RIDING THE WAVES OF UNCERTAINTY: DISTRIBUTIONAL VALUE LEARNING FOR LLMS WITH VERIFIABLE REWARDS

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning with verifiable rewards (RLVR) trains large language models from feedback on completed responses, yet learning a useful value baseline from sparse terminal outcomes remains difficult. A conventional PPO critic regresses a scalar expected return; a multi-output value objective offers an alternative way to learn from the same feedback. We propose DISTRLVR, a distributional actor–critic framework that predicts prefix-conditioned returns and uses their expectation as the PPO baseline. Dual Sample Replacement (dSR) constructs multi-step critic targets from existing trajectories and target-critic predictions, supplying additional supervision without generating additional responses. The predicted quantiles also support optional tail-aware advantage reweighting. Experiments on mathematical reasoning and executable code generation show improvements over PPO and GRPO, including gains over VAPO in mean code reward; instruction-following results further examine transfer to graded programmatic feedback. Scalar-target and capacity controls support an optimization-based interpretation of the gains: multi-output losses can improve value learning without requiring accurate reconstruction of the complete return law. DistRLVR with dSR provides an effective RLVR training recipe, while tail reweighting yields a task-dependent trade-off between average quality and repeated-sampling success. Code is available in the anonymous repository.

1 INTRODUCTION

Verifiable rewards make reinforcement learning practical for mathematical reasoning and code generation: a completed response can be checked without labeling every intermediate step (Shao et al., 2024; Guo et al., 2025). For actor–critic RLVR, terminal verification places substantial credit-assignment pressure on value estimation. The critic must value every token prefix from a reward observed after a long response, even though stochastic continuations of the same prefix may succeed or fail. Estimation errors then perturb the advantages used by PPO (Schulman et al., 2017; 2016).

A conventional PPO critic regresses the expected return of each prefix. An accurately estimated conditional mean provides the standard state-value baseline used by GAE, but its supervision remains sparse: a trajectory supplies one realized suffix per visited prefix, and bootstrapping propagates critic error backward. Independent completions improve coverage at the cost of additional generation. Distributional prediction offers another training formulation: it models the uncertain outcomes of a prefix while retaining their expectation as the actor baseline.

The role of the distribution depends on the reward. With deterministic binary terminal feedback and $\gamma = 1$, the prefix return is Bernoulli, so its calibrated mean determines the population law. This motivates an optimization view of distributional value learning: replacing scalar regression with a multi-output loss can improve how the critic learns from sparse samples, without requiring accurate reconstruction of the full return distribution. Under graded rewards, prefixes with the same mean can additionally differ in dispersion or tails. We study this critic-training formulation across both reward regimes (Figure 1).

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

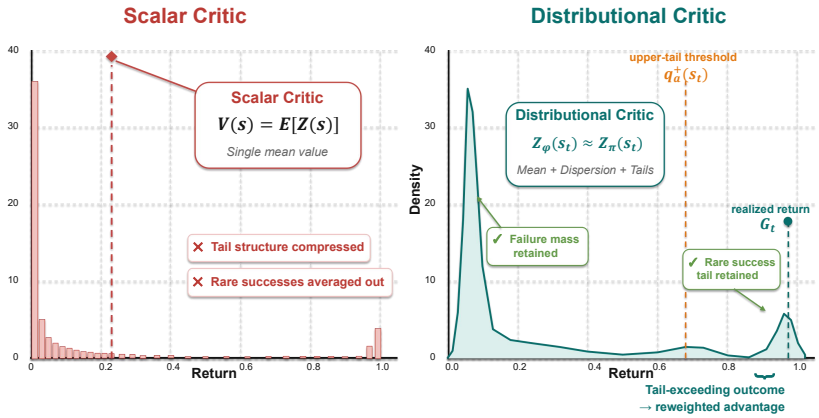


Figure 1: Scalar and distributional critics. A scalar critic retains only the mean of a return law, whereas a distributional critic represents its dispersion and tails and can supply the threshold used by optional Tail reweighting. The plotted law is schematic rather than a measured critic prediction. It illustrates graded returns; with deterministic binary terminal reward and $\gamma = 1$, the Bernoulli law is fully determined by its mean.

DistRLVR pairs PPO with a categorical or quantile return predictor at each token prefix (Bellemare et al., 2017; Dabney et al., 2018a). Its expectation supplies the usual mean-value baseline. To train the predictor from terminal feedback, we adapt GMAC's Sample Replacement construction (Nam et al., 2021) into dual Sample Replacement (dSR). A backward sampler combines target-critic draws with multi-step returns, using the observed verifier reward at the terminal boundary. The K completions broaden prompt-level trajectory coverage, whereas M controls sampling from the conditional target at each visited prefix. Thus, dSR adds critic-target samples without additional response generation. An optional quantile-threshold rule reweights selected advantages; the default dSR configuration uses ordinary mean-value advantages.

Across mathematical reasoning benchmarks, dSR improves aggregate accuracy and Pass@128 over PPO and GRPO under a common rollout protocol. The gains also extend to programmatic constraint satisfaction and executable code, including a controlled VAPO comparison. Longer-response and cross-model experiments show that the method transfers beyond the primary configuration. The generation-budget study distinguishes the benefit of new trajectories from that of additional target samples, while measured runtime shows that dSR can be integrated into PPO training at comparable end-to-end cost.

Contributions. We develop a token-prefix distributional value learner for RLVR that retains PPO's mean-value advantage calculation; introduce dSR supervision that samples multi-step targets from existing completions without additional rollout generation; and demonstrate policy improvements across binary mathematics and graded instruction-following and code rewards. Scalar-target and capacity controls, together with budget and runtime measurements, characterize the practical value of this critic-training formulation.

2 Related Work

Distributional value learning for LLM PPO. C51, quantile regression, and IQN establish return representations for distributional reinforcement learning (Bellemare et al., 2017; Dabney et al., 2018b;a). DisPPO demonstrates the practical benefits of quantile critics and distributional estimation in PPO recently (Zhou et al., 2026a). Building on this, DistRLVR addresses a concrete bottleneck in critic supervision under terminal verification in LLM RL post-training: each visited prefix typically has only one realized continuation, while one-step targets at zero-reward intermediate states depend

entirely on bootstrapped critic predictions, supporting both categorical and quantile critics, with the learned mean supplying the standard PPO baseline.

Distributional modeling and tail-aware policy updates. Quantile reward models estimate attribute-score distributions and aggregate them using preference supervision (Dorka, 2024). DVPO combines token-level value distributions with asymmetric risk control (Zhu et al., 2025a), DFPO develops value-flow modeling with risk and consistency constraints (Zhu et al., 2026), and Q# uses reference-policy return distributions for KL-regularized generation (Zhou et al., 2025). In RLVR, FlowRL matches the response distribution to a reward-induced target through trajectory balance (Zhu et al., 2025b), while RS-GRPO derives risk-sensitive advantages estimated from grouped outcomes (Jiang et al., 2026). DistRLVR connects critic learning to tail use through an optional Tail gate: a learned upper-quantile threshold at each prefix determines how realized outcomes reweight PPO advantages. This enables prefix-conditioned use of tail information alongside the default mean-value update, with task-dependent trade-offs evaluated separately from dSR.

Evaluation of the PPO training recipe. GRPO, DAPO, and VAPO improve RLVR through group-relative updates, refined optimization, and value-based training (Shao et al., 2024; Yu et al., 2025; Yue et al., 2025). Recent categorical-critic work also highlights the role of the value-learning loss (Zhou et al., 2026b). Our study combines rollout-matched PPO/GRPO comparisons with VAPO comparisons on code, and spans binary mathematical verification and graded instruction-following and code rewards. Our evaluation also connects distributional value learning to the practical demands of LLM PPO, demonstrating policy gains while examining critic supervision, tail use, and computational cost.

3 Problem Formulation

A prompt x and its generated prefix define the state $s_t = (x, y_{<t})$. The policy samples a token $a_t \sim \pi_\theta(\cdot \mid s_t)$, and the next state appends that token. An episode has actions $a_0, \ldots, a_{T-1}$ and terminates at s_T. We write transition rewards as r_t for $t < T$ and an additional verifier reward as r_T; there is no action a_T. This convention attaches the verifier reward exactly once. For $\gamma \in (0, 1]$, the return and value distribution are

$$G_t = \sum_{j=t}^{T} \gamma^{j-t} r_j, \qquad Z^\pi(s_t) \overset{D}{=} G_t \mid s_t, \qquad V^\pi(s_t) = \mathbb{E}[Z^\pi(s_t)]. \tag{1}$$

For terminal-only binary verification with $\gamma = 1$, $Z^\pi(s_t) \sim \text{Bernoulli}(V^\pi(s_t))$. This identity rules out a population-information advantage over a calibrated scalar critic. We evaluate how the finite-sample critic-training formulation affects downstream policy performance.

Actor update. Given a value estimate V_t, generalized advantage estimation (GAE) uses

$$\delta_t = r_t + \gamma V_{t+1} - V_t, \qquad \hat{A}_t = \sum_{l=0}^{T-t-1} (\gamma \lambda_{\text{GAE}})^l \delta_{t+l}. \tag{2}$$

Under the terminal-state reward convention, set $V_T = r_T$ when constructing these targets. PPO maximizes

$$\mathcal{J}_{\text{PPO}}(\theta) = \mathbb{E}_t \left[\min\{u_t(\theta)\hat{A}_t, \text{clip}(u_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\} \right], \tag{3}$$

where $u_t(\theta) = \pi_\theta(a_t \mid s_t)/\pi_{\text{old}}(a_t \mid s_t)$ and advantages are held fixed during the actor update. We use λ_{GAE} for actor-side traces and λ_{SR} for critic-target sampling; they are separate parameters.

