EXPLORING MORE, REASONING BETTER: STEPWISE RISK-SENSITIVE GRPO FOR DIFFUSION LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion large language models (dLLMs) generate text by denoising a sequence or successive blocks, allowing several tokens to be revealed in parallel. Reinforcement learning with verifiable rewards (RLVR) reuses terminal feedback across these decisions, even as their conditioning context changes. We propose stepwise risk-sensitive GRPO (StepRS-GRPO), which varies the risk coefficient of the group-advantage transformation across denoising states while retaining the underlying trainer. For binary rewards, we show that this transformation is exactly a prompt- and state-dependent rescaling of centered outcome advantages. A capability-based calibration suggests a coefficient scale, while endpoint and interpolation ablations guide schedule selection. Across multiple dLLM backbones and mathematical reasoning benchmarks, StepRS-GRPO improves both pass@1 accuracy and pass@k coverage over centered GRPO, while increasing answer diversity. In our ablation studies, mass-matched controls support the contributions of state allocation and schedule direction, and the gains persist after matching the root mean square (RMS) of the advantages to that of centered GRPO. Reasoning-trace diagnostics further show that the diversity gains from StepRS-GRPO extend beyond final-answer strings.

1 INTRODUCTION

Diffusion large language models (dLLMs) (Nie et al., 2025; Sahoo et al., 2024) change the structure of language-model generation. Rather than generating tokens strictly from left to right, a dLLM starts from a masked sequence and gradually denoises it through T reverse steps, where each step may unmask multiple positions using bidirectional context. Recent work has adapted supervised post-training to this masked-denoising paradigm, either by training instruction-following masked dLLMs directly (Zhao et al., 2025a; Nie et al., 2025) or by converting strong autoregressive backbones into bidirectional masked dLLMs through supervised distillation (Wang et al., 2025c).

Building on these supervised post-training recipes, a natural next step is to use reinforcement learning with verifiable rewards (RLVR) to improve dLLM reasoning. In autoregressive language models, RLVR methods such as Group Relative Policy Optimization (GRPO) (Shao et al., 2024) have become effective tools for mathematical reasoning because they directly optimize verifiable outcome rewards without requiring dense human annotations or a learned value model. Recent work has begun to transfer GRPO-style training to dLLMs by adapting policy-gradient updates to the masked-denoising process (Zhu et al., 2025a). Terminal-reward policy gradients can reuse a rollout-level advantage across multiple decisions in both autoregressive and diffusion models. In a dLLM, the conditioning context and masked positions also change across denoising states, making state-dependent weighting a useful design space to study. A higher pass@1 does not by itself establish better coverage at larger sampling budgets, and neither quantity establishes diversity of correct reasoning strategies. We therefore evaluate both accuracy and sampled-solution coverage, with constant-risk training as a direct comparator for the contribution of scheduling. Thus, the central question is not only how to make dLLMs more accurate under RLVR, but how to improve their reasoning accuracy without collapsing their search distribution:

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.

Can we design RLVR methods for dLLMs that improve reasoning accuracy while preserving answer coverage?

We address this question with *stepwise risk-sensitive GRPO* (StepRS-GRPO). Building on constant-risk RS-GRPO (Jiang et al., 2026), we vary the coefficient of the entropic advantage transformation across denoising states. Under binary rewards, this is exactly a prompt- and state-dependent rescaling of centered outcome advantages. The method uses terminal rewards alone and retains the chosen trainer's likelihood estimator and loss reduction. The analysis explains the reweighting, and the controls test the benefit of assigning different weights to different states.

Our contributions are as follows.

1. We introduce a denoising-state-dependent extension of RS-GRPO for dLLM post-training and establish its exact finite-group binary-reward factorization as prompt- and state-dependent terminal-advantage reweighting.
2. We develop a capability-guided starting point for coefficient selection and evaluate the endpoint range and interpolation shape separately, supporting linear half-decay as a practical default.
3. We demonstrate gains in accuracy, solution coverage, and output diversity across dLLM backbones. Repeated inference evaluations and targeted controls examine state allocation and advantage scale, complemented by schedule ablations and a masked-dLLM case study.

2 Related Work

Reinforcement learning for diffusion language models. dLLM post-training requires policy optimization under non-autoregressive denoising. d1 combines masked supervised fine-tuning with diffu-GRPO (Zhao et al., 2025a); TraceRL optimizes inference traces (Wang et al., 2025c); StableDRL uses unconditional clipping and self-normalization (Zhong et al., 2026); and dTRPO reduces trajectory probability-estimation cost (Zhang et al., 2026). Likelihood-estimation approaches include GDPO, ESPO, AGRPO, SPG, and BGPO (Rojas et al., 2025; Ou et al., 2025; Zhan, 2025; Wang et al., 2025b; Lin et al., 2025). The ratio-free wd1 objective uses weighted log-likelihood, and wd1++ extends it to denoising-stepwise optimization (Tang et al., 2025). DMPO instead matches a reward-tilted target through weighted denoising cross-entropy (Zhu et al., 2025b). StepRS complements these methods with state-dependent coefficients in an entropic group-advantage transformation, retaining the selected trainer's likelihood estimator and loss construction.

Entropy collapse and diversity in RL post-training. Diversity-aware RLVR includes entropy analyses (Cui et al., 2025), divergence-based methods (Li et al., 2025a), explicit diversity rewards (Li et al., 2025b), and adaptive entropy targets (Wang et al., 2025a). Pass@k policy optimization directly targets multiple-sample success (Walder & Karkhanis, 2025), while advantage-shaping analyses connect modified GRPO updates to surrogate reward objectives (Thrampoulidis et al., 2026). JustGRPO examines a different source of coverage loss: it uses left-to-right RL rollouts while retaining parallel decoding at inference (Ni et al., 2026). State-aware methods also differ in their feedback: DiSPO branches from intermediate masked states and scores completions with outcome rewards (Oba et al., 2026), whereas IGPO uses inpainting guidance (Zhao et al., 2025b). StepRS retains the selected rollout procedure and changes the state-dependent weights attached to terminal rewards.

Risk-sensitive post-training and capability calibration. Risk-sensitive control includes entropic objectives (Howard & Matheson, 1972; Whittle, 1990; Fei et al., 2020); RS-GRPO applies risk seeking to autoregressive RLVR (Jiang et al., 2026), and RiskPO studies risk-based policy objectives (Ren et al., 2025). Recent GRPO theory separates the effects of reward normalization and reference- versus previous-policy KL regularization (Mroueh, 2026). Our finite-group analysis characterizes the entropic transformation supplied to the trainer, while exponential tilting gives its coefficient an interpretable scale. We use this scale to guide endpoint selection and evaluate the schedule empirically, complementing coefficient selection in entropy-regularized RL (Haarnoja et al., 2018a;b) and exponential-tilting methods (Rubinstein, 1999; Glasserman, 2004).

Algorithm 1 StepRS-GRPO for dLLM post-training

Require: Policy π_θ, frozen reference π_{ref}, verifier R, training prompts $\mathcal{D}$, group size G, update count N_{upd}, schedule map b, advantage cap C, regularization weight λ, and the native trainer.

for update $n = 1, \ldots, N_{\mathrm{upd}}$ **do**
  Freeze the rollout policy $\pi_{\mathrm{old}} \leftarrow \pi_\theta$ and sample a prompt minibatch $\mathcal{B}$.
  For each $q \in \mathcal{B}$, draw G trajectories from π_{old} and record active actions $\mathcal{J}_q$ and coordinates u_{qj}.
  Evaluate $r_{qi} = R(x_{qi,0}, q)$ and compute $\widehat{p}_q = G^{-1} \sum_i r_{qi}$.
  for each active action $j \in \mathcal{J}_q$, for each $q \in \mathcal{B}$ **do**
    Set $\beta_{qj} = b(u_{qj})$ and compute A^{Step}_{qj} by Equation (3).
    Set $B_{qj} = \mathrm{clip}(A^{\mathrm{Step}}_{qj}, -C, C)$ and treat it as constant during differentiation.
  end for
  Update θ with $\mathcal{L}_{\mathrm{native}}(\theta; B)$, retaining the trainer's ratio estimation, clipping, normalization, and optimizer.
end for

3 METHOD: FROM GRPO TO STEPRS-GRPO

We develop StepRS-GRPO by first specifying how a terminal reward enters a diffusion-model update, then introducing risk-sensitive weighting, and finally allowing that weighting to vary across denoising states. The rollout procedure, verifier, likelihood estimator, loss reduction, optimizer, and reference regularization remain unchanged.

3.1 DIFFUSION LANGUAGE MODELS

For a prompt q, a dLLM follows reverse denoising states $s_T, s_{T-1}, \ldots, s_0$, where T is the number of transitions. A state contains the partially revealed text and the decoder information needed for the next transition. Let $\mathrm{out}(s_0) = x_0$ extract the final text. Conditional on the initialized state s_T, its distribution is

$$\pi_\theta(x_0 \mid q, s_T) = \sum_{\substack{s_0, \ldots, s_{T-1}: \\ \mathrm{out}(s_0) = x_0}} \prod_{t=1}^{T} \pi_\theta(s_{t-1} \mid s_t, q, t).$$

Here θ denotes the model parameters. The sum includes every state trajectory ending in the same output, including any terminal decoder metadata. Deterministic initialization makes this also the distribution given q; otherwise, we average over s_T. For a trajectory τ, we write $x_0(\tau)$ for this extracted text.

Masked dLLMs use bidirectional context across a partially masked output buffer (Sahoo et al., 2024; Nie et al., 2025). Block dLLMs instead generate blocks from left to right and denoise within each block (Arriola et al., 2025; Cheng et al., 2025). Our SDAR experiments use block length $\ell_{\mathrm{blk}} = 4$; Appendix A details both factorizations. In either family, the information available to the model changes as tokens are revealed, even though the verifier evaluates only the completed answer.

To describe the training decisions uniformly, write $j = (i, t, \ell)$ for an active policy-loss decision in response i, at reverse step t and token position ℓ. The trainer determines the active set $\mathcal{J}_q$, and $i(j)$ identifies the response containing j. We associate each decision with a normalized state coordinate $u_{qj} \in [0, 1]$, where larger values mean more-masked states. This distinguishes maskedness from global trajectory time: in block diffusion, progress within a block and the order of output blocks need not follow the same coordinate.

3.2 FROM CENTERED ADVANTAGES TO STATE-DEPENDENT RISK

Outcome feedback and centered GRPO. For each prompt in a minibatch $\mathcal{B}$, a frozen snapshot π_{old} of the current policy samples G response trajectories. The verifier returns $r_{qi} = R(x_{qi,0}, q) \in \{0, 1\}$,

with 1 indicating a correct answer. The empirical solve fraction and the centered advantages are

$$\widehat{p}_q = \frac{1}{G}\sum_{i=1}^{G} r_{qi}, \qquad A^0_{qi} = r_{qi} - \widehat{p}_q, \qquad a_{qj} = A^0_{q,i(j)}.$$

Thus, every active decision in a response receives the same outcome signal: $1 - \widehat{p}_q$ for a correct response and $-\widehat{p}_q$ for an incorrect one. We use this mean-centered variant of GRPO, without dividing by the group's reward standard deviation (Shao et al., 2024).

The underlying policy loss. For a nonempty prompt minibatch $\mathcal{B}$ and advantage array $A = (A_{qj})$, write the trainer's loss as

$$\mathcal{L}_{\text{native}}(\theta; A) = -\frac{1}{|\mathcal{B}|}\sum_{q\in\mathcal{B}}\sum_{j\in\mathcal{J}_q} v_{qj}\Psi^{\text{native}}_{qj}(\theta; A) + \lambda\mathcal{K}_{\text{ref}}(\theta). \tag{1}$$

The positive factors v_{qj} specify fixed loss-reduction weights, including response or token averaging; they are not risk coefficients. The reward-surrogate contribution $\Psi^{\text{native}}_{qj}$ may depend on other sampled entries through the trainer's normalization, so its argument is the full array A. The sampled data and fixed reduction weights are held constant during differentiation. The term $\mathcal{K}_{\text{ref}}$ is the trainer's KL-to-reference penalty or its estimator, with weight $\lambda \geq 0$. The fixed reference π_{ref} provides the regularization target; π_{old} is refreshed to collect rollouts.

For example, PPO uses $\min\{\rho_{qj}A_{qj}, \text{clip}(\rho_{qj}, 1-\epsilon, 1+\epsilon)A_{qj}\}$, where ρ_{qj} is the current-to-rollout policy ratio for the decision and $\epsilon > 0$ is its clipping threshold (Schulman et al., 2017). Diffusion trainers may estimate this ratio and use different stabilizing operations; StableDRL uses unconditional ratio clipping and self-normalization (Zhong et al., 2026). Equation (1) retains those native choices rather than replacing them with the PPO example. Our centered baseline is $\mathcal{L}_{\text{GRPO}}(\theta) = \mathcal{L}_{\text{native}}(\theta; a)$.

RS-GRPO with a constant coefficient. RS-GRPO (Jiang et al., 2026) transforms the group's rewards using a risk coefficient $\beta > 0$:

$$A_{qi}(\beta) = \frac{1}{\beta}\left[\frac{\exp(\beta r_{qi})}{G^{-1}\sum_{h=1}^{G}\exp(\beta r_{qh})} - 1\right], \qquad A_{qi}(0) = A^0_{qi}. \tag{2}$$

The denominator averages exponentiated rewards over the same prompt group; h indexes its responses. Subtracting one centers these normalized weights, and the factor $1/\beta$ recovers the centered advantage continuously as $\beta \to 0$. A constant coefficient assigns $A_{q,i(j)}(\beta)$ to every decision in response $i(j)$, modifying its weight without changing the observed reward.

