p@128	avg	p@128
Base Model	0.000	0.000	0.000	0.000	3.256	4.819	0.296	8.296	0.888	3.279
PPO	5.104	43.333	3.264	26.667	31.903	86.345	23.012	**65.975**	15.821	55.580
DistRLVR (0)	7.535	46.667	5.452	**37.778**	**46.233**	88.353	28.611	**65.531**	**21.958**	59.582
DistRLVR (SR)	**8.125**	**53.333**	**6.042**	35.556	43.737	**90.361**	**29.241**	65.126	21.786	**61.094**
DistRLVR (Tail)	**9.549**	**54.444**	**7.361**	**36.667**	**44.829**	**89.157**	**31.451**	62.914	**23.298**	**60.796**

Method	MATH500		Minerva		M2M(4k)		CountDown	Avg.	
	avg.32	p@128	avg.32	p@128	avg.128	p@128	avg.128	avg	p@128
Base Model	3.256	4.200	1.241	26.103	13.45	54.70	15.98	8.482	28.334
PPO	62.590	**94.667**	18.053	**64.951**	17.17	62.18	64.97	40.696	73.933
DistRLVR (0)	69.969	**95.067**	21.944	61.520	20.43	66.88	68.94	45.321	**74.489**
DistRLVR (SR)	**71.025**	94.533	**22.522**	**62.377**	**20.80**	**69.23**	**69.85**	**46.049**	**75.380**
DistRLVR (Tail)	**72.623**	94.267	**22.794**	57.721	**21.29**	**67.09**	**69.80**	**46.627**	73.026

dominated by the bootstrap term, i.e., $Z_t^{\mathrm{tgt}} \approx \gamma Z_{\bar{\phi}}(s_{t+1})$. This makes highly correlated supervision across adjacent prefixes and can compound critic errors over long horizons. Moreover, because the verification signal is concentrated at the terminal step, TD(0) propagates distributional information backward through a long chain of local bootstraps, providing weak and noisy supervision for learning return dispersion and tail behavior (e.g., rare successes under stochastic decoding).

A natural fix is to expose prefix states to longer-horizon backups whose endpoints are closer to the terminal verification step. However, explicitly constructing and training on all n-step distributional targets is computationally prohibitive at LLM horizons and would destroy computational efficiency. We therefore follow GMAC Nam et al. (2021) and adopt a bootstrapped distributional λ-return that mixes backup depths to increase target diversity while moderating error amplification, together with Sample Replacement as an efficient sampler that generates many per-token targets via a single backward sweep. We next define the target distribution and then describe our instantiation to improve trajectory coverage and per-state target diversity.

Bootstrapped multi-step distributional targets Let $H_t \triangleq T - t$ denote the remaining horizon from timestep t. For any $n \in \{1, \ldots, H_t\}$, define the n-step bootstrapped return distribution

$$\widetilde{Z}_t^{(n)} \overset{D}{=} \sum_{i=0}^{n-1} \gamma^i r_{t+i} \; + \; \gamma^n Z_{\bar{\phi}}(s_{t+n}), \tag{3}$$

where $Z_{\bar{\phi}}$ is stop-gradient and $Z_{\bar{\phi}}(s_T)$ is degenerate at r_T. Larger n anchors the target closer to terminal verification, reducing reliance on purely local bootstrapping.

Following GMAC Nam et al. (2021), we mix backup depths via a truncated geometric distribution controlled by $\lambda_{\mathrm{SR}} \in [0, 1)$, where SR continues the backup with probability λ_{SR} and performs replacement with probability $1 - \lambda_{\mathrm{SR}}$. Concretely, sample a backup depth $N \in \{1, \ldots, H_t\}$ with $\Pr(N = n) = (1 - \lambda_{\mathrm{SR}})\lambda_{\mathrm{SR}}^{n-1}$ for $n < H_t$ and $\Pr(N = H_t) = \lambda_{\mathrm{SR}}^{H_t - 1}$, and define the target draw as $\widetilde{Z}_t^{(\lambda_{\mathrm{SR}})} \sim \widetilde{Z}_t^{(N)}$. This randomized mixture increases target diversity while reducing sensitivity to long-horizon error compounding under terminal-sparse verification.

Directly materializing all n-step targets is costly for long horizons. SR instead generates M target draws per visited prefix state in $O(MT)$ time via a single backward sweep. For each $m \in [M]$, set $g_T^{(m)} \leftarrow r_T$ and for $t = T - 1, \ldots, 0$ sample $b_t^{(m)} \sim \mathrm{Bernoulli}(\lambda_{\mathrm{SR}})$ and (when $b_t^{(m)} = 0$) a replacement draw $\tilde{z}_{t+1}^{(m)} \sim Z_{\bar{\phi}}(s_{t+1})$, then

$$g_t^{(m)} \leftarrow r_t + \gamma\Big(b_t^{(m)}\, g_{t+1}^{(m)} + (1 - b_t^{(m)})\, \tilde{z}_{t+1}^{(m)}\Big). \tag{4}$$

Table 2: Comparison of distributional critic heads in DistRLVR, under vanilla TD(0) and dSR targets. We report mean over 3 seeds.

Method	AIME24		AIME25		AMC23		MATH500		Minerva		Olympiad		Avg.
	avg.32	p@128	avg.32	p@128	avg.32	p@128	avg.32	p@128	avg.32	p@128	avg.32	p@128	
Vanilla TD(0) targets													
C51	7.118	46.667	5.694	**37.778**	42.934	88.755	**71.031**	94.000	22.047	**62.500**	29.295	64.938	29.687
QR	7.639	**47.778**	5.590	35.556	42.432	87.550	70.746	94.533	22.093	60.172	**29.466**	64.889	29.661
IQN	7.535	46.667	5.451	**37.778**	42.633	88.353	69.969	**95.067**	21.944	61.520	28.611	**65.531**	29.357
dSR targets													
C51	7.778	45.556	5.139	34.444	42.495	88.755	70.242	94.533	**22.610**	**63.725**	29.023	**65.481**	29.548
QR	**8.090**	**50.000**	**5.972**	**41.111**	**44.089**	**89.157**	70.767	**94.867**	21.569	61.275	**29.531**	64.938	**30.003**
IQN	**8.125**	**53.333**	**6.042**	35.556	**43.737**	**90.361**	**71.025**	94.533	**22.522**	62.377	29.241	65.136	**30.115**

[SR samples the truncated-geometric mixture target] Conditioning on a trajectory and $Z_{\bar{\phi}}$ with $Z_{\bar{\phi}}(s_T)$ degenerate at r_T, the recursion in Eq. equation 4 yields $g_t^{(m)} \overset{D}{=} \widetilde{Z}_t^{(\lambda_{\mathrm{SR}})}$ for any t. In terminal-sparse RLVR, this turns each sampled trajectory into dense per-prefix distributional supervision beyond TD(0). A proof is deferred to Appendix C.

SR yields M i.i.d. samples $\{g_t^{(m)}\}_{m=1}^M$ whose marginal matches the target return distribution $Z_t^{\mathrm{tgt}} \triangleq \widetilde{Z}_t^{(\lambda_{\mathrm{SR}})}$ (Lemma 3.2). We update the distributional critic by minimizing the standard distributional loss as in Eq. 2, implemented by plugging $\{g_t^{(m)}\}$ as target draws.

Practical instantiation: dual Sample Replacement To make SR effective for RLVR, we instantiate it with a two-level sampling protocol, denoted dSR. First, to better approximate the return distribution under terminal verification, we draw a group of K independent trajectories for each prompt $\mathbf{x}$: $\{\tau^{(k)}\}_{k=1}^K \sim \pi_\theta(\cdot \mid \mathbf{x})$. This improves trajectory coverage conditioned on the same prompt, making rare tail events more observable and stabilizing estimation of distributional quantities. Second, given a sampled trajectory $\tau^{(k)} = \{(s_t^{(k)}, r_t^{(k)})\}_{t=0}^{T_k}$, we apply the SR recursion to generate M SR targets $\{g_t^{(m,k)}\}_{m=1}^M$ for each visited token state $s_t^{(k)}$ via a single backward sweep.

In other words, dSR produces a per-token, multi-sample target set for each prefix state, which is a small empirical return distribution rather than a single scalar or single-sample target. The resulting training batch is

$$\mathcal{D}_{\mathrm{SR}} = \bigcup_{k=1}^{K} \bigcup_{t=0}^{T_k-1} \left\{ \left(s_t^{(k)},\ g_t^{(m,k)} \right) \right\}_{m=1}^{M}, \tag{5}$$

with K controlling completion coverage per prompt and M controlling the number of target draws per visited prefix.

In this way, dSR replaces TD(0) supervision in terminal-sparse RLVR with a randomized mixture over multiple backup depths, stabilizing distribution learning at long horizons. Computationally, SR generates M targets for all prefixes in $O(MT)$ time via a single backward sweep, avoiding the $O(T^2)$ cost of explicitly materializing all n-step targets. The analysis details are deferred to Appendix C. Besides, due to page limitations, the discussion on better leveraging tail information has been moved to the appendix B.

Training Step Summary.
At each iteration, we perform the following updates:

(i) **Critic Update:** Build distributional critic targets using TD(0) (Eq. equation 1) or dSR (Eq. equation 4) and update Z_ϕ with the head-specific loss (C51/QR/IQN, see Appendix D).
(ii) **Advantage Estimation:** Compute scalar values via $V_\phi(s) = \mathbb{E}[Z_\phi(s)]$ and obtain GAE advantages $\hat{A}_t$ (Sec. A.3).
(iii) **Tail Shaping:** Apply tail reweighting to obtain $\hat{A}_t^{\mathrm{tail}}$ (Eqs. equation 11); setting $w \equiv 1$ recovers the default update relying solely on improved value estimation.
(iv) **Policy Update:** Update the actor using the PPO clipped objective (Eq. equation 7).

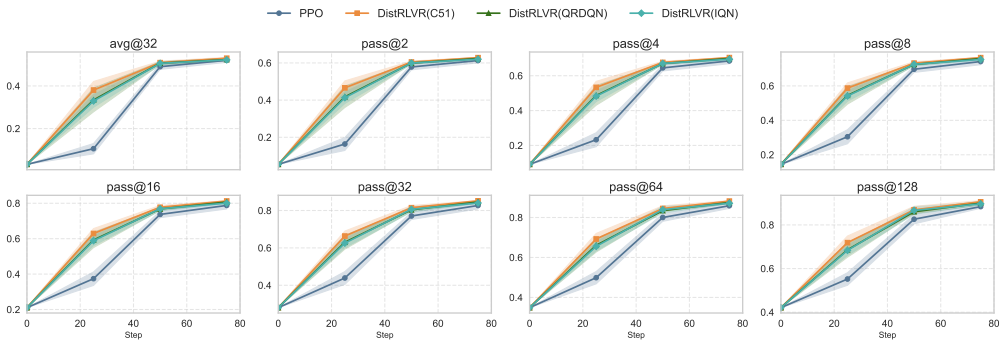

Figure 1: The performance versus training step for PPO and DistRLVR with various heads.

Table 3: Performance comparison of tail-shaping variants (mean over 3 seeds).