4 DistRLVR

4.1 A distributional state-value critic

Figure 2 shows the training workflow. We replace PPO's scalar critic by $Z_\phi(s)$, a return distribution conditioned on the prompt and generated prefix. The default actor baseline is its mean, $V_\phi(s) =$

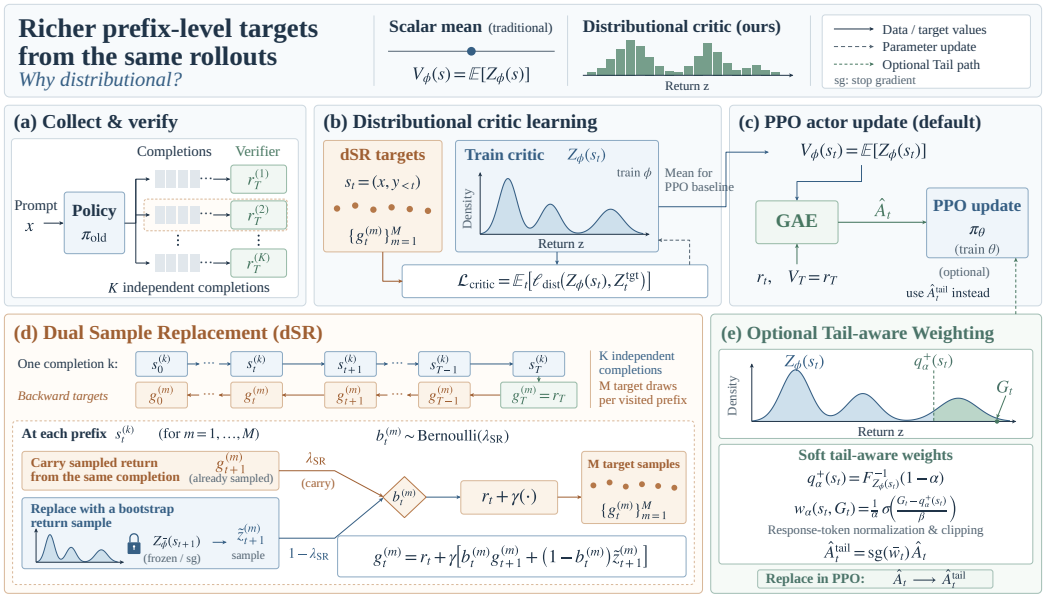


Figure 2: DistRLVR workflow. (a–c) Verified completions supervise a return critic whose mean supplies the PPO/GAE baseline. (d) dSR draws M targets per prefix from K completions and frozen-critic predictions. (e) Tail optionally reweights advantages. Densities illustrate graded returns; binary terminal returns with $\gamma = 1$ are determined by their mean.

$\mathbb{E}[Z_\phi(s)]$. This retains the standard clipped PPO surrogate, with advantages computed from the critic mean.

We use three established parameterizations: C51 predicts probabilities on fixed support atoms, QR predicts fixed quantile locations, and IQN conditions a quantile predictor on a sampled fraction (Bellemare et al., 2017; Dabney et al., 2018b;a). Our default is IQN. These are state-value critics; no greedy action-value target or DQN control rule is used. The head-specific losses are given in Appendix B.

With a frozen target critic $Z_{\bar{\phi}}$, the one-step target is

$$Z_t^{\text{tgt}} \overset{D}{=} r_t + \gamma Z_{\bar{\phi}}(s_{t+1}), \qquad \mathcal{L}_{\text{critic}} = \mathbb{E}_t[\ell_{\text{dist}}(Z_\phi(s_t), Z_t^{\text{tgt}})]. \tag{4}$$

At termination we set $Z_{\bar{\phi}}(s_T) = \delta_{r_T}$, where δ_c denotes a point mass at c. Targets are detached from gradient computation.

4.2 Distributional Multi-Step Targets

At a nonterminal prefix with zero immediate reward, a one-step target is obtained entirely from the next prefix's critic prediction. We use the Sample Replacement (SR) construction of Nam et al. (2021). Our RLVR instantiation combines K completions per prompt with M SR draws per visited prefix; we call this sampling protocol dual Sample Replacement (dSR).

Condition on a sampled trajectory and a fixed target critic. Write $H_t = T - t$. For $1 \le n \le H_t$, define the conditional target law

$$\tilde{Z}_t^{(n)} \overset{D}{=} \sum_{i=0}^{n-1} \gamma^i r_{t+i} + \gamma^n Z_{\bar{\phi}}(s_{t+n}), \qquad Z_{\bar{\phi}}(s_T) = \delta_{r_T}. \tag{5}$$

For $\lambda_{\text{SR}} \in [0, 1)$, the SR target is a mixture of these laws with depth probabilities

$$p_t(n) = \begin{cases} (1 - \lambda_{\text{SR}})\lambda_{\text{SR}}^{n-1}, & 1 \le n < H_t, \\ \lambda_{\text{SR}}^{H_t - 1}, & n = H_t. \end{cases} \tag{6}$$

The resulting target law is

$$\mathcal{L}(\widetilde{Z}_t^{(\lambda_{\mathrm{SR}})}) = \sum_{n=1}^{H_t} p_t(n)\mathcal{L}(\widetilde{Z}_t^{(n)}). \tag{7}$$

Here $\mathcal{L}(X)$ denotes the probability law of X; the sum mixes probability laws, rather than averaging sampled returns.

SR samples this mixture in one backward sweep. Independently for each $m = 1, \ldots, M$, set $g_T^{(m)} = r_T$ and compute

$$\begin{aligned} g_t^{(m)} &= r_t + \gamma \left[b_t^{(m)} g_{t+1}^{(m)} + (1 - b_t^{(m)}) \widetilde{z}_{t+1}^{(m)} \right], \\ b_t^{(m)} &\sim \mathrm{Bernoulli}(\lambda_{\mathrm{SR}}), \qquad \widetilde{z}_{t+1}^{(m)} \sim Z_{\bar{\phi}}(s_{t+1}). \end{aligned} \tag{8}$$

Replacement draws and Bernoulli variables are independent conditional on the trajectory and target critic. Replacement draws need only be generated when used.

Lemma 4.1 (Conditional sampling identity)**.** *For a fixed trajectory, target critic, and prefix index $t < T$, the M draws in Eq. (8) are conditionally independent and have common law $\mathcal{L}(\widetilde{Z}_t^{(\lambda_{\mathrm{SR}})})$.*

The proof is in Appendix A. The identity characterizes the conditional bootstrapped target law. Targets at different prefix indices within the same backward sweep remain dependent.

For each prompt, dSR applies this construction to K independently sampled completions. The resulting targets supervise the corresponding visited prefix states. Distinct completions generally visit distinct nonempty prefixes, so K is a prompt-level coverage parameter, whereas M controls conditional SR sampling at a visited state. Thus K adds real trajectories and M adds conditional bootstrapped supervision.

Constructing the targets takes $O(M \sum_{k=1}^{K} T_k)$ arithmetic operations, excluding neural-network evaluation and distributional-loss computation. Longer backup depths are available, but their probability decreases geometrically: $\Pr(N = H_t) = \lambda_{\mathrm{SR}}^{H_t - 1}$ and $\mathbb{E}[N] = (1 - \lambda_{\mathrm{SR}}^{H_t})/(1 - \lambda_{\mathrm{SR}})$. Consequently, the default $\lambda_{\mathrm{SR}} = 0.9$ gives an expected depth of at most ten tokens. Early prefixes receive terminal information mainly through successive critic updates; a target includes the realized terminal reward directly when its sampled backup reaches the terminal boundary.

4.3 Optional outcome-dependent tail reweighting

The default method uses the mean baseline and ordinary PPO advantages. An optional rule gives different weights to outcomes relative to the predicted upper quantile. Define

$$\begin{aligned} q_\alpha^+(s) &= F_{Z_\phi(s)}^{-1}(1 - \alpha), \qquad w_\alpha(s, G) = \frac{1}{\alpha} \sigma\left(\frac{G - q_\alpha^+(s)}{\beta} \right), \\ \hat{A}_t^{\mathrm{tail}} &= \mathrm{sg}(\bar{w}_t) \hat{A}_t, \end{aligned} \tag{9}$$

where $0 < \alpha < 1$, $\beta > 0$, G_t is the realized return, σ is the sigmoid, and sg stops gradients. Here $\bar{w}_t$ is the response-token-normalized and clipped weight applied to the actor. The outcome-dependent gate defines a weighted PPO surrogate. For discrete returns, probability atoms at the threshold affect the selected mass; response-token normalization controls the scale of the applied weights.

Changing a state-only baseline to a tail functional is a different intervention. With an unbiased Monte Carlo return, an action-independent baseline does not change the exact expected policy gradient. We discuss this distinction and discrete-safe tail functionals in Appendix E.2.

Training sequence. Generate and verify K completions per prompt under π_{old}; freeze target predictions; draw M SR targets at each valid response prefix; fit the critic; compute mean-value GAE; and update PPO with Eq. (3), optionally using Tail. Response masks and completion boundaries ensure terminal rewards enter once and targets never cross sequences.

5 EXPERIMENTS

We compare rollout-matched baselines, scalar-target and capacity controls, and graded-reward methods. Differences are percentage points (pp).

5.1 SETUP AND EVALUATION

The primary policy is DeepSeek-R1-Distill-Qwen-1.5B. Training uses VERL (Sheng et al., 2025) and vLLM (Kwon et al., 2023). Table 1 summarizes the settings. Dataset protocols, metric definitions, and detailed results are given in Appendices C and D. PPO, GRPO, and DistRLVR use the same model, rollout count ($K = 8$), and evaluation settings in the primary comparison.

Table 1: Training and evaluation protocol.