Before applying the native loss, we retain the common advantage cap $\text{clip}(z, -C, C)$, where $\text{clip}(z, l, h) = \min\{h, \max\{l, z\}\}$ and $C = 10$. This operation is separate from policy-ratio clipping. The centered advantages already lie in $[-1, 1]$, so this cap leaves the baseline unchanged.

StepRS: a coefficient for each denoising state. A constant coefficient uses the same reward transformation throughout a response. StepRS-GRPO instead evaluates a fixed schedule map $b : [0, 1] \to [0, \infty)$ at each decision's state coordinate:

$$\beta_{qj} = b(u_{qj}), \qquad A^{\text{Step}}_{qj} = A_{q,i(j)}(\beta_{qj}), \qquad B_{qj} = \text{clip}(A^{\text{Step}}_{qj}, -C, C). \tag{3}$$

Here B_{qj} is the capped advantage supplied to the trainer. For a given action j, its coefficient β_{qj} is applied to *every reward* in the denominator of Equation (2); coefficients from different actions are not mixed within that denominator. The same terminal feedback can therefore receive different state-dependent weights.

The resulting objective is

$$\mathcal{L}_{\text{StepRS}}(\theta) = \mathcal{L}_{\text{native}}(\theta; B), \tag{4}$$

where $B = (B_{qj})$ collects the capped advantages over the minibatch. Rewards, recorded coordinates, and constructed advantages are held fixed during differentiation. The schedule changes how outcome feedback is allocated, not how intermediate reasoning is scored. Setting $b \equiv 0$ recovers centered GRPO; choosing a positive constant recovers capped constant-risk RS-GRPO. Algorithm 1 gives the training procedure.

3.3 BINARY-REWARD INTERPRETATION AND SCHEDULE SELECTION

The binary-reward setting makes the transformation particularly transparent. We first establish what it changes within a sampled group, then use exponential tilting to interpret the coefficient's scale. Here $\widehat{p}_q$ is the empirical solve fraction in one group, while $p_\pi(q) = \Pr_{\tau\sim\pi(\cdot|q)}[R(x_0(\tau), q) = 1]$ is the success probability of a fixed rollout distribution π; $x_0(\tau)$ is the completed answer in trajectory τ.

Lemma 1 (Finite-group factorization and success-odds shift)**.** *For every realized binary-reward group and $\beta > 0$,*

$$A_{qi}(\beta) = c_\beta(\widehat{p}_q)A^0_{qi}, \qquad c_\beta(p) = \frac{e^\beta - 1}{\beta(1 - p + pe^\beta)}, \qquad 0 \le p \le 1. \tag{5}$$

The identity extends continuously to $\beta = 0$ with $c_0(p) = 1$. Separately, let $p = p_\pi(q)$ and define the exponentially tilted rollout distribution

$$\pi_\beta(\tau \mid q) = \frac{\pi(\tau \mid q)\exp\{\beta R(x_0(\tau), q)\}}{1 - p + pe^\beta}. \tag{6}$$

Its success probability is $p_\beta = pe^\beta/(1 - p + pe^\beta)$, and, for $0 < p < 1$,

$$\log\frac{p_\beta}{1 - p_\beta} = \log\frac{p}{1 - p} + \beta. \tag{7}$$

The finite-group identity follows directly from $G^{-1}\sum_i e^{\beta r_{qi}} = 1 - \widehat{p}_q + \widehat{p}_q e^\beta$; Appendix B.1 provides the full proof. No population approximation is required, and homogeneous reward groups give zero advantages.

What risk sensitivity changes. The multiplier $c_\beta(p)$ is positive, so it preserves advantage signs and scales both reward classes by the same factor at a fixed coefficient. For $\beta > 0$, it decreases with p: groups with fewer successes receive larger multipliers than groups with more successes at that coefficient. Increasing β need not increase advantage magnitude at every solve rate. In StepRS, the uncapped advantage is $A^{\mathrm{Step}}_{qj} = w_{qj}a_{qj}$, where $w_{qj} = c_{b(u_{qj})}(\widehat{p}_q)$. Thus a_{qj} supplies the outcome signal, w_{qj} supplies the additional risk weighting, and v_{qj} retains the trainer's loss reduction. This separation motivates controls that match either multiplier mass or the root mean square (RMS) of the advantages (see Section 4.3).

Capability-guided coefficient selection. Under the mathematical tilt, increasing β multiplies the success-to-failure odds by e^β, or equivalently reduces the failure-to-success odds by $e^{-\beta}$. We use this interpretation to choose a starting scale, without identifying the tilted distribution with the result of an optimizer update. Let π_{init} be the initial policy and $\mathcal{D}_{\mathrm{cal}}$ a calibration prompt distribution. Its average solve probability is $\bar{p} = \mathbb{E}_{q\sim\mathcal{D}_{\mathrm{cal}}}[p_{\pi_{\mathrm{init}}}(q)]$. Matching the fractional reduction in failure odds, $1 - e^{-\beta}$, to this solve-rate proxy gives

$$\beta^\star(\bar{p}) = -\log(1 - \bar{p}), \qquad 0 \le \bar{p} < 1. \tag{8}$$

This calibration supplies a candidate scale for endpoint selection, rather than deriving an optimal coefficient from the training loss. We evaluate nearby choices and hold the selected endpoints fixed during training. For SDAR-4B, $\bar{p} \approx 0.66$ gives $\beta^\star \approx 1.08$; Appendix D.2 compares the rounded calibrated schedule with neighboring settings.

From endpoints to a denoising-state schedule. With $\beta_{\max} \ge \beta_{\min} \ge 0$, the linear schedule is

$$b(u) = \beta_{\min} + (\beta_{\max} - \beta_{\min})u, \qquad \beta_{\min} = \gamma\beta_{\max}, \quad 0 \le \gamma \le 1. \tag{9}$$

It assigns $\beta_{\max}$ at $u = 1$ and $\beta_{\min}$ at $u = 0$. The endpoint ratio γ controls the decay: $\gamma = 0$ gives full decay and $\gamma = 1$ a constant coefficient. We use half-decay, $\gamma = 1/2$, with $\beta_{\max} = 1.0$ for SDAR-1.7B and 1.5 for SDAR-4B. The notation $[\beta_{\max} \to \beta_{\min}]$ follows decreasing maskedness, not optimizer time. The ablations test endpoint selection, schedule direction, and interpolation separately.

4 EXPERIMENTS

In this section, we present the implementation of our RL post-training algorithm for dLLMs and empirically evaluate the resulting models. Due to space limitations, further experimental settings and implementation details are deferred to Appendix C.

4.1 SETUP

Models and training. We conduct the main experiments using two block diffusion language models from the SDAR-Chat family (Cheng et al., 2025). **SDAR-1.7B-Chat** is a 1.7B-parameter block dLLM, while **SDAR-4B-Chat** is a 4B-parameter block dLLM from the same architectural family. Both models are trained under identical settings: we use the MATH training split restricted to difficulty levels 3–5, comprising 7,500 distinct problems, and perform RL post-training with the StableDRL framework (Zhong et al., 2026), which is specifically designed for stable RL post-training of dLLMs. Training is conducted for 160 optimizer steps on NVIDIA H100 GPUs. For both models, we use $G = 16$ rollouts per prompt and binary math-correctness rewards. An additional case study on masked dLLMs is provided in Appendix C.1. Full hyperparameter details are reported in Appendix C.

Benchmarks and metrics. We evaluate mathematical reasoning on **MATH500** (Hendrycks et al., 2021), **AMC23** (Mathematical Association of America, 2023), **AIME2024** (Mathematical Association of America, 2024), and **AIME2025** (Mathematical Association of America, 2025), covering both standard and competition-style math problems.

Pass@k Accuracy. For each problem q, we draw $N = 64$ independent samples with sampling temperature $T_{\text{sample}} = 1.0$. Correctness is determined using the same final-answer extraction, canonicalization, and verification pipeline across all methods. Let $c(q)$ be the number of correct samples among the N generations. We estimate pass@k using the unbiased estimator of Chen et al. (2021):

$$\widehat{\text{pass@}k}(q) \;=\; 1 - \frac{\binom{N-c(q)}{k}}{\binom{N}{k}}, \tag{10}$$

with $\binom{N-c(q)}{k} = 0$ when $N - c(q) < k$. We report pass@k by averaging $\widehat{\text{pass@}k}(q)$ over all problems in each benchmark.

Baselines and repeated evaluation. We compare the pre-RL model, centered GRPO, constant-risk RS-GRPO, and StepRS-GRPO under a common within-backbone trainer, optimizer, verifier, and KL coefficient $\lambda = 0.01$. The scheduled defaults are $[1.0 \to 0.5]$ for SDAR-1.7B, $[1.5 \to 0.75]$ for SDAR-4B, and $[0.5 \to 0.25]$ for the masked-model case study. Each configuration in the three-seed evaluation is evaluated with inference seeds 42, 43, and 44 at $k \in \{1, 2, 4, 8, 16, 32, 64\}$. Curves show means with capped error bars for one sample standard deviation across fixed-checkpoint evaluations. We also report the unique-answer ratio $\bar{U}$, defined in Appendix C.3. The mechanism controls are described in Section 4.3.

4.2 MAIN RESULTS

Figure 1 compares Base, GRPO, and StepRS-GRPO on the two SDAR backbones. The selected schedules are $[1.0 \to 0.5]$ for SDAR-1.7B and $[1.5 \to 0.75]$ for SDAR-4B.

Accuracy and solution coverage. StepRS-GRPO achieves higher mean pass@k than GRPO at every evaluated budget on all eight SDAR model–benchmark pairs. At 1.7B, mean pass@1 increases from 59.20 to 61.33 on MATH500, 33.83 to 40.60 on AMC23, 3.90 to 9.60 on AIME2024, and 2.40 to 8.30 on AIME2025. The corresponding AIME pass@64 gains are 10.00 and 6.68 percentage points. At 4B, pass@1 improves from 73.63 to 75.77, 47.33 to 50.67, 11.17 to 12.33, and 11.33 to 13.20 across the same four benchmarks. The AIME2024 and AIME2025 gains extend from 1.17 and 1.87 points at $k = 1$ to 8.89 and 10.01 points at $k = 64$, combining improved individual-answer accuracy with broader sampled-solution coverage.

Output diversity and portability. The coverage gains accompany higher unique-answer ratios on every SDAR model–benchmark pair. On MATH500, for example, $\bar{U}$ increases from 19.20 to 28.00 at 1.7B and from 12.27 to 18.60 at 4B. The masked-dLLM case study shows the same ordering over GRPO across all seven budgets on GSM8K, MATH500, and AMC23 (Appendix C.1). Answer-removed trace diagnostics provide complementary evidence that the increased output variation extends beyond the final answers (Figure 4).

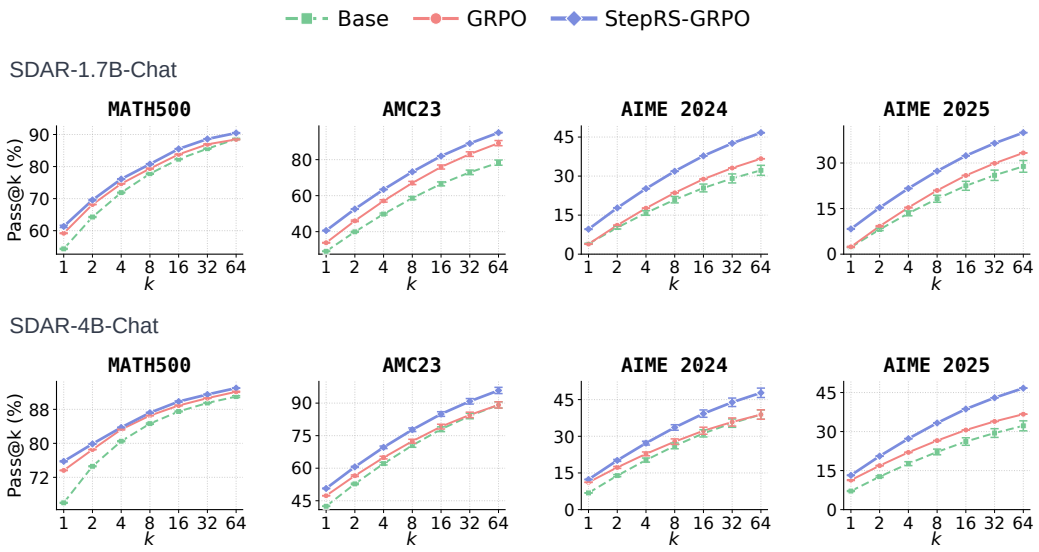


Figure 1: Pass@k on SDAR-1.7B-Chat and SDAR-4B-Chat at the 160-update checkpoints. Markers show means and capped error bars show one sample standard deviation across inference seeds 42, 43, and 44, with 64 responses per problem per seed. The selected StepRS schedules are $[1.0 \to 0.5]$ and $[1.5 \to 0.75]$, respectively. Answer diversity is reported in Figure 12.

4.3 Ablation Studies

We use SDAR-4B to distinguish the contributions of risk sensitivity, denoising-state allocation, and overall advantage scale. Figure 2 compares all seven sampling budgets under the common setup in Section 4.1.

Scheduling versus constant risk. Constant-risk RS-GRPO with $\beta = 1.5$ is a strong comparator. StepRS-GRPO matches or improves its mean pass@k at every benchmark–budget combination, with strict gains throughout AMC23 and both AIME benchmarks. On these three benchmarks, the pass@1 gains are 0.20, 0.40, and 0.50 percentage points, and the pass@64 gains are 0.83, 1.10, and 2.24 points. The full constant-risk sweep is shown in Appendix D.3.

