AN IMPERFECT VERIFIER IS GOOD ENOUGH: LEARNING WITH NOISY REWARDS

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement Learning with Verifiable Rewards (RLVR) is widely used for post-training Large Language Models, but practical verifiers can make errors. We study how the rate and structure of reward noise affect RLVR in code generation, with a preliminary scientific-reasoning check. In multi-seed Qwen3 8B experiments on MBPP, mean validation reward over two fixed late evaluations is within 1 percentage point of the clean baseline at the tested resampled group-rollout noise rates through 20%, and within about 2 points at 30%. Confidence intervals allow larger losses; these point estimates do not establish a general tolerance threshold. We also examine full-program pass@k, four controlled noise structures, two model-based verifiers, and policy models from three families spanning 4B–9B parameters. We derive conditional advantage distributions for symmetric and asymmetric group noise, including retained format penalties, and show why clipping limits simple gradient-scaling arguments. The analysis identifies information preserved by whole-group corruption and limits on interpreting our asymmetric sweep as a precision–recall comparison. Overall, the results indicate that imperfect verification can support effective RLVR in the tested settings, while aggregate error rates alone do not characterize the learning signal.

1 INTRODUCTION

Reinforcement learning (RL) has grown in popularity as a post-training method to improve large language models (LLMs) in various domains, particularly after the release of DeepSeek-R1 (DeepSeek-AI et al., 2025), which demonstrated that Reinforcement Learning with Verifiable Rewards (RLVR) and Group Relative Policy Optimization (GRPO) (Shao et al., 2024) can produce a frontier-level model at relatively low cost. Early efforts focused on verifiable domains such as math and coding, where deterministic checks provide the reward signal. Since then, RLVR has been extended to semi-verifiable domains such as finance and law, where rubrics and LLM-as-a-Judge provide the reward signal (Viswanathan et al., 2025; Zhou et al., 2025).

RLVR is motivated by the ideal of a *perfect* verifier: a deterministic oracle that consistently rewards good outputs and penalizes bad ones. In practice, though, such a verifier does not exist. While some datasets, such as GPQA (answers are multiple-choice) and AIME (answers are integers between 000 and 999), admit highly accurate verifiers, verification can be challenging and error-prone even in ostensibly verifiable domains such as mathematics. For example, a string-equality-based verifier might incorrectly reject $\frac{1}{e^2} - \frac{1}{6}$ as an answer with respect to a ground truth $\frac{6-e^2}{6e^2}$, despite mathematical equivalence (Xu et al., 2025b; Huang et al., 2025). This problem is exacerbated in semi-verifiable domains where non-deterministic model-based judges are used (Tan et al., 2025; Pan et al., 2026).

Verifier accuracy has improved through domain-specific techniques, such as Math-Verify (Hugging Face, 2025b), and fine-tuned judges (Zhu et al., 2025a; He et al., 2025). A practical question remains: *under which error structures and training conditions can an imperfect verifier support effective RLVR?* This question also matters for LLM-as-a-Judge and Agent-as-a-Judge (Vidgen et al., 2026; Zhuge et al., 2025), where errors may depend systematically on the generated response.

To study these questions in a controlled setting, we focus on RLVR for coding, and we measure the impact of verifier noise on model training. We focus on coding for two reasons: because the reward structure mirrors rubric-based rewards (a set of unit tests is analogous to a set of text-based rubric

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.

criteria), and because the verifiers can be highly accurate, providing a noise-free baseline as a point of comparison for controlled and realistic noise.

Our multi-seed Qwen3 8B experiments on MBPP show small observed losses under resampled group-rollout noise: the mean over two fixed late evaluations is within 1 percentage point of the clean baseline through 20% noise, and within about 2 points at 30%. The confidence intervals allow larger losses, so these point estimates do not establish a one-point tolerance guarantee. We also compare noise structures and two model-based verifiers, and include a preliminary scientific-reasoning experiment. Our analysis explains why whole-group corruption preserves useful advantage information in expectation and why our asymmetric design cannot isolate a general precision–recall trade-off.

2 Related work

Reinforcement learning with verifiable rewards. Silver & Sutton (2025) articulated a vision for scaling RL to language agents, crystallizing an emerging trend of increasing model abilities through RL (OpenAI, 2024; Zeng et al., 2025). A key example of this is RLVR, which was popularized by DeepSeek-R1 (DeepSeek-AI et al., 2025). They showed that GRPO with outcome-based rewards on math and code tasks can produce strong reasoning LLMs while having less risk of reward hacking than learned reward models (Gao et al., 2023). Verification can also be applied at the process level rather than just the outcome to provide stronger learning signals (Lightman et al., 2023; Hübotter et al., 2026). Our work takes RLVR as a given (see Appendix E or DeepSeek-AI et al. (2025) for background) and investigates what happens when the outcome-based reward signal is imperfect.

LLM-as-a-Judge / model-based verifiers. Using models as evaluators has become standard practice for evaluation and training (Bai et al., 2022; Gu et al., 2026). Zheng et al. (2023) introduced the LLM-as-a-Judge paradigm for evaluation, and Lambert et al. (2025) created a benchmark to rank judges' alignment with human preferences. Liu et al. (2026) show that reasoning models are better verifiers than non-reasoning models. For training, rubric-based approaches use model-based verifiers to provide reward signals in domains without deterministic verifiers (Gunjal et al., 2025; He et al., 2025).

Model-based verifiers have well-documented failure modes, including positional bias (Thakur et al., 2025) and sensitivity to prompt phrasing (Chen et al., 2024; Shankar et al., 2024), meaning that model-based verifier noise is neither uniform nor independent.

Reinforcement learning with noisy rewards. The problem of learning under corrupted/noisy rewards is not new. Everitt et al. (2017) formalize the corrupted reward channel as a POMDP (Kaelbling et al., 1998) and identify conditions under which optimal behavior is still recoverable. Wang et al. (2020) propose estimators robust to reward perturbation in deep RL. In the preference-based setting, Gong et al. (2025) study noisy feedback and Li et al. (2026) analyze feature-dependent noise. Separately, Fortunato et al. (2018) add parametric noise directly to network weights to drive exploration.