Item	Setting
Model / training	DeepSeek-R1-Distill-Qwen-1.5B; VERL+vLLM; DeepScaleR math training
Primary math	$K = 8$ for matched PPO/GRPO/DistRLVR; 1,024-token cap; temperature/top-$p = 1.0/1.0$
Math reward	Extracted-answer correctness in $\{0, 1\}$; unparsable/truncated outputs receive 0; KL is separate
Math aggregate	Six DeepScaleR benchmarks plus M2M and Countdown; accuracy uses 8 settings, Pass@128 uses 7
Graded tasks	IF score; code mean test fraction and strict executable Pass@k; LoCoMo average QA score

For binary outcomes, average success estimates single-sample accuracy and Pass@k measures repeated-sampling coverage. The mathematical evaluation uses the combinatorial Pass@128 estimator defined in Appendix C.

5.2 MATCHED-ROLLOUT RESULTS AND SCALAR CONTROLS

Table 2: Mathematical results (%). PPO at $K = 1$ provides a rollout reference; other rows use $K = 8$. Accuracy uses eight settings and Pass@128 seven.

Method	Aggregate accuracy	Pass@128
PPO ($K = 1$)	28.258	63.445
PPO ($K = 8$)	32.040	64.820
GRPO	32.600	65.300
Scalar TD(0)	29.200	63.800
Scalar Bernoulli BCE	30.460	64.400
Scalar TD(λ)	31.110	65.000
Scalar Monte Carlo	30.400	64.900
Capacity-matched scalar	31.000	64.800
DistRLVR (dSR)	33.918	**67.217**
DistRLVR (Tail)	**34.962**	66.037

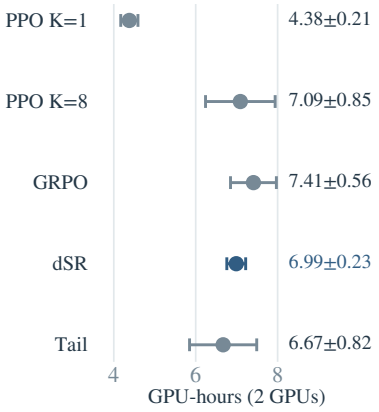


Figure 3: Two-GPU cost. Whiskers show across-run SD.

dSR improves aggregate accuracy over PPO by 1.878 pp and Pass@128 by 2.397 pp (Table 2). Relative to GRPO, the gains are 1.318 and 1.917 pp. The historical PPO-$K = 1$ result is separated from the rollout-matched rows. Figure 3 shows the corresponding training cost and across-run standard deviation.

Scalar TD(λ) scores 1.910 pp above scalar TD(0) in aggregate accuracy and 1.200 pp above it in Pass@128, yet remains below IQN+dSR. Bernoulli BCE, Monte Carlo targets, and the capacity-matched scalar head likewise score below dSR. Distributional TD(0) already reaches 33.639/65.971

before dSR adds 0.279/1.246 pp. We interpret these results primarily through the critic objective: multi-output losses offer an effective alternative to scalar regression under sparse feedback, without requiring accurate recovery of the complete return law. Full target comparisons are in Appendix D.2.

Table 3: Task-level Avg@32 (%). PPO uses $K = 1$ and DistRLVR uses $K = 8$. The final column averages the six displayed tasks.

Method	AIME24	AIME25	AMC23	MATH500	Minerva	Olympiad	6-task mean
PPO ($K = 1$)	5.104	3.264	31.903	62.590	18.053	23.012	23.988
dSR ($K = 8$)	8.125	6.042	43.737	71.025	22.522	29.241	30.115
Tail ($K = 8$)	9.549	7.361	44.829	72.623	22.794	31.451	31.435

Task-level results reveal the different difficulty of the benchmarks (Table 3). dSR reaches 8.125% and 6.042% Avg@32 on AIME 2024/2025, 43.737% on AMC 2023, and 71.025% on MATH500. The corresponding Minerva and OlympiadBench scores are 22.522% and 29.241%. Under their separate training protocols, M2M at a 4K response limit reaches 20.80% and Countdown reaches 69.85%. Full per-benchmark accuracy, Pass@128, and seed standard deviations are in Appendices D.1 and D.3, including the task-level differences between dSR and Tail.

5.3 Real completions and target samples

A separate experiment holds generated tokens fixed by adjusting prompt allocation and varies real completions K and SR target draws M (Table 4). Increasing either quantity improves both metrics, but the two axes are not interchangeable. Moving from $(1, 1)$ to $(8, 1)$ raises aggregate accuracy/Pass@128 from 31.2/64.0 to 33.0/66.0; $(1, 8)$ reaches 32.6/65.4. The K completions broaden prompt-level trajectory coverage, whereas M controls sampling from the conditional critic target. Over the displayed ranges, increasing K gives the larger gain; additional target draws remain useful without new response generation. The complete allocation experiment is in Appendix D.6.

Table 4: Allocation of a fixed generation budget between real completions K and SR target draws M. Every row uses the same number of generated tokens; accuracy and Pass@128 are percentages.

Real K	Targets M	Relative tokens	Accuracy	Pass@128
1	1	1.00	31.2	64.0
2	1	1.00	32.0	64.8
4	1	1.00	32.6	65.4
8	1	1.00	**33.0**	**66.0**
1	2	1.00	31.8	64.5
1	4	1.00	32.3	65.0
1	8	1.00	32.6	65.4

5.4 Runtime and transfer across settings

Rollout generation accounts for most training time (Figure 4). PPO-$K = 8$ and dSR take 212m50s (7.09 GPU-hours) and 209m33s (6.99), respectively. SR construction takes 10.5 minutes, approximately 5% of the dSR run. Thus the additional critic-target sampling fits within an end-to-end cost comparable to PPO; component measurements and scaling ranges are in Appendix D.7.

Table 5 summarizes the principal transfer results. The reference is GRPO for 1.5B-8K and Qwen3-8B and PPO otherwise; the 1.5B-8K row is a six-task macro-average, and protocols differ across rows.

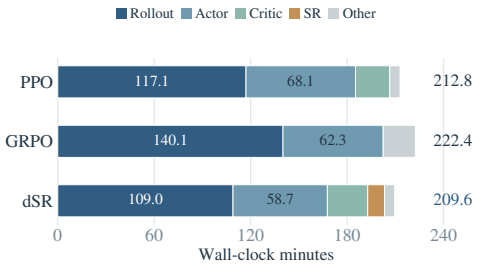


Figure 4: Two-GPU runtime by component (min).

Table 5: Transfer results (%).

Setting	Metric	Ref.	dSR	Tail
1.5B, 8K	Acc.	44.10	45.68	**45.73**
	P@128	78.37	**78.92**	77.30
Qwen3-8B	Acc.	53.24	54.88	**55.40**
	P@128	76.28	**78.24**	76.67
Phi-3-mini	Acc.	17.2	**19.2**	18.5
	P@128	55.0	**57.6**	52.9
Non-RLVR init.	Acc.	11.0	**13.5**	–
	P@128	35.0	**38.0**	–

At an 8K response limit, dSR exceeds GRPO by 1.58 pp in six-task average accuracy and 0.55 pp in Pass@128. The gains include AMC 2023 (66.4% versus 61.2%) and AIME 2024 (27.5% versus 25.3%), while Minerva is 27.8% versus 28.7%. On Qwen3-8B, the aggregate gains over GRPO are 1.64/1.96 pp. Pass@128 rises from 68.67% to 72.67% on AIME 2024 and from 50.00% to 52.67% on AIME 2025; AIME 2025 accuracy also rises from 22.45% to 27.52%.

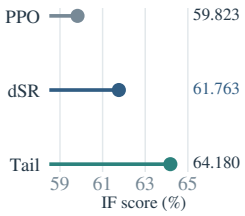
On Phi-3-mini, dSR exceeds PPO by 2.0 pp in average accuracy and 2.6 pp in Pass@128. MATH500 accuracy rises from 46.9% to 51.0%, and OlympiadBench from 13.8% to 16.4%. Post-training from a non-RLVR initialization also improves over PPO (+2.5/+3.0 pp). The full task-level results and seed deviations for these settings are in Appendix D.8.

5.5 Graded rewards beyond mathematics

Instruction following and code generation use graded programmatic training rewards: the fraction of satisfied constraints and unit tests passed, respectively. We report IF scores, mean code reward, and strict code Pass@k, which requires a fully correct solution. LoCoMo evaluates conversational-memory QA. Task results are detailed in Appendix D.9.

Table 6: Instruction-following, executable-code, and conversational-QA results (%).

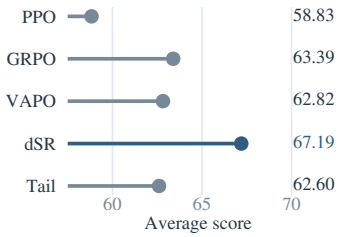
(a) Instruction following



(b) Executable code

Method	Mean	P@2	P@4	P@8
PPO	6.060	9.195	11.999	14.768
GRPO	6.308	9.354	12.556	16.409
VAPO	7.204	11.800	12.400	15.200
dSR	**14.203**	**20.223**	**26.908**	**33.746**

(c) Conversational QA



Code Mean is the fraction of tests passed; P@k requires all tests to pass. LoCoMo reports average QA score.

Instruction following. On `allenai/IF_multi_constraints_upto5`, the IF score rises from 59.823% for PPO to 61.763% for dSR (+1.940 pp). Tail reaches 64.180%, exceeding dSR by 2.417 pp. The training verifier gives credit for partial constraint satisfaction. This dataset is distinct from the official out-of-distribution IFBench test set; detailed scores are in Appendix D.9.

Executable code generation. On the Skywork/LiveCodeBench protocol (Jain et al., 2024), dSR improves mean test fraction over GRPO by 7.895 pp and over VAPO by 6.999 pp. Its strict-success advantage over GRPO grows from 10.869 pp at Pass@2 to 17.337 pp at Pass@8; the Pass@8 gain over VAPO is 18.546 pp. Thus dSR improves partial execution quality and strict success against both baselines. Detailed scores are in Appendix D.9.