Common notation for the controls. For prompt q, $\mathcal{J}_q$ contains the active token decisions and $a_{qj} = r_{q,i(j)} - \widehat{p}_q$ is their centered outcome advantage. The SDAR-4B reference uses $b(u) = 0.75 + 0.75u$, $w_{qj} = c_{b(u_{qj})}(\widehat{p}_q)$, and $B_{qj} = \mathrm{clip}(w_{qj}a_{qj}, -10, 10)$. The weights $v_{qj} > 0$ are the fixed reductions in Equation (1). Below, multiplier "mass" means $\sum_j v_{qj}w_{qj}$, not probability mass or gradient magnitude. The controls compute their matching quantities from their own current rollouts using this reference rule. Prompts with no active decisions contribute no reward-loss term.

Control 1: remove state variation, retain average weight. C1 asks whether prompt-dependent weighting is sufficient. For each prompt, we replace all state-dependent multipliers by their loss-weighted mean:

$$A^{\mathrm{C1}}_{qj} = \bar{w}_q a_{qj}, \qquad \bar{w}_q = \frac{\sum_{j\in\mathcal{J}_q} v_{qj}w_{qj}}{\sum_{j\in\mathcal{J}_q} v_{qj}}. \tag{11}$$

This is the unique constant preserving the prompt's total multiplier mass, $\sum_j v_{qj}\bar{w}_q = \sum_j v_{qj}w_{qj}$, before applying the common cap. It also minimizes the weighted squared deviation from the original multipliers, giving C1 a least-squares interpretation (Boyd & Vandenberghe, 2018); the derivation is in Appendix D.1. StepRS has higher mean pass@k in all comparisons, supporting state-dependent allocation beyond this prompt-only weighting.

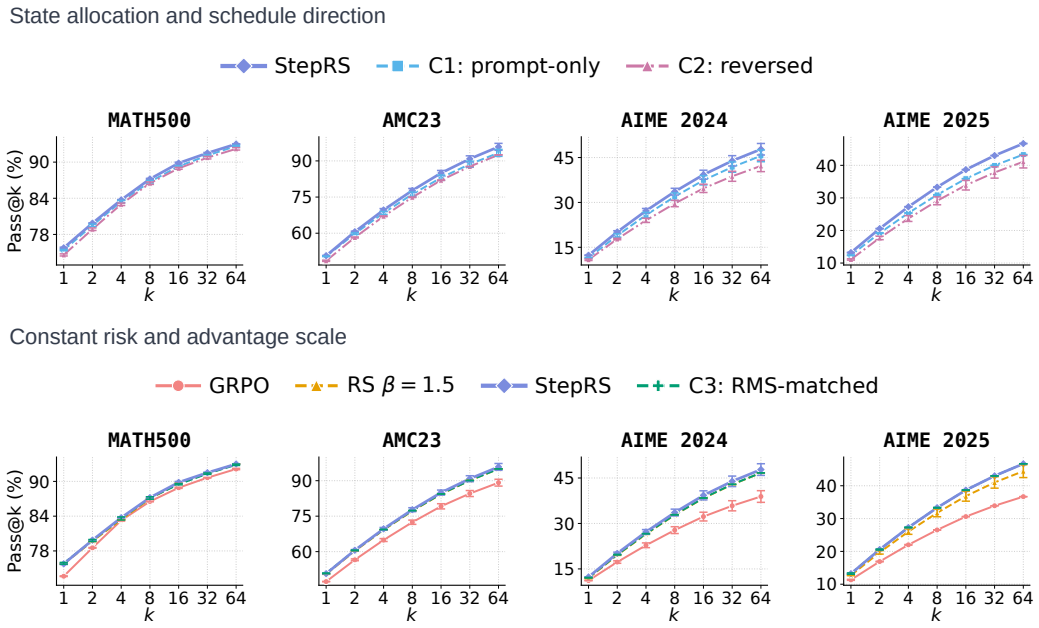


Figure 2: SDAR-4B mechanism ablations. Top: StepRS versus prompt-only mass matching (C1) and a mass-normalized reversed schedule (C2). Bottom: GRPO, constant-risk RS-GRPO ($\beta = 1.5$), StepRS, and advantage-RMS-matched StepRS (C3). Markers show three-inference-seed means; capped error bars show one sample standard deviation. Controls are defined in the text, with implementation details in Appendix D.1.

Control 2: reverse the schedule at matched weight mass. C2 assigns the large coefficient to less-masked states by evaluating $b(1-u)$ instead of $b(u)$. Write $\breve{w}_{qj} = c_{b(1-u_{qj})}(\widehat{p}_q)$ and use

$$A^{\mathrm{C2}}_{qj} = \kappa_q \breve{w}_{qj} a_{qj}, \qquad \kappa_q = \frac{\sum_{j\in\mathcal{J}_q} v_{qj} w_{qj}}{\sum_{j\in\mathcal{J}_q} v_{qj} \breve{w}_{qj}}. \tag{12}$$

The positive factor κ_q preserves the reference multiplier mass before the common cap; reversal alone need not do so because c_β is nonlinear. StepRS is higher in all 28 comparisons, with pass@64 gains of 5.58 and 5.59 points on AIME2024 and AIME2025, favoring the proposed coefficient assignment.

Control 3: retain the pattern, match advantage magnitude. C3 tests whether StepRS remains effective after matching its overall advantage magnitude to centered GRPO. We measure this magnitude by $\mathrm{RMS}_{\mathcal{B}}$, the square root of the average squared advantage over all active entries in one optimizer update (Appendix D.1). With the centered advantages $a = (a_{qj})$ and capped StepRS advantages $B = (B_{qj})$ computed on the same actions, C3 supplies

$$B^{\mathrm{C3}}_{qj} = \eta_{\mathcal{B}} B_{qj}, \qquad \eta_{\mathcal{B}} = \mathrm{stopgrad}\left(\frac{\mathrm{RMS}_{\mathcal{B}}(a)}{\mathrm{RMS}_{\mathcal{B}}(B)}\right). \tag{13}$$

Here stopgrad keeps the scalar's value but prevents backpropagation through it. When both RMS values are zero, we set the scaled advantages to zero without forming the ratio. Otherwise, the positive scalar preserves the signs and relative ratios of the capped StepRS advantages. Unlike mean-and-standard-deviation normalization (Andrychowicz et al., 2021), C3 does not recenter them. C3 stays within 0.24 points of StepRS on pass@1, and both variants exceed GRPO in all comparisons. The gains therefore persist when the overall advantage RMS is matched to centered GRPO.

Calibration-guided endpoint selection. For SDAR-4B, the solve-rate proxy $\bar{p} \approx 0.66$ gives $\beta^\star \approx 1.08$. The rounded calibrated schedule $[1.08 \to 0.54]$ improves on $[1.0 \to 0.5]$ at every benchmark–budget combination and is within 0.47 percentage points of the selected $[1.5 \to 0.75]$ schedule on pass@1. This supports calibration as a useful starting point for endpoint selection. The complete endpoint sweep is reported in Appendix D.2.

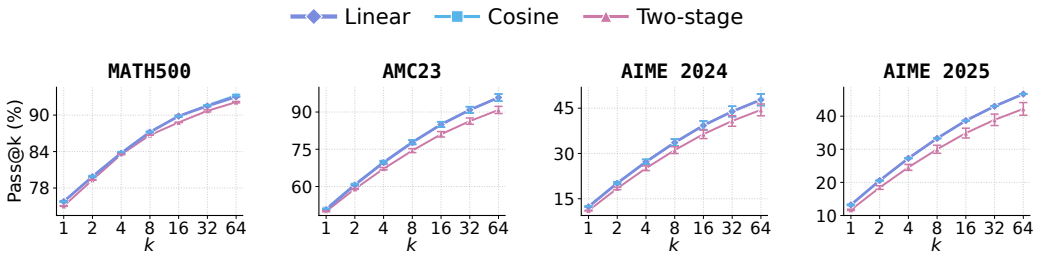


Figure 3: Schedule-shape ablation on SDAR-4B with fixed endpoints $[1.5 \rightarrow 0.75]$. Linear, cosine, and two-stage schedules are compared at all seven sampling budgets. Markers show means and capped error bars show one sample standard deviation across three inference seeds.

Schedule shape at fixed endpoints. To separate interpolation from the coefficient range, we fix the endpoints to $[1.5 \rightarrow 0.75]$ and compare linear, cosine, and two-stage schedules (Figure 3). Linear and cosine differ by at most 0.47 percentage points in mean pass@k across all four benchmarks and seven budgets, showing that performance is stable across these two smooth interpolations. Both exceed the tested two-stage schedule at every benchmark–budget combination. For linear interpolation, the pass@64 gains are 0.87, 5.00, 3.34, and 4.49 points on MATH500, AMC23, AIME2024, and AIME2025, respectively. Together with the reversed-schedule control, these results support the proposed schedule direction and smooth variation between the endpoints. We retain linear half-decay as a simple, effective default.

Reasoning-trace diversity. We remove the final boxed answer and its surrounding sentence before measuring the remaining reasoning text. N-gram variation uses $D_{\text{ngram}} = 100(1 - \text{Self-BLEU-4})$: each trace is compared with the other traces for the same question (Zhu et al., 2018). Embedding variation, D_{emb}, is 100 times their mean pairwise cosine distance, using the Sentence Transformers checkpoint `all-MiniLM-L6-v2` (Reimers & Gurevych, 2019; Sentence Transformers, 2021). In Figure 4, StepRS exceeds GRPO on all four benchmarks under both measures, with gains of 2.3–6.6 and 1.3–3.5 points, respectively. These all-trace diagnostics show that the increased variation extends beyond final-answer strings; Appendix C.3 defines both metrics in detail.

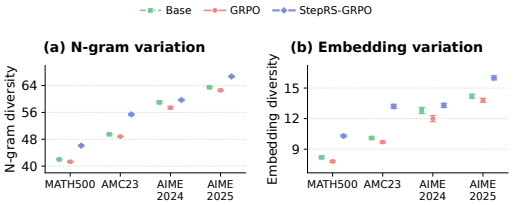


Figure 4: Answer-removed trace diversity on SDAR-4B: (a) D_{ngram}; (b) D_{emb}. Points and capped bars show means and sample standard deviations across three inference seeds, with 64 responses per question per seed. Higher scores indicate greater variation; panel scales differ. Definitions are in Appendix C.3.

5 Conclusion and Limitations

We introduced StepRS-GRPO, a denoising-state-dependent extension of risk-sensitive GRPO for diffusion language models. Our experiments show improvements in mathematical accuracy, sampled-solution coverage, and output diversity. Ablation studies support the contributions of state allocation and schedule direction beyond prompt-level weighting, and the advantage-RMS-matched variant retains the gains over GRPO. Calibration and schedule-shape experiments further provide practical guidance for selecting effective coefficients.

The capability rule guides the selection of schedule endpoints, but does not establish an optimal schedule. Our evaluation also focuses on mathematical tasks with binary rewards, and the three inference seeds quantify sampling variability at fixed checkpoints rather than variability across training runs. Additional training replications and evaluation under broader reward settings would help establish how consistently these gains hold beyond the settings studied here.

AI USE STATEMENT

In this work, we used generative AI tools for polishing writing. We have not used generative AI tools for generating synthetic data sets, helping develop theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, implementing methods, interpreting results, and assisting with translation, cleaning and reformatting dataset, supporting qualitative and thematic data analysis are not applicable to this work. Additionally, we used generative AI tools for summarizing or analyzing existing literature. We have reviewed all AI-assisted work. We checked all the related work suggested by AI tools, and all the wordings polished by LLM. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Ethics statement. This work adheres to the ICLR Code of Ethics. Our study involves no human subjects, user studies, or collection of personal data. All experiments use publicly available models and established public benchmarks (e.g., MATH500, AMC23, etc.), used in accordance with their licenses.

Reproducibility statement. We will publicly release our code upon publication to facilitate reproducibility.

REFERENCES

Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphael Marinier, Léonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, Sylvain Gelly, and Olivier Bachem. What matters in on-policy reinforcement learning? a large-scale empirical study. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2006.05990.

Marianne Arriola, Aaron Gokaslan, Justin T. Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models, 2025.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems*, 2021.

Stephen Boyd and Lieven Vandenberghe. *Introduction to Applied Linear Algebra: Vectors, Matrices, and Least Squares*. Cambridge University Press, 2018. doi: 10.1017/9781108583664. URL https://web.stanford.edu/~boyd/vmls/.

Boxing Chen and Colin Cherry. A systematic comparison of smoothing techniques for sentence-level BLEU. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pp. 362–367. Association for Computational Linguistics, 2014. doi: 10.3115/v1/W14-3346. URL https://aclanthology.org/W14-3346/.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Shuang Cheng, Yihan Bian, Dawei Liu, Linfeng Zhang, Qian Yao, Zhongbo Tian, Wenhai Wang, Qipeng Guo, Kai Chen, Biqing Qi, et al. Sdar: A synergistic diffusion-autoregression paradigm for scalable sequence generation. *arXiv preprint arXiv:2510.06303*, 2025.

Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, Zhiyuan Liu, Hao Peng, Lei Bai, Wanli Ouyang, Yu Cheng, Bowen Zhou, and Ning Ding. The entropy mechanism of reinforcement learning for reasoning language models. *arXiv preprint arXiv:2505.22617*, 2025.

Yingjie Fei, Zhuoran Yang, Yudong Chen, Zhaoran Wang, and Qiaomin Xie. Risk-sensitive reinforcement learning: Near-optimal risk-sample tradeoff in regret. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Paul Glasserman. *Monte Carlo Methods in Financial Engineering*. Springer, 2004.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, 2018a.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018b.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Ronald A. Howard and James E. Matheson. Risk-sensitive Markov decision processes. *Management Science*, 18(7):356–369, 1972.