Method	AIME24		AIME25		AMC23		MATH500		Minerva		Olympiad		Avg.	P@k
	avg.32	p@128	avg.32	p@128	avg.32	p@128	avg.32	p@128	avg.32	p@128	avg.32	p@128		
IQN(SR)	8.125	**53.333**	6.042	35.556	43.737	**90.361**	71.025	**94.533**	22.522	**62.377**	29.241	**65.136**	30.115	**66.883**
Averse	8.021	47.778	**7.569**	34.444	**45.582**	87.952	72.196	93.533	**23.231**	55.025	30.918	61.877	31.253	63.435
Seeking	**9.688**	52.222	6.910	**41.111**	44.691	88.353	71.890	**94.067**	23.081	**57.721**	30.761	**63.457**	31.170	**66.155**
Tail	**9.549**	**54.444**	**7.361**	**36.667**	**44.829**	**89.157**	**72.623**	94.267	**22.794**	**57.721**	**31.451**	62.914	**31.434**	65.861

4 Experiments

In this section, we conduct experiments to address key questions: (i) Do distributional critics improve both final performance and sample efficiency over scalar critics in RLVR? (ii) Can explicitly leveraging tail information from the learned return distribution further improve performance?

4.1 Experimental Setup

Datasets We focus on mathematical reasoning tasks and use three distinct datasets for training and evaluation. **(i) DeepScaleR training corpus.** We train on the *DeepScaleR-Preview-Dataset* Luo et al. (2025), a widely used RLVR math corpus containing ∼40K problem–answer pairs sourced from AIME (1984–2023), AMC (pre-2023), Omni-MATH Gao et al. (2024), and Still Min et al. (2024). The result models are evaluated on a diverse set of held-out benchmarks, including AIME'24, AIME'25, AMC'23, Minerva Lewkowycz et al. (2022), OlympiadBench He et al. (2024), and MATH500 Lightman et al. (2023). **(ii) Low-data distribution shift (M2M).** To probe generalization under limited training data and distribution shift, we train on a subset of MATH500 and evaluate on Minerva (denoted MATH500→Minerva, M2M), where the evaluation set differs in problem formulation and difficulty distribution. **(iii) In-domain i.i.d. (Countdown).** We additionally train and evaluate on Countdown using disjoint splits, representing an in-domain i.i.d. setting.

Training and evaluating details We use DeepSeek-R1-Distill-Qwen-1.5B as the base policy. Unless otherwise specified, the maximum generation length is 1,024 tokens. All experiments are implemented on top of VeRL Sheng et al. (2025). Starting from its PPO implementation, we replace the scalar critic with a distributional critic. For risk-related variants in Sec. B, we only modify the actor update as specified there. We train each method with 3 random seeds. To ensure a fair comparison, we match all shared hyperparameters across runs (optimizer, batch size, rollout temperature, etc., see Appendix E).

Inference and reporting For each method, we evaluate 3 checkpoints (one per seed), sampling 128 rollouts per question and report the mean across seeds, while standard deviations are provided in the Appendix E. Unless otherwise stated, DistRLVR(0) and DistRLVR(SR) refer to the methods in Secs. 3.1-3.2 using TD(0) targets and the proposed dSR target construction, respectively, both with the standard PPO actor update. DistRLVR(Tail) further incorporates the tail-aware advantage shaping in Sec. B. Unless explicitly noted, all DistRLVR variants use the IQN parameterization for the critic by default.

Figure 2: The results of different rollout length.

4.2 Incorporating Value Distributions Improves Performance

Distributional critics improve final performance Across all three training settings, DistRLVR consistently outperforms scalar-critic PPO on both average and Pass@k performance (Table 1). These gains support our central claim: under terminal, near-binary verification and stochastic decoding, the prompt-conditional return is often heavy-tailed or bimodal, and learning the return distribution yields a more informative and better-generalizing baseline than a mean-only value estimate. In terms of cost, since rollout generation dominates wall-clock time, the addition of distributional critic learning results in minimal extra time overhead, with performance gains achieved without a significant increase in training duration (Appendix E.3).

Our dSR targets stabilize distribution learning Replacing distributional TD(0) with dSR further improves performance and yields more stable critic training. On average, distributional TD(0) attains an overall score of 33.639, improving over PPO (28.258) by 19.4%. Our dSR further increases the average gain to 20.4% relative to PPO. Moreover, dSR outperforms TD(0) on the majority of benchmarks, indicating more consistent and stable critic learning under terminal-sparse verification. The results also show that the improvements brought by DistRLVR are more pronounced on harder tasks, suggesting that better distributional credit assignment becomes increasingly valuable as reasoning difficulty increases.

DistRLVR significantly improves sample efficiency We assess sample efficiency from the training learning curves. Figure 2 shows that DistRLVR improves more rapidly than PPO across all three distributional heads (C51/QR-DQN/IQN). At the same training steps, DistRLVR yields consistently higher avg@32 and Pass@k for all k, and the separation from PPO persists (and often widens) as training proceeds. This pattern indicates that modeling return distributions provides more informative token-level credit assignment under stochastic completions, allowing the actor to exploit useful learning signals earlier and translate each rollout budget into larger performance gains.

Effect of different distributional critic parameterizations We compare standard distributional heads (C51/QR/IQN), and within each head contrast vanilla TD(0) targets with dSR targets. As shown in Table 2, dSR yields consistent (albeit sometimes modest) improvements over TD(0) across parameterizations on most benchmarks, supporting the benefit of multi-step distributional targets for more effective credit propagation and more stable distribution learning under terminal-sparse verification. Among the three heads, IQN combined with dSR achieves the strongest and most consistent gains. We therefore adopt IQN+dSR as the DistRLVR default configuration in our experiments.

Leveraging Tail Information Further Improves RLVR Table 1 shows that incorporating tail information on top of DistRLVR yields further gains. In particular, the tail-reweighting variant achieves an average improvement of 24.1% over PPO, indicating that explicitly emphasizing informative tail rollouts can substantially amplify the benefits of distributional value modeling.

Method	M2M(1k)		M2M(1k16n)	
	avg.	p@128	avg.	p@128
Base Model	1.20	20.51	1.20	20.51
PPO	13.05	56.84	13.05	56.84
DistRLVR (0)	14.86	**62.25**	15.79	**61.68**
DistRLVR (SR)	**14.93**	59.69	**16.07**	**60.97**
DistRLVR (Tail)	**14.87**	**60.11**	**15.84**	60.68

Figure 3: M2M ablations across context lengths.

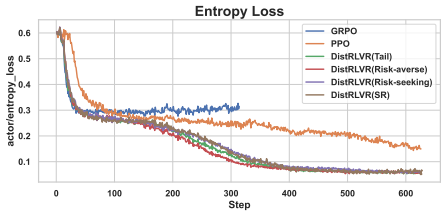

Figure 4: Entropy dynamics.

We also evaluate two standard risk-sensitive alternatives, *risk-seeking* and *risk-averse*, which directly modify the advantage estimator rather than reweighting a fixed advantage (definitions in the Appendix). Table 3 shows that *risk-seeking* is generally competitive with tail reweighting, consistent with both strategies prioritizing high-return tails. In contrast, *risk-averse* achieves comparable average scores but yields noticeably lower Pass@k, suggesting that conservative, low-tail-focused updates improve mean behavior only weakly while reducing the probability of sampling high-quality solutions under best-of-k evaluation. Taken together, these results indicate that distributional critics provide consistent gains in RLVR, and that explicitly emphasizing the high-return tail further improves best-of-k performance.

4.3 Discussion and Ablation Study

Effect of rollout count Figure 2 also highlights the effect of rollout count. Increasing the number of rollouts consistently improves performance and accelerates learning, which aligns with our motivation that richer Monte-Carlo coverage provides the critic with more informative return samples and is particularly important for accurately fitting return distributions. More importantly, when controlling for rollout count, DistRLVR still yields clear gains. In particular, under the same group-rollout budget (n=8), DistRLVR consistently outperforms GRPO, demonstrating that the improvements are not merely due to using more rollouts, but come from stronger distributional value modeling and the resulting credit-assignment signal.

Robustness to output length We evaluate robustness under long-horizon rollouts by training and testing with maximum generation lengths of 1K and 4K under the same protocol. While the 4K setting generally yields higher absolute performance, the gains from distributional value modeling remain consistent across both horizons.

Entropy dynamics Figure 4 reveals a clear separation in exploration dynamics. All methods reduce entropy rapidly at the beginning, consistent with quickly learning basic solution patterns. Compared with PPO, DistRLVR variants exhibit a substantially faster and deeper entropy decline and converge to a much lower entropy level, indicating earlier commitment to a more deterministic policy. In contrast, PPO decreases entropy more gradually, while GRPO remains at a relatively high-entropy plateau. Importantly, this accelerated entropy reduction is accompanied by higher Pass@k throughout training (Figure 1/Figure 2), suggesting improved learning efficiency rather than harmful mode collapse.

5 Summary

This paper investigates how value distribution matters for RLVR in LLM post-training. We propose DistRLVR, a unified framework that learns a token-level distributional critic, incorporating typical distribution modeling approaches and an improved dSR target construction. Our results support two takeaways. First, accurate token-level return-distribution modeling improves learning effectiveness and sample efficiency in terminal-sparse RLVR. Second, explicitly exploiting distributional structure—e.g., via tail reweighting or other principled emphasis on salient regions of the return distribution—yields additional gains beyond stronger critics. While we learn rich return distributions that expose informative tail behavior, our current exploitation mechanisms are still preliminary. How to best use tail information is likely task-dependent across downstream objectives. Developing principled, adaptive, and task-aware tail strategies is an important direction left for future work.

REFERENCES

Anonymous. Pass@k training for adaptively balancing exploration and exploitation of large reasoning models, 2026. URL https://openreview.net/forum?id=eslxxopXTF.

Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International conference on machine learning*, pp. 449–458. PMLR, 2017.

Will Dabney, Georg Ostrovski, David Silver, and Rémi Munos. Implicit quantile networks for distributional reinforcement learning. In *International conference on machine learning*, pp. 1096–1105. PMLR, 2018a.

Will Dabney, Mark Rowland, Marc Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018b.

Nicolai Dorka. Quantile regression for distributional reward models in rlhf. *arXiv preprint arXiv:2409.10164*, 2024.

Sicheng Feng, Gongfan Fang, Xinyin Ma, and Xinchao Wang. Efficient reasoning models: A survey. *arXiv preprint arXiv:2504.10903*, 2025.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, et al. Omni-math: A universal olympiad level mathematic benchmark for large language models. *arXiv preprint arXiv:2410.07985*, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. 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)*, pp. 3828–3850, 2024.

Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Gheshlaghi Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In Sheila A. McIlraith and Kilian Q. Weinberger (eds.), *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pp. 3215–3222. AAAI Press, 2018.

Yuhua Jiang, Jiawei Huang, Yufeng Yuan, Xin Mao, Qianchuan Zhao, Lin Yan, et al. Risk-sensitive reinforcement learning for alleviating exploration dilemmas in large language models. In *The 5th Workshop on Mathematical Reasoning and AI at NeurIPS 2025*, 2025.

Timo Kaufmann, Paul Weng, Viktor Bengs, and Eyke Hüllermeier. A survey of reinforcement learning from human feedback. *Trans. Mach. Learn. Res.*, 2025, 2025. URL https://openreview.net/forum?id=f7OkIurx4b.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *Advances in neural information processing systems*, 35:3843–3857, 2022.

Jing Liang, Hongyao Tang, Yi Ma, Jinyi Liu, Yan Zheng, Shuyue Hu, Lei Bai, and Jianye Hao. Squeeze the soaked sponge: Efficient off-policy reinforcement finetuning for large language model. *CoRR*, abs/2507.06892, 2025.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023.