Conversational QA. On LoCoMo (Maharana et al., 2024), dSR obtains an average score of 67.19, compared with 58.83 for PPO, 63.39 for GRPO, and 62.82 for VAPO. The gains are 3.80

pp over GRPO and 4.37 pp over VAPO. Tail scores 62.60. This auxiliary result complements the programmatic tasks; the full table is in Appendix D.10.

5.6 Architecture and hyperparameter sensitivity

C51, QR, and IQN provide alternative distributional heads. Their responses to dSR differ: the C51 ablation slightly decreases its six-task mean, whereas QR increases it. We use IQN in the main configuration and report the full head comparison in Appendix D.4. Table 7 summarizes the local M2M sweep (1K responses, 16 completions); Appendix D.5 gives full settings and outcomes.

Table 7: Local M2M sensitivity: accuracy / Pass@128 (%).

Parameter	Low	Low A/P	Mid	Mid A/P	High	High A/P
α	.05	15.131/60.256	.25	15.840/60.680	.50	15.736/62.678
λ_{SR}	.10	15.705/59.829	.90	16.070/60.970	.95	16.431/61.538
M	2	16.079/59.829	8	16.070/60.970	16	16.164/62.108

The displayed configurations span 15.131–16.431% accuracy and 59.829–62.678% Pass@128. Increasing M from 2 to 16 changes accuracy by 0.085 pp and Pass@128 by 2.279 pp, while the upper-tail parameter primarily changes the balance between the two metrics.

5.7 Tail reweighting across tasks

Tail raises aggregate mathematical accuracy from 33.918 to 34.962 (+1.044 pp), while changing Pass@128 from 67.217 to 66.037 (-1.180 pp). On Qwen3-8B, Tail improves average accuracy over dSR by 0.52 pp and changes Pass@128 from 78.24% to 76.67%. On Phi-3-mini, dSR exceeds Tail on both metrics: 19.2% versus 18.5% accuracy and 57.6% versus 52.9% Pass@128. In instruction following, Tail raises the IF score over dSR by 2.417 pp; the LoCoMo average favors dSR. dSR remains the default; Tail is task-dependent (Appendix D.8).

6 Limitations and conclusion

DistRLVR learns token-prefix return distributions and uses their mean as the PPO baseline. dSR combines existing completions with frozen-critic samples to construct multi-step targets, propagating terminal rewards through critic updates. Optional Tail reweights upper-return outcomes.

With rollout count matched, dSR improves aggregate mathematical accuracy and Pass@128 over PPO and GRPO. Among the listed scalar controls, none matches dSR on both metrics. On executable code, dSR improves mean test fraction over GRPO by 7.895 pp and over VAPO by 6.999 pp, and improves Pass@8 over GRPO by 17.337 pp. In the fixed-generation-token study, increasing either M or K raises the policy scores, with a larger gain from increasing K over the displayed ranges. End-to-end training cost remains comparable to PPO in the measured configuration. These findings motivate dSR as the default, with Tail selected according to the task and evaluation metric.

Limitations. For deterministic binary terminal rewards with $\gamma = 1$, a calibrated mean determines the Bernoulli return law. We interpret the gains primarily as an optimization benefit of replacing scalar regression with a multi-output critic loss; this benefit does not require an accurately reconstructed return distribution. The contribution lies in learning a useful baseline under sparse feedback. Tail introduces an outcome-dependent risk preference, so its coefficient should be selected on held-out data using both average quality and repeated-sampling success. The current evaluation covers moderate-scale models and selected verification tasks.

Future work can compare critic objectives on identical prefixes with fresh continuations and extend the study to larger models and controlled verifier noise. The results establish multi-output value learning with dSR as a practical approach to actor–critic RLVR.

AI USE STATEMENT

AI tools assisted with manuscript organization and language polishing.

REPRODUCIBILITY STATEMENT

The target construction and conditional sampling proof are given in Appendix A, critic losses in Appendix B, and evaluation settings and detailed numerical results in Appendices C–D.8. Anonymous code repository: https://anonymous.4open.science/r/DistRLVR_Anonymous-CD2E/.

ETHICS STATEMENT

The study uses existing language models and task datasets to examine post-training algorithms. Improved optimization of an imperfect verifier can amplify errors or exploitable reward features; this is particularly relevant to outcome-dependent reweighting. Programmatic task success does not establish reliable reasoning or safe deployment. Dataset terms, evaluation contamination, and the risks of executing generated code should be considered when reproducing or extending the experiments.

REFERENCES

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.

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.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024. URL https://arxiv.org/abs/2403.07974.

Yuhua Jiang, Jiawei Huang, Yufeng Yuan, Xin Mao, YuYue, Qianchuan Zhao, and Lin Yan. Risk-Sensitive Reinforcement Learning for Alleviating Exploration Dilemmas in Large Language Models. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 47117–47141, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/4e6d14709eae0cbc49a1d19d87fb8b21-Paper-Conference.pdf.

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.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating Very Long-Term Conversational Memory of LLM Agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13851–13870. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.747. URL https://aclanthology.org/2024.acl-long.747/.

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. URL https://proceedings.mlr.press/v139/nam21a.html.

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.

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.

Yu Yue, Yufeng Yuan, Qiying Yu, Xiaochen Zuo, Ruofei Zhu, Wenyuan Xu, Jiaze Chen, Chengyi Wang, TianTian Fan, Zhengyin Du, Xiangpeng Wei, Xiangyu Yu, Gaohong Liu, Juncai Liu, Lingjun Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Ru Zhang, Xin Liu, Mingxuan Wang, Yonghui Wu, and Lin Yan. VAPO: Efficient and reliable reinforcement learning for advanced reasoning tasks. *arXiv preprint arXiv:2504.05118*, 2025. URL https://arxiv.org/abs/2504.05118.

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. *arXiv preprint arXiv:2502.20548*, 2025. URL https://arxiv.org/abs/2502.20548.

Zhijian Zhou, Long Li, Xuan Zhang, Zongkai Liu, Yanting Miao, Yuchen Liu, Deshu Chen, Ke Li, Xing Sun, Ruoxi Jiang, Xiaoyu Tan, Chao Qu, and Yuan Qi. DisPPO: Quantile-based distributional reinforcement learning for large language models. In *International Conference on Machine Learning*, 2026a. URL https://chaos-q.github.io/publication/2026-disppo.

Zhijian Zhou, Long Li, Xuan Zhang, Zongkai Liu, Yulei Qin, Ke Li, Xing Sun, Xiaoyu Tan, Chao Qu, and Yuan Qi. Start Classifying: Categorical Critics for LLM Reinforcement Learning. *arXiv preprint arXiv:2608.02181*, 2026b. URL https://arxiv.org/abs/2608.02181.

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. URL https://arxiv.org/abs/2512.03847.

Dingwei Zhu, Zhiheng Xi, Shihan Dou, Jiahan Li, Chenhao Huang, Junjie Ye, Sixian Li, Mingxu Chai, Yuhui Wang, Yajie Yang, Ming Zhang, Jiazheng Zhang, Shichun Liu, Caishuang Huang, Yunke Zhang, Yuran Wang, Tao Gui, Xipeng Qiu, Qi Zhang, and Xuanjing Huang. DFPO: Scaling value modeling via distributional flow towards robust and generalizable LLM post-training, 2026. URL https://arxiv.org/abs/2602.05890.

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. URL https://arxiv.org/abs/2509.15207.

A Sample Replacement: sampling identity and endpoint error

Conditional target construction. Fix a realized trajectory τ and a target distribution mapping $Z_{\bar{\phi}}$. All probabilities in the sampling identity are conditional on these objects. The terminal reward is attached only once, through $g_T = r_T$ or the equivalent endpoint distribution $Z_{\bar{\phi}}(s_T) = \delta_{r_T}$. In particular, when intermediate rewards vanish, the full return from s_t is $\gamma^{T-t} r_T$ under this convention.

Define the stopping depth without inspecting the final, immaterial coin:

$$N_t^{(m)} = \min\left(\{n \in \{1, \dots, H_t - 1\} : b_{t+n-1}^{(m)} = 0\} \cup \{H_t\}\right). \tag{10}$$

For $n < H_t$, this event consists of $n-1$ continue outcomes followed by a replacement, hence has probability $\lambda_{\mathrm{SR}}^{n-1}(1-\lambda_{\mathrm{SR}})$. The event $N_t^{(m)} = H_t$ requires only $b_t^{(m)} = \cdots = b_{T-2}^{(m)} = 1$, hence has probability $\lambda_{\mathrm{SR}}^{H_t-1}$. The variable $b_{T-1}^{(m)}$ is unrestricted: both branches at that step equal r_T. For $H_t = 1$, the required set of continue outcomes is empty and the terminal depth has probability one.