Noisy rewards in RLVR. Rad et al. (2026) analyze reward noise in GRPO and validate their analysis on Qwen2.5 3B for code generation. Their prediction that positively informative noise can primarily affect learning speed is consistent with our observations; our study adds comparisons across noise structures and model families at finite training budgets. Cai et al. (2025) study asymmetric synthetic and model-based verifier noise in mathematical reasoning and evaluate reward corrections. Differences in task and evaluation protocol limit direct numerical comparisons with our results (Appendix M). Shao et al. (2025) study spurious rewards in RLVR for math, finding that Qwen models can learn from incorrect reward signals. Zhu & Kang (2026) follow up with a rebuttal saying that noise does hurt model performance. Chen et al. (2026) revisit spurious rewards through the lens of exploration vs. exploitation, and Xu et al. (2025a) propose a two-stage entropy-based approach for noise-tolerant RLVR training.

These studies cover different corruption mechanisms and noise levels. We focus on observed performance under specified noise structures. Mansouri et al. (2025) model Bernoulli reward

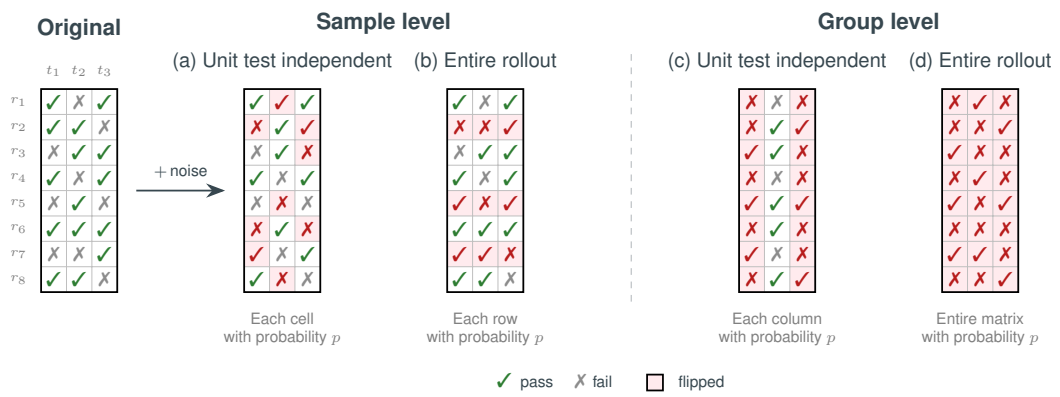

Figure 1: The four controlled noise injection modes. Rows are rollouts (r_1–r_8), columns are unit tests (t_1–t_3). Red cells indicate flipped outcomes. **(a)** Each cell flipped independently. **(b)** Entire rows flipped. **(c)** Entire columns flipped. **(d)** Entire matrix flipped.

corruption and derive a gradient correction. We evaluate uncorrected training; without a correction baseline, our experiments do not determine when such methods would improve performance.

3 METHODOLOGY

We begin with a noise-free baseline of RLVR in the coding domain, where the verifier executes the supplied unit tests. We check that the ground truth solutions pass all tests. Throughout, "error-free" refers to these test outcomes; a finite test suite does not establish program correctness on every input.

Next, we train models using noisy verifiers, evaluating the trained model against the held-out validation split *using the error-free verifier*. For the verifier used during training, we vary noise rates as well as noise *type*: we study a more controlled setting of specific noise patterns as well as more realistic noise distributions by using a model-based verifier.

3.1 CONTROLLED NOISE FOR RLVR

In the code-generation setting we study, each prompt has T unit tests, and each training step generates a group of G rollouts per prompt. Test outcomes are represented by a $G \times T$ binary matrix $\mathbf{M}$, where $M_{i,j} = 1$ if rollout i passes test j; the test-based reward is the row mean. Execution failure receives a clean reward of 0. Missing a fenced Python code block instead receives a format penalty of -0.25, which is retained unchanged under corruption and is excluded from the test-outcome model below.

We inject noise by flipping entries of $\mathbf{M}$ with probability p, independently of whether the original value is pass or fail. This bitflipping means the false positive rate and false negative rate are both equal to p. We identify two orthogonal axes that determine the *structure* of the noise, yielding four modes (illustrated in Figure 1):

- **(a) Sample × unit test.** Each cell $M_{i,j}$ is flipped independently with probability p. This noise is the most fine-grained; every test outcome for every rollout has an independent chance of corruption.
- **(b) Sample × rollout.** Each row (rollout) is selected with probability p, and all test outcomes in that row are flipped. This introduces perfectly correlated errors across tests within a rollout.
- **(c) Group × unit test.** Each column (test index) is selected with probability p, and that test's outcome is flipped for *all* rollouts in the group. This introduces perfectly correlated errors across rollouts for a test within the sampled group.
- **(d) Group × rollout.** The entire matrix is flipped with probability p. When triggered, every test outcome for every rollout is inverted; otherwise, the matrix is left intact. This noise is the coarsest; it either corrupts everything or nothing.

Noise is resampled independently on each encounter with a prompt across training epochs. The main sweep uses equal false-negative and false-positive rates; an exploratory asymmetric sweep appears in Appendix C. These controlled experiments do not model persistent verifier blind spots. Appendix F derives the group-noise advantage distributions, including retained format penalties and the limits of gradient identities under clipping.

3.2 Model-based verifier

Controlled corruption specifies error rates and correlations but does not capture all verifier failure modes. We therefore also replace the unit-test executor with a model-based verifier that predicts whether each test would pass given the generated code. It receives the code and unit tests as assert statements and returns binary judgments, aggregated into a per-rollout reward (Appendix D). Comparing two verifier models changes both the rate and structure of their errors; it is not an intervention on error rate alone.

This setup resembles rubric-based post-training (Gunjal et al., 2025; He et al., 2025), where a judge grades individual criteria and aggregates them into a score. Here, executing the tests provides reference labels for computing verifier accuracy, precision, recall, and F1 throughout training. These metrics describe each judge on the evolving outputs of its own trained policy; they do not compare judges on a shared, fixed response distribution. Predicting test execution also need not reproduce the error structure of rubric judgments in other domains.

4 Experimental setup

Dataset. Our experiments focus on the Mostly Basic Python Problems (MBPP) dataset (Austin et al., 2021). Each problem provides a natural-language description and three unit tests. We train on the standard MBPP training split (374 problems) and evaluate on the validation split (90 problems) using a maximum response length of 8192 tokens. We report mean unit-test reward, including format penalties, and pass@k, where a successful program must pass all three tests (Appendix C).

Models and compute. We use an internal training framework built on SLIME (Zhu et al., 2025b). For our main experiments, we focus on Qwen3 8B (Yang et al., 2025) and GLM4 9B (`GLM-Z1-9B-0414`) (Team GLM et al., 2024), with ablations using Llama 3.1 8B and Qwen3 4B. Training uses Group Relative Policy Optimization (GRPO) (Shao et al., 2024); full hyperparameters are in Appendix B.