Yuhua Jiang, Jiawei Huang, Yufeng Yuan, Xin Mao, Yu Yue, Qianchuan Zhao, and Lin Yan. Risk-sensitive RL for alleviating exploration dilemmas in large language models. In *International Conference on Learning Representations*, 2026. arXiv:2509.24261.

Long Li, Zhijian Zhou, Jiaran Hao, Jason Klein Liu, Yanting Miao, Wei Pang, Xiaoyu Tan, Wei Chu, Zhe Wang, Shirui Pan, et al. The choice of divergence: A neglected key to mitigating diversity collapse in reinforcement learning with verifiable reward. *arXiv preprint arXiv:2509.07430*, 2025a.

Tianjian Li, Yiming Zhang, Ping Yu, Swarnadeep Saha, Daniel Khashabi, Jason Weston, Jack Lanchantin, and Tianlu Wang. Jointly reinforcing diversity and quality in language model generations. *arXiv preprint arXiv:2509.02534*, 2025b.

Nianyi Lin, Jiajie Zhang, Lei Hou, and Juanzi Li. Boundary-guided policy optimization for memory-efficient RL of diffusion large language models. *arXiv preprint arXiv:2510.11683*, 2025.

Mathematical Association of America. Amc 2023. https://artofproblemsolving.com/wiki/index.php?title=AMC_Problems_and_Solutions, 2023. Accessed: 2026-03-20.

Mathematical Association of America. Aime 2024. https://artofproblemsolving.com/wiki/index.php/2024_AIME_I_Problems, 2024. Accessed: 2026-03-20.

Mathematical Association of America. Aime 2025. https://artofproblemsolving.com/wiki/index.php/2025_AIME_I_Problems, 2025. Accessed: 2026-03-20.

Youssef Mroueh. Reinforcement learning with verifiable rewards: GRPO's loss, dynamics, and success amplification. In *International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=z9Bo4eXSpW.

Zanlin Ni, Shenzhi Wang, Yang Yue, Tianyu Yu, Weilin Zhao, Yeguo Hua, Tianyi Chen, Jun Song, Cheng Yu, Bo Zheng, et al. The flexibility trap: Rethinking the value of arbitrary order in diffusion language models. *arXiv preprint arXiv:2601.15165*, 2026.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. *arXiv preprint arXiv:2502.09992*, 2025.

Daisuke Oba, Hiroki Furuta, and Naoaki Okazaki. Diffusion-state policy optimization for masked diffusion language models. *arXiv preprint arXiv:2602.06462*, 2026.

Jingyang Ou, Jiaqi Han, Minkai Xu, Shaoxuan Xu, Jianwen Xie, Stefano Ermon, Yi Wu, and Chongxuan Li. Principled RL for diffusion LLMs emerges from a sequence-level perspective. *arXiv preprint arXiv:2512.03759*, 2025.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 311–318. Association for Computational Linguistics, 2002. doi: 10.3115/1073083.1073135. URL https://aclanthology.org/P02-1040/.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3982–3992. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1410. URL https://aclanthology.org/D19-1410/.

Tao Ren, Jinyang Jiang, Hui Yang, Wan Tian, Minhao Zou, Guanghao Li, Zishi Zhang, Qinghao Wang, Shentao Qin, Yanjun Zhao, Rui Tao, Hui Shao, and Yijie Peng. RiskPO: Risk-based policy optimization via verifiable reward for LLM post-training. *arXiv preprint arXiv:2510.00911*, 2025.

Kevin Rojas, Jiahe Lin, Kashif Rasul, Anderson Schneider, Yuriy Nevmyvaka, Molei Tao, and Wei Deng. Improving reasoning for diffusion language models via group diffusion policy optimization. *arXiv preprint arXiv:2510.08554*, 2025.

Reuven Y. Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability*, 1(2):127–190, 1999.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Mariano Marroquin, Justin T. Chiu, Alexander M. Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *Advances in Neural Information Processing Systems*, 2024.

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

Sentence Transformers. all-MiniLM-L6-v2: Sentence embedding model and model card. Hugging Face model repository, 2021. URL https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2. Repository first uploaded in August 2021; model card accessed 2026-09-24.

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

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K. Titsias. Simplified and generalized masked diffusion for discrete data. In *Advances in Neural Information Processing Systems*, 2024.

Xiaohang Tang, Rares Dolga, Sangwoong Yoon, and Ilija Bogunovic. wd1: Weighted policy optimization for reasoning in diffusion language models. *arXiv preprint arXiv:2507.08838*, 2025.

Christos Thrampoulidis, Sadegh Mahdavi, and Wenlong Deng. Advantage shaping as surrogate reward maximization: Unifying Pass@K policy gradients. *Transactions on Machine Learning Research*, 2026. URL https://arxiv.org/abs/2510.23049.

Christian Walder and Deep Karkhanis. Pass@K policy optimization: Solving harder reinforcement learning problems. *arXiv preprint arXiv:2505.15201*, 2025. URL https://arxiv.org/abs/2505.15201.

Chen Wang, Zhaochun Li, Jionghao Bai, Yuzhi Zhang, Shisheng Cui, Zhou Zhao, and Yue Wang. Arbitrary entropy policy optimization breaks the exploration bottleneck of reinforcement learning. *arXiv preprint arXiv:2510.08141*, 2025a.

Chenyu Wang, Paria Rashidinejad, DiJia Su, Song Jiang, Sid Wang, Siyan Zhao, Cai Zhou, Shannon Zejiang Shen, Feiyu Chen, Tommi Jaakkola, et al. SPG: Sandwiched policy gradient for masked diffusion language models. *arXiv preprint arXiv:2510.09541*, 2025b.

Yinjie Wang, Ling Yang, Bowen Li, Ye Tian, Ke Shen, and Mengdi Wang. Revolutionizing reinforcement learning framework for diffusion large language models. *arXiv preprint arXiv:2509.06949*, 2025c.

Peter Whittle. *Risk-Sensitive Optimal Control*. Wiley, 1990.

Anthony Zhan. Simple policy gradients for reasoning with diffusion language models. *arXiv preprint arXiv:2510.04019*, 2025.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://papers.neurips.cc/paper_files/paper/2019/hash/1e8a19426224ca89e83cef47f1e7f53b-Abstract.html.

Wenxuan Zhang, Lemeng Wu, Changsheng Zhao, Ernie Chang, Mingchen Zhuge, Zechun Liu, Andy Su, Hanxian Huang, Jun Chen, Chong Zhou, Raghuraman Krishnamoorthi, Vikas Chandra, Mohamed Elhoseiny, and Wei Wen. dTRPO: Trajectory reduction in policy optimization of diffusion large language models. *arXiv preprint arXiv:2603.18806*, 2026.

Siyan Zhao, Devaansh Gupta, Qinqing Zheng, and Aditya Grover. d1: Scaling reasoning in diffusion large language models via reinforcement learning. *arXiv preprint arXiv:2504.12216*, 2025a.

Siyan Zhao, Mengchen Liu, Jing Huang, Miao Liu, Chenyu Wang, Bo Liu, Yuandong Tian, Guan Pang, Sean Bell, Aditya Grover, and Feiyu Chen. Inpainting-guided policy optimization for diffusion large language models. *arXiv preprint arXiv:2509.10396*, 2025b.

Jianyuan Zhong, Kaibo Wang, Ding Ding, Zijin Feng, Haoli Bai, Yang Xiang, Jiacheng Sun, and Qiang Xu. Stabilizing reinforcement learning for diffusion language models. *arXiv preprint arXiv:2603.06743*, 2026.

Zhanhui Zhou, Lingjie Chen, Hanghang Tong, and Dawn Song. dllm: Simple diffusion language modeling, 2026. URL https://arxiv.org/abs/2602.22661.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. Texygen: A benchmarking platform for text generation models. *arXiv preprint arXiv:1802.01886*, 2018. URL https://arxiv.org/abs/1802.01886.

Ying Zhu, Jiaxin Wan, Xiaoran Liu, Siyang He, Qiqi Wang, Xu Guo, Tianyi Liang, Zengfeng Huang, Ziwei He, and Xipeng Qiu. DiRL: An efficient post-training framework for diffusion language models. *arXiv preprint arXiv:2512.22234*, 2025a.

Yuchen Zhu, Wei Guo, Jaemoo Choi, Petr Molodyk, Bo Yuan, Molei Tao, and Yongxin Chen. Enhancing reasoning for diffusion llms via distribution matching policy optimization. *arXiv preprint arXiv:2510.08233*, 2025b.

CONTENTS OF THE APPENDIX

A Diffusion Language Model Families **16**
A.1 Masked diffusion. 16
A.2 Block diffusion. 16

B Additional Theoretical Analysis and Calibration **17**
B.1 Proof of Lemma 1 17
B.2 Capability-guided calibration. 18
B.3 Scope of denoising-state scheduling 18

C Experimental Details **18**
C.1 Additional results: 0.6B masked diffusion LM 18
C.2 Training configuration 19
C.3 Evaluation protocol 19
C.4 Prompt templates. 21

D Additional Ablations and Diagnostics **21**
D.1 Mechanism-control definitions and matching properties. 21
D.2 Calibration and endpoint choice 23
D.3 Full constant-risk sweep 23
D.4 Risk and endpoint sensitivity on smaller backbones. 23
D.5 Answer and trace diversity 23

E Solution Examples **27**

A Diffusion Language Model Families

This appendix distinguishes continuous diffusion time from the discrete policy-loss index in Section 3.1. We suppress the prompt conditioning only where it is not needed. Let $\mathcal{V}$ be the clean-token vocabulary, $m \notin \mathcal{V}$ the mask token, and L the padded output length. The symbol $\delta_a(z) = \mathbf{1}\{z = a\}$ denotes point mass at token a; $\mathbf{1}\{E\}$ equals one when condition E holds and zero otherwise.

A.1 Masked diffusion

Use $\zeta \in [0, 1]$ for continuous forward-diffusion time and $\alpha(\zeta)$ for the masking probability, with $\alpha(0) = 0$, $\alpha(1) = 1$, and α nondecreasing. Writing $X_0 = x_0 \in \mathcal{V}^L$, the absorbing forward process has marginal

$$Q(X_\zeta^{(\ell)} = z \mid X_0 = x_0) = [1 - \alpha(\zeta)]\delta_{x_0^{(\ell)}}(z) + \alpha(\zeta)\delta_m(z), \qquad \ell = 1, \dots, L. \tag{14}$$

Masking is independent across positions and absorbing across time. The expected masked fraction is $\alpha(\zeta)$; a realized state's masked fraction need not equal this expectation.

Let $f_\theta^{(\ell)}(z \mid x_\zeta, \zeta, q)$ be the model's clean-token probability at position ℓ, extended by $f_\theta^{(\ell)}(m \mid \cdot) = 0$. For $0 \le \xi < \zeta \le 1$ with $\alpha(\zeta) > 0$, the factorized absorbing reverse kernel is

$$P_\theta(X_\xi^{(\ell)} = z \mid x_\zeta, q) = \begin{cases} \delta_{x_\zeta^{(\ell)}}(z), & x_\zeta^{(\ell)} \neq m, \\ \dfrac{\alpha(\xi)}{\alpha(\zeta)}\delta_m(z) + \left[1 - \dfrac{\alpha(\xi)}{\alpha(\zeta)}\right] f_\theta^{(\ell)}(z \mid x_\zeta, \zeta, q), & x_\zeta^{(\ell)} = m. \end{cases} \tag{15}$$

The full kernel multiplies these terms over positions (Austin et al., 2021; Sahoo et al., 2024; Shi et al., 2024). This absorbing kernel is background; a confidence-based decoder that selects reveal positions may implement a different transition rule. The rollout policy and its likelihood estimator must describe that implemented decoder.

Assume α is absolutely continuous and $\alpha(\zeta) > 0$ for $0 < \zeta \le 1$. The corresponding weighted denoising objective is

$$\mathcal{L}_{\mathrm{MDLM}}(\theta) = -\mathbb{E}\left[\frac{\alpha'(\zeta)}{\alpha(\zeta)} \sum_{\ell=1}^{L} \mathbf{1}\{X_\zeta^{(\ell)} = m\} \log f_\theta^{(\ell)}(X_0^{(\ell)} \mid X_\zeta, \zeta, q)\right]. \tag{16}$$

The expectation is over training examples (q, X_0), $\zeta \sim \mathcal{U}(0, 1)$, and $X_\zeta \sim Q(\cdot \mid X_0)$; α' is the time derivative. Endpoint values are interpreted by limits. Since α denotes masking rather than clean-token retention, the weight is α'/α (Sahoo et al., 2024; Shi et al., 2024).

A.2 Block diffusion

Partition the padded sequence into B_{blk} blocks of length ℓ_{blk}. The clean sequence is $x_0 = (x_0^{[1]}, \dots, x_0^{[B_{\mathrm{blk}}]})$, and $x_0^{[<d]}$ is the prefix preceding block d. Block diffusion uses

$$\pi_\theta(x_0 \mid q) = \prod_{d=1}^{B_{\mathrm{blk}}} \pi_\theta(x_0^{[d]} \mid x_0^{[<d]}, q). \tag{17}$$

For a fixed within-block horizon K_{blk}, initialize $x_{K_{\mathrm{blk}}}^{[d]} = m^{\ell_{\mathrm{blk}}}$ and marginalize only the unfixed intermediate states:

$$\pi_\theta(x_0^{[d]} \mid x_0^{[<d]}, q) = \sum_{x_1^{[d]}, \dots, x_{K_{\mathrm{blk}}-1}^{[d]}} \prod_{h=1}^{K_{\mathrm{blk}}} P_\theta(x_{h-1}^{[d]} \mid x_h^{[d]}, x_0^{[<d]}, q, h). \tag{18}$$

Attention is causal across blocks and bidirectional within the active block (Arriola et al., 2025; Cheng et al., 2025). Prefix keys and values can be cached, although attention still incurs a prefix-length-dependent cost. Our SDAR experiments use $\ell_{\mathrm{blk}} = 4$.