Proof of Lemma 4.1. If $N_t^{(m)} = n < H_t$, unrolling the recursion gives

$$g_t^{(m)} = \sum_{i=0}^{n-1} \gamma^i r_{t+i} + \gamma^n \widetilde{z}_{t+n}^{(m)}.$$

The replacement draw at s_{t+n} is independent of the stopping event, so this expression has law $\mathcal{L}(\widetilde{Z}_t^{(n)})$. If $N_t^{(m)} = H_t$, unrolling gives

$$g_t^{(m)} = \sum_{i=0}^{H_t-1} \gamma^i r_{t+i} + \gamma^{H_t} r_T,$$

regardless of the last Bernoulli outcome. This equals the full-depth target in Eq. (5). Marginalizing over the stopping depth gives Eq. (7). Independent randomness across m gives conditional independence for fixed t. □

Discrepancy from exact endpoint distributions. We give a conditional bound in the 1-Wasserstein distance W_1, assuming finite first moments. Write $\lambda = \lambda_{\mathrm{SR}}$ in this paragraph. Define an exact-endpoint comparator on the same fixed trajectory by

$$Z_{t,\tau}^{(n),*} \overset{D}{=} \sum_{i=0}^{n-1} \gamma^i r_{t+i} + \gamma^n Z^\pi(s_{t+n}), \qquad \mathcal{L}(Z_{t,\tau}^{(\lambda),*}) = \sum_{n=1}^{H_t} p_t(n) \mathcal{L}(Z_{t,\tau}^{(n),*}).$$

The subscript τ emphasizes that sampled intermediate actions and rewards are fixed. This comparator is not, by itself, the unconditional on-policy return law at s_t. Let $\Delta(s) = W_1(Z_{\bar{\phi}}(s), Z^\pi(s))$. Then

$$W_1(\widetilde{Z}_t^{(\lambda)}, Z_{t,\tau}^{(\lambda),*}) \le \sum_{n=1}^{H_t} p_t(n) \gamma^n \Delta(s_{t+n}). \tag{11}$$

Proof. For each depth, translation invariance and positive homogeneity of W_1 give $W_1(\widetilde{Z}_t^{(n)}, Z_{t,\tau}^{(n),*}) = \gamma^n \Delta(s_{t+n})$. Couple both mixtures using the same depth N, then use an optimal coupling of the corresponding endpoint distributions conditional on N. The expected coupling cost is the right-hand side of Eq. (11), which upper-bounds W_1. □

With deterministic verification, $Z^\pi(s_T) = \delta_{r_T}$ and the terminal term is zero. With a stochastic verifier, a single observed terminal reward is not the full verifier distribution conditional on an otherwise fixed completion; the conditional terminal term need not vanish. The bound controls discrepancy inherited from the endpoint predictors. Its dependence on λ_{SR} reflects both the depth probabilities and the critic errors encountered along the trajectory.

Sampling depth and computational scope. The tail-sum identity gives

$$\mathbb{E}[N] = \sum_{j=1}^{H_t} \Pr(N \geq j) = \sum_{j=1}^{H_t} \lambda_{\mathrm{SR}}^{j-1} = \frac{1 - \lambda_{\mathrm{SR}}^{H_t}}{1 - \lambda_{\mathrm{SR}}}.$$

At $\lambda_{\mathrm{SR}} = 0.9$ and $H_t = 1024$, the probability that one draw reaches the terminal endpoint is approximately 1.55×10^{-47}. The $O(MT)$ complexity counts target-construction arithmetic only. Materializing all targets costs $O(MT)$ storage per trajectory; streaming the recursion needs $O(M)$ working storage, excluding critic outputs, model activations, and the training system's batch storage.

B Distributional critic parameterizations

All heads predict state-value return distributions. Given a state s and detached target draws $g^{(1)}, \ldots, g^{(M)}$, the actor baseline is the mean of the predicted distribution. The following definitions specialize established distributional losses to this setting.

C51. For L equally spaced support points $v_\ell \in [v_{\min}, v_{\max}]$, the head predicts probabilities $p_\phi^\ell(s)$ and represents $Z_\phi(s) = \sum_\ell p_\phi^\ell(s)\delta_{v_\ell}$. Each target draw is clipped to the support and linearly projected onto its neighboring atoms; a draw exactly on an atom assigns all its mass to that atom. If m^ℓ is the mean projected mass over draws, the loss and baseline are

$$\mathcal{L}_{\mathrm{C51}}(s) = -\sum_{\ell=1}^{L} m^\ell \log p_\phi^\ell(s), \qquad V_\phi(s) = \sum_{\ell=1}^{L} p_\phi^\ell(s) v_\ell.$$

SR replacement draws sample an atom with its predicted probability. The C51 runs use support $[0, 1]$ with $L = 51$; this support would need reconsideration if reward penalties outside that interval were included in the return.

QR and IQN. QR predicts $z_\phi(s; \tau_i)$ at fixed fractions $\tau_i = (i - 1/2)/N_q$. IQN samples fractions uniformly in $(0, 1)$ and conditions its predictor on a quantile embedding of the transformer hidden state. For a smoothing parameter $\kappa > 0$, both use a pairwise quantile Huber loss:

$$\mathcal{L}_{\mathrm{quantile}}(s) = \frac{1}{N_q M} \sum_{i=1}^{N_q} \sum_{m=1}^{M} \rho_{\tau_i}^\kappa(g^{(m)} - z_\phi(s; \tau_i)), \tag{12}$$

$$\rho_\tau^\kappa(u) = |\tau - \mathbf{1}\{u < 0\}|\, \mathcal{H}_\kappa(u)/\kappa, \qquad \mathcal{H}_\kappa(u) = \begin{cases} u^2/2, & |u| \leq \kappa, \\ \kappa(|u| - \kappa/2), & |u| > \kappa. \end{cases} \tag{13}$$

For QR, the mean of its equally weighted atoms is the induced scalar value. For IQN, averaging independently sampled quantile predictions estimates the mean of its induced distribution. QR replacement draws select uniformly among its predicted atoms; IQN draws sample a fresh uniform fraction and evaluate the quantile predictor. N_q counts predicted quantiles and M counts SR target draws; neither is the number K of independent completions. The quantile Huber loss is a smoothed objective and should not be assumed to share the exact population minimizers of unsmoothed quantile regression.

Multi-step target comparison. The distributional TD(λ) ablation uses the same IQN backbone and PPO update as the dSR configuration. Appendix D.2 compares its policy outcomes with one-step targets and dSR. Together with the scalar controls, these results examine the optimization benefit of the critic objective and target construction.

C Evaluation protocol and implementation details

Binary evaluation metrics. For Q questions with binary outcomes I_{qi}, average success over n draws is

$$\text{Avg}\,@n = \frac{100}{Q}\sum_{q=1}^{Q}\frac{1}{n}\sum_{i=1}^{n} I_{qi}.$$

It estimates single-draw accuracy; it is not success on any of n draws. With $N = 128$ sampled completions and c_q correct completions, the standard estimator is

$$\widehat{\text{Pass}\,@k} = \frac{100}{Q}\sum_{q=1}^{Q}\left(1 - \frac{\binom{N-c_q}{k}}{\binom{N}{k}}\right), \qquad 1 \le k \le N,$$

using a zero numerator when $N - c_q < k$. Pass@128 is the fraction of questions with any observed success and measures oracle success coverage under verification. For graded rewards, mean score measures partial task satisfaction, while strict Pass@k uses binary complete success.

Aggregation. The eight accuracy settings are AIME 2024, AIME 2025, AMC 2023, MATH500, Minerva, and OlympiadBench after DeepScaleR training, plus M2M and Countdown under their separate training protocols. The original table uses Avg@32 for the six DeepScaleR benchmarks and Avg@128 for M2M and Countdown. We therefore label their macro-average "aggregate accuracy", rather than Avg@32. The Pass@128 macro-average excludes Countdown and uses seven settings. All methods in the matched-K comparison use these settings and the same macro-average weights.

Table 8: Primary mathematical-run settings.

Parameter	Setting
Policy	DeepSeek-R1-Distill-Qwen-1.5B
Serving engine	vLLM 0.8.4
Sampling temperature / top-p / top-k	1.0 / 1.0 / disabled
Maximum prompt / response length	766 / 1,024 tokens (default)
Actor learning rate	10^{-6}
PPO clipping / entropy coefficient	0.2 / 0.001
Critic warm-up steps	0
Global batch / minibatch size	128 / 32
GRPO KL coefficient	0.001
C51 support / atoms	$[0, 1]$ / 51
Fixed QR fractions	32
Completions K / SR draws M	8 / 8
SR continuation probability	0.9
Tail mass α / sigmoid temperature β	0.25 / 0.5

Quantile and target samples. N_q denotes the number of predicted quantile fractions, M the number of SR target draws, and K the number of independently generated completions. They control the critic output, target sampling, and rollout generation, respectively. The main IQN formulation conditions token-prefix hidden states on sampled quantile fractions; C51 and fixed QR provide alternative output heads.

Datasets and splits. The DeepScaleR training corpus contains approximately 40K mathematical problem–answer pairs. M2M trains on a subset of MATH500 and evaluates on Minerva; Countdown uses disjoint training and evaluation splits. The 1K default and the 8K generation setting use the decoding parameters in Table 8. A short response cap can truncate reasoning before the final answer, so comparisons across generation budgets must retain the cap and decoding settings alongside the scores.

Statistical summaries. Appendix D.3 gives three-seed means and standard deviations for the original mathematical runs. The additional scalar controls and graded-task tables give point estimates. Whiskers in Figure 3 denote across-run runtime standard deviations.

D Additional experimental results

D.1 Original per-task mathematical results

The following tables give task-level results with PPO at $K = 1$ and DistRLVR at $K = 8$. Table 2 gives the primary rollout-matched comparison.

Table 9: Original average accuracy (percent; three-seed means). PPO used $K = 1$ and the DistRLVR variants used $K = 8$. The first six settings use Avg@32; M2M and Countdown use Avg@128.

Setting	PPO ($K = 1$)	TD(0)	dSR	Tail
AIME 2024	5.104	7.535	8.125	9.549
AIME 2025	3.264	5.452	6.042	7.361
AMC 2023	31.903	46.233	43.737	44.829
OlympiadBench	23.012	28.611	29.241	31.451
MATH500	62.590	69.969	71.025	72.623
Minerva	18.053	21.944	22.522	22.794
M2M (4K)	17.17	20.43	20.80	21.29
Countdown	64.97	68.94	69.85	69.80

Table 10: Task-level Pass@128 (%; three-seed means). PPO uses $K = 1$ and DistRLVR uses $K = 8$.