Each training run requires approximately 64 GPU-hours on H100 GPUs. Multi-seed experiments are therefore limited to the most important configurations to manage compute costs. Unless otherwise noted, results are from a single seed. The expanded Qwen3 8B group-rollout sweep uses 3–6 seeds per setting, including 4 baseline seeds; pass@k uses 2–3 seeds. Other configurations retain the seed counts shown in Table 1.

5 Results

We first report the group-rollout noise sweep, then compare the four controlled noise types at p=0.10 and the two model-based verifiers. The comparisons concern the recorded training budgets and validation problems.

5.1 Observed tolerance to group-rollout noise

The recorded group-rollout sweeps cover noise rates up to p=0.50. For the expanded Qwen3 8B cohort, the mean of evaluation steps 239 and 259 (Table 8) is within 1 percentage point of the clean baseline (0.890) at the tested rates p=0.05–0.20, with observed losses of 1.4 points at p=0.25 and 2.0 points at p=0.30. Uncertainty is wider than the one-point margin: the 95% interval for the difference at p=0.20 is $[-2.9, +1.8]$ points, and at p=0.05 it is $[-4.3, +2.7]$. Peak scores show a similar pattern (Figure 2), but include unequal checkpoint-selection opportunities. Performance drops substantially at p=0.40 and above. At p=0.50, the test-outcome signal is uninformative in expectation before normalization, but the unchanged format penalties still provide a signal.

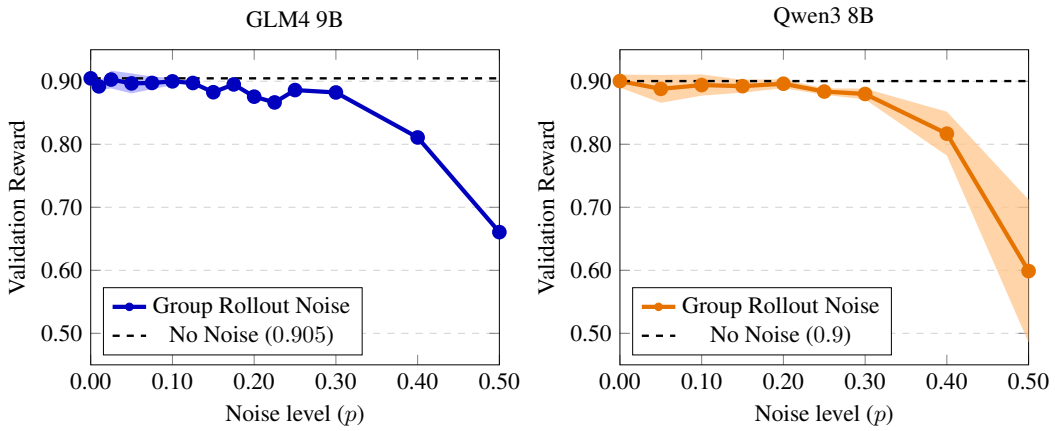

Figure 2: Best validation reward across noise levels for group rollout noise. Shaded regions indicate ± 1 standard deviation across seeds. Qwen3 8B uses the expanded 3–6-seed results, including later checkpoints where available; GLM4 9B uses recovered peaks through rollout step 259. These peak results include different run lengths; Figure 3 compares a common rollout step.

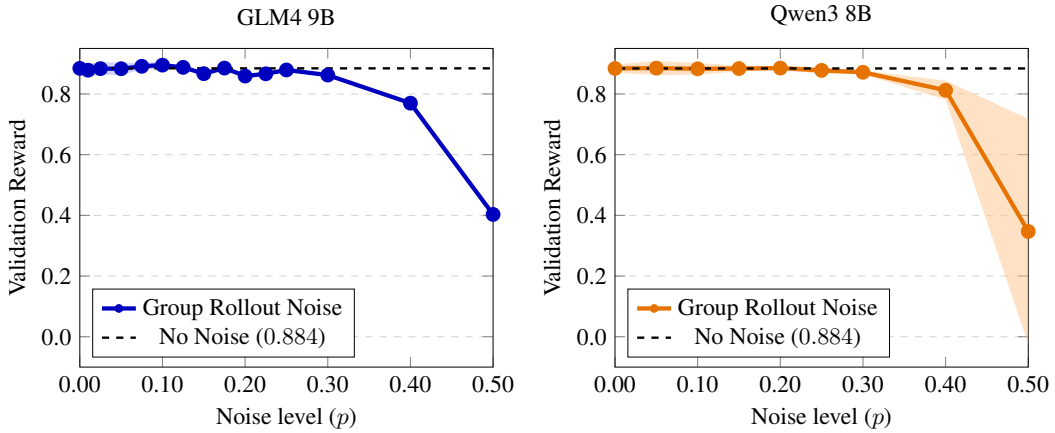

Figure 3: Validation reward at the common evaluation checkpoint, rollout step 259, for the recovered group-rollout runs. Shaded regions indicate ± 1 sample standard deviation where multiple seeds are available. Qwen3 8B uses the expanded 3–6-seed cohort, including 4 baseline seeds; GLM4 9B uses 3 through p=0.10 and 1 above.

Tables 5 and 6 report the expanded results. For peak unit-test rewards, two-sided independent equal-variance t-tests against the clean baseline give unadjusted p-values of 0.309 at 15% noise, 0.028 at 25%, and 0.023 at 30%; a nonsignificant result does not establish equivalence. Peak pass@1 is 0.816 at 15% noise versus 0.814 for its two-seed baseline. At 40%, pass@1 falls to 0.756 even though pass@16 matches the baseline, illustrating that tolerance depends on the evaluation metric. We complement peak scores with final-checkpoint results and training curves (Figure 3 and appendix I).

Figure 3 shows that low to moderate noise can remain close to the clean baseline at a common checkpoint. At rollout step 259, GLM4 9B with group-level entire-rollout noise at p=0.10 achieves 0.895, compared to 0.884 for the clean baseline. Qwen3 8B reaches 0.883 versus 0.884 without noise. Averaging the last two common evaluations gives 0.884 at 10% noise and 0.890 for the Qwen3 8B baseline (Table 8).

These descriptive comparisons do not establish that noise improves generalization or prevents overfitting. The seed intervals concern a fixed set of 90 validation problems, and no untouched test-set or problem-resampling analysis is available. We discuss the information retained by group-rollout corruption in Section 6.2.