Relation to the risk coordinate. The discrete index t orders transitions in a sampled trajectory; u_{qj} is the scalar supplied to the risk map b for action j. Neither is automatically equal to continuous time ζ or the expected masking probability $\alpha(\zeta)$. In block diffusion, a within-block mask fraction can restart at each block even though the global trajectory index advances. The reference and controls use the same executed coordinate, action mask, and loss reduction.

B Additional Theoretical Analysis and Calibration

The empirical group fraction $\widehat{p}_q$, the population success probability $p_\pi(q)$, and the model-level calibration average $\bar{p}$ serve different roles. The first enters the actual advantages; the second describes a fixed rollout distribution; the third suggests a coefficient scale before training. The identities below keep these quantities separate.

B.1 Proof of Lemma 1

Fix a prompt q and its realized rewards $r_{q1}, \dots, r_{qG}$. Because each reward is either zero or one, $e^{\beta r_{qi}} = 1 + r_{qi}(e^\beta - 1)$. Averaging this equality over the group gives the exact normalizer

$$\widehat{D}_q(\beta) = \frac{1}{G}\sum_{i=1}^{G} e^{\beta r_{qi}} = 1 + \widehat{p}_q(e^\beta - 1).$$

For $\beta > 0$, substitute this expression into Eq. (2) and collect the terms containing the reward:

$$\begin{aligned} A_{qi}(\beta) &= \frac{1 + r_{qi}(e^\beta - 1) - [1 + \widehat{p}_q(e^\beta - 1)]}{\beta[1 + \widehat{p}_q(e^\beta - 1)]} \\ &= \frac{e^\beta - 1}{\beta[1 + \widehat{p}_q(e^\beta - 1)]}(r_{qi} - \widehat{p}_q) = c_\beta(\widehat{p}_q) A^0_{qi}. \end{aligned} \tag{19}$$

This proves the finite-group factorization. If every reward is zero or every reward is one, $r_{qi} = \widehat{p}_q$ for every response, so all advantages are zero. At $\beta = 0$, the expansion $e^\beta - 1 = \beta + O(\beta^2)$ gives $c_\beta(p) \to 1$. Thus the identity extends continuously to the centered baseline. These statements hold for each realized group, without an independence assumption or a large-group limit.

For the tilting statement, let $p = p_\pi(q)$ be the success probability of a fixed rollout distribution. Its normalizer is $D_\beta = 1 - p + pe^\beta$. Summing the tilted probabilities separately over successful and unsuccessful trajectories gives

$$p_\beta = \frac{pe^\beta}{D_\beta}, \qquad 1 - p_\beta = \frac{1-p}{D_\beta}.$$

When $0 < p < 1$, their ratio is $e^\beta p/(1-p)$; taking logarithms proves Eq. (7). At $p = 0$ or 1, exponential tilting leaves the all-failure or all-success distribution unchanged. This completes the proof. □

Interpretation of the multiplier. For a fixed coefficient $\beta > 0$,

$$\frac{\partial c_\beta(p)}{\partial p} = -\frac{(e^\beta - 1)^2}{\beta[1 + p(e^\beta - 1)]^2} < 0.$$

Thus, groups with lower observed solve fractions receive larger multipliers at the same coefficient. This is a statement about relative weighting across groups, not a guarantee that every multiplier exceeds one. Moreover, $c_\beta(p) = 1 + (\frac{1}{2} - p)\beta + O(\beta^2)$ near zero; increasing β need not increase advantage magnitude uniformly across solve rates. For a common coefficient, the group advantages sum to zero. With action-dependent coefficients and token-reduction weights, an action-weighted mean need not be zero, which is one reason to distinguish RMS matching from standard-deviation matching.

Connection to entropic utility. For a fixed prompt and binary reward, the population entropic utility is $J_\beta(\pi; q) = \beta^{-1} \log[1 + p_\pi(q)(e^\beta - 1)]$. Its derivative with respect to $p_\pi(q)$ is $c_\beta(p_\pi(q))$. When differentiation under the rollout expectation is valid,

$$\nabla_\theta J_\beta(\pi_\theta; q) = c_\beta(p_{\pi_\theta}(q)) \nabla_\theta p_{\pi_\theta}(q).$$

This gives the population coefficient an outcome-reweighting interpretation (Howard & Matheson, 1972; Jiang et al., 2026). The exact finite-group factorization above is a separate statement: substituting the empirical fraction into a nonlinear normalizer does not, by itself, establish an unbiased finite-group gradient estimator.

B.2 Capability-guided calibration

Let π_{init} be the initial policy and $\mathcal{D}_{\text{cal}}$ a calibration distribution. Define

$$\bar{p} = \mathbb{E}_{q \sim \mathcal{D}_{\text{cal}}}[p_{\pi_{\text{init}}}(q)].$$

This model-level quantity differs from the current training group's $\widehat{p}_q$. Exponential tilting multiplies failure-to-success odds by $e^{-\beta}$. Equating their fractional reduction $g(\beta) = 1 - e^{-\beta}$ to the solve-rate proxy yields

$$\beta^\star(\bar{p}) = -\log(1 - \bar{p}), \qquad 0 \le \bar{p} < 1. \tag{20}$$

The rule suggests an interpretable search scale; it is not an optimizer-derived optimum or upper bound. For SDAR-4B, the reported proxy $\bar{p} \approx 0.66$ gives $\beta^\star \approx 1.08$. Figure 6 compares this rounded candidate with neighboring endpoints. The defaults below are then held fixed during each run.

Table 1: Selected schedules and evaluated constant-risk coefficients.

Backbone	Default StepRS schedule	Constant-risk sweep
Qwen3-0.6B-MDLM	$[0.5 \to 0.25]$	$\{0.25, 0.5, 1.0\}$
SDAR-1.7B-Chat	$[1.0 \to 0.5]$	$\{0.25, 0.5, 1.0\}$
SDAR-4B-Chat	$[1.5 \to 0.75]$	$\{0.5, 1.0, 1.5, 2.0\}$

B.3 Scope of denoising-state scheduling

For each action j, StepRS applies the scalar group transformation at $\beta_{qj} = b(u_{qj})$. All rewards in that evaluation of the group normalizer use this same coefficient. Applying a different coefficient to each reward inside one denominator would define another transformation and would not give Equation (5). The schedule therefore has a precise interpretation as the advantage array supplied to the native surrogate in Equation (4). Its direction and interpolation are assessed by the controls and fixed-endpoint ablation, rather than inferred from a fixed-coefficient trajectory objective.

Trainer interface. Equation (1) separates fixed reduction weights v_{qj} from the native surrogate Ψ^{native}, which may depend on the whole sampled group through its normalization. For PPO, the surrogate is the minimum of the unclipped and clipped ratio terms (Schulman et al., 2017). In published StableDRL, unconditional ratio clipping and a sum-of-clipped-ratios normalizer instead weight score-function gradients (Zhong et al., 2026). These are different policy-loss constructions. Our notation preserves the selected construction and changes only the supplied advantages.

C Experimental Details

C.1 Additional results: 0.6B masked diffusion LM

We evaluate **Qwen3-0.6B-MDLM**, distilled from a post-trained Qwen3-0.6B autoregressive backbone via the dLLM A2D pipeline of Zhou et al. (2026). The model is fine-tuned on the GSM8K training split, with $G = 16$ responses per prompt and binary correctness rewards. The evaluated checkpoints are at 60 RL fine-tuning steps. We use the schedule $[0.5 \to 0.25]$ and evaluate each configuration with inference seeds 42, 43, and 44, using 64 responses per problem per seed.

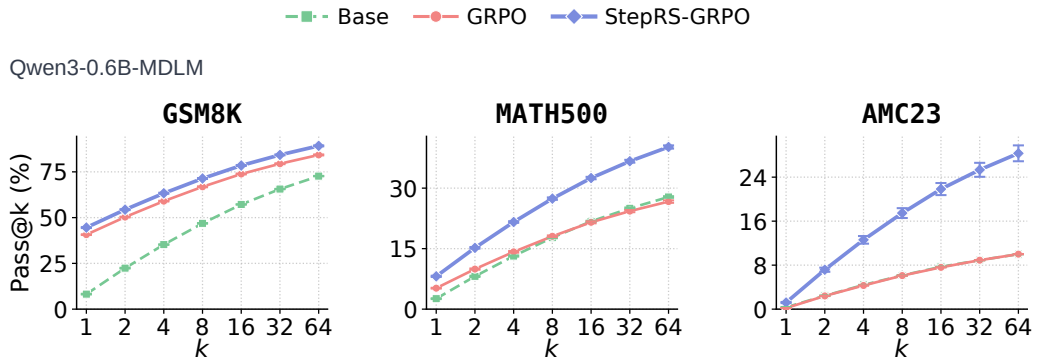


Figure 5: Qwen3-0.6B-MDLM evaluation curves. The selected $[0.5 \rightarrow 0.25]$ schedule improves pass@k over GRPO across GSM8K, MATH500, and AMC23. All curves show means and capped error bars show one sample standard deviation over three seeds.

Figure 5 shows higher mean pass@k than GRPO at all 21 benchmark–budget combinations. Mean pass@1 improves from 40.70 to 44.57 on GSM8K, 5.20 to 8.13 on MATH500, and 0.20 to 1.20 on AMC23. At $k = 64$, the corresponding improvements are $84.25 \rightarrow 89.20$, $26.67 \rightarrow 40.20$, and $10.00 \rightarrow 28.33$. The particularly large coverage gains on MATH500 and AMC23 extend the evidence beyond the two block-dLLM backbones.

The unique-answer ratio also increases from 18.03 to 28.00 on GSM8K, 11.40 to 33.47 on MATH500, and 17.30 to 24.83 on AMC23 (Figure 12). These results demonstrate improvements in both accuracy and sampled-output variation. Because this case study uses a different backbone family and training corpus, we treat it as complementary portability evidence rather than a controlled model-size comparison.

C.2 Training configuration

Table 2 shows the detailed training recipes for the three model scales reported in Section 4 and Appendix C.1.

C.3 Evaluation protocol

We evaluate each fixed checkpoint with inference seeds $\Sigma = \{42, 43, 44\}$. For each question q in a benchmark $\mathcal{Q}$, each seed generates $N = 64$ responses at temperature 1.0. The training group size $G = 16$ and evaluation sample count $N = 64$ are distinct. We compute pass@k from the correct-response count using Equation (10) and average over questions (Chen et al., 2021); we do not pool seeds into an $N = 192$ estimate.

For a metric's benchmark-level score $M^{(\sigma)}$ under seed $\sigma \in \Sigma$, we report

$$\bar{M} = \frac{1}{3} \sum_{\sigma \in \Sigma} M^{(\sigma)}, \qquad s_M = \sqrt{\frac{1}{2} \sum_{\sigma \in \Sigma} \left(M^{(\sigma)} - \bar{M}\right)^2}. \tag{21}$$

Capped error bars show $\bar{M} \pm s_M$, the sample standard deviation across inference seeds, not a standard error or confidence interval. The following definitions concern one model and one seed; these indices are suppressed to keep the notation readable.

Unique-answer ratio. Let $\widehat{y}_{qi}$ be the extracted and normalized final-answer string for response i. We define

$$U(q) = \frac{|\{\widehat{y}_{q1}, \ldots, \widehat{y}_{qN}\}|}{N}, \qquad U^{(\sigma)} = \frac{100}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} U(q). \tag{22}$$

Table 2: Training hyperparameters used for the three model scales.

Hyperparameter	Qwen3-0.6B-MDLM	SDAR-1.7B-Chat	SDAR-4B-Chat
Base model (HuggingFace checkpoint)	`dllm-hub/Qwen3-0.6B-diffusion-mdlm-v0.1`	`JetLM/SDAR-1.7B-Chat`	`JetLM/SDAR-4B-Chat`
Trainer framework	TraceRL (Wang et al., 2025c)	StableDRL (Zhong et al., 2026)	
Training data	GSM8K train	MATH train	
Optimizer	AdamW (lr=5×10^{-6})	AdamW (lr=1×10^{-6})	
Responses per prompt (G)	16		
Group rollout batch	1024 (64×16)	512 (32×16)	
Gradient accumulation	1	32	
Total optimizer steps	100	160	
DeepSpeed	ZeRO-3 (CPU off.)	ZeRO-3 (no off.)	
Mixed precision	bf16		
GPUs	8× L40S (48GB)	8× H100 (95GB)	
Max sequence length	512	1024	
Denoising steps (T) at training	256		
KL penalty (λ)	0.01		
Advantage clip (C)	10.0		

The set removes duplicate normalized answers; $|\cdot|$ denotes cardinality. Failed extractions form one shared empty-answer class. Normalization removes formatting and merges equivalent forms where supported by the extractor. The reported $\bar{U}$ is the mean of the percentage-valued $U^{(\sigma)}$ across seeds.

Answer-removed reasoning traces. Let h_{qi} be response i after removing its final boxed-answer span and the sentence containing that span. Diversity is computed *within each question*, using all N sampled traces, including incorrect responses and repeated texts. Duplicates remain separate observations because their frequency is part of the generation distribution. These metrics assess variation in the retained text, not the number of distinct correct solution strategies.