Jinyi Liu, Zhi Wang, Yan Zheng, Jianye Hao, Chenjia Bai, Junjie Ye, Zhen Wang, Haiyin Piao, and Yang Sun. Ovd-explorer: Optimism should not be the sole pursuit of exploration in noisy environments. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 13954–13962, 2024.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Li Erran Li, et al. Deepscaler: Surpassing o1-preview with a 1.5 b model by scaling rl. *Notion Blog*, 2025.

Xiaoteng Ma, Junyao Chen, Li Xia, Jun Yang, Qianchuan Zhao, and Zhengyuan Zhou. Dsac: Distributional soft actor-critic for risk-sensitive reinforcement learning. *Journal of Artificial Intelligence Research*, 83, 2025.

Yingqian Min, Zhipeng Chen, Jinhao Jiang, Jie Chen, Jia Deng, Yiwen Hu, Yiru Tang, Jiapeng Wang, Xiaoxue Cheng, Huatong Song, et al. Imitate, explore, and self-improve: A reproduction report on slow-thinking reasoning systems. *arXiv preprint arXiv:2412.09413*, 2024.

Daniel W Nam, Younghoon Kim, and Chan Y Park. Gmac: A distributional perspective on actor-critic framework. In *International Conference on Machine Learning*, pp. 7927–7936. PMLR, 2021.

Ruotian Peng, Yi Ren, Zhouliang Yu, Weiyang Liu, and Yandong Wen. Simko: Simple pass@k policy optimization. *arXiv preprint arXiv:2510.14807*, 2025.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In Yoshua Bengio and Yann LeCun (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1506.02438.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, 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 *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 1279–1297, 2025.

Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang. Learning to reason under off-policy guidance. *CoRR*, abs/2504.14945, 2025.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Weinan Dai, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. DAPO: an open-source LLM reinforcement learning system at scale. *CoRR*, abs/2503.14476, 2025.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *arXiv preprint arXiv:2504.13837*, 2025.

Jin Peng Zhou, Kaiwen Wang, Jonathan D Chang, Zhaolin Gao, Nathan Kallus, Kilian Q Weinberger, Kianté Brantley, and Wen Sun. q♯: Provably optimal distributional rl for llm post-training. *CoRR*, 2025.

Dingwei Zhu, Zhiheng Xi, Shihan Dou, Yuhui Wang, Sixian Li, Junjie Ye, Honglin Guo, Shichun Liu, Chenhao Huang, Yajie Yang, et al. Dvpo: Distributional value modeling-based policy optimization for llm post-training. *arXiv preprint arXiv:2512.03847*, 2025a.

Xuekai Zhu, Daixuan Cheng, Dinghuai Zhang, Hengli Li, Kaiyan Zhang, Che Jiang, Youbang Sun, Ermo Hua, Yuxin Zuo, Xingtai Lv, et al. Flowrl: Matching reward distributions for llm reasoning. *arXiv preprint arXiv:2509.15207*, 2025b.

A Preliminaries

In this section, we formalize LLM generation under RLVR as a token-level episodic Markov Decision Process (MDP), review the fundamentals of Distributional RL, and summarize PPO with scalar advantage estimation.

A.1 Token-Level Episodic MDP for RLVR

We model LLM generation under RLVR as a finite-horizon episodic MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma \rangle$ with horizon T. Given a prompt $\mathbf{x}$, an episode corresponds to generating a token sequence $y_{0:T-1} = (a_0, \ldots, a_{T-1})$.

- **States.** At step $t \in \{0, \ldots, T-1\}$, the state is the prompt-prefix $s_t \triangleq (\mathbf{x}, y_{<t})$, where $y_{<t} = (a_0, \ldots, a_{t-1})$.
- **Actions.** The action $a_t \in \mathcal{V}$ is the next token chosen from the vocabulary $\mathcal{V}$ via the policy $\pi_\theta(\cdot \mid s_t)$.
- **Transitions.** The next state deterministically appends the chosen token: $s_{t+1} = (\mathbf{x}, y_{\leq t})$, i.e., $\mathcal{P}(s_{t+1} \mid s_t, a_t) = \mathbb{I}\{s_{t+1} = (\mathbf{x}, y_{\leq t})\}$.
- **Rewards.** RLVR feedback is typically sparse. We write $r_t = r(s_t, a_t)$ and assume terminal verification reward is received at completion.

We define the (discounted) return from timestep t as $G_t \triangleq \sum_{k=t}^{T} \gamma^{k-t} r_k$. In RLVR settings, we often set $\gamma = 1$. Although the environment dynamics are deterministic, G_t remains stochastic because rollouts are sampled from π_θ and in some settings, and the verification procedure may also introduce randomness. Therefore, we treat the return as a random variable, denoted as $Z^\pi(s_t)$[1] (see Sec. 3).

A.2 Distributional Value Function

Standard RL algorithms often estimate the scalar state-value function $V^\pi(s) = \mathbb{E}[G_t \mid s_t = s]$. In contrast, distributional RL models the return distribution and learns a critic $Z_\phi(s) \approx Z^\pi(s)$. The return distribution satisfies the distributional Bellman equation:

$$Z^\pi(s) \overset{D}{=} r(s,a) + \gamma Z^\pi(s'), a \sim \pi(\cdot \mid s),\ s' \sim \mathcal{P}(\cdot \mid s, a), \tag{6}$$

where $\overset{D}{=}$ denotes equality in distribution.

A.3 Proximal Policy Optimization

PPO Schulman et al. (2017) optimizes a clipped surrogate objective:

$$\mathcal{L}^{\text{CLIP}}(\theta) = \mathbb{E}_t\Big[\min\big(\rho_t(\theta)\,\hat{A}_t,\ \text{clip}(\rho_t(\theta), 1-\epsilon, 1+\epsilon)\,\hat{A}_t\big)\Big], \tag{7}$$

[1] We use $Z^\pi(s)$ to denote the return random variable (and by abuse of notation, its distribution); $F_{Z^\pi(s)}$ denotes its CDF.

where $\rho_t(\theta) \triangleq \pi_\theta(a_t \mid s_t)/\pi_{\theta_{\text{old}}}(a_t \mid s_t)$ is the importance ratio and the expectation is taken over timesteps in a batch collected under $\pi_{\theta_{\text{old}}}$. In practice, $\hat{A}_t$ is commonly computed by generalized advantage estimation (GAE) Schulman et al. (2016) using a scalar critic V_ϕ:

$$\hat{A}_t^{\text{GAE}} = \sum_{l=0}^{T-t-1} (\gamma\lambda)^l \, \delta_{t+l}, \delta_t = r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t). \tag{8}$$

Here $\lambda \in [0, 1]$ is the GAE trace-decay parameter. In our distributional setting, PPO/GAE still requires a scalar baseline, which we induce from the return distribution via a functional $\rho(\cdot)$ (mean by default): $V(s) \triangleq \rho(Z(s)) = \mathbb{E}[Z(s)]$.

B Tail-Aware Advantage Shaping

A key benefit of learning a distributional critic is that it provides more accurate and lower-variance token-level value estimates than a scalar critic, yielding more reliable advantages. More importantly, distributional modeling exposes uncertainty and tail structure of returns. In RLVR, tail events, especially rare successes, are disproportionately important for learning.

Tail statistics naturally induce different risk preferences. Emphasizing the tail (e.g., lower-quantile / lower-CVaR) corresponds to risk-averse updates that improve robustness and worst-case behavior, whereas emphasizing the upper tail (e.g., upper-quantile / upper-CVaR) corresponds to risk-seeking updates that accelerate the discovery and consolidation of rare high-return trajectories. We discuss such variants in the Appendix F.1, directly replacing baselines by risk functionals in DistRLVR. But these approaches often change the optimization scale and training dynamics and can be less stable in long-horizon RLVR.

To introduce controllable tail sensitivity while preserving the PPO advantage structure, we propose a simple Tail-Aware Advantage Shaping mechanism. The core idea is to reallocate gradient budget toward tokens whose realized outcome falls into the state-conditional upper tail predicted by the distributional critic.

Let $\hat{A}_t$ be a standard advantage estimator (Eq. 8) with return target G_t. We first extract an upper-tail threshold,

$$q_\alpha^+(s_t) \triangleq \text{Quantile}_{1-\alpha}(Z_\phi(s_t)), \tag{9}$$

and then compute an *outcome-gated* weight measuring how "tail" the realized return is relative to this threshold:

$$w_\alpha(s_t, G_t) \triangleq \frac{1}{\alpha} \sigma\left(\frac{G_t - q_\alpha^+(s_t)}{\beta}\right), \tag{10}$$

where $\sigma(\cdot)$ is the sigmoid function, β controls gate sharpness, and α specifies the targeted upper-tail mass (with $1/\alpha$ compensating for sparsity). Finally, we shape the advantage in Eq. 7 by a stop-gradient multiplicative reweighting:

$$\hat{A}_t^{\text{tail}} \triangleq \text{sg}\big(w_\alpha(s_t, G_t)\big) \, \hat{A}_t. \tag{11}$$

In practice, we normalize and clip w_α over response tokens to keep the overall update scale stable. Consequently, we do not merely upweight successful episodes. Instead, we assign more credit to tokens whose outcomes are tail-exceeding relative to their predicted state-conditional return distributions, sharpening credit assignment and accelerating the propagation of rare success signals to early tokens.

C Theory Analysis for Terminal-Sparse TD(0) and Sample Replacement

This appendix restates the target construction in Sec. 3.2 using consistent indexing and provides proofs omitted in the main text (in particular, Lemma 3.2). Throughout, λ_{SR} denotes the *continue probability* of the truncated-geometric backup-depth distribution used by Sample Replacement (SR), and is unrelated to the GAE trace parameter λ_{GAE} in Eq. equation 8.

In many RLVR setups with terminal-only verification rewards, it is common to set $\gamma = 1$ (and often $\lambda_{\text{GAE}} = 1$ when using Monte-Carlo-style advantage estimation). All SR sampling-correctness

statements (e.g., Lemma 3.2) hold verbatim for any $\gamma \in (0, 1]$. When $\gamma = 1$, the bootstrapping-discrepancy bound in Lemma C.5 remains valid but loses the γ^n attenuation, yielding a more conservative (yet appropriate) characterization of long-horizon bootstrapping bias in RLVR.

C.1 Setup

We adopt the token-level episodic MDP in Sec. A.1. An episode generates a token sequence $y_{0:T-1} = (a_0, \ldots, a_{T-1})$, reaching the terminal state s_T after sampling a_{T-1}. We use the following indexing convention, consistent with the main text:

- **Actions and transitions:** for $t = 0, \ldots, T-1$, an action a_t sampled from $\pi_\theta(\cdot \mid s_t)$ deterministically transitions to s_{t+1}.
- **Rewards:** we allow intermediate rewards $\{r_t\}_{t=0}^{T-1}$ (often zero), and additionally allow an *extra terminal-state reward* r_T produced by the verifier upon reaching s_T (i.e., after the completion is formed). Importantly, there is *no* action a_T.