Table 1: Noise-type comparison runs on MBPP, evaluated against ground-truth unit tests. Best and final denote the highest and last checkpoint validation reward; for the recovered Qwen3 8B and GLM4 9B baselines and group-rollout runs, both are restricted to evaluation steps through 259, and final is step 259. The Qwen3 8B baseline and group-rollout rows use the expanded seed cohort of Table 8. $\pm$ values are standard deviations across seeds when available.

Model	Setup	Seeds	Best	Final	Steps-to-best
Base model (before training)					
GLM4 9B	Base model	3	0.609 ± 0.010	0.609 ± 0.010	NA
Qwen3 8B	Base model	3	0.487 ± 0.011	0.487 ± 0.011	NA
Baselines (ground-truth unit tests)					
GLM4 9B	Baseline	3	0.905 ± 0.002	0.884 ± 0.006	159 ± 34.6
Qwen3 8B	Baseline	4	0.900 ± 0.010	0.884 ± 0.013	209 ± 60.0
Llama 3.1 8B	Baseline	2	0.658 ± 0.001	0.505 ± 0.025	59 ± 28.2
Controlled noise (p=0.10)					
GLM4 9B	Group rollout	3	0.900 ± 0.006	0.895 ± 0.012	212 ± 80.8
	Group unit test	1	0.891	0.869	139
	Sample rollout	1	0.866	0.835	59
	Unit test	1	0.875	0.864	119
Qwen3 8B	Group rollout	6	0.891 ± 0.015	0.883 ± 0.019	209 ± 62.6
	Group unit test	2	0.893 ± 0.003	0.881 ± 0.007	229 ± 14.1
	Sample rollout	2	0.864 ± 0.011	0.863 ± 0.012	249 ± 14.1
	Unit test	2	0.854 ± 0.013	0.852 ± 0.014	229 ± 14.1
Llama 3.1 8B	Group rollout	1	0.651	0.542	19
Model-based verifier					
Qwen3 8B	Verifier: Qwen3 4B	1	0.704	0.652	39
	Verifier: Qwen3 30B-A3B	1	0.871	0.869	179

5.2 The effect of noise structure

Table 1 shows the final and best validation reward for each noise type. We compare the four controlled noise modes at a fixed noise rate of p=0.10, a moderate level, and the model-based verifiers, to see how the type of noise impacts the results.

Controlled noise. For Qwen3 8B and GLM4 9B, group-level noise (modes c and d) gives higher recorded best and final rewards than sample-level noise (modes a and b). Most comparisons use only 1–2 seeds, so the magnitude and reliability of these differences remain uncertain. The Llama experiment covers only group-rollout noise and does not independently compare the four structures.

Model noise. With the 30B verifier, peak validation reward is 0.871, compared to 0.900 for the clean baseline; with the 4B verifier, it is 0.704. These single-seed comparisons demonstrate different training outcomes for two judges. Their accuracy and precision also differ, but the comparison does not isolate which properties of the judges cause the performance gap (Section 6.3).

We include results for convergence of GLM4 9B and Qwen3 8B in Appendix I, Llama 3.1 8B in Table 1 and appendix L, and Qwen3 4B in Appendix J.

6 DISCUSSION

We examine the scope and interpretation of these results: their preliminary extension to scientific reasoning, the learning signal retained by group-level corruption, and what the model-based verifier metrics can establish.

6.1 DO THE RESULTS HOLD FOR OTHER DOMAINS?

For a preliminary check beyond coding, we consider Graduate-Level Google-Proof Q&A (GPQA) (Rein et al., 2023), which contains graduate-level scientific reasoning problems with multiple-choice answers. We do not include mathematics; prior work reports varied noise effects for Qwen models in that domain (Shao et al., 2025; Zhu & Kang, 2026), and rule-based equivalence checks can reject valid answers (Xu et al., 2025b).

Experimental setup. We remove the diamond problems (198 problems) from the main subset (448 problems before, 250 problems after). We then use the 250 remaining problems for training and the 198 diamond problems for validation. We train Qwen3 8B with group rollout noise at p=0.05 and p=0.30 along with a noise-free baseline. GPQA uses an 8192-token training response limit, compared with 4096 for MBPP, and Group Sequence Policy Optimization (GSPO) (Zheng et al., 2025) instead of GRPO, with the aim of stabilizing training (Appendix E). We do not compare the optimizers on this task.

Table 2: Best validation reward on GPQA Diamond for Qwen3 8B trained with GSPO under group rollout noise (one seed per setting). Both noisy settings are close to the clean baseline.

Setup	Best
Base model	0.540
No noise	0.600
Noise p=0.05	0.604
Noise p=0.30	0.603

Results. Table 2 reports peak validation reward: 0.540 for the base model, 0.600 after clean training, 0.604 at p=0.05, and 0.603 at p=0.30. The small differences between trained models are descriptive single-seed observations subject to checkpoint selection. They do not establish a benefit from noise or greater robustness of binary rewards than unit-test rewards.

Observation. These results suggest that imperfect verification can also support post-training in scientific reasoning. One model, two noise settings, and the change from GRPO to GSPO are insufficient to establish a common robustness threshold across domains.

6.2 WHY CAN MODELS LEARN WITH NOISY REWARDS?

Whole-group corruption has a simple relationship to the clean advantages. We derive it below and separate this property from the hypothesis that noise improves optimization or generalization.

Controlled noise. For a fixed sampled group without format penalties, a full flip changes rewards from r_i to $1 - r_i$, exactly negating its normalized advantage vector $\mathbf{A}$. Thus $\mathbf{A}'$ equals $\mathbf{A}$ with probability $1-p$ and $-\mathbf{A}$ with probability p, giving $\mathbb{E}[\mathbf{A}' \mid \mathbf{r}] = (1-2p)\mathbf{A}$. The expected unclipped policy-gradient contribution therefore remains aligned with the clean contribution for $p < 0.5$. This explains why useful learning signals can survive substantial group-level corruption, without implying a benefit from adding noise. Appendix F gives the full distributions with retained format penalties and shows where clipping invalidates simple gradient scaling.

All four symmetric modes have the same conditional expected test reward, $\mathbb{E}[r'_i \mid \mathbf{M}] = p+(1-2p)r_i$, for a test-scored response. Their differences arise from correlations between corrupted outcomes and from normalizing each realized group. Since normalization is nonlinear, equal expected rewards do not imply equal expected advantages. Sample-level corruption can alter within-group rankings without preserving or reversing the entire advantage vector. This distinction provides an algebraic reason to analyze noise structure, but does not by itself prove which structure gives better generalization. Moreover, the unchanged -0.25 format penalty preserves information about producing a code block even at $p = 0.5$. Since early responses often fail to finish within the token budget, the measured gains cannot be fully separated into formatting and coding improvements with the available aggregate records.