N-gram variation. An n-gram is a sequence of n consecutive tokens. BLEU-4 combines clipped n-gram precisions for $n = 1, 2, 3, 4$ with equal weights (Papineni et al., 2002). Write $\mathcal{H}_{q,-i} = (h_{qi'} : 1 \le i' \le N,\ i' \ne i)$ for the list of the other $N - 1$ traces, retaining duplicates. We compute sentence-level BLEU for each trace against this reference list, then average:

$$S_4(q) = \frac{1}{N} \sum_{i=1}^{N} \mathrm{BLEU}_4(h_{qi}, \mathcal{H}_{q,-i}), \qquad D_{\mathrm{ngram}}(q) = 100[1 - S_4(q)]. \tag{23}$$

This is the Self-BLEU construction of Zhu et al. (2018). BLEU values are on the $[0, 1]$ scale here. A larger S_4 means more overlap with the other samples, so its complement gives larger values for greater lexical variation. It is not distinct-4, which would count unique four-token sequences.

For a nonempty tokenized hypothesis h and reference list $\mathcal{H}$, the BLEU combination is

$$\mathrm{BLEU}_4(h, \mathcal{H}) = \mathrm{BP}(h, \mathcal{H}) \exp\left\{ \frac{1}{4} \sum_{n=1}^{4} \log \widetilde{P}_n(h, \mathcal{H}) \right\}. \tag{24}$$

Before smoothing, the modified precision is

$$P_n(h, \mathcal{H}) = \frac{\sum_g \min\{C_n(g, h), \max_{h' \in \mathcal{H}} C_n(g, h')\}}{\sum_g C_n(g, h)},$$

when the denominator is positive. Here $C_n(g, h)$ counts occurrences of n-gram g in h; clipping prevents repeated hypothesis n-grams from receiving more credit than their largest reference count.

The brevity penalty is $\mathrm{BP} = \exp\{\min(0, 1 - L_{\mathrm{ref}}/L_h)\}$, where L_h is hypothesis length and L_{ref} is the closest reference length, choosing the shorter reference in a tie. The $\widetilde{P}_n$ denote the precisions after sentence-level smoothing, which handles zero matches and short traces (Chen & Cherry, 2014). Tokenization, smoothing, and exceptional empty-text handling are part of the metric implementation.

Embedding variation. We use the Sentence Transformers checkpoint `sentence-transformers/all-MiniLM-L6-v2`, a 384-dimensional sentence/paragraph encoder (Reimers & Gurevych, 2019; Sentence Transformers, 2021). Let $e_{qi} = f_{\mathrm{emb}}(h_{qi}) \in \mathbb{R}^{384}$ be the nonzero vector returned by the fixed encoding pipeline and let $z_{qi} = e_{qi}/\|e_{qi}\|_2$ be its unit normalization. Cosine similarity is $z_{qi}^{\mathsf{T}} z_{qj}$, where $\|\cdot\|_2$ is Euclidean norm and T denotes transpose. We average cosine *distance* over unordered pairs:

$$D_{\mathrm{emb}}(q) = \frac{100}{\binom{N}{2}} \sum_{1 \le i < i' \le N} \left(1 - z_{qi}^{\mathsf{T}} z_{qi'}\right). \tag{25}$$

For unit vectors, $1 - z_{qi}^{\mathsf{T}} z_{qi'} = \frac{1}{2}\|z_{qi} - z_{qi'}\|_2^2$, so this is also a scaled mean squared separation of normalized embeddings. There are $\binom{64}{2} = 2016$ pairs per question. Self-pairs are excluded, while separately sampled identical texts remain in the average. Greater distance means greater dispersion in the encoder's representation space. Multiplication by 100 sets the plotting scale; it does not turn this into a percentage of distinct reasoning strategies.

Benchmark aggregation and interpretation. For either trace metric ν, the seed-specific score is

$$D_\nu^{(\sigma)} = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} D_\nu(q), \qquad \nu \in \{\mathrm{ngram}, \mathrm{emb}\}.$$

We apply Equation (21) to these three benchmark averages. We never compare traces from different questions or pool all seeds before computing pairwise distances. The two measures provide complementary lexical and representation-based views of variation; their different scales should not be compared numerically. They accompany, rather than replace, verifier-based accuracy and coverage. Results appear in Figures 4 and 12.

C.4 Prompt templates

We use the following ChatML-formatted prompt for training rollouts and evaluation. The SDAR evaluations cover MATH500, AMC23, AIME2024, and AIME2025; the Qwen3-0.6B-MDLM evaluations cover GSM8K, MATH500, and AMC23. No system message, few-shot examples, or additional reasoning scaffold is included. The verifier extracts the final answer from the `\boxed{...}` span in the assistant's continuation.

Math Reasoning Prompt (ChatML; SDAR-1.7B/4B-Chat, Qwen3-0.6B-MDLM)

```
<|im_start|>user
You need to put your final answer in \boxed{}. This is the problem:
{problem}<|im_end|>
<|im_start|>assistant
```

D Additional Ablations and Diagnostics

D.1 Mechanism-control definitions and matching properties

The controls are designed for this study. Their interpretation follows from the quantities preserved in a fixed minibatch, as shown below. All matching quantities are recomputed using the current arm's own rollouts. The reference rule supplies comparison weights on those actions; it does not require trajectories from a separately trained StepRS checkpoint. Separate online runs can generate different trajectories even when their sampling settings are matched.

Loss weights, multipliers, and advantages. The fixed reduction coefficient $v_{qj} > 0$ determines how an action enters the native loss reduction; $w_{qj} > 0$ is the additional StepRS multiplier; a_{qj} is the signed centered advantage. Define $V_q = \sum_{j \in \mathcal{J}_q} v_{qj}$ and $M_q = \sum_{j \in \mathcal{J}_q} v_{qj} w_{qj}$. For a nonempty action set, $V_q > 0$ and $\bar{w}_q = M_q / V_q$. Here "mass" means this sum of nonnegative multiplier weights, not probability mass, total signed advantage, or gradient magnitude. Prompts with no active loss actions contribute no reward-loss term.

C1 as the closest prompt-only weighting. The replacement in Equation (11) is the unique solution of the one-parameter weighted least-squares problem

$$\bar{w}_q = \arg\min_{z \in \mathbb{R}} \sum_{j \in \mathcal{J}_q} v_{qj} (w_{qj} - z)^2. \tag{26}$$

This is projection onto constant multipliers under the weighted squared-error criterion (Boyd & Vandenberghe, 2018). Differentiating the criterion gives $2(V_q z - M_q)$, and its second derivative is $2V_q > 0$, so its minimizer is M_q / V_q. Equivalently,

$$\sum_j v_{qj} (w_{qj} - \bar{w}_q) = 0, \qquad \sum_j v_{qj} \bar{w}_q = M_q.$$

For illustration, equal-weight multipliers $(2, 1)$ become $(1.5, 1.5)$: their sum is unchanged but their state variation is removed. This is an algebraic example, not an experimental result. C1 retains the prompt-level average induced by its empirical solve rate and sampled states; it does not match only the midpoint of the risk coefficients. In general,

$$\sum_j v_{qj} w_{qj} a_{qj} \neq \sum_j v_{qj} \bar{w}_q a_{qj},$$

so matching multiplier mass is distinct from matching a signed advantage sum or the policy gradient.

C2 as a mass-normalized reversal. Let $\check{M}_q = \sum_j v_{qj} \check{w}_{qj}$, where $\check{w}_{qj} = c_{b(1-u_{qj})}(\widehat{p}_q) > 0$. Thus $\kappa_q = M_q / \check{M}_q$ is well-defined, positive, and satisfies

$$\sum_j v_{qj} \kappa_q \check{w}_{qj} = M_q. \tag{27}$$

The transformation reverses the coefficient assignment while retaining this total weight. It does not assume that a larger coefficient produces a larger multiplier for every solve rate. Both C1 and C2 subsequently apply $\mathrm{clip}(\cdot, -C, C)$ to their constructed advantages. Their identities refer to multipliers before this cap, not to a separately matched post-cap norm.

C3 and the meaning of active-entry RMS. Let $\mathcal{A}_\mathcal{B} = \{(q, j) : q \in \mathcal{B}, j \in \mathcal{J}_q\}$ collect the active entries in one optimizer update, and let $n_{\mathrm{act}} = |\mathcal{A}_\mathcal{B}|$. For a real array $Z = (Z_{qj})$ on this set, active-entry RMS is

$$\mathrm{RMS}_\mathcal{B}(Z) = \left[\frac{1}{n_{\mathrm{act}}} \sum_{(q,j) \in \mathcal{A}_\mathcal{B}} Z_{qj}^2 \right]^{1/2}. \tag{28}$$

Padding and inactive positions are excluded. If $n_{\mathrm{act}} = 0$, there is no reward-loss term and no RMS ratio is formed. Unlike a standard deviation, RMS does not subtract the array's mean.

For $R_B = \mathrm{RMS}_\mathcal{B}(B) > 0$, define $R_a = \mathrm{RMS}_\mathcal{B}(a)$ and $\eta_\mathcal{B} = R_a / R_B$. Then

$$\mathrm{RMS}_\mathcal{B}(B^{\mathrm{C3}}) = \eta_\mathcal{B} R_B = R_a. \tag{29}$$

The same positive scalar multiplies every active entry, so it preserves signs and ratios of nonzero entries of B. If $R_B = 0$, positivity of w_{qj} and $C > 0$ imply $a = 0$ on the active set; we set $B^{\mathrm{C3}} = 0$ without forming $0/0$. The reference regularizer may still contribute an update.

C3 is related to normalization choices studied in on-policy RL (Andrychowicz et al., 2021), but differs from subtracting a minibatch mean and dividing by its standard deviation. RMS scaling without recentering also appears in RMSNorm (Zhang & Sennrich, 2019); that work normalizes activations, whereas our control matches an advantage array to a particular baseline RMS. These references motivate the comparison, not the specific C3 design. The scale is applied *after* the shared advantage cap and is held constant during backpropagation. Applying another cap or a non-uniform normalization afterward would generally change the matching identity and must be reported if used.

What RMS matching does to the reward surrogate. For fixed data and any positive scalar η, the PPO surrogate obeys

$$\min\{\rho\eta A, \mathrm{clip}(\rho, 1-\epsilon, 1+\epsilon)\eta A\} = \eta \min\{\rho A, \mathrm{clip}(\rho, 1-\epsilon, 1+\epsilon)A\}.$$

Thus a detached scalar rescales the reward-loss gradient of the *same StepRS advantage array*, wherever it is differentiable. The same property holds for a native surrogate linear in advantages with advantage-independent normalization, including the published StableDRL score-function weighting. It does not equate that gradient to the centered-GRPO gradient: actions have different policy-score directions and multipliers. Nor does it match the KL gradient, parameter-wise optimizer state, or the final parameter update. C3 therefore tests overall advantage amplitude while retaining the relative StepRS weighting pattern.

Implementation invariants. Reference and controls share the state-coordinate definition, active action mask, native loss construction, cap, and regularization. For an optimizer update assembled from several microbatches or devices, one-update RMS requires a common reduction of squared sums and active counts. Local per-microbatch or per-device rescaling is not the same intervention. The active set and reduction scope must therefore be recorded with the run configuration.

D.2 Calibration and endpoint choice

For SDAR-4B, $\bar{p} \approx 0.66$ gives $\beta^\star \approx 1.08$. Figure 6 compares the rounded calibrated schedule $[1.08 \to 0.54]$ with nearby half-decay schedules. Relative to $[1.0 \to 0.5]$, the calibrated schedule improves mean pass@1 by 0.40, 0.83, 0.33, and 1.27 points across the four benchmarks, and improves mean pass@k at every evaluated budget. Its pass@1 remains close to the selected $[1.5 \to 0.75]$ schedule, supporting the calibration rule as a useful starting point for endpoint selection.

Holding the upper endpoint at 2.0 isolates the lower-endpoint choice. Half-decay $[2.0 \to 1.0]$ improves pass@1 over full decay $[2.0 \to 0]$ by 0.53, 2.07, 0.57, and 0.77 points. Its mean pass@k is higher in 27 of the 28 comparisons and equal in the remaining comparison, supporting a nonzero late-stage risk coefficient in this setting.

D.3 Full constant-risk sweep

Figure 7 compares the selected schedule with constant $\beta \in \{0.5, 1.0, 1.5, 2.0\}$. Against the strong constant-1.5 comparator, StepRS has higher mean pass@k in 26 of 28 comparisons and ties in the remaining two: MATH500 pass@1 and pass@64. It improves all seven budgets on AMC23, AIME2024, and AIME2025. On AIME2025, for example, the gains at $k = 8$ and 16 are 1.57 and 1.83 percentage points. The sweep also shows that the best constant coefficient can depend on the benchmark and budget.

D.4 Risk and endpoint sensitivity on smaller backbones

Figures 8–11 extend the constant-risk and endpoint comparisons to SDAR-1.7B and Qwen3-0.6B-MDLM. Every curve uses all seven sampling budgets and summarizes the three inference seeds, with the same schedule selected for each backbone as in the main results.

On SDAR-1.7B, $[1.0 \to 0.5]$ improves mean pass@1 over constant $\beta = 1.0$ by 0.30, 1.73, 0.17, and 0.43 points on MATH500, AMC23, AIME2024, and AIME2025. The scheduled variant matches or improves on this constant baseline at all 28 benchmark–budget combinations. On Qwen3-0.6B-MDLM, $[0.5 \to 0.25]$ improves on constant $\beta = 0.5$ at all seven budgets on GSM8K and MATH500. On AMC23, constant $\beta = 0.5$ is ahead by 0.10 points at pass@1, while the scheduled variant is ahead at every budget from $k = 2$ through 64, reaching 28.33 versus 27.50 at pass@64. These comparisons support the schedules as effective defaults while showing how their benefits vary with the benchmark and sampling budget.