Accordingly, we write a realized rollout as

$$\tau \;=\; \{(s_t, a_t, r_t)\}_{t=0}^{T-1} \;\cup\; \{(s_T, r_T)\}, \tag{12}$$

and define the (discounted) return random variable from prefix state s_t as

$$Z^\pi(s_t) \;\triangleq\; \sum_{k=t}^{T} \gamma^{k-t} r_k, \qquad \gamma \in (0, 1]. \tag{13}$$

Terminal-sparse verification (common RLVR case). In many RLVR tasks, rewards are terminal and sparse:

$$r_t = 0 \text{ for all } t < T, \qquad r_T \in \mathbb{R}. \tag{14}$$

This assumption is used only to simplify several expressions and to formalize why TD(0) becomes weakly supervised at early prefixes; the SR sampling correctness results do not rely on it.

Terminal bootstrap convention. At the terminal step, we use the *observed* verification reward as the bootstrap target:

$$Z_{\bar\phi}(s_T) \;\triangleq\; \delta_{r_T}. \tag{15}$$

This is a training-time convention for constructing targets (rather than an assumption that the target network "predicts" r_T).

C.2 Why TD(0) targets are weak in terminal-sparse RLVR

For $t < T$, the one-step distributional TD target (Eq. equation 1) is

$$Z_t^{\mathrm{tgt}} \;=\; r_t + \gamma Z_{\bar\phi}(s_{t+1}).$$

Under terminal sparsity (Eq. equation 14), this reduces to

$$Z_t^{\mathrm{tgt}} \;=\; \gamma Z_{\bar\phi}(s_{t+1}), \qquad t < T, \tag{16}$$

so most prefix targets are dominated by the *local bootstrap* from s_{t+1}. As a consequence, adjacent prefixes receive highly correlated supervision, and distributional information about rare terminal outcomes must be propagated backward through an $O(T - t)$ chain of local bootstraps. This motivates multi-step targets with endpoints closer to verification, while remaining computationally feasible at long horizons.

C.3 Bootstrapped multi-step targets and truncated-geometric mixing

Let $H_t \triangleq T - t$ denote the remaining horizon from timestep t. For any $n \in \{1, \ldots, H_t\}$, define the bootstrapped n-step return distribution (Eq. equation 3):

$$\widetilde{Z}_t^{(n)} \;\overset{D}{=}\; \sum_{i=0}^{n-1} \gamma^i r_{t+i} \;+\; \gamma^n Z_{\bar\phi}(s_{t+n}), \tag{17}$$

where the endpoint distribution at $t + n = T$ is set by Eq. equation 15. Larger n anchors the target closer to the terminal verification reward, reducing reliance on purely local bootstrapping.

We mix backup depths using a truncated geometric distribution controlled by $\lambda_{\mathrm{SR}} \in [0, 1)$ (main text, Sec. 3.2):

$$\Pr(N = n) = (1 - \lambda_{\mathrm{SR}})\lambda_{\mathrm{SR}}^{n-1} \text{ for } n < H_t, \qquad \Pr(N = H_t) = \lambda_{\mathrm{SR}}^{H_t - 1}. \tag{18}$$

Define the (bootstrapped) mixture target random variable by

$$\widetilde{Z}_t^{(\lambda_{\mathrm{SR}})} \triangleq \widetilde{Z}_t^{(N)}, \qquad N \sim \text{Eq. equation 18}. \tag{19}$$

Equivalently, $\widetilde{Z}_t^{(\lambda_{\mathrm{SR}})}$ is a mixture over $\{\widetilde{Z}_t^{(n)}\}_{n=1}^{H_t}$ with weights given in Eq. equation 18.

C.4 SR is an unbiased sampler of the bootstrapped mixture target

We now prove Lemma 3.2 under the indexing convention in Eq. equation 12.

Conditioning and sources of randomness. Condition on a realized trajectory τ (thus $\{(s_t, r_t)\}_{t=0}^{T}$ are fixed) and on a fixed target-network distribution mapping $s \mapsto Z_{\bar{\phi}}(s)$. The only randomness in SR comes from (i) independent Bernoulli variables $\{b_t^{(m)}\}$ and (ii) conditionally independent replacement draws $\tilde{z}_{t+1}^{(m)} \sim Z_{\bar{\phi}}(s_{t+1})$.

SR recursion. For each sample index $m \in [M]$, initialize $g_T^{(m)} \leftarrow r_T$ and for $t = T - 1, \ldots, 0$ sample

$$b_t^{(m)} \sim \mathrm{Bernoulli}(\lambda_{\mathrm{SR}}), \qquad \tilde{z}_{t+1}^{(m)} \sim Z_{\bar{\phi}}(s_{t+1}) \ \text{(used only if } b_t^{(m)} = 0),$$

then apply the SR recursion (Eq. equation 4):

$$g_t^{(m)} \leftarrow r_t + \gamma\Big(b_t^{(m)}\, g_{t+1}^{(m)} + (1 - b_t^{(m)})\, \tilde{z}_{t+1}^{(m)}\Big). \tag{20}$$

All $\{b_t^{(m)}\}$ are independent across t and m, and replacement draws are conditionally independent given τ and $Z_{\bar{\phi}}$.

Stopping time. Define the truncated stopping time for sample path m:

$$N_t^{(m)} \triangleq \min\{n \in \{1, \ldots, H_t\} : b_{t+n-1}^{(m)} = 0\}, \quad \text{with } N_t^{(m)} = H_t \text{ if no such } n. \tag{21}$$

[Distribution of the stopping time] For $n \in \{1, \ldots, H_t - 1\}$,

$$\Pr\Big(N_t^{(m)} = n\Big) = (1 - \lambda_{\mathrm{SR}})\lambda_{\mathrm{SR}}^{n-1}, \qquad \Pr\Big(N_t^{(m)} = H_t\Big) = \lambda_{\mathrm{SR}}^{H_t - 1}.$$

Proof. For $n < H_t$, the event $\{N_t^{(m)} = n\}$ requires $b_t^{(m)} = \cdots = b_{t+n-2}^{(m)} = 1$ and $b_{t+n-1}^{(m)} = 0$. Independence gives probability $\lambda_{\mathrm{SR}}^{n-1}(1 - \lambda_{\mathrm{SR}})$. If no replacement occurs before terminal, all $b_t^{(m)}, \ldots, b_{T-1}^{(m)} = 1$, which has probability $\lambda_{\mathrm{SR}}^{H_t - 1}$. □

[Unrolling SR recursion conditioned on $N_t^{(m)}$] Condition on $\{N_t^{(m)} = n\}$. If $n < H_t$, then

$$g_t^{(m)} \overset{D}{=} \sum_{i=0}^{n-1} \gamma^i r_{t+i} \ + \ \gamma^n \tilde{z}_{t+n}^{(m)}, \qquad \tilde{z}_{t+n}^{(m)} \sim Z_{\bar{\phi}}(s_{t+n}).$$

If $n = H_t$, then $g_t^{(m)} = \sum_{i=0}^{H_t} \gamma^i r_{t+i}$ (i.e., the terminal-anchored return that includes r_T).

Proof. If $n < H_t$, then $b_t^{(m)} = \cdots = b_{t+n-2}^{(m)} = 1$ and $b_{t+n-1}^{(m)} = 0$. Recursively substituting Eq. equation 20 for n steps yields

$$g_t^{(m)} = r_t + \gamma r_{t+1} + \cdots + \gamma^{n-1} r_{t+n-1} + \gamma^n \tilde{z}_{t+n}^{(m)}.$$

If $n = H_t$, SR always takes the "continue" branch until reaching T, where $g_T^{(m)} = r_T$, giving $g_t^{(m)} = \sum_{i=0}^{H_t} \gamma^i r_{t+i}$. □

Proof of Lemma 3.2. By Lemma C.4, conditional on $N_t^{(m)} = n$, $g_t^{(m)}$ is distributed as the bootstrapped n-step return $\widetilde{Z}_t^{(n)}$ in Eq. equation 17. By Lemma C.4, the mixture weights over n match Eq. equation 18. Therefore, marginalizing over $N_t^{(m)}$ yields

$$g_t^{(m)} \overset{D}{=} \widetilde{Z}_t^{(\lambda_{\mathrm{SR}})},$$

where $\widetilde{Z}_t^{(\lambda_{\mathrm{SR}})}$ is defined in Eq. equation 19. □

I.i.d. clarification. For each *fixed* prefix timestep t, SR generates M conditionally i.i.d. target draws $\{g_t^{(m)}\}_{m=1}^M$ whose marginal matches $\widetilde{Z}_t^{(\lambda_{\mathrm{SR}})}$. (Within a single sample path m, the sequence $\{g_t^{(m)}\}_t$ is of course temporally coupled by the backward recursion.)

C.5 Discrepancy to the true on-policy mixture (bootstrapping bias)

SR is unbiased for the *bootstrapped* mixture target $\widetilde{Z}_t^{(\lambda_{\mathrm{SR}})}$. It is not, in general, unbiased for the true on-policy mixture unless $Z_{\bar{\phi}}$ is exact. We formalize this gap under a generic distributional discrepancy $d(\cdot,\cdot)$ that is (i) shift-invariant, (ii) positively homogeneous, and (iii) convex under mixtures.

Define the true n-step return distribution

$$Z_t^{(n)} \overset{D}{=} \sum_{i=0}^{n-1} \gamma^i r_{t+i} \;+\; \gamma^n Z^\pi(s_{t+n}),$$

and the corresponding truncated-geometric mixture $Z_t^{(\lambda_{\mathrm{SR}})} \triangleq Z_t^{(N)}$ with the same N as in Eq. equation 18. Let $\Delta_{\bar{\phi}}(s) \triangleq d(Z_{\bar{\phi}}(s), Z^\pi(s))$.

[Bootstrapping-induced discrepancy bound] For each t,

$$d\Big(\widetilde{Z}_t^{(\lambda_{\mathrm{SR}})},\, Z_t^{(\lambda_{\mathrm{SR}})}\Big) \;\le\; (1-\lambda_{\mathrm{SR}}) \sum_{n=1}^{H_t-1} \lambda_{\mathrm{SR}}^{n-1}\gamma^n \Delta_{\bar{\phi}}(s_{t+n}) \;+\; \lambda_{\mathrm{SR}}^{H_t-1}\gamma^{H_t}\Delta_{\bar{\phi}}(s_T). \tag{22}$$

Under the terminal bootstrap convention in Eq. equation 15, the terminal endpoint incurs no additional approximation from critic bootstrapping in the target construction, since $Z_{\bar{\phi}}(s_T)$ is set to the degenerate distribution at the observed r_T. Any remaining discrepancy to the true terminal return distribution $Z^\pi(s_T)$ (and hence $\Delta_{\bar{\phi}}(s_T)$) depends solely on verifier stochasticity at termination.

Proof. For each n, shift-invariance and positive homogeneity imply

$$d\Big(\widetilde{Z}_t^{(n)}, Z_t^{(n)}\Big) = d\left(\sum_{i=0}^{n-1}\gamma^i r_{t+i} + \gamma^n Z_{\bar{\phi}}(s_{t+n}),\; \sum_{i=0}^{n-1}\gamma^i r_{t+i} + \gamma^n Z^\pi(s_{t+n})\right) = \gamma^n \Delta_{\bar{\phi}}(s_{t+n}).$$

Applying convexity of d under the truncated-geometric mixture over n yields Eq. equation 22. □

C.6 Replacement sampling under different distributional heads

SR requires generating replacement draws $\tilde{z} \sim Z_{\bar{\phi}}(s)$. We specify an implementation-consistent sampling rule for each head in Sec. 3.1.

Categorical (C51). If $Z_{\bar{\phi}}(s) = \sum_{\ell=1}^L p_{\bar{\phi}}^\ell(s)\delta_{v_\ell}$, sample $\tilde{z} = v_\ell$ with probability $p_{\bar{\phi}}^\ell(s)$.