Setting	PPO ($K = 1$)	TD(0)	dSR	Tail
AIME 2024	43.333	46.667	53.333	54.444
AIME 2025	26.667	37.778	35.556	36.667
AMC 2023	86.345	88.353	90.361	89.157
OlympiadBench	65.975	65.531	65.126	62.914
MATH500	94.667	95.067	94.533	94.267
Minerva	64.951	61.520	62.377	57.721
M2M (4K)	62.18	66.88	69.23	67.09

D.2 Distributional target controls

Table 11 compares the IQN target constructions under the common $K = 8$ protocol. Distributional TD(0) already improves aggregate accuracy/Pass@128 over PPO by 1.599/1.151 percentage points. Thus the one-step distributional formulation contributes an initial improvement before dSR or Tail is introduced. dSR adds 0.279/1.246 points over distributional TD(0), and 0.578/0.917 points over the distributional TD(λ) control. These policy outcomes motivate dSR as the default target recipe; the architecture-specific ablations follow in Appendix D.4.

Table 11: IQN target comparison at $K = 8$ (%).

Method	Aggregate accuracy	Pass@128
PPO	32.040	64.820
DistRLVR (TD(0))	33.639	65.971
DistRLVR (TD(λ))	33.340	66.300
DistRLVR (dSR)	33.918	67.217

D.3 Seed variation in the original mathematical runs

The following tables report three-seed variation under the original protocol. They complement the task-level means with seed standard deviations.

Table 12: Original Avg@32 means and standard deviations over three seeds. PPO used $K = 1$; DistRLVR used $K = 8$.

Benchmark	PPO ($K = 1$)	TD(0)	dSR	Tail
AIME 2024	5.104 ± 0.9	7.535 ± 0.4	8.125 ± 0.6	9.549 ± 1.2
AIME 2025	3.264 ± 0.7	5.452 ± 0.7	6.042 ± 0.3	7.361 ± 0.8
AMC 2023	31.903 ± 2.3	46.233 ± 1.6	43.737 ± 0.7	44.829 ± 0.3
MATH500	62.59 ± 0.8	69.969 ± 0.9	71.025 ± 0.4	72.623 ± 0.5
Minerva	18.053 ± 0.8	21.944 ± 0.8	22.522 ± 0.4	22.794 ± 0.4
OlympiadBench	23.012 ± 0.8	28.611 ± 1.2	29.241 ± 0.2	31.451 ± 0.3

Table 13: Original Pass@128 means and standard deviations over three seeds, under the original rollout protocol.

Benchmark	PPO ($K = 1$)	TD(0)	dSR	Tail
AIME 2024	43.333 ± 3.3	46.667 ± 5.8	53.333 ± 3.3	54.444 ± 1.9
AIME 2025	26.667 ± 5.8	37.778 ± 7.7	35.556 ± 7.7	36.667 ± 5.8
AMC 2023	86.345 ± 0.7	88.353 ± 2.8	90.361 ± 2.1	89.157 ± 1.2
MATH500	94.667 ± 0.6	95.067 ± 0.2	94.533 ± 0.3	94.267 ± 0.5
Minerva	64.951 ± 1.9	61.52 ± 4.9	62.377 ± 0.2	57.721 ± 1.9
OlympiadBench	65.975 ± 1.6	65.531 ± 2.2	65.126 ± 1.1	62.914 ± 0.6

D.4 Head and baseline-function ablations

C51 and QR have different responses to dSR: their six-task accuracy means change by -0.139 and $+0.342$ points, respectively. IQN is the primary configuration. The following tables compare the heads with one-step and dSR supervision.

Table 14: Head ablation: Avg@32 (%; three-seed means). Suffix 0 denotes TD(0), and SR denotes dSR; dashes indicate omitted cells.

Benchmark	C51-0	QR-0	IQN-0	C51-SR	QR-SR	IQN-SR
AIME 2024	7.118	7.639	7.535	7.778	8.090	8.125
AIME 2025	5.694	5.590	5.451	5.139	5.972	6.042
AMC 2023	42.934	42.432	–	42.495	44.089	43.737
MATH500	71.031	70.746	69.969	70.242	70.767	71.025
Minerva	22.047	22.093	21.944	22.610	21.569	22.522
OlympiadBench	29.295	29.466	28.611	29.023	29.531	29.241
Six-task mean	29.687	29.661	–	29.548	30.003	30.115

The "Averse" and "Seeking" run labels in the next table denote lower- and upper-tail baseline variants. An action-independent baseline does not by itself optimize a risk-sensitive objective; see Appendix E.2. Tail has the highest average score in this table, but a lower Pass@128 than the neutral dSR configuration.

Table 15: Head ablation: Pass@128 (%).

Benchmark	C51-0	QR-0	IQN-0	C51-SR	QR-SR	IQN-SR
AIME 2024	46.667	47.778	46.667	45.556	50.000	53.333
AIME 2025	37.778	35.556	37.778	34.444	41.111	35.556
AMC 2023	88.755	87.550	88.353	88.755	89.157	90.361
MATH500	94.000	94.533	95.067	94.533	94.867	94.533
Minerva	62.500	60.172	61.520	63.725	61.275	62.377
OlympiadBench	64.938	64.889	65.531	65.481	64.938	–

Table 16: Tail and baseline-function ablations: Avg@32 (percent; three-seed means). Lower/upper-tail correspond to the historical Averse/Seeking run labels.

Benchmark	dSR	Lower-tail	Upper-tail	Tail
AIME 2024	8.125	8.021	9.688	9.549
AIME 2025	6.042	7.569	6.910	7.361
AMC 2023	43.737	45.582	44.691	44.829
MATH500	71.025	72.196	71.890	72.623
Minerva	22.522	23.231	23.081	22.794
OlympiadBench	29.241	30.918	30.761	31.451
Six-task mean	30.115	31.253	31.170	31.434

Table 17: Tail and baseline-function ablations: Pass@128 (%).

Benchmark	dSR	Lower-tail	Upper-tail	Tail
AIME 2024	53.333	47.778	52.222	54.444
AIME 2025	35.556	34.444	41.111	36.667
AMC 2023	90.361	87.952	88.353	89.157
MATH500	94.533	93.533	94.067	94.267
Minerva	62.377	55.025	57.721	57.721
OlympiadBench	–	61.877	63.457	62.914
Six-task mean	–	63.435	66.155	65.861

D.5 Generation limits and hyperparameter sensitivity

The M2M comparison fixes a 1K generation limit and varies the completion-count configuration. The local sensitivity sweep uses the 1K, 16-completion M2M setting. Increasing M from 2 to 16 raises Pass@128 from 59.829 to 62.108 while average accuracy changes from 16.079 to 16.164.

Table 18: M2M at a 1K generation limit, with the original completion count and a 16-completion variant. Metrics are average score and Pass@128.

Method	1K average	1K P@128	1K/$K = 16$ avg.	1K/$K = 16$ P@128
PPO	13.05	56.84	13.05	56.84
TD(0)	14.86	62.25	15.79	61.68
dSR	14.93	59.69	16.07	60.97
Tail	14.87	60.11	15.84	60.68

Table 19: Hyperparameter sensitivity of DistRLVR. We report avg@32 and pass@128.

Parameter	**Value**	**avg@32**	**pass@128**
α	0.050	15.131	60.256
	0.200	15.371	59.402
	0.250	15.840	60.680
	0.500	15.736	62.678
λ_{SR}	0.1	15.705	59.829
	0.8	15.825	60.684
	0.9	16.070	60.970
	0.95	16.431	61.538
M	2	16.079	59.829
	4	16.070	60.970
	8	16.070	60.970
	16	16.164	62.108

D.6 Allocation of completions and SR draws

Table 20: Fixed-generation-token allocation. Prompt allocation holds generated tokens constant; metrics are Avg@32 and Pass@128 (%).

K	M	Relative generated tokens	Avg@32	Pass@128
1	1	1.00	31.2	64.0
2	1	1.00	32.0	64.8
4	1	1.00	32.6	65.4
8	1	1.00	33.0	66.0
1	2	1.00	31.8	64.5
1	4	1.00	32.3	65.0
1	8	1.00	32.6	65.4

This separate experiment holds generated tokens fixed by adjusting prompt allocation. Increasing either K or M improves both reported metrics; $(K, M) = (8, 1)$ reaches 33.0/66.0, compared with 32.6/65.4 for $(1, 8)$. K supplies new trajectories, whereas M supplies conditional bootstrapped supervision on visited prefixes. GPU time and prompt coverage may differ at the same generated-token budget.

D.7 Measured runtime

The precise end-to-end durations are 131m25s for the original PPO-$K = 1$, 212m50s for PPO-$K = 8$, 222m22s for GRPO, 203m43s for TD(0), 209m33s for dSR, and 200m02s for Tail. Multiplying elapsed hours by two gives 4.38, 7.09, 7.41, 6.79, 6.99, and 6.67 GPU-hours after rounding.

Table 21: Runtime breakdown in minutes for the matched-$K = 8$ runs. Component times are rounded, so totals may differ by 0.1 minute from their sum. Two GPUs are used.

Method	Rollout	Actor	Critic	SR	Other	Total
PPO	117.1	68.1	21.3	0	6.4	212.8
GRPO	140.1	62.3	0	0	20.0	222.4
DistRLVR (TD(0))	110.0	61.1	26.5	0	6.1	203.7
DistRLVR (dSR)	109.0	58.7	25.1	10.5	6.3	209.6

Table 22: Runtime and peak-memory scaling ranges across configurations. Ranges denote observed configuration dependence, not confidence intervals.