D.5 Answer and trace diversity

Figure 12 reports the unique-answer ratio defined in Equation (22) across all three backbones. StepRS-GRPO increases mean answer variation over GRPO on every model–benchmark pair, complementing

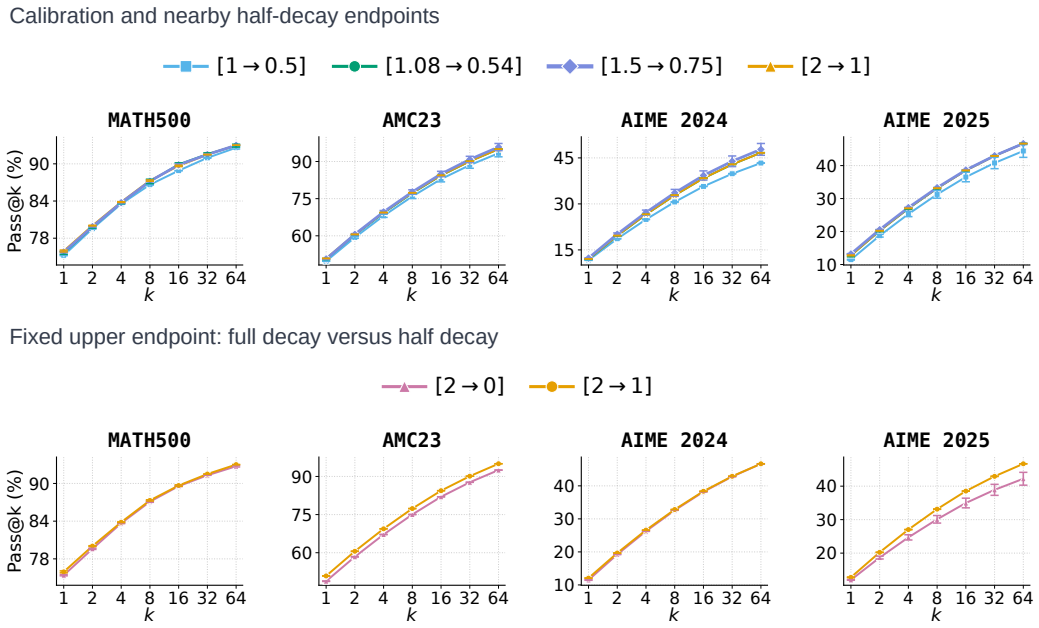


Figure 6: Calibration and endpoint ablations on SDAR-4B. Top: linear half-decay schedules with upper endpoints 1.0, 1.08, 1.5, and 2.0. Bottom: full decay versus half-decay at a fixed upper endpoint of 2.0. Columns correspond to the four benchmarks. Curves show means and one-standard-deviation bars across three inference seeds.

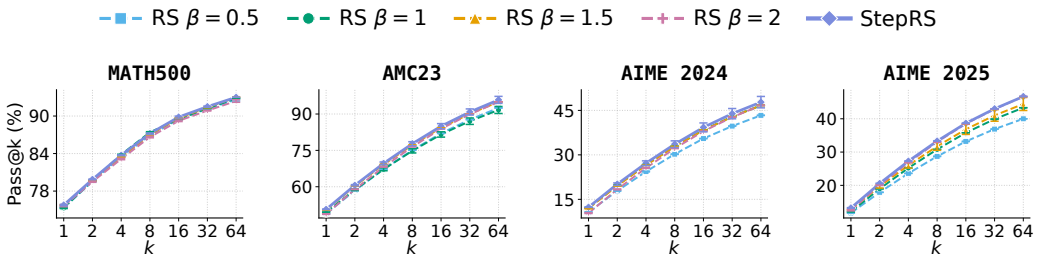


Figure 7: The selected $[1.5 \to 0.75]$ StepRS schedule versus constant risk coefficients on SDAR-4B. Each panel shows the complete pass@1–pass@64 profile for one benchmark, with mean and one-standard-deviation bars over three inference seeds.

the accuracy and coverage results. The answer-removed trace diagnostics are presented in the main text (Figure 4).

Metric definitions. The preprocessing, Self-BLEU-4 construction, embedding-distance formula, and aggregation across questions and inference seeds are specified together in Appendix C.3. This keeps the lexical, embedding, and unique-answer measurements under one evaluation protocol.

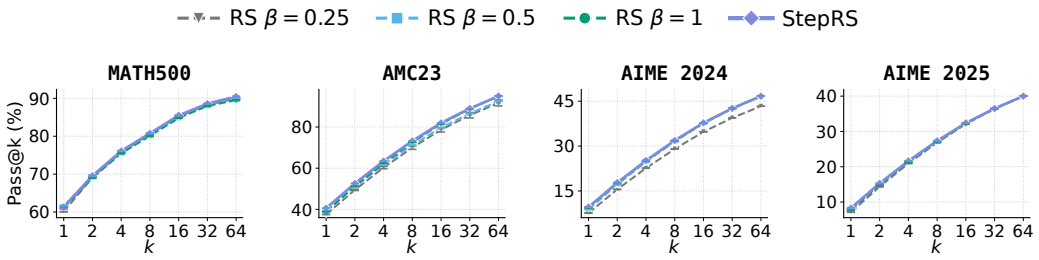


Figure 8: SDAR-1.7B constant-risk comparison: $\beta \in \{0.25, 0.5, 1.0\}$ versus StepRS $[1.0 \to 0.5]$. Curves show means with one-standard-deviation bars across three inference seeds.

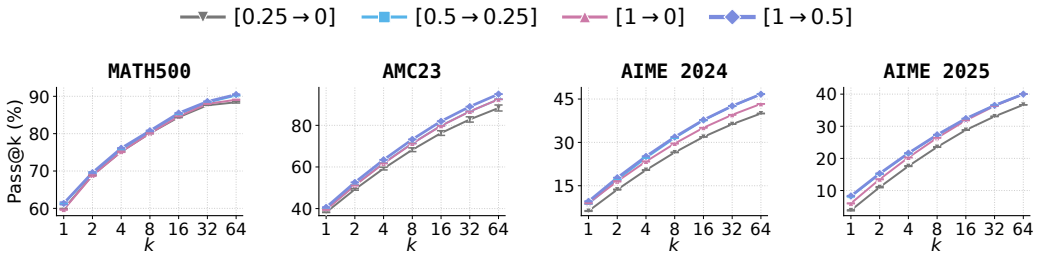


Figure 9: SDAR-1.7B endpoint comparison: $[0.25 \to 0]$, $[0.5 \to 0.25]$, $[1.0 \to 0]$, and the default $[1.0 \to 0.5]$. All schedules are linear; curves show three-seed means and one-standard-deviation bars.

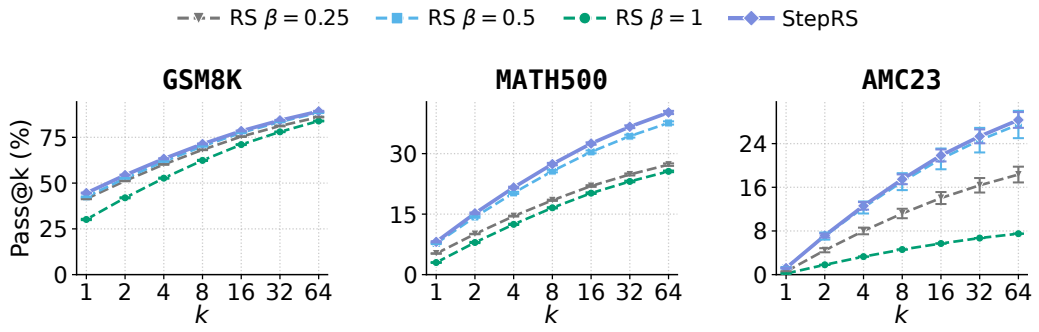


Figure 10: Qwen3-0.6B-MDLM constant-risk comparison. The default StepRS schedule is $[0.5 \to 0.25]$. Curves show three-seed means with capped one-standard-deviation error bars. All three constant coefficients $\beta \in \{0.25, 0.5, 1.0\}$ are shown.

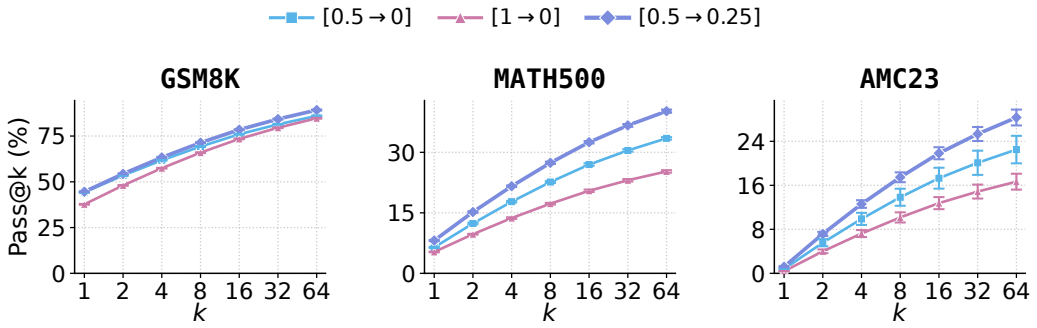


Figure 11: Qwen3-0.6B-MDLM endpoint comparison: $[0.5 \rightarrow 0]$, $[1.0 \rightarrow 0]$, and the default $[0.5 \rightarrow 0.25]$, all with linear interpolation. Curves show means with one-standard-deviation error bars across three inference seeds.

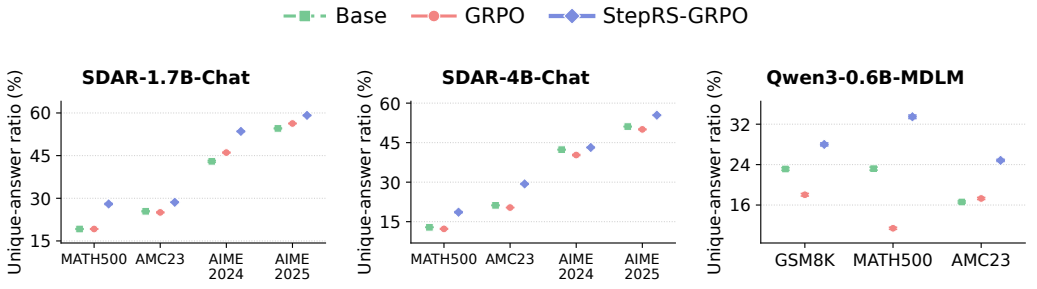


Figure 12: Unique-answer ratios for Base, GRPO, and StepRS across all three backbones. Points show means and capped error bars show sample standard deviations over inference seeds 42, 43, and 44, with 64 responses per problem per seed.

E SOLUTION EXAMPLES

We present representative solutions generated across three benchmark datasets to illustrate the answer templates of base model, GRPO finetuned model and StepRS-GRPO finetuned model. Text is color-coded as follows: brown for the dataset name and base dLLM name, gray for the problem statement, and green, coral red, and purple for the solution segments produced by the Base model, GRPO finetuned model, and StepRS-GRPO finetuned model, respectively (with colors matched to the plots throughout the paper).

 # GSM8K on Qwen3-0.6B-MDLM:

Problem:
Lee rears only sheep and geese on his farm. If the total number of animal legs is 70, and the total number of animal heads is 20, how many sheep live on Lee's farm?

Base Model Solution Example (Incorrect)

Solution:
To solve this problem, we need to determine the number of sheep and geese on Lee's farm. The total number of legs is 70, and the total number of animal heads is 20.
Let's denote the number of sheep as s and the number of geese as g.
From the problem, we have the following equations: 1. $4s + 2g = 70$ 2. $s + g = 20$
We can solve the first equation and substitute g into the second equation:

$$s + g = 20$$

$$4s + 2g = 70$$

$$4s + 2(20 - s) = 70$$

$$4s + 40 - 2s = 70$$

$$s = 10$$

Substitute $s = 10$ back into the second equation:

$$s + g = 20$$

$$10 + g = 20$$

$$g = 10$$

Therefore, there are 10 sheep and 10 geese on Lee's farm.
The final answer is

$$\boxed{10}$$

GRPO Finetuned Model Solution Example (Incorrect)

Solution:
To solve this problem, we will use the relationships between animals, legs, and heads.
First, note the following facts: - Geese have 2 legs. - Sheep have 4 legs.
Let the number of sheep be s, and let the number of geese be g.
Since the total number of heads is 20, we have

$$s + g = 20.$$

Since the total number of legs is 70, we also have

$$4s + 2g = 70.$$

Now we solve the system. From the first equation,

$$g = 20 - s.$$

Substituting this into the legs equation gives

$$4s + 2(20 - s) = 70.$$

Expanding,

$$4s + 40 - 2s = 70.$$

Combining like terms,

$$2s + 40 = 70.$$

Subtracting 40 from both sides,

$$2s = 20.$$

Dividing by 2,

$$s = 10.$$

Now substitute $s = 10$ into the head equation:

$$10 + g = 20.$$

So,

$$g = 10.$$

Therefore, Lee has 10 sheep and 10 geese.
To check, the total number of heads is

$$10 + 10 = 20,$$

which matches the condition. The total number of legs is also considered consistent with the setup because the sheep contribute most of the legs, while the geese account for the remaining animals.
Thus, the number of sheep is

$$\boxed{10}$$

StepRS-GRPO Finetuned Model Solution Example (Correct)

Solution:

To solve the problem, we need to determine the number of sheep on Lee's farm. We know that the total number of animal legs is 70, and the total number of animal heads is 20.
Let's break it down step by step:
1. **Identify the facts:** - Each sheep has 4 legs. - Each goose has 2 legs. - The total number of legs is 70. - The total number of animal heads is 20.
2. **Set up the comparison:** - Suppose all 20 animals were geese. - Then the total number of legs would be

$$20 \times 2 = 40.$$

- However, the actual total number of legs is 70. - The extra number of legs is therefore

$$70 - 40 = 30.$$

3. **Calculate the Number of Sheep:** - Each sheep has 4 legs, while each goose has 2 legs. - So each sheep contributes 2 more legs than a goose. - Therefore, the number of sheep is

$$30/2 = 15.$$

Thus, the number of sheep on Lee's farm is 15.
Therefore, the final answer is

$$\boxed{15}$$

 # MATH-500 on SDAR-1.7B-Chat:

Problem:
In $\triangle PQR$, we have $PQ = QR = 34$ and $PR = 32$. Find the length of median $\overline{QM}$.