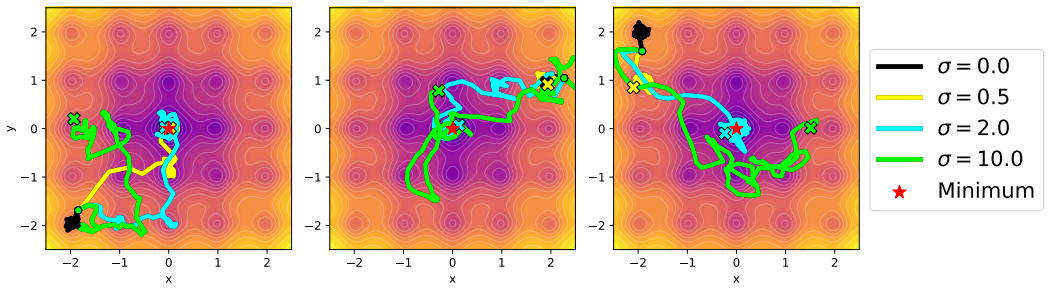

Figure 4: Recorded Ackley optimization trajectories from three starting points under additive reward noise. Here σ denotes the plotted reward-noise scale. Some noisy trajectories leave the starting basin; these examples do not test whole-group corruption or generalization.

Equal error rates can produce different learning signals. For a fixed group without format penalties, the asymmetric implementation draws independent shared decisions with $a = \mathrm{FNR}$ and $b = \mathrm{FPR}$. Its normalized advantage vector satisfies

$$\mathbf{A}' = \begin{cases} \mathbf{A}, & \text{with probability } (1-a)(1-b), \\ -\mathbf{A}, & \text{with probability } ab, \\ \mathbf{0}, & \text{with probability } a+b-2ab. \end{cases}$$

Exactly one decision makes all rewards identical, removing every within-group contrast. Consequently, exchanging a and b leaves this distribution unchanged. When $a = b = p$, the mean is $(1-2p)\mathbf{A}$, as in the main sweep, but the conditional covariance is half as large. Thus identical marginal error rates can produce different distributions of learning signals: group comparisons may be preserved, reversed, or erased. Verifier accuracy alone does not describe these differences. Retained penalties and clipping require the fuller analysis in Appendix F.

The Ackley function. We include a two-dimensional illustration of optimization with noisy rewards. A Gaussian policy with fixed sampling variance is optimized on the Ackley function using a vanilla policy-gradient objective with group-normalized advantages (Appendix N). The recorded trajectories run for 500 steps from three starting points on a circle around the global minimum.

In Figure 4, the noiseless trajectories remain near their starting basins, while some noisy trajectories travel toward the global minimum. This experiment uses additive reward noise, rather than the whole-group inversions of our main sweep. It has no held-out objective and therefore does not test generalization, prevention of overfitting, or the proposed mechanism in LLM training. The trajectories are an illustration, not a replicated estimate of optimization benefit.

Observation. Resampled group corruption can retain useful advantage information. Whether noise also improves generalization through regularization remains untested; related optimization results are discussed in Appendix G.1.

6.3 What do the verifier metrics establish?

Figure 5 shows the two model-based verifier runs on MBPP; we did not run analogous judge experiments on GPQA. The 4B run peaks early and subsequently declines, whereas the 30B run reaches a final reward of 0.869, compared with the clean baseline's 0.884. Both judges have recall above 90% on their respective policy outputs, while the 4B judge has lower precision. Because precision depends on the prevalence of correct outputs, it can change as a consequence of policy quality. Without evaluating both judges on shared response pools, these curves do not isolate intrinsic judge quality from the policies they train.

The rising judge reward alongside declining clean validation reward in the 4B run is consistent with reward overoptimization (Gao et al., 2023). However, the curves compare training prompts scored by a judge with validation prompts scored by execution. They do not establish a particular exploitation mechanism or show that false negatives encourage useful exploration.

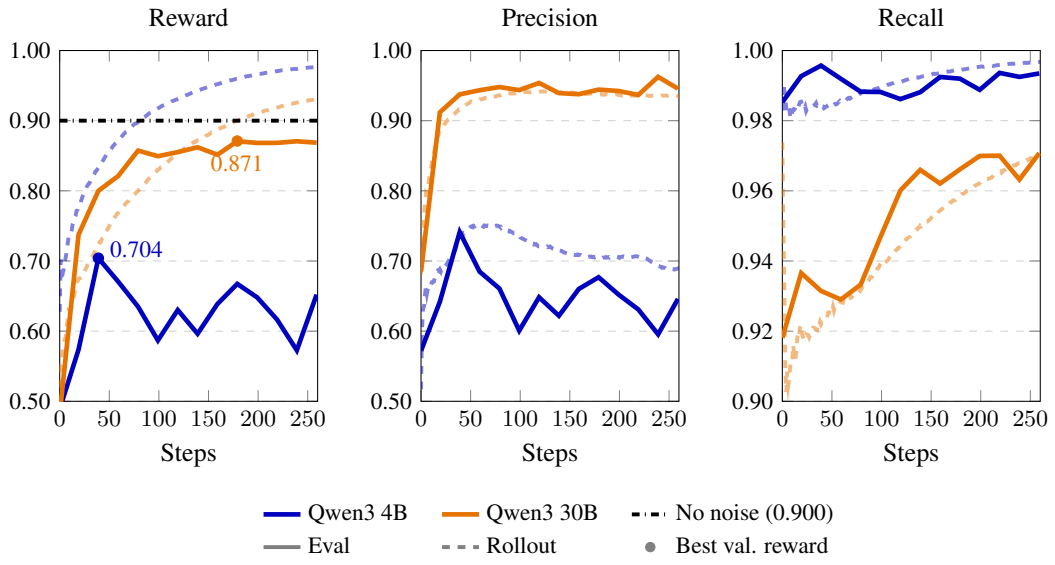

Figure 5: Training Qwen3 8B with model-based verifier rewards on MBPP (one seed per judge). **Left:** validation reward against executed unit tests (solid) and rollout reward from the judge (dashed). Peak validation rewards are 0.871 with the 30B judge and 0.704 with the 4B judge, versus 0.900 for the clean baseline. **Center and right:** verifier precision and recall on each judge's corresponding policy outputs; these are different response distributions.