Implicit quantiles (IQN). IQN parameterizes an implicit quantile function $\hat{z}_{\bar{\phi}}(s;\tau)$. A replacement draw is obtained by sampling $\tau \sim \mathcal{U}(0,1)$ and setting $\tilde{z} \leftarrow \hat{z}_{\bar{\phi}}(s;\tau)$, which samples from the induced return distribution.

Quantile regression (QR). QR predicts fixed quantile atoms $\{\hat{z}_{\bar{\phi}}(s;\tau_i)\}_{i=1}^{N_q}$ at fractions $\tau_i = \frac{i-\frac{1}{2}}{N_q}$. In SR, we define the induced return distribution of the QR head as the discrete uniform measure over these atoms:

$$Z_{\bar{\phi}}(s) \triangleq \frac{1}{N_q}\sum_{i=1}^{N_q}\delta_{\hat{z}_{\bar{\phi}}(s;\tau_i)}.$$

A replacement draw is then obtained by sampling $i \sim \mathrm{Unif}\{1,\dots,N_q\}$ and setting $\tilde{z} \leftarrow \hat{z}_{\bar{\phi}}(s;\tau_i)$. With this induced distribution, SR sampling is exact with respect to the target distribution used by the QR head.

C.7 Computational cost and memory footprint

For a trajectory of length T, generating one SR sample path $\{g_t^{(m)}\}_{t=0}^{T}$ requires a single backward sweep over $t = T-1,\dots,0$, i.e., $O(T)$ time. Generating M independent SR sample paths costs $O(MT)$ per trajectory.

In dSR (Sec. 3.2), we additionally draw K trajectories per prompt. The total target-generation time per prompt is therefore $O(KMT)$ (up to variation in completion lengths). If SR targets are materialized for training, storage is $O(KMT)$. Alternatively, SR targets can be generated and consumed on-the-fly during the backward sweep, keeping only $\{g_{t+1}^{(m)}\}_{m=1}^{M}$ in memory at each step, which reduces the working memory to $O(M)$ per trajectory (plus model activations required by the training pipeline).

D Distributional RL Preliminaries (Extended)

This appendix summarizes three classical distributional RL parameterizations (C51, QR, and IQN) and clarifies their instantiation in our PPO-based RLVR setting. The original algorithms are commonly presented in the DQN/control formulation, which models a return distribution over state–action pairs $Z_\phi(s,a)$ and performs greedy control through $Q_\phi(s,a) = \mathbb{E}[Z_\phi(s,a)]$. Figure 5, drawn from Dabney et al. (2018a), provides an intuitive visualization of the differences among the three distributional reinforcement learning algorithms.

In contrast, our method learns a token-level state-value return distribution $Z_\phi(s)$ as a critic for PPO; consequently, action selection and control-specific targets are not used, and the critic is trained by distributional regression toward value targets Z_t^{tgt} (constructed by TD(0) or Sample Replacement in Sec. 3.2 and Appendix C). Throughout this section, N_q denotes the number of categorical atoms (C51) or quantile fractions (QR), consistent with the main text.

Categorical (C51). C51 Bellemare et al. (2017) represents the return distribution on a fixed support $\{v_\ell\}_{\ell=1}^{N_q}$ spanning $[V_{\min}, V_{\max}]$:

$$Z_\phi(s,a) = \sum_{\ell=1}^{N_q} p_\phi^\ell(s,a)\,\delta_{v_\ell}, \qquad \sum_{\ell=1}^{N_q} p_\phi^\ell(s,a) = 1. \tag{23}$$

In the control setting, given a transition (s_t, a_t, r_t, s_{t+1}) and a greedy next action $a^* \triangleq \arg\max_a \mathbb{E}[Z_{\bar{\phi}}(s_{t+1},a)]$, the one-step distributional target is

$$Z_t^{\mathrm{tgt}} \triangleq r_t + \gamma\, Z_{\bar{\phi}}(s_{t+1}, a^*). \tag{24}$$

Since the shifted target generally does not lie on the fixed support, C51 applies a projection operator Π to obtain a categorical target $m_t \triangleq \Pi Z_t^{\mathrm{tgt}} \in \Delta^{N_q-1}$ and minimizes cross-entropy:

$$\mathcal{L}_{\mathrm{C51}}(\phi) = -\sum_{\ell=1}^{N_q} m_t^\ell \log p_\phi^\ell(s_t,a_t). \tag{25}$$

In our state-value formulation, we parameterize $Z_\phi(s) = \sum_{\ell=1}^{N_q} p_\phi^\ell(s)\delta_{v_\ell}$ and regress toward the value target distribution Z_t^{tgt} using the same projected cross-entropy:

$$\mathcal{L}_{\mathrm{C51}}^V(\phi) = -\sum_{\ell=1}^{N_q} m_t^\ell \log p_\phi^\ell(s_t), \qquad m_t \triangleq \Pi Z_t^{\mathrm{tgt}}. \tag{26}$$

Quantile regression (QR). QR-DQN Dabney et al. (2018b) predicts N_q quantile locations $\{z_\phi(s,a;\tau_i)\}_{i=1}^{N_q}$ at fixed fractions

$$\tau_i \triangleq \frac{i-\frac{1}{2}}{N_q}, \qquad i = 1,\dots,N_q, \tag{27}$$

inducing an empirical return distribution $Z_\phi(s,a) \approx \frac{1}{N_q}\sum_{i=1}^{N_q} \delta_{z_\phi(s,a;\tau_i)}$. In the control setting, one forms target samples $y_t^j \triangleq r_t + \gamma\, z_{\bar\phi}(s_{t+1}, a^*; \tau_j)$ for $j = 1,\dots,N_q$ and minimizes the pairwise quantile Huber objective

$$\mathcal{L}_{\mathrm{QR}}(\phi) = \frac{1}{N_q^2}\sum_{i=1}^{N_q}\sum_{j=1}^{N_q} \rho_{\tau_i}^\kappa\Big(y_t^j - z_\phi(s_t,a_t;\tau_i)\Big), \tag{28}$$

where $\rho_\tau^\kappa(\cdot)$ uses the Huber penalty $\mathcal{H}_\kappa(\cdot)$.

In our state-value formulation, we predict quantiles $\{z_\phi(s;\tau_i)\}_{i=1}^{N_q}$ and regress them to target draws $\{g_t^{(m)}\}_{m=1}^M$ produced by TD(0) or Sample Replacement:

$$\mathcal{L}_{\mathrm{QR}}^V(\phi) = \frac{1}{N_q M}\sum_{i=1}^{N_q}\sum_{m=1}^{M} \rho_{\tau_i}^\kappa\Big(g_t^{(m)} - z_\phi(s_t;\tau_i)\Big). \tag{29}$$

Implicit quantiles (IQN). IQN Dabney et al. (2018a) parameterizes a continuous quantile function $z_\phi(s,a;\tau)$ with $\tau \sim \mathcal{U}(0,1)$. In the control setting, one draws $\{\tau_\ell\}_{\ell=1}^K \sim \mathcal{U}(0,1)$ and $\{\tau'_j\}_{j=1}^{K'} \sim \mathcal{U}(0,1)$, forms target samples $y_t^j \triangleq r_t + \gamma\, z_{\bar\phi}(s_{t+1}, a^*; \tau'_j)$, and minimizes

$$\mathcal{L}_{\mathrm{IQN}}(\phi) = \frac{1}{KK'}\sum_{\ell=1}^{K}\sum_{j=1}^{K'} \rho_{\tau_\ell}^\kappa\Big(y_t^j - z_\phi(s_t,a_t;\tau_\ell)\Big). \tag{30}$$

In our state-value formulation, we parameterize $z_\phi(s;\tau)$ and regress to target draws $\{g_t^{(m)}\}_{m=1}^M$. Specifically, draw $\{\tau_\ell\}_{\ell=1}^K \sim \mathcal{U}(0,1)$ and minimize

$$\mathcal{L}_{\mathrm{IQN}}^V(\phi) = \frac{1}{KM}\sum_{\ell=1}^{K}\sum_{m=1}^{M} \rho_{\tau_\ell}^\kappa\Big(g_t^{(m)} - z_\phi(s_t;\tau_\ell)\Big). \tag{31}$$

These head-specific losses are concrete instantiations of the distributional regression objective in Eq. equation 2, where Z_t^{tgt} is constructed by TD(0) (Eq. equation 1) or Sample Replacement (Eq. equation 4 and Appendix C). Finally, PPO uses the induced scalar baseline $V_\phi(s) = \mathbb{E}[Z_\phi(s)]$ while retaining $Z_\phi(s)$ for distributional tail statistics.

E Experiment Details

Hyperparameters We implement our method on top of the VeRL Sheng et al. (2025) framework using vLLM v0.8.4 Kwon et al. (2023). The major hyperparameter choices are shown in the Table 4.

Code available at https://anonymous.4open.science/r/DistRLVR_Anonymous-CD2E/.

E.1 Evaluation Metrics

In this study, we primarily use two evaluation metrics: pass@k and avg@32. For pass@k, we report results for $k \in \{1,2,4,8,16,32,64,128\}$. During evaluation, we sample 128 responses for each prompt.

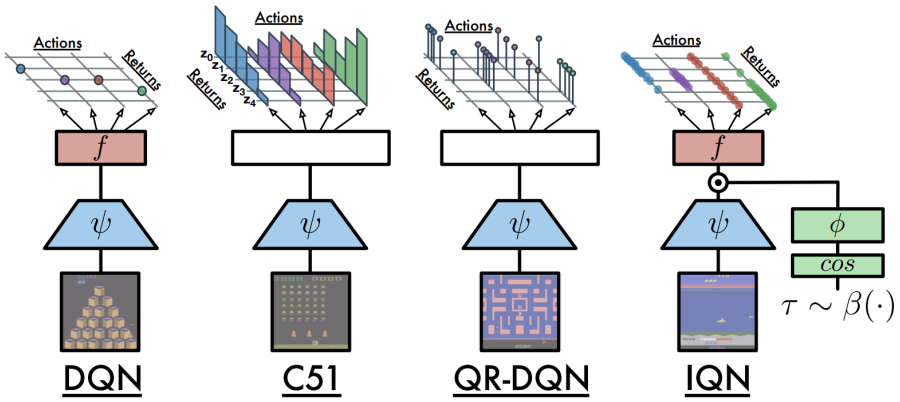

Figure 5: Three classical distributional reinforcement learning algorithms. Dabney et al. (2018a)

Table 4: Hyperparameter settings for PPO, GRPO, and DistRLVR.

Hyperparameter	**Symbol**	**Value**
Rollout & Decoding		
Sampling Temperature	T	1.0
Nucleus Sampling Threshold	p	1.0
Top-k Cutoff	k	-1
Max Prompt Length	L_{prompt}	766
Max Response Length	L_{resp}	1024 (by default)
PPO Optimization		
Actor Learning Rate	η_{actor}	1×10^{-6}
Critic Warmup Steps	N_{warmup}	0
Entropy Coefficient	c_{ent}	0.001
Clip Range	ϵ	0.2
Global Batch Size	B	128
Mini Batch Size	b	32
GRPO-Specific		
KL Penalty Coefficient	β_{KL}	0.001
KL Approximation Method	–	low_var_kl
DistRLVR: Distributional Critic		
Support Range (C51)	$[v_{\min}, v_{\max}]$	$[0, 1]$
Number of Atoms	L	51
Number of Quantiles	N_q	32
IQN Samples per Update	N_τ	8
DistRLVR: Tail Shaping		
Upper-tail Quantile	α	0.25
Tail Shaping Beta (Gate Sharpness)	β_{tail}	0.5
Soft Beta (Sigmoid Temperature)	soft_β	0.5
dSR: Sample Replacement		
Number of Trajectories per Prompt	K	8
Number of SR Targets per Token State	M	8
SR Continue Probability	λ_{SR}	0.9

Table 5: Performance comparison with standard deviation. We report mean±std over three seeds.