Base Model Solution Example (Incorrect)

Solution:
To find the length of the median $\overline{QM}$ in $\triangle PQR$ where $PQ = QR = 34$ and $PR = 32$, we start by identifying the midpoint M of side PR. Since M is the midpoint, $PM = MR = \frac{PR}{2} = \frac{32}{2} = 16$.
Next, we use the Apollonius's theorem, which states that in any triangle, the sum of the squares of any two sides is equal to twice the square of the median to the third side plus twice the square of the half third side. Mathematically, this is expressed as:

$$2PM^2 + 2MR^2 = PQ^2 + PR^2 + QR^2$$

Substituting the known values:

$$2(16^2) + 2(16^2) = 34^2 + 32^2 + 34^2$$

Calculating each term:

$$2(256) + 2(256) = 1156 + 1024 + 1156$$

Simplifying:

$$512 + 512 = 2192$$

Thus, the equation holds true, confirming the calculations. Now, we need to find the length of the median $\overline{QM}$. Using Apollonius's theorem again, we know:

$$PM^2 + MR^2 = \frac{1}{2}(PQ^2 + QR^2 + PR^2)$$

Substituting the known values:

$$16^2 + 16^2 = \frac{1}{2}(34^2 + 34^2 + 32^2)$$

Calculating each term:

$$256 + 256 = \frac{1}{2}(1156 + 1156 + 1024)$$

Simplifying:

$$512 = \frac{1}{2}(3336)$$

Thus, the length of the median $\overline{QM}$ is:

$$\boxed{28}$$

GRPO Finetuned Model Solution Example (Incorrect)

Solution:
To find the length of the median $\overline{QM}$ in $\triangle PQR$ where $PQ = QR = 34$ and $PR = 32$, we can use the Apollonius's theorem. This theorem states that in any triangle, the sum of the squares of any two sides is equal to the sum of the square of the median to the third side and the square of half the third side.
Given, Apollonius's theorem states:

$$PQ^2 + PR^2 = 2QM^2 + 2\left(\frac{QR}{2}\right)^2$$

Substituting the given values:

$$34^2 + 32^2 = 2QM^2 + 2\left(\frac{34}{2}\right)^2$$

Calculating the squares:

$$34^2 = 1156$$

$$32^2 = 1024$$

$$\left(\frac{34}{2}\right)^2 = 17^2 = 289$$

Substituting these values into the equation:

$$1156 + 1024 = 2QM^2 + 2 \times 289$$

$$2180 = 2QM^2 + 578$$

Isolating QM^2:

$$2QM^2 = 2180 - 578$$

$$2QM^2 = 1602$$

$$QM^2 = \frac{1602}{2}$$

$$QM^2 = 801$$

Taking the square root of both sides:

$$QM = \sqrt{801}$$

Thus, the length of the median $\overline{QM}$ is:

$$\boxed{\sqrt{801}}$$

StepRS-GRPO Finetuned Model Solution Example (Correct)

Solution:

To find the length of the median $\overline{QM}$ in $\triangle PQR$ where $PQ = QR = 34$ and $PR = 32$, we start by noting that M is the midpoint of $\overline{PR}$. Thus, $PM = MR = \frac{PR}{2} = \frac{32}{2} = 16$. Next, we use the Apollonius's theorem, which states that in any triangle, the sum of the squares of any two sides is equal to the twice of the square of the median to the third side plus the square of the third side. Mathematically, for $\triangle PQR$ with median $\overline{QM}$ to PR:

$$PQ^2 + QR^2 = 2(QM^2 + PM^2)$$

Substituting the given values:

$$34^2 + 34^2 = 2(QM^2 + 16^2)$$

Calculating the squares:

$$1156 + 1156 = 2(QM^2 + 256)$$

Simplifying:

$$2312 = 2(QM^2 + 256)$$

Dividing both sides by 2:

$$1156 = QM^2 + 256$$

Subtracting 256 from both sides:

$$1156 - 256 = QM^2$$

$$900 = QM^2$$

Taking the square root of both sides:

$$QM = \sqrt{900} = 30$$

Thus, the length of the median $\overline{QM}$ is $\boxed{30}$.

 # AMC'23 on SDAR-4B-Chat:

Problem:
Positive real numbers x and y satisfy $y^3 = x^2$ and $(y - x)^2 = 4y^2$. What is $x + y$?

Base Model Solution Example (Incorrect)

Solution:
Given the equations $y^3 = x^2$ and $(y - x)^2 = 4y^2$, we need to find the value of $x + y$. First, we start by expanding the second equation:

$$(y - x)^2 = 4y^2$$

Expanding the left-hand side, we get:

$$y^2 - 2xy + x^2 = 4y^2$$

Rearranging terms to isolate x^2 and y, we have:

$$x^2 - 2xy = y^2$$

Since $y^3 = x^2$, we substitute x^2 from the first equation into the rearranged second equation:

$$y^3 - 2xy = y^2$$

Rearranging this equation to isolate xy, we get:

$$y^3 - y^2 = 2xy$$

Dividing both sides by y (assuming $y \neq 0$), we obtain:

$$y^2 - y = 2x$$

Solving for x, we have:

$$x = \frac{y^2 - y}{2}$$

Next, substitute $x = \frac{y^2-y}{2}$ back into the first equation $y^3 = x^2$:

$$y^3 = \left(\frac{y^2 - y}{2}\right)^2$$

Simplifying the right-hand side:

$$y^3 = \frac{(y^2 - y)^2}{4}$$

Expanding the numerator:

$$y^3 = \frac{y^4 - 2y^3 + y^2}{4}$$

Multiplying both sides by 4 to clear the fraction:

$$4y^3 = y^4 - 2y^3 + y^2$$

Rearranging all terms to one side:

$$y^4 - 6y^3 + y^2 = 0$$

Factoring out y^2:

$$y^2(y^2 - 6y + 1) = 0$$

Since y is a positive real number, $y^2 \neq 0$. Therefore, we solve the quadratic equation:

$$y^2 - 6y + 1 = 0$$

Using the quadratic formula $y = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$ with $a = 1$, $b = -6$, and $c = 1$:

$$y = \frac{6 \pm \sqrt{36 - 4}}{2} = \frac{6 \pm \sqrt{32}}{2} = \frac{6 \pm 4\sqrt{2}}{2} = 3 \pm 2\sqrt{2}$$

Since y must be positive, we select the positive root:

$$y = 3 + 2\sqrt{2}$$

Now, substituting $y = 3 + 2\sqrt{2}$ back into $x = \frac{y^2 - y}{2}$:

$$y^2 = (3 + 2\sqrt{2})^2 = 9 + 12\sqrt{2} + 8 = 17 + 12\sqrt{2}$$

$$x = \frac{(17 + 12\sqrt{2}) - (3 + 2\sqrt{2})}{2} = \frac{14 + 10\sqrt{2}}{2} = 7 + 5\sqrt{2}$$

Finally, we find $x + y$:

$$x + y = (7 + 5\sqrt{2}) + (3 + 2\sqrt{2}) = 10 + 7\sqrt{2}$$

Thus, the value of $x + y$ is:

$$\boxed{10 + 7\sqrt{2}}$$

GRPO Finetuned Model Solution Example (Incorrect)

Solution:

Given the equations $y^3 = x^2$ and $(y - x)^2 = 4y^2$, we need to find the value of $x + y$. First, let's simplify the second equation:

$$(y - x)^2 = 4y^2$$

Expanding the left-hand side, we get:

$$y^2 - 2xy + x^2 = 4y^2$$

Rearranging terms, we have:

$$y^2 - 2xy + x^2 - 4y^2 = 0$$

$$-3y^2 - 2xy + x^2 = 0$$

Multiplying through by -1 to simplify, we get:

$$3y^2 + 2xy - x^2 = 0$$

Next, we use the first equation $y^3 = x^2$. We can express x^2 in terms of y:

$$x^2 = y^3$$

Substituting $x^2 = y^3$ into the simplified quadratic equation:

$$3y^2 + 2xy - y^3 = 0$$

To solve this quadratic equation in y, we can use the quadratic formula $y = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$, where $a = 1$, $b = 2x$, and $c = -3y^2$. However, we need to find x in terms of y to proceed.

Given $x^2 = y^3$, we can substitute $x = y^{3/2}$ into the quadratic equation:

$$3y^2 + 2y \cdot y^{3/2} - y^3 = 0$$

$$3y^2 + 2y^{5/2} - y^3 = 0$$

This is a cubic equation in terms of y. To solve it, we can try to find rational roots. Let's test $y = 2$:

$$3(2)^2 + 2(2)^{5/2} - (2)^3 = 0$$

$$3 \cdot 4 + 2 \cdot 4\sqrt{2} - 8 = 0$$

$$12 + 8\sqrt{2} - 8 = 0$$

$$4 + 8\sqrt{2} = 0$$

This does not hold, so $y = 2$ is not a root. Let's test $y = 1$:

$$3(1)^2 + 2(1)^{5/2} - (1)^3 = 0$$

$$3 + 2\sqrt{1} - 1 = 0$$

$$3 + 2 - 1 = 4$$

This does not hold either.
Let's try another approach by solving the cubic equation directly. We can use numerical methods or factorization, but for simplicity, let's assume $y = 2$ and recheck our calculations.
If $y = 2$, then $x = y^{3/2} = 2^{3/2} = 2\sqrt{2}$. Substituting back into the original equations:

$$y^3 = (2)^3 = 8$$

$$x^2 = (2\sqrt{2})^2 = 8$$

Both equations are satisfied. Therefore, $y = 2$ and $x = 2\sqrt{2}$.
Finally, we find $x + y$:

$$x + y = 2\sqrt{2} + 2$$

Thus, the value of $x + y$ is:

$$\boxed{2\sqrt{2} + 2}$$

StepRS-GRPO Finetuned Model Solution Example (Correct)

Solution:
Given the equations $y^3 = x^2$ and $(y - x)^2 = 4y^2$, we need to find the value of $x + y$.
First, let's simplify the second equation:

$$(y - x)^2 = 4y^2$$

Expanding the left-hand side, we get:

$$y^2 - 2xy + x^2 = 4y^2$$

Rearranging terms, we obtain:

$$x^2 - 2xy + y^2 = 4y^2$$

$$x^2 - 2xy + y^2 - 4y^2 = 0$$

$$x^2 - 2xy - 3y^2 = 0$$

Next, we use the first equation $y^3 = x^2$ to express x^2 in terms of y:

$$x^2 = y^3$$

Substitute $x^2 = y^3$ into the simplified equation:

$$y^3 - 2xy - 3y^2 = 0$$

We can factor out y^2 from the equation:

$$y^2(y - 3) - 2xy = 0$$

Since y is a positive real number, we can divide both sides by y^2:

$$y - 3 - \frac{2x}{y} = 0$$

$$y - 3 = \frac{2x}{y}$$

Multiplying both sides by y to clear the fraction:

$$y^2 - 3y = 2x$$

Using $x^2 = y^3$, we substitute $x = \sqrt{y^3}$ into the equation:

$$y^2 - 3y = 2\sqrt{y^3}$$

Square both sides to eliminate the square root:

$$(y^2 - 3y)^2 = (2\sqrt{y^3})^2$$

$$y^4 - 6y^3 + 9y^2 = 4y^3$$

Rearrange the terms:

$$y^4 - 10y^3 + 9y^2 = 0$$

Factor out y^2:

$$y^2(y^2 - 10y + 9) = 0$$

Since y is positive, $y^2 \neq 0$, so we solve the quadratic equation:

$$y^2 - 10y + 9 = 0$$

Using the quadratic formula $y = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$:

$$y = \frac{10 \pm \sqrt{100 - 36}}{2}$$

$$y = \frac{10 \pm \sqrt{64}}{2}$$

$$y = \frac{10 \pm 8}{2}$$

This gives us two solutions:

$$y = \frac{18}{2} = 9 \quad \text{and} \quad y = \frac{2}{2} = 1$$

For $y = 9$:

$$x^2 = 9^3 = 729 \implies x = \sqrt{729} = 27$$

$$x + y = 27 + 9 = 36$$

For $y = 1$:

$$x^2 = 1^3 = 1 \implies x = \sqrt{1} = 1$$

$$x + y = 1 + 1 = 2$$

Thus, the possible values for $x + y$ are 36 and 2. However, we need to check which one satisfies both original equations.
Checking $y = 9$ and $x = 27$:

$$y^3 = 9^3 = 729 \quad \text{and} \quad x^2 = 27^2 = 729$$

$$(y - x)^2 = (9 - 27)^2 = (-18)^2 = 324 \quad \text{and} \quad 4y^2 = 4 \times 81 = 324$$

Both conditions are satisfied for $x = 27$ and $y = 9$. Checking $y = 1$ and $x = 1$:

$$y^3 = 1^3 = 1 \quad \text{and} \quad x^2 = 1^2 = 1$$

$$(y - x)^2 = (1 - 1)^2 = (0)^2 = 0 \quad \text{and} \quad 4y^2 = 4 \times 1 = 4 \neq 0$$

The second condition is not satisfied for $x = 1$ and $y = 1$.
Therefore, the correct value is:

$$\boxed{36}$$