Xu et al. (2025b) find benefits from reducing false negatives in mathematical verification. Our experiments do not contradict that result: the task and verifier interventions differ, and neither judge comparison identifies a general precision–recall trade-off (Appendix G.2).

Observation. High observed recall alone does not ensure effective training. The asymmetric sweep (Table 7) also cannot isolate the relative effects of FPR and FNR: swapping them leaves the advantage distribution unchanged for groups without format penalties. Appendix F treats mixed-format groups and the normalization stabilizer explicitly.

7 Conclusion

We studied reward corruption in coding RLVR, with a preliminary scientific-reasoning check. In the expanded Qwen3 8B group-rollout sweep, observed mean reward over two fixed late evaluations differs from the clean baseline by less than 1 percentage point at the tested rates through 20%, and by about 2 points at 30%. Confidence intervals allow larger losses. One model-based verifier run reaches a peak reward of 0.871, compared to 0.900 for clean training. These observations show that imperfect verification can support improvement in the tested settings. Our analysis identifies the advantage information preserved by whole-group corruption and the restrictions on interpreting asymmetric group noise. These findings motivate assessing verifiers through both aggregate error rates and the reward contrasts they induce within a rollout group, while measuring downstream quality against a separate reference.

Limitations. Most replication concerns one model and one resampled corruption process. MBPP has only 90 validation problems; seed intervals do not quantify generalization to new problems, and peak selection can be optimistic. GPQA changes the optimizer and uses one seed per setting. Pass@k records and exact per-run source revisions are incomplete. Retained format penalties confound the separation of formatting and correctness gains. We do not establish a verifier-accuracy threshold, robustness to persistent biases, a general precision–recall preference, or improved generalization from noise.

REPRODUCIBILITY STATEMENT

The supplementary archive provides anonymized evaluation records, recorded configurations, scripts for reproducing the reported statistics, and reference implementations of the reward and noise mechanisms. Training and evaluation settings, cohort selection, metric definitions, and remaining provenance limits are described in the appendix. The expanded peak, fixed-checkpoint, and asymmetric results can be checked from the supplied records without access to private experiment tracking services. The pass@k summaries are included with their stated limitations.

AI USE STATEMENT

Generative AI tools assisted with recovering and organizing existing experiment records, checking statistical calculations, implementing analysis and figure code, reviewing methods and result interpretations, and revising manuscript text. This assistance included analysis of existing results and feedback on the presentation of hypotheses and limitations. This assistance used existing experimental records and did not run model training or model evaluation. Numerical outputs were checked against the supplied spreadsheet and recorded histories. The authors are responsible for the final text, claims, and accompanying artifacts.

REFERENCES

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program Synthesis with Large Language Models, August 2021.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosuite, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. Constitutional AI: Harmlessness from AI Feedback, December 2022.

Xin-Qiang Cai, Wei Wang, Feng Liu, Tongliang Liu, Gang Niu, and Masashi Sugiyama. Reinforcement Learning with Verifiable yet Noisy Rewards under Imperfect Verifiers, December 2025.

Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-SGD: Biasing Gradient Descent Into Wide Valleys. In *International Conference on Learning Representations*, February 2017.

Guiming Hardy Chen, Shunian Chen, Ziche Liu, Feng Jiang, and Benyou Wang. Humans or LLMs as the Judge? A Study on Judgement Bias. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 8301–8327, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.474.

Peter Chen, Xiaopeng Li, Ziniu Li, Wotao Yin, Xi Chen, and Tianyi Lin. Exploration vs Exploitation: Rethinking RLVR through Clipping, Entropy, and Spurious Reward, January 2026.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang

Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *Nature*, 645(8081): 633–638, September 2025. ISSN 0028-0836, 1476-4687. doi: 10.1038/s41586-025-09422-z.

Tom Everitt, Victoria Krakovna, Laurent Orseau, and Shane Legg. Reinforcement learning with a corrupted reward channel. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, pp. 4705–4713, Melbourne, Australia, August 2017. AAAI Press. ISBN 978-0-9992411-0-3.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware Minimization for Efficiently Improving Generalization. In *International Conference on Learning Representations*, October 2020.

Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Matteo Hessel, Ian Osband, Alex Graves, Volodymyr Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, Charles Blundell, and Shane Legg. Noisy Networks For Exploration. In *International Conference on Learning Representations*, February 2018.

Leo Gao, John Schulman, and Jacob Hilton. Scaling Laws for Reward Model Overoptimization. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 10835–10866. PMLR, July 2023.

Yuhao Gong, Zhenbo Lu, Wanxuan Lu, Wengang Zhou, and Houqiang Li. Adaptive Confidence-aware Preference-based Reinforcement Learning with Noisy Feedback. In *Companion Proceedings of the ACM on Web Conference 2025*, WWW '25, pp. 2120–2128, New York, NY, USA, May 2025. Association for Computing Machinery. ISBN 979-8-4007-1331-6. doi: 10.1145/3701716.3717570.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Saizhuo Wang, Kun Zhang, Zhouchi Lin, Bowen Zhang, Lionel Ni, Wen Gao, Yuanzhuo Wang, and Jian Guo. A survey on LLM-as-a-Judge. *The Innovation*, pp. 101253, January 2026. ISSN 26666758. doi: 10.1016/j.xinn.2025.101253.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean Hendryx. Rubrics as Rewards: Reinforcement Learning Beyond Verifiable Domains, October 2025.

Yun He, Wenzhe Li, Hejia Zhang, Songlin Li, Karishma Mandyam, Sopan Khosla, Yuanhao Xiong, Nanshu Wang, Xiaoliang Peng, Beibin Li, Shengjie Bi, Shishir G. Patil, Qi Qi, Shengyu Feng, Julian Katz-Samuels, Richard Yuanzhe Pang, Sujan Gonugondla, Hunter Lang, Yue Yu, Yundi Qian, Maryam Fazel-Zarandi, Licheng Yu, Amine Benhalloum, Hany Awadalla, and Manaal Faruqui. AdvancedIF: Rubric-Based Benchmarking and Reinforcement Learning for Advancing LLM Instruction Following, November 2025.

Yuzhen Huang, Weihao Zeng, Xingshan Zeng, Qi Zhu, and Junxian He. From accuracy to robustness: A study of rule- and model-based verifiers in mathematical reasoning. arXiv:2505.22203v2 [cs.LG], October 2025. URL https://arxiv.org/abs/2505.22203v2.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, and Andreas Krause. Reinforcement Learning via Self-Distillation, January 2026.