Method	AIME24		AIME25		AMC23	
	avg.32	p@128	avg.32	p@128	avg.32	p@128
Base Model	0	0	0	0	3.256	4.819
PPO	5.104 ± 0.9	43.333 ± 3.3	3.264 ± 0.7	26.667 ± 5.8	31.903 ± 2.3	86.345 ± 0.7
DistRL (dep)	5.174 ± 1.5	41.111 ± 3.8	2.743 ± 1	26.667 ± 12	30.999 ± 1.6	85.141 ± 2.8
DistRL (0)	7.535 ± 0.4	46.667 ± 5.8	5.452 ± 0.7	37.778 ± 7.7	46.233 ± 1.6	88.353 ± 2.8
DistRL (SR)	8.125 ± 0.6	53.333 ± 3.3	6.042 ± 0.3	35.556 ± 7.7	43.737 ± 0.7	90.361 ± 2.1
DistRL (Tail)	9.549 ± 1.2	54.444 ± 1.9	7.361 ± 0.8	36.667 ± 5.8	44.829 ± 0.3	89.157 ± 1.2

Method	MATH500		Minerva		Olympiad	
	avg.32	p@128	avg.32	p@128	avg.32	p@128
Base Model	3.256	4.2	1.241	26.103	0.296	8.296
PPO	62.59 ± 0.8	94.667 ± 0.6	18.053 ± 0.8	64.951 ± 1.9	23.012 ± 0.8	65.975 ± 1.6
DistRLVR (dep)	62.854 ± 1.1	95 ± 0.5	17.816 ± 0.8	64.461 ± 1.7	23.414 ± 0.8	66.42 ± 1.9
DistRLVR (0)	69.969 ± 0.9	95.067 ± 0.2	21.944 ± 0.8	61.52 ± 4.9	28.611 ± 1.2	65.531 ± 2.2
DistRLVR (SR)	71.025 ± 0.4	94.533 ± 0.3	22.522 ± 0.4	62.377 ± 0.2	29.241 ± 0.2	65.126 ± 1.1
DistRLVR (Tail)	72.623 ± 0.5	94.267 ± 0.5	22.794 ± 0.4	57.721 ± 1.9	31.451 ± 0.3	62.914 ± 0.6

avg@32 This metric is primarily used to evaluate the average "single-response correctness" over 32 sampled outputs per example. It is typically computed as a Monte Carlo mean, providing an approximation of the model's expected performance under the current sampling strategy. Specifically, for each prompt we sample 128 responses, and each response is associated with a reward. To compute avg@32, we take the reward s_i of the 32 sampled responses (out of 128) responses, and define $avg@32$:

$$avg@32 = \frac{1}{32} \sum_{i=1}^{32} s_i \tag{32}$$

In RLVR, rewards are typically binary, i.e., $\{0, 1\}$. Therefore, when $s_i \in \{0, 1\}$, avg@32 corresponds to the proportion of correct answers among the 32 samples, which approximates the expected accuracy of a single response drawn at random from the model's sampling distribution. This metric can be used to evaluate the model's average answer accuracy, reflecting the overall quality and stability of its responses. In this study, we assess different algorithms using avg@32 to demonstrate that DistRL achieves superior performance, attaining higher accuracy under the same sampling budget. Moreover, the avg@32 curves indicate that DistRL exhibits greater training stability and improved sample efficiency.

pass@k pass@k is a metric used to measure the probability that a model generates at least one correct answer given a sampling budget of k, under the setting of "sampling multiple answers for the same prompt." It represents the model's success rate within a given sampling budget. The physical meaning of pass@k is that for the same prompt, the sampling distribution of the model is$P(y|x)$,and definding the decision function $I(y) \in \{0, 1\}$ determines whether the answer is correct. If k samples are taken for a question, yielding k responses, then pass@k is defined as:

$$\text{pass@}k = \Pr\left(\max_{i=1,\ldots,k} I(y_i) = 1 \right) = 1 - \Pr(\forall i,\ I(y_i) = 0)\,. \tag{33}$$

In this study, we sampled $n = 128$ responses for each question, with k taken from $\{1, 2, 4, 8, 16, 32, 64, 128\}$. During actual evaluation, we treat the process of selecting k samples from these n samples as sampling without replacement, where c samples receive the correct reward. Following the standard pass@k estimator used in prior work, pass@k is calculated as follows:

$$\text{pass@}k = 1 - \frac{\binom{n-c}{k}}{\binom{n}{k}} \qquad (k \le n). \tag{34}$$

In our RLVR setting, rewards are binary, i.e., $\{0, 1\}$: each sampled response is either correct (1) or incorrect (0). Under this regime, *pass@k* directly quantifies the probability of obtaining *at least one*

correct solution given a sampling budget of k attempts. This metric is a best-of-(k) metric. Thus,it better reflects the model's usability under a fixed compute budget. Moreover, pass@k is sensitive to improvements in exploration and coverage of diverse solution paths: even when single-sample accuracy is modest, an algorithm that increases the chance of producing a correct candidate within k samples will yield a higher pass@k. Therefore, pass@k is particularly well-suited for evaluating RLVR methods on tasks with binary success signals.

In this study, we primarily focus on pass@128 because it reflects the model's best-of-128 capability under a fixed sampling budget, indicating that DistRL can improve performance more effectively under the same sampling budget. In contrast, avg@32 is mainly used to show that DistRL leads to responses with better overall quality and greater stability.

E.2 More results

To facilitate reproducibility, we release our full training and evaluation code in an anonymous repository. This appendix reports additional results omitted from the main text due to space constraints, including complete mean$\pm$std numbers. Concretely, we train *three* independent models with different random seeds; for each trained model, we evaluate using 128 sampled completions per prompt and compute the corresponding metrics. We then report the mean and standard deviation over the three seeds, which provides a direct measure of run-to-run variability and highlights the training stability of our approach. Compared to PPO, our method consistently exhibits smaller standard deviations, indicating improved stability across independent runs.

E.3 Training Time Analysis and Advantages

A thorough analysis reveals that the overall time cost during training is predominantly influenced by the rollout process. Specifically, when all algorithms are configured with `rollout.n=8`, as shown in Figure 6, the time costs for GRPO, PPO, and DistRLVR converge to nearly identical levels, indicating that the additional complexity introduced by the value distribution is negligible in LLM post-training.

Upon closer inspection, it becomes evident that PPO and GRPO incur slightly higher time costs under `n=8`. We attribute this difference primarily to the strategies learned during the training process. In particular, both PPO and GRPO tend to generate longer sequences, as compared to those guided by the value distribution in DistRLVR. This results in a higher computational burden, as longer sequences naturally demand more processing power and time.

These observations underscore the efficiency of DistRLVR, where the value distribution contributes not only to improved performance but also to a more efficient generation process, striking a balance between computational cost and response length.

F Additional Mechanisms

F.1 Risk-Sensitive Policy Updates via Risk-Aware Baselines

This section describes an auxiliary mechanism not expanded in the main text: risk sensitivity applied to the *baseline* rather than to the policy objective. Throughout, we keep the standard PPO/GAE structure in Sec. A.3, but replace the scalar baseline induced from the distributional critic by a general risk functional of the return distribution.

Setup. Let $Z_\phi(s_t)$ be the learned return distribution at prefix state s_t. In the main text, PPO/GAE uses a scalar value baseline $V_\phi(s_t)$ (by default the mean $\mathbb{E}[Z_\phi(s_t)]$). Here we generalize the baseline through a risk functional $\rho(\cdot)$:

$$V_\phi^\rho(s_t) \triangleq \rho(Z_\phi(s_t)), \tag{35}$$

and keep the actor update identical to PPO except that advantages are computed relative to V_ϕ^ρ. Concretely, define TD residuals

$$\delta_t^\rho \triangleq r_t + \gamma V_\phi^\rho(s_{t+1}) - V_\phi^\rho(s_t), \tag{36}$$

and the corresponding GAE advantage

$$\hat{A}_t^{\rho} \triangleq \sum_{l=0}^{T-t-1} (\gamma\lambda_{\text{GAE}})^l \, \delta_{t+l}^{\rho}. \tag{37}$$

Equivalently, under $\lambda_{\text{GAE}} = 1$ and $\gamma = 1$ (common in terminal-only RLVR), one may view this as

$$\hat{A}_t^{\rho} \approx G_t - \rho(Z_\phi(s_t)), \tag{38}$$

i.e., the realized return is centered by a risk-aware baseline rather than by the mean. The PPO objective in Eq. equation 7 is then applied with $\hat{A}_t \leftarrow \hat{A}_t^{\rho}$.

Neutral, risk-averse, and risk-seeking baselines. We consider the following choices of $\rho(\cdot)$, all of which depend only on the state (prefix) and thus define valid baselines.

Neutral (mean baseline). The default choice recovers standard PPO/GAE:

$$\rho_{\text{mean}}(Z) \triangleq \mathbb{E}[Z], \qquad V_\phi^{\rho}(s) = \mathbb{E}[Z_\phi(s)]. \tag{39}$$

Risk-averse (lower-tail baseline). A risk-averse baseline uses a lower-tail functional, e.g., a lower CVaR:

$$\rho_{\text{LCVaR}}(Z) \triangleq \text{CVaR}_\alpha^-(Z) \approx \mathbb{E}[\, Z \mid Z \le \text{Quantile}_\alpha(Z)], \tag{40}$$

which emphasizes the worst α-fraction outcomes. Because $\rho_{\text{LCVaR}}(Z)$ is typically *more pessimistic* than $\mathbb{E}[Z]$, it shifts the baseline downward. Intuitively, this tends to (i) reduce the magnitude of negative advantages for bad outcomes (stabilizing updates when failures are common) and (ii) enlarge positive advantages for moderately good outcomes (since they are measured relative to a pessimistic reference). As a result, updates become indirectly biased toward policies that improve lower-tail performance and robustness: behaviors that frequently produce extremely poor outcomes are less likely to be reinforced, while behaviors that reliably avoid worst-case failures receive stronger positive signals. In practice, this variant often yields more conservative generations and raises the lower tail of the reward/return distribution.

Risk-seeking (upper-tail baseline). Conversely, a risk-seeking baseline uses an upper-tail functional, e.g., an upper CVaR:

$$\rho_{\text{UCVaR}}(Z) \triangleq \text{CVaR}_\alpha^+(Z) \approx \mathbb{E}\big[\, Z \,\big|\, Z \ge \text{Quantile}_{1-\alpha}(Z)\big], \tag{41}$$

which emphasizes the best α-fraction outcomes. Here $\rho_{\text{UCVaR}}(Z)$ is *more optimistic* than $\mathbb{E}[Z]$, shifting the baseline upward. Consequently, most "average" outcomes yield smaller (often negative) advantages because G_t rarely exceeds an optimistic reference; only when a rollout attains a rare high return does G_t approach or surpass the upper-tail baseline, producing a comparatively strong positive advantage. This concentrates learning signal on rare high-reward events, indirectly encouraging exploration and exploitation of behaviors that can achieve extreme successes, even if they are infrequent.