Configuration	Relative time	Relative peak memory
$K = 1$ (relative to $K = 8$)	0.62	0.85–0.95
$K = 2$	0.67	0.88–0.96
$K = 4$	0.78	0.92–0.98
$K = 8$	1.00	1.00
1,024 tokens (length reference)	1.00	1.00
2,048 tokens	1.6–2.0	1.15–1.35
4,096 tokens	2.8–4.0	1.4–1.9
8,192 tokens	5.0–8.0	2.0–3.0

D.8 Additional model and generation settings

At the 8K generation limit, dSR improves average scores over GRPO on five of six benchmarks and has a higher six-task Pass@128 mean. GRPO has the higher Minerva score. The task-level tables show how the aggregate improvement is distributed across benchmarks.

Table 23: 1.5B model with an 8K generation limit: average scores (percent).

Benchmark	GRPO	TD(0)	dSR	Tail
AIME 2024	25.3	27.8	27.5	28.6
AIME 2025	22.6	21.8	23.5	22.3
AMC 2023	61.2	67.1	66.4	66.8
MATH500	82.6	83.6	83.5	83.4
Minerva	28.7	27.1	27.8	27.9
OlympiadBench	44.2	45.3	45.4	45.4

For Qwen3-8B, dSR reaches 78.24% six-task mean Pass@128, compared with GRPO's 76.28%, while Tail has the highest mean accuracy (55.40%). dSR's Pass@128 is 72.67% on AIME 2024 and 52.67% on AIME 2025, compared with 68.67% and 50.00% for GRPO. On MATH500, its Pass@128 is 98.60% compared with 97.66%.

On Phi-3-mini-4k-instruct, dSR has the strongest aggregate accuracy and Pass@128. Its MATH500 average improves from PPO's 46.9 to 51.0, and its OlympiadBench average improves from 13.8 to 16.4.

A separate non-RLVR-initialization comparison reports 11.0/35.0 for PPO-$K = 8$ and 13.5/38.0 for dSR (Avg@32/Pass@128). This evaluates post-training from a non-RLVR initialization.

Table 24: 1.5B model with an 8K generation limit: Pass@128 (percent).

Benchmark	GRPO	TD(0)	dSR	Tail
AIME 2024	73.3	75.6	75.6	75.6
AIME 2025	61.1	65.6	66.7	60.0
AMC 2023	94.8	94.8	94.8	94.0
MATH500	96.8	96.7	96.9	96.9
Minerva	72.3	66.7	68.5	66.7
OlympiadBench	71.9	70.6	71.0	70.6

Table 25: Qwen3-8B: average scores (percent).

Benchmark	PPO	GRPO	TD(0)	dSR	Tail
AIME 2024	18.96	29.65	28.77	29.65	31.20
AIME 2025	10.25	22.45	26.84	27.52	26.60
AMC 2023	68.22	75.68	76.76	78.82	79.06
MATH500	92.65	93.42	93.62	93.86	94.10
Minerva	38.96	40.46	40.93	40.78	42.22
OlympiadBench	54.12	57.78	58.20	58.63	59.22
Six-task mean	47.19	53.24	54.19	54.88	55.40

Table 26: Qwen3-8B: Pass@128 (percent). Six-task means and differences are computed before rounding.

Benchmark	PPO	GRPO	TD(0)	dSR	Tail
AIME 2024	70.00	68.67	70.00	72.67	70.67
AIME 2025	48.67	50.00	50.67	52.67	49.33
AMC 2023	94.22	93.24	93.20	94.22	94.80
MATH500	97.46	97.66	97.42	98.60	98.00
Minerva	68.81	68.46	68.69	69.72	66.26
OlympiadBench	80.21	79.66	79.20	81.58	80.95
Six-task mean	76.56	76.28	76.53	78.24	76.67

Table 27: Phi-3-mini-4k-instruct: average scores and standard deviations (percent).

Benchmark	PPO	TD(0)	dSR	Tail
AIME 2024	1.53 ± 0.6	1.98 ± 0.3	3.0 ± 0.4	2.26 ± 0.7
AIME 2025	0.83 ± 0.3	0.21 ± 0.3	1.0 ± 0.3	0.35 ± 0.4
AMC 2023	18.0 ± 1.1	19.9 ± 0.5	20.5 ± 0.9	20.4 ± 1.5
MATH500	46.9 ± 1.2	47.8 ± 0.6	51.0 ± 0.1	49.3 ± 1.8
Minerva	22.2 ± 2.3	24.8 ± 0.7	23.1 ± 1.1	23.6 ± 1.1
OlympiadBench	13.8 ± 1	14.9 ± 0.4	16.4 ± 0.7	15.0 ± 1.3
Six-task mean	17.2 ± 1.1	18.3 ± 0.5	19.2 ± 0.6	18.5 ± 1.2

Table 28: Phi-3-mini-4k-instruct: Pass@128 (percent).

Benchmark	PPO	TD(0)	dSR	Tail
AIME 2024	23.3	20.0	30.0	23.3
AIME 2025	34.4	13.3	32.2	18.9
AMC 2023	73.5	71.9	79.1	75.1
MATH500	86.6	85.9	87.5	86.7
Minerva	59.7	59.4	60.3	59.4
OlympiadBench	52.7	50.4	56.2	53.7
Six-task mean	55.0	50.2	57.6	52.9

D.9 GRADED-TASK EVALUATION DETAILS

The code experiment uses Skywork training data and evaluates on LiveCodeBench with fraction-of-tests-passed training rewards and strict all-tests-passed success. Table 6 compares mean code reward and strict-success metrics. The following table gives the strict-success results for PPO, GRPO, VAPO, and dSR.

Table 29: Executable-code Pass@k (%).

Method	Pass@2	Pass@4	Pass@8
PPO	9.195	11.999	14.768
GRPO	9.354	12.556	16.409
VAPO	11.800	12.400	15.200
DistRLVR (dSR)	20.223	26.908	33.746

For programmatic instruction following, Table 30 gives the IF scores. dSR improves over PPO by 1.9400 pp and over distributional TD(0) by 0.3696 pp. Tail reaches 64.1798%, exceeding dSR by 2.4172 pp. Training uses a graded reward for partial constraint satisfaction.

Table 30: Instruction-following scores (%).

Metric	PPO	TD(0)	dSR	Tail
IF score	59.8226	61.3930	61.7626	**64.1798**

D.10 CONVERSATIONAL QA

LoCoMo (Maharana et al., 2024) evaluates conversational-memory QA. dSR has the highest average among the listed configurations: 67.19 versus 63.39 for GRPO and 62.82 for VAPO. Tail lowers the average to 62.60, showing that stronger outcome reweighting can reduce average QA quality.

Table 31: LoCoMo average QA score.

Method	Average score
PPO	58.83
GRPO	63.39
VAPO	62.82
DistRLVR (dSR)	67.19
DistRLVR (Tail)	62.60

D.11 HISTORICAL ROLLOUT AND TRAINING TRACES

Table 32 transcribes the two-decimal values displayed in the original rollout-count plot, where n denotes completions per prompt. The effect of SR depends on n and task: at $n = 2$ it is lower than IQN(0) on all six tasks, while at $n = 8$ it is higher on five of six. This is a descriptive rollout-count sweep.

Figure 5 shows training trajectories for PPO, C51, QR, and IQN across average accuracy and repeated-sampling success metrics.

Table 32: Rollout-count sensitivity: average score (%). Here n is the number of completions per prompt.

	$n = 1$		$n = 2$		$n = 8$	
Benchmark	IQN(0)	IQN(SR)	IQN(0)	IQN(SR)	IQN(0)	IQN(SR)
AIME 2024	4.48	4.31	6.29	5.31	7.54	8.12
AIME 2025	2.85	2.74	4.62	3.44	5.45	6.04
AMC 2023	32.93	30.37	38.50	36.32	46.23	43.74
MATH500	62.96	62.03	66.46	65.38	69.97	71.03
Minerva	20.64	17.64	20.24	19.80	21.94	22.52
OlympiadBench	21.70	22.49	25.82	25.30	28.61	29.24