Hugging Face. Chat templates. https://huggingface.co/docs/transformers/en/chat_templating, 2025a. Accessed: 2026-03-24.

Hugging Face. Math-Verify, January 2025b. URL https://github.com/huggingface/Math-Verify.

Chi Jin, Rong Ge, Praneeth Netrapalli, Sham M. Kakade, and Michael I. Jordan. How to Escape Saddle Points Efficiently. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 1724–1732. PMLR, July 2017.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1):99–134, May 1998. ISSN 0004-3702. doi: 10.1016/S0004-3702(98)00023-X.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. In *International Conference on Learning Representations*, February 2017.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. URL https://arxiv.org/abs/1412.6980.

Bobby Kleinberg, Yuanzhi Li, and Yang Yuan. An Alternative View: When Does SGD Escape Local Minima? In *Proceedings of the 35th International Conference on Machine Learning*, pp. 2698–2707. PMLR, July 2018.

Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, Noah A. Smith, and Hannaneh Hajishirzi. RewardBench: Evaluating Reward Models for Language Modeling. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1755–1797, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-195-7. doi: 10.18653/v1/2025.findings-naacl.96.

Samuel L. Smith and Quoc V. Le. A Bayesian Perspective on Generalization and Stochastic Gradient Descent. In *International Conference on Learning Representations*, February 2018.

Yuxuan Li, Harshith Reddy Kethireddy, and Srijita Das. Evaluating Feature Dependent Noise in Preference-based Reinforcement Learning, January 2026.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's Verify Step by Step, May 2023.

Yixin Liu, Yue Yu, DiJia Su, Sid Wang, Xuewei Wang, Song Jiang, Bo Liu, Arman Cohan, Yuandong Tian, and Zhengxing Chen. Examining Reasoning LLMs-as-Judges in Non-Verifiable LLM Post-Training, 2026.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding R1-Zero-Like Training: A Critical Perspective, October 2025.

Omar El Mansouri, Mohamed El Amine Seddik, and Salem Lahlou. Noise-corrected GRPO: From Noisy Rewards to Unbiased Gradients, 2025.

OpenAI. Learning to reason with LLMs. https://openai.com/index/learning-to-reason-with-llms/, September 2024.

Tianjun Pan, Xuan Lin, Wenyan Yang, Qianyu He, Shisong Chen, Licai Qi, Wanqing Xu, Hongwei Feng, Bo Xu, and Yanghua Xiao. RubricEval: A rubric-level meta-evaluation benchmark for LLM judges in instruction following. arXiv:2603.25133 [cs.AI], March 2026. URL https://arxiv.org/abs/2603.25133.

Ali Rad, Khashayar Filom, Darioush Keivan, Peyman Mohajerin Esfahani, and Ehsan Kamalinejad. Rate or Fate? RLV$^{\varepsilon}$R: Reinforcement Learning with Verifiable Noisy Rewards, January 2026.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A Graduate-Level Google-Proof Q&A Benchmark, November 2023.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, August 2017.

Shreya Shankar, J.D. Zamfirescu-Pereira, Bjoern Hartmann, Aditya Parameswaran, and Ian Arawjo. Who Validates the Validators? Aligning LLM-Assisted Evaluation of LLM Outputs with Human Preferences. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*, UIST '24, pp. 1–14, New York, NY, USA, October 2024. Association for Computing Machinery. ISBN 979-8-4007-0628-8. doi: 10.1145/3654777.3676450.

Rulin Shao, Shuyue Stella Li, Rui Xin, Scott Geng, Yiping Wang, Sewoong Oh, Simon Shaolei Du, Nathan Lambert, Sewon Min, Ranjay Krishna, Yulia Tsvetkov, Hannaneh Hajishirzi, Pang Wei Koh, and Luke Zettlemoyer. Spurious Rewards: Rethinking Training Signals in RLVR, June 2025.

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, April 2024.

David Silver and Richard S Sutton. Welcome to the Era of Experience, 2025.

Sijun Tan, Siyuan Zhuang, Kyle Montgomery, William Y. Tang, Alejandro Cuadron, Chenguang Wang, Raluca Ada Popa, and Ion Stoica. JudgeBench: A benchmark for evaluating LLM-based judges. In *Proceedings of the 13th International Conference on Learning Representations (ICLR)*, Singapore, April 2025.

Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Dan Zhang, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, Hao Yu, Hongning Wang, Jiadai Sun, Jiajie Zhang, Jiale Cheng, Jiayi Gui, Jie Tang, Jing Zhang, Jingyu Sun, Juanzi Li, Lei Zhao, Lindong Wu, Lucen Zhong, Mingdao Liu, Minlie Huang, Peng Zhang, Qinkai Zheng, Rui Lu, Shuaiqi Duan, Shudan Zhang, Shulin Cao, Shuxun Yang, Weng Lam Tam, Wenyi Zhao, Xiao Liu, Xiao Xia, Xiaohan Zhang, Xiaotao Gu, Xin Lv, Xinghan Liu, Xinyi Liu, Xinyue Yang, Xixuan Song, Xunkai Zhang, Yifan An, Yifan Xu, Yilin Niu, Yuantao Yang, Yueyan Li, Yushi Bai, Yuxiao Dong, Zehan Qi, Zhaoyu Wang, Zhen Yang, Zhengxiao Du, Zhenyu Hou, and Zihan Wang. Chatglm: A family of large language models from glm-130b to glm-4 all tools, 2024. URL https://arxiv.org/abs/2406.12793.

Aman Singh Thakur, Kartik Choudhary, Venkat Srinik Ramayapally, Sankaran Vaidyanathan, and Dieuwke Hupkes. Judging the Judges: Evaluating Alignment and Vulnerabilities in LLMs-as-Judges. In Ofir Arviv, Miruna Clinciu, Kaustubh Dhole, Rotem Dror, Sebastian Gehrmann, Eliya Habba, Itay Itzhak, Simon Mille, Yotam Perlitz, Enrico Santus, João Sedoc, Michal Shmueli Scheuer, Gabriel Stanovsky, and Oyvind Tafjord (eds.), *Proceedings of the Fourth Workshop on Generation, Evaluation and Metrics (GEM^2)*, pp. 404–430, Vienna, Austria and virtual meeting, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-261-9.