Relation to tail-aware advantage reweighting (main text). Risk-aware baselines and tail-aware reweighting both leverage $Z_\phi(s)$ but intervene at different points of the policy update:

- **Baseline risk sensitivity (this section)** modifies the *centering* of advantages by replacing $\mathbb{E}[Z_\phi(s)]$ with $\rho(Z_\phi(s))$ in Eqs. equation 35–equation 37. The PPO objective remains unchanged. This effect is typically *mild and indirect*: it primarily changes the variance, scale, and stability of the advantage signal by using a more pessimistic or optimistic reference.
- **Tail-aware advantage shaping (Sec. B)** keeps the baseline (and thus the advantage definition) standard, but *reallocates gradient weight* across tokens/outcomes via an outcome-gated multiplicative factor $w_\alpha(s_t, G_t)$ (Eqs. equation 10–equation 11). This is a *more direct* mechanism for emphasizing state-conditional tail events in the policy gradient.

Practically, baseline risk sensitivity is a gentle variant that can improve stability (especially in long-horizon, terminal-sparse RLVR), while tail reweighting explicitly focuses optimization on tail outcomes. The two mechanisms are complementary: one changes the reference point of advantages, the other changes how much each advantage contributes to the actor update.

On expected gradients. Crucially, $V_\phi^\rho(s)$ depends only on the state s and not on the sampled action. Therefore, in the exact policy-gradient view, replacing the baseline by any state-dependent functional $\rho(Z_\phi(s))$ does not change the *expected* gradient direction, and mainly affects gradient variance and optimization dynamics. In our setting, this implies that risk-aware baselines provide an indirect risk preference chiefly through stabilizing (or sharpening) the advantage signal under heavy-tailed, bimodal return distributions typical of RLVR.

Empirical evidence. The effect of applying risk sensitivity only through the baseline is empirically modest but consistent with the variance-reduction interpretation above. Ablations in Table 3 include these results. Compared to the neutral baseline ($\rho(Z) = \mathbb{E}[Z]$), risk-averse and risk-seeking baselines typically change training stability and the relative emphasis on tail outcomes, while producing smaller gains than directly shaping the policy update. These results support the view that baseline-level risk sensitivity primarily acts as an indirect control knob (through advantage normalization and noise structure), whereas stronger behavioral shifts require risk to enter the policy-gradient weighting more explicitly (as in tail-aware shaping in the main text).

F.2 A Degenerate Scalar Target for Distributional-Critic Updates

This section records a commonly used but *degenerate* way of updating a distributional critic: using a single scalar return target to supervise *all* quantile/atom outputs. In our RLVR setting, such a scalar target typically comes from a standard scalar-return pipeline (e.g., Monte-Carlo return, λ-return, or GAE-based return), which we denote by $R_t \in \mathbb{R}$ for each prefix state s_t.

Quantile-based heads (IQN / fixed-quantile regression). Consider a quantile critic parameterized by a quantile function $\hat{z}_\phi(s;\tau)$, and a fixed set of quantile fractions $\{\tau_i\}_{i=1}^{N_q}$ (for QR, $\tau_i = \frac{i-\frac{1}{2}}{N_q}$; for IQN, τ may be sampled from $\mathcal{U}(0,1)$). A degenerate scalar-target update applies the quantile regression loss by pairing the *same* scalar target R_t with every quantile output:

$$\min_\phi \ \mathbb{E}\left[\sum_{i=1}^{N_q} \rho_{\tau_i}^\kappa (R_t - \hat{z}_\phi(s_t;\tau_i))\right], \tag{42}$$

where $\rho_\tau^\kappa(\cdot)$ is the standard (Huber) quantile regression penalty.

This construction collapses the intended return-distribution supervision into a single point target.

[Scalar target collapses quantiles] Let the target distribution be the degenerate distribution δ_{R_t} at a scalar return R_t. Then, for any $\tau \in (0,1)$, the (population) minimizer of the quantile regression objective is $\hat{z}^*(s_t;\tau) = R_t$. In particular, under Eq. equation 42, the optimal solution satisfies

$$\hat{z}_\phi^*(s_t;\tau_1) = \cdots = \hat{z}_\phi^*(s_t;\tau_{N_q}) = R_t, \tag{43}$$

so the learned distribution collapses to a point mass (up to function-approximation error).

Quantile regression is Fisher-consistent: the minimizer at fraction τ equals the τ-quantile of the target distribution. For the degenerate target δ_{R_t}, all quantiles coincide at R_t, hence $\hat{z}^*(s_t;\tau) = R_t$ for every $\tau \in (0,1)$, which implies Eq. equation 43. □

Categorical head (C51). For C51, the critic predicts a categorical distribution on a fixed support $\{v_\ell\}_{\ell=1}^L$:

$$Z_\phi(s_t) = \sum_{\ell=1}^L p_\phi^\ell(s_t)\, \delta_{v_\ell}, \qquad \sum_{\ell=1}^L p_\phi^\ell(s_t) = 1.$$

A degenerate scalar-target update first projects the scalar return onto the categorical support and then minimizes cross-entropy:

$$y_t \ \triangleq \ \Pi_Z(R_t) \ \in \ \Delta^{L-1}, \qquad \min_\phi \ \mathrm{CE}(y_t,\ p_\phi(\cdot \mid s_t))\,. \tag{44}$$

Since y_t is a deterministic projection of a *single* scalar, the optimal predictor matches this projected point target, yielding a highly concentrated distribution (one-hot if R_t lies on the support, or mass on at most two neighboring atoms under linear interpolation). Thus, C51 also collapses distributional supervision to an (almost) point mass.

Interpretation and relation to terminal-sparse TD(0). Using a scalar target R_t to supervise all quantile/atom outputs is a degenerate approximation to distributional learning: it discards dispersion and tail structure and trains the critic to represent (approximately) δ_{R_t} rather than a non-trivial return distribution. In terminal-sparse RLVR, this degeneration aligns with the weakness of TD(0) supervision discussed in Sec. C.2: when early-prefix targets are dominated by bootstrapping, and the bootstrapped endpoint distribution itself has been collapsed by scalar supervision, distributional information about rare terminal outcomes cannot be propagated to early tokens. This motivates using genuinely distributional targets (e.g., the truncated-geometric bootstrapped mixture and SR sampling in Sec. C.4) instead of scalarizing the target before updating the distributional critic.

Empirical evidence. The degenerate scalar-target update discussed above is also reflected in our experiments. As reported in Table 5, using a scalar return to supervise all quantile/atom outputs yields consistently worse performance (and does not realize the benefits of distributional learning), aligning with the theoretical implication that the target collapses toward a point mass and thus fails to preserve dispersion and tail information needed for effective credit assignment under terminal-sparse RLVR.

G Case Study

We analyze the differences in model outputs after training with different algorithms. We find that GRPO tends to produce longer responses, while PPO performs worse in terms of answer accuracy. In contrast, our proposed DistRL (SR) and DistRL (Tail) achieve higher accuracy with greater stability. DistRL (seeking) and DistRL (averse) are more likely to exhibit distinct reasoning patterns.

Case: Samples from the MATH500 dataset

[Prompt] Find the number of ordered pairs (a, b) of integers such that $|a + bi| \leq 5$. Let's think step by step and output the final answer within `\boxed{}`.

[Ground Truth] $\boxed{81}$

Case: Inference trajectory of the model trained by GRPO

[Question] Find the number of ordered pairs (a, b) of integers such that $|a + bi| \leq 5$. Let's think step by step and output the final answer within \boxed{}.

[Inference] First, I need to understand what the problem is asking. We're to find the number of ordered integer pairs (a, b) such that the absolute value or the modulus of the complex number $a + bi$ is less than or equal to 5. The modulus is equal to the square root of $(a^2 + b^2)$, so we're looking for all integer solutions where this square root is ≤ 5.
I'll start by counting all possible ordered pairs (a, b) where a and b are non-negative. For each modulus value k (where k is an integer from 0 to 5), there is a number of solutions: $2k$. The total number of ordered pairs for each k is $2k$. Adding these up from $k = 0$ to $k = 5$, we get a total of $0 + 2 + 4 + 6 + 8 + 10 = 26$ ordered pairs where only one of a or b is zero. Since we want all ordered pairs, including those where both a and b are non-zero, I'll consider their symmetrical counterparts in each quadrant. However, for cases where a or b are zero, we must only count them as pairs once. But an alternative approach is to recognize that for each a and b, whether they are positive or negative doesn't change the modulus condition, so we can calculate the total for non-negative a and b and then adjust by considering their multiples.
Rather than separating into cases for a and b being positive or negative, it's simpler to note that each non-zero modulus k (where $k > 0$) contributes 4 ordered pairs if both a and b are non-zero, and 2 ordered pairs based on the signs of b or a.
So, for k from 1 to 5, the count is $4(2k - 1)$, because excluding the cases where a or b is zero, each modulus k contributes $2k$ solutions based solely on each a and b being positive or negative, but I'm about the exact count of multiplicative factors. Hence, the total number of pairs should be $(2k)$ for modulus zero together with $4(2k - 1)$ for moduli greater than zero. Therefore, calculating this sum gives the total count as desired. </think>
We are to find the number of ordered pairs (a, b) such that $|a + bi| = \sqrt{a^2 + b^2} \leq 5$. This is equivalent to finding integer solutions (a, b) satisfying $a^2 + b^2 \leq 25$.
Each integer pair (a, b) corresponds to a lattice point within or on a circle of radius 5 centered at the origin. The number of such points can be calculated using the formula for circle lattice points, but since we're dealing with integers:
For each integer a from -5 to 5, count the number of integers b such that $b^2 \leq 25 - a^2$. Sum these counts.
Compute for each a: - $a = 0$: b can be -5 to $5 \Rightarrow 11$ points. - $a = \pm 1$: $b^2 \leq 24 \Rightarrow b = -4, \ldots, 4 \Rightarrow 9$ points each $\Rightarrow 18$. - $a = \pm 2$: $b^2 \leq 21 \Rightarrow 9$ points each $\Rightarrow 18$. - $a = \pm 3$: $b^2 \leq 16 \Rightarrow 9$ points each $\Rightarrow 18$. - $a = \pm 4$: $b^2 \leq 9 \Rightarrow 7$ points each $\Rightarrow 14$. - $a = \pm 5$: $b = 0 \Rightarrow 1$ point each $\Rightarrow 2$.
Total: $11 + 18 + 18 + 18 + 14 + 2 = 81$.

[Answer] $\boxed{81}$

Case: Inference trajectory of the model trained by PPO

[Question] Find the number of ordered pairs (a, b) of integers such that $|a + bi| \leq 5$. Let's think step by step and output the final answer within \boxed{}.