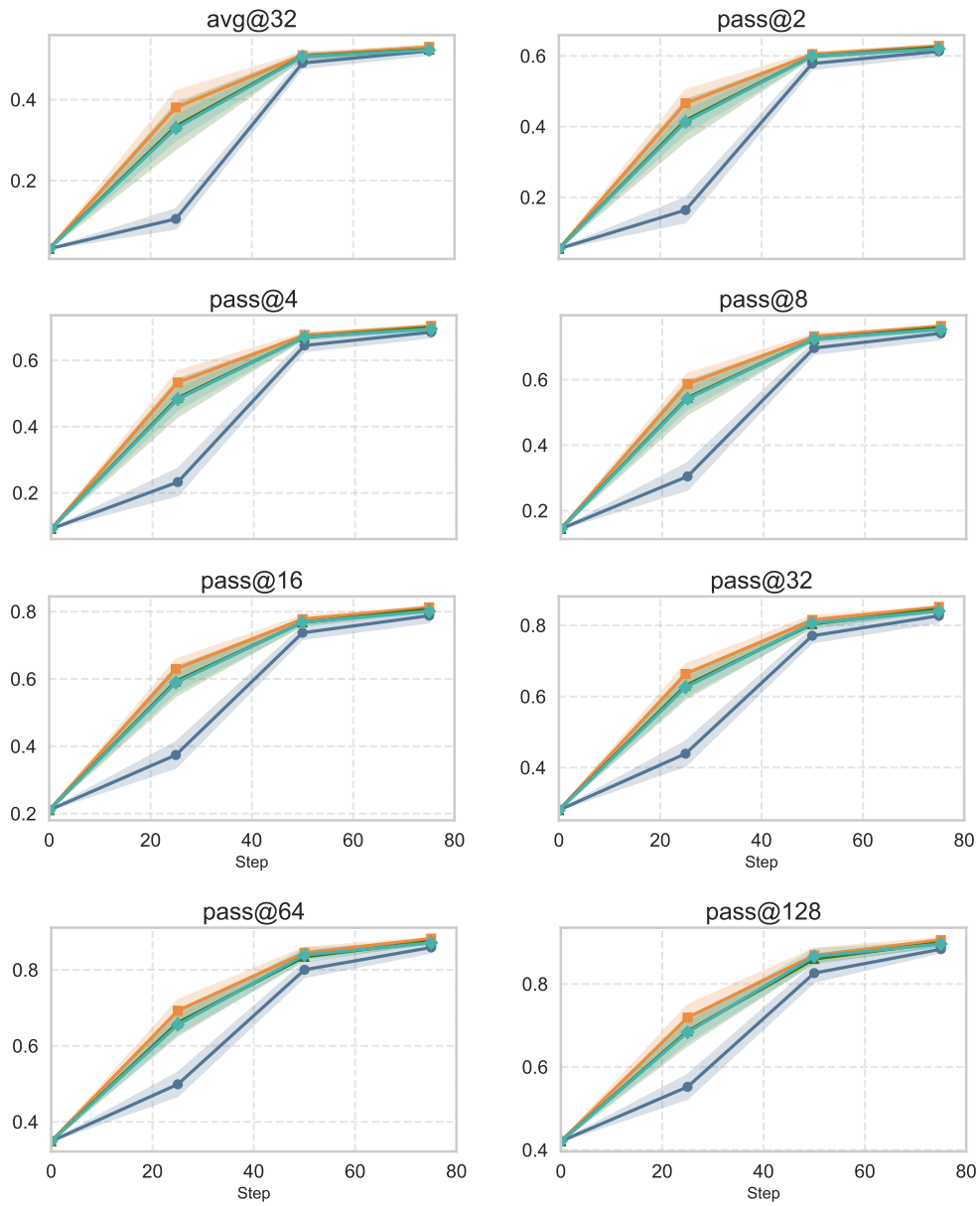


Figure 5: Training trajectories across evaluation metrics. Blue: PPO; orange: C51; green: QR; teal: IQN. Vertical scales differ across panels.

Figure 6 shows generated response length during training under a 1,024-token cap. The PPO curves use different rollout counts; runtime measurements are given in Figure 4 and Appendix D.7.

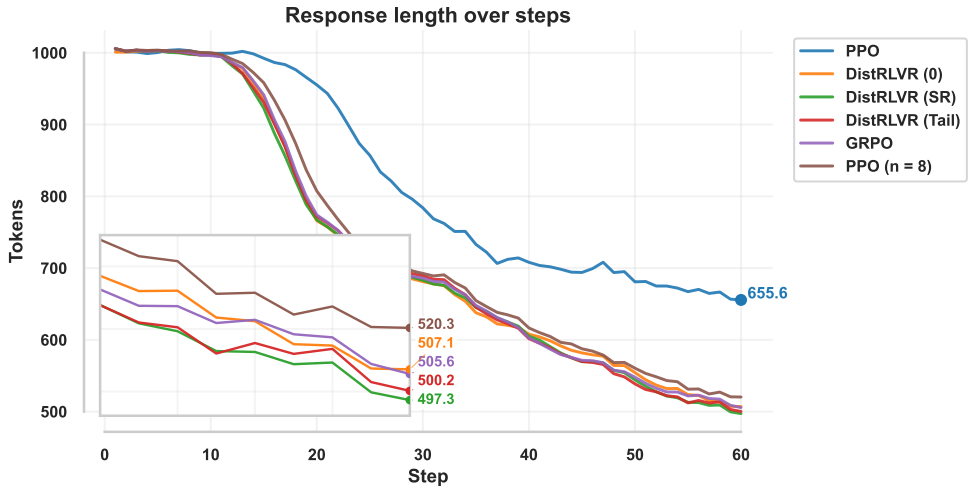


Figure 6: Generated response length during training with a 1,024-token cap. PPO ($n = 8$) uses eight completions per prompt.

D.12 Prefix-conditioned outcome diagnostics

The fixed-prefix analysis samples 120 MATH500 problems, two prefixes per problem, and 64 continuations per prefix (Table 33). Of 240 prefixes, 74 have at least 20% empirical mass on each binary outcome. Stochastic continuations therefore produce both success and failure at many visited prefixes, despite deterministic verification.

Table 33: Original fixed-prefix sampling counts on MATH500. The threshold refers to empirical binary outcomes among 64 continuations of the same prefix.

Quantity	Count
Problems	120
Prefix stages per problem	2
Continuations per prefix	64
Total sampled prefixes	240
Prefixes with at least 20% of each outcome	74

Comparing critic predictions with their own dSR targets measures agreement with bootstrapped supervision. Independent real continuations instead evaluate the prediction against realized outcomes. For per-prefix empirical success rates $\hat{p}_s$, RMSE and Brier loss on the same continuations and weights satisfy

$$\text{Brier} - \text{RMSE}^2 = \mathbb{E}_s[\hat{p}_s(1 - \hat{p}_s)].$$

This identity separates prediction error in the conditional mean from the empirical variability of binary continuations.

E Interpretation of auxiliary updates

E.1 A scalar sample is not a deterministic target law

A single scalar sample per update can provide valid distributional supervision. For the pinball loss $\rho_\tau(u) = u(\tau - \mathbf{1}\{u < 0\})$, consider

$$\mathcal{L}_\tau(z; s) = \mathbb{E}[\rho_\tau(G - z) \mid s].$$

Its minimizers are conditional τ-quantiles of $G \mid s$. A fresh draw G gives an unbiased stochastic estimate of this loss. Pairing one draw with all quantile fractions therefore does not force their population minimizers to coincide. For example, when $G \mid s \sim \mathrm{Bernoulli}(p)$ with $0 < p < 1$, the quantile is zero for $\tau < 1 - p$ and one for $\tau > 1 - p$. This is a nondegenerate law learned from scalar observations.

Collapse instead follows if the *conditional target law* is deterministic, $G = c(s)$ almost surely. All pinball-loss quantiles then equal $c(s)$. A finite training set with one stored return per unique prefix can also be memorized, but that is a finite-data overfitting issue, not a general inconsistency of Monte Carlo distributional learning. The same distinction applies to categorical cross-entropy: random one-hot labels recover their class probabilities in expectation, whereas one deterministic repeated label produces a point target. Quantile Huber losses introduce smoothing and need not have exactly the same population minimizers as the unsmoothed pinball loss. Fresh return draws, fixed stored labels, conditional means, and scalar bootstrapped estimates therefore induce different learning objectives.

E.2 Tail-derived baselines and outcome reweighting

For an integrable return Z and $0 < \alpha \leq 1$, define

$$\mathrm{CVaR}^-_\alpha(Z) = \frac{1}{\alpha}\int_0^\alpha F_Z^{-1}(u)\,du, \qquad \mathrm{CVaR}^+_\alpha(Z) = \frac{1}{\alpha}\int_{1-\alpha}^1 F_Z^{-1}(u)\,du. \tag{14}$$

These definitions allocate fractional probability at a quantile boundary and therefore apply to discrete rewards. Conditioning naively on $Z \leq q_\alpha$ or $Z \geq q_{1-\alpha}$ can include more than an α fraction of probability when the boundary has an atom. For $Z \sim \mathrm{Bernoulli}(p)$,

$$\mathrm{CVaR}^-_\alpha(Z) = \max\{0, (p + \alpha - 1)/\alpha\}, \qquad \mathrm{CVaR}^+_\alpha(Z) = \min\{1, p/\alpha\}.$$

Thus these functionals also contain no information beyond p in the binary case.

For any action-independent, stopped-gradient baseline $b(s)$,

$$\mathbb{E}_{a\sim\pi_\theta(\cdot|s)}[\nabla_\theta \log \pi_\theta(a \mid s) b(s)] = b(s)\nabla_\theta \sum_a \pi_\theta(a \mid s) = 0. \tag{15}$$

Consequently, replacing the mean by a lower- or upper-tail baseline does not change the exact expected-return policy gradient when using an unbiased Monte Carlo return. PPO clipping, finite batches, normalization, and bootstrapped GAE affect the practical behavior of a baseline. We describe these variants as lower-tail and upper-tail baselines, using "Averse" and "Seeking" as the corresponding run labels.

Outcome-dependent reweighting is different: $w(s, G)(G - b(s))$ generally changes the expected gradient because w depends on the sampled continuation. The processed gate in the main text defines a weighted PPO surrogate, whose expected update generally differs from a CVaR policy gradient. For discrete rewards, $q_{1-\alpha}$ need not delimit an event with probability exactly α, and the factor $1/\alpha$ alone does not normalize the gate's expectation. Response-token masks, normalization, and clipping determine the applied weights in this surrogate.

F AN ILLUSTRATIVE VERIFICATION CASE

The worked MATH500 example asks for the number of integer ordered pairs (a, b) satisfying $|a + bi| \leq 5$. Direct enumeration gives

$$\sum_{a=-5}^{5} \left(2 \left\lfloor \sqrt{25 - a^2} \right\rfloor + 1\right) = 11 + 18 + 18 + 18 + 14 + 2 = 81.$$

The following table records the final answers of individual completions.

Table 34: Final-answer verification for the worked example.

Configuration	Final answer	Matches verifier
PPO	169	No
GRPO	81	Yes
DistRLVR (dSR)	81	Yes
DistRLVR (Tail)	81	Yes
Lower-tail baseline	81	Yes
Upper-tail baseline	81	Yes

A correct final answer can coexist with an erroneous intermediate step. The upper-tail-baseline completion states the formula $1 + 4 \sum_{k=1}^{5} \lfloor \sqrt{25 - k^2} \rfloor$, which evaluates to 61, before arriving at 81 by a separate enumeration. This example illustrates the distinction between final-answer correctness and validity of the reasoning trace.