[Inference] To find the number of integer ordered pairs (a, b) such that $|a + bi| \leq 5$, I start by recognizing that the expression $|a + bi|$ represents the magnitude of the complex number $a + bi$, which is given by $\sqrt{a^2 + b^2}$.
The inequality $\sqrt{a^2 + b^2} \leq 5$ can be squared to remove the square root, resulting in $a^2 + b^2 \leq 25$. This means we're looking for all integer pairs (a, b) that lie within or on the circle centered at the origin with a radius of 5.
Next, I consider the possible integer values for a. Since $a^2 \leq 25$, a can range from -5 to 5, inclusive. For each value of a, I determine the maximum and minimum values of b such that $b^2 \leq 25 - a^2$.
For each a, the range of b will be from $-\sqrt{25 - a^2}$ to $\sqrt{25 - a^2}$, rounded to integers. I calculate this for each a and sum the number of possible b values, considering all four quadrants (both positive and negative b). This approach ensures that all possible integer pairs within the circle are accounted for, leading to the final count of 169 ordered pairs.
</think>
Let a and b be integers such that $|a + bi| \leq 5$. The condition translates to $a^2 + b^2 \leq 25$.
To find the number of integer solutions, we count all pairs (a, b) where a and b are integers satisfying the above inequality.
For each integer a such that $-5 \leq a \leq 5$, the possible integer values of b range from $-\lfloor\sqrt{25 - a^2}\rfloor$ to $\lfloor\sqrt{25 - a^2}\rfloor$.
Summing the number of possible b values for each a gives the total number of ordered pairs.
Thus, the total number of ordered pairs is:

$$\boxed{169}$$

[Answer] $\boxed{169}$

Case: Inference trajectory of the model trained by DistRL(SR)

[Question] Find the number of ordered pairs (a, b) of integers such that $|a + bi| \leq 5$. Let's think step by step and output the final answer within `\boxed{}`.

[Inference] First, I need to determine the conditions for the integer pairs (a, b) such that the magnitude of the complex number $a + bi$ is less than or equal to 5.
Next, I'll identify all possible integer values of a and b that satisfy this condition.
Then, I'll count the number of valid pairs for each value of a and sum them up to get the final result. `</think>`
To find the number of ordered pairs (a, b) of integers such that $|a + bi| \leq 5$, we start by noting that the magnitude of the complex number $a + bi$ is given by:

$$|a + bi| = \sqrt{a^2 + b^2}.$$

We need this to be less than or equal to 5:

$$\sqrt{a^2 + b^2} \leq 5.$$

Squaring both sides:

$$a^2 + b^2 \leq 25.$$

This is the equation of a circle centered at the origin with radius 5. We need to find all integer pairs (a, b) that lie inside or on this circle.
Steps to Find the Number of Pairs:
1. **Determine the range for a:** Since $a^2 \leq 25$, a can range from -5 to 5.
2. **For each integer a in -5 to 5, determine the range for b:** For each a, b must satisfy:

$$b^2 \leq 25 - a^2, \quad |b| \leq \sqrt{25 - a^2}.$$

Since b must be an integer, b ranges from $-\lfloor\sqrt{25 - a^2}\rfloor$ to $\lfloor\sqrt{25 - a^2}\rfloor$.
3. **Count the number of valid b for each a:** For each a, the number of b values is $2\lfloor\sqrt{25 - a^2}\rfloor + 1$.
Calculation:
- When $a = 0$: $b = -5, \ldots, 5 \Rightarrow 11$ values. - When $a = \pm 1$: $b = -4, \ldots, 4 \Rightarrow 9$ values each. - When $a = \pm 2$: $b = -4, \ldots, 4 \Rightarrow 9$ values each. - When $a = \pm 3$: $b = -4, \ldots, 4 \Rightarrow 9$ values each. - When $a = \pm 4$: $b = -3, \ldots, 3 \Rightarrow 7$ values each. - When $a = \pm 5$: $b = 0 \Rightarrow 1$ value each.
Total:

$$11 + 18 + 18 + 18 + 14 + 2 = 81.$$

Thus, the number of ordered pairs (a, b) is $\boxed{81}$.

[Answer] $\boxed{81}$

Case: Inference trajectory of the model trained by DistRL(Tail)

[Question] Find the number of ordered pairs (a, b) of integers such that $|a + bi| \leq 5$. Let's think step by step and output the final answer within \boxed{}.

[Inference] First, I need to understand the problem requirements.
The question asks for the number of ordered pairs of integers (a, b) such that the absolute value of $a + bi$ is less than or equal to 5.
I recognize that $|a + bi|$ represents the magnitude of a complex number, which is calculated as the square root of $(a^2 + b^2)$.
To simplify the problem, I'll square both sides of the inequality to eliminate the square root, resulting in $a^2 + b^2$ being less than or equal to 25.
Next, I'll determine the possible integer values for a and b that satisfy this inequality.
Finally, I'll count all valid ordered pairs to find the solution. </think>
Let's determine the number of ordered pairs (a, b) of integers such that $|a + bi| \leq 5$.
The magnitude of the complex number $a + bi$ is:

$$|a + bi| = \sqrt{a^2 + b^2}.$$

We need:

$$\sqrt{a^2 + b^2} \leq 5.$$

Squaring both sides:

$$a^2 + b^2 \leq 25.$$

We are looking for all integer pairs (a, b) that satisfy this inequality. This is equivalent to finding all integer points inside or on the circle with radius 5 centered at the origin.
For each integer $a \in \{-5, -4, \ldots, 5\}$: - Calculate $25 - a^2$. - Count integer b with $b^2 \leq 25 - a^2$.
The number of integer b values for each a is $2\lfloor\sqrt{25 - a^2}\rfloor + 1$.
Compute: 1. $a = 0$: $b \in [-5, 5] \Rightarrow 11$. 2. $a = \pm 1$: $b \in [-4, 4] \Rightarrow 9$ each. 3. $a = \pm 2$: $b \in [-4, 4] \Rightarrow 9$ each. 4. $a = \pm 3$: $b \in [-4, 4] \Rightarrow 9$ each. 5. $a = \pm 4$: $b \in [-3, 3] \Rightarrow 7$ each. 6. $a = \pm 5$: $b = 0 \Rightarrow 1$ each.
Sum:

$$11 + 18 + 18 + 18 + 14 + 2 = 81.$$

Thus, the number of ordered pairs (a, b) is $\boxed{81}$.

[Answer] $\boxed{81}$

Case: Inference trajectory of the model trained by DistRL(seeking)

[Question] Find the number of ordered pairs (a, b) of integers such that $|a + bi| \leq 5$. Let's think step by step and output the final answer within \boxed{}.

[Inference] First, the problem requires finding the number of ordered pairs (a, b) of integers such that the magnitude of the complex number $a + bi$ is less than or equal to 5. The magnitude is given by $\sqrt{a^2 + b^2}$, so we need $\sqrt{a^2 + b^2} \leq 5$. Squaring both sides, we get $a^2 + b^2 \leq 25$.
Next, we need to count all integer pairs (a, b) that satisfy this inequality. This is equivalent to finding all integer points inside or on the circle of radius 5 centered at the origin.
We can approach this by considering all possible integer values of a from -5 to 5 and for each a, determining the range of b such that $b^2 \leq 25 - a^2$. For each a, the number of valid b values is $2\sqrt{25 - a^2} + 1$ (including zero).
Summing up the valid b values for each a from -5 to 5 gives the total number of ordered pairs. </think>
To solve the problem, we need to find the number of ordered pairs (a, b) of integers such that $|a + bi| \leq 5$. This is equivalent to finding the number of integer solutions to:

$$a^2 + b^2 \leq 25.$$

This represents all integer points inside or on a circle of radius 5 centered at the origin. We can count by each a.
For each integer a with $-5 \leq a \leq 5$, the maximum integer b satisfies:

$$b^2 \leq 25 - a^2, \quad |b| \leq \sqrt{25 - a^2}.$$

Thus, for each a, the number of integer b is $2\lfloor\sqrt{25 - a^2}\rfloor + 1$.
We can calculate this for each a from 0 to 5 and then account for symmetry.
A more straightforward approach is to recognize:

$$\text{Total points} = 1 + 4\sum_{k=1}^{5} \left\lfloor \sqrt{25 - k^2} \right\rfloor.$$

Computing: - $a = 0$: 11 - $a = \pm 1$: 18 - $a = \pm 2$: 18 - $a = \pm 3$: 18 - $a = \pm 4$: 14 - $a = \pm 5$: 2
Total $= 81$.
Thus, the number of ordered pairs (a, b) is $\boxed{81}$.

[Answer] $\boxed{81}$

Case: Inference trajectory of the model trained by DistRL(averse)

[Question] Find the number of ordered pairs (a, b) of integers such that $|a + bi| \leq 5$. Let's think step by step and output the final answer within \boxed{}.

[Inference] First, I identify that the problem involves finding integer pairs (a, b) such that the absolute value of $a + bi$ is less than or equal to 5.
Next, I recognize that the absolute value of a complex number $a + bi$ is given by the square root of $(a^2 + b^2)$.
Then, I set up the inequality $\sqrt{a^2 + b^2} \leq 5$ and square both sides to eliminate the square root.
Finally, I determine the integer solutions (a, b) that satisfy the inequality within the range defined by the square of 5. </think>
To find the number of ordered pairs (a, b) of integers such that $|a + bi| \leq 5$, we start by noting that $|a + bi| = \sqrt{a^2 + b^2}$. Therefore, the condition translates to $\sqrt{a^2 + b^2} \leq 5$, or equivalently, $a^2 + b^2 \leq 25$.
We need to count the number of integer pairs (a, b) that satisfy this inequality. This is equivalent to finding the number of lattice points inside or on the circle of radius 5 centered at the origin.
To do this, we consider all integers a and b such that $a^2 + b^2 \leq 25$. We can break this down by checking each possible value of a and determining the corresponding possible values of b.
The possible values for a are integers from -5 to 5. For each a, the maximum value of b is $\lfloor\sqrt{25 - a^2}\rfloor$.
Let's calculate the number of valid b for each a: - $a = 0$: $b = -5, \ldots, 5 \Rightarrow 11$ - $a = \pm 1$: $b = -4, \ldots, 4 \Rightarrow 9$ each - $a = \pm 2$: $b = -4, \ldots, 4 \Rightarrow 9$ each - $a = \pm 3$: $b = -4, \ldots, 4 \Rightarrow 9$ each - $a = \pm 4$: $b = -3, \ldots, 3 \Rightarrow 7$ each - $a = \pm 5$: $b = 0 \Rightarrow 1$ each
Total: $11 + 18 + 18 + 18 + 14 + 2 = 81$.
Thus, the number of ordered pairs (a, b) is $\boxed{81}$.

[Answer] $\boxed{81}$

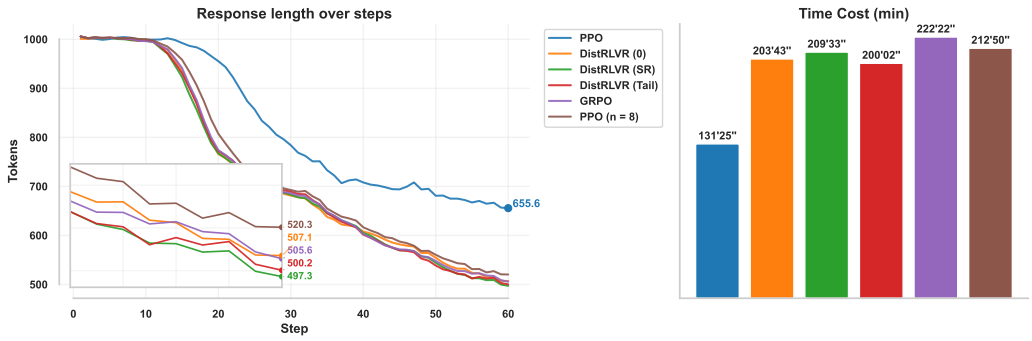

Figure 6: Training time and response length analysis for PPO, GRPO, and DistRLVR.