Bertie Vidgen, Austin Mann, Abby Fennelly, John Wright Stanly, Lucas Rothman, Marco Burstein, Julien Benchek, David Ostrofsky, Anirudh Ravichandran, Debnil Sur, Neel Venugopal, Alannah Hsia, Isaac Robinson, Calix Huang, Olivia Varones, Daniyal Khan, Michael Haines, Zach Richards, Chirag Mahapatra, Brendan Foody, and Osvald Nitski. APEX-Agents, January 2026.

Vijay Viswanathan, Yanchao Sun, Xiang Kong, Meng Cao, Graham Neubig, and Tongshuang Wu. Checklists Are Better Than Reward Models For Aligning Language Models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=RPRqKhjrr6.

Jingkang Wang, Yang Liu, and Bo Li. Reinforcement Learning with Perturbed Rewards. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):6202–6209, April 2020. ISSN 2374-3468. doi: 10.1609/aaai.v34i04.6086.

Donglai Xu, Hongzheng Yang, Yuzhi Zhao, Pingping Zhang, Jinpeng Chen, Wenao Ma, Zhijian Hou, Mengyang Wu, Xiaolei Li, Senkang Hu, Ziyi Guan, Jason Chun Lok Li, and Lai Man Po. From Exploration to Exploitation: A Two-Stage Entropy RLVR Approach for Noise-Tolerant MLLM Training, November 2025a.

Zhangchen Xu, Yuetai Li, Fengqing Jiang, Bhaskar Ramasubramanian, Luyao Niu, Bill Yuchen Lin, and Radha Poovendran. TinyV: Reducing False Negatives in Verification Improves RL for LLM Reasoning, May 2025b.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-Math Technical Report: Toward Mathematical Expert Model via Self-Improvement, September 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 Technical Report, May 2025.

Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. SimpleRL-Zoo: Investigating and Taming Zero Reinforcement Learning for Open Base Models in the Wild. In *Second Conference on Language Modeling*, August 2025.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group Sequence Policy Optimization, 2025.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and Chatbot Arena. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, pp. 46595–46623, Red Hook, NY, USA, December 2023. Curran Associates Inc.

Yang Zhou, Sunzhu Li, Shunyu Liu, Wenkai Fang, Kongcheng Zhang, Jiale Zhao, Jingwen Yang, Yihe Zhou, Jianwei Lv, Tongya Zheng, Hengtong Lu, Wei Chen, Yan Xie, and Mingli Song. Breaking the Exploration Bottleneck: Rubric-Scaffolded Reinforcement Learning for General LLM Reasoning, October 2025.

Lianghui Zhu, Xinggang Wang, and Xinlong Wang. JudgeLM: Fine-tuned large language models are scalable judges. In *Proceedings of the 13th International Conference on Learning Representations (ICLR)*, Singapore, April 2025a.

Yuxuan Zhu and Daniel Kang. Noisy Data is Destructive to Reinforcement Learning with Verifiable Rewards, March 2026.

Zhanxing Zhu, Jingfeng Wu, Bing Yu, Lei Wu, and Jinwen Ma. The Anisotropic Noise in Stochastic Gradient Descent: Its Behavior of Escaping from Sharp Minima and Regularization Effects, June 2019.

Zilin Zhu, Chengxing Xie, Xin Lv, and slime Contributors. slime: An LLM post-training framework for RL scaling. https://github.com/THUDM/slime, 2025b. GitHub repository. Corresponding author: Xin Lv.

Mingchen Zhuge, Changsheng Zhao, Dylan Ashley, Wenyi Wang, Dmitrii Khizbullin, Yunyang Xiong, Zechun Liu, Ernie Chang, Raghuraman Krishnamoorthi, Yuandong Tian, Yangyang Shi, Vikas Chandra, and Jürgen Schmidhuber. Agent-as-a-Judge: Evaluate agents with agents. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, pp. 80569–80611, Vancouver, Canada, July 2025.

A REPRODUCIBILITY

Qwen3 before our RL training performs worse under our evaluation protocol than reported in the Qwen team's technical report (Yang et al., 2025). This cross-protocol difference does not itself establish an evaluation error in either study. The supplementary archive provides the recorded evaluation settings and reward implementations. Our evaluation applies the model's chat template (Hugging Face, 2025a); the reported comparisons use this prompting protocol.

Run records and selection. The accompanying data include individual seed scores, run manifests and configuration records, and scripts for reproducing the tables from recorded W&B histories. All 37 expanded Qwen3 8B peak scores match runs in the original and later projects. The clean baseline uses seeds 1, 3, 11 and 12; 5% uses seeds 1–3; 10% uses seeds 1–3 and 10–12; the remaining settings use one original seed and seeds 10–12. We preserve this recorded cohort, with additional and repeated run candidates documented separately; this is a retrospective cohort, not a prospectively specified inclusion rule. All 37 runs have evaluations at steps 239 and 259. Our retrospective fixed-step analysis averages these two scores within each run, then reports the mean and sample standard deviation across seeds, without selecting a checkpoint by validation performance. The archives do not identify exact per-run source revisions; recovered source snapshots and their limitations are documented with the analysis scripts. Reconstructing aggregate results from these records does not establish that every historical training run can be reproduced exactly.

Evaluation uncertainty. All MBPP evaluations use the same 90 validation problems. The reported standard deviations and confidence intervals describe variation across training seeds on this fixed problem set; they do not include uncertainty from sampling new problems. The 16 generations per problem reduce generation-sampling uncertainty but are not independent problems. No untouched test-set evaluation or prespecified noninferiority margin is available, so small observed mean differences do not establish equivalence within a chosen practical tolerance.

B TRAINING DETAILS

B.1 FINAL HYPERPARAMETERS

Table 3 lists the hyperparameters used across all experiments unless otherwise noted.

The original MBPP runs and later clean baselines evaluate every 20 rollout steps; most later noisy runs evaluate every 10. The curves use the shared index grid 0, 19, 39, and so on through 259. Runs were configured for 261 or 500 rollout iterations; the expanded peak table retains all recorded maxima, including later evaluations through step 479 where available. Matching rollout steps controls the nominal sampling budget, not exact generated-token counts or wall-clock time. The recovered GLM checkpoint is `GLM-Z1-9B-0414`, referred to as GLM4 9B in the paper.

B.2 HYPERPARAMETER SENSITIVITY

Table 4 compares training performance under alternative sampling hyperparameters for GLM4 9B and Qwen3 8B. The main setting (top-p=1.00, temperature=1.00) is reproduced from Table 1 for reference.

In the recorded runs, reducing top-p from 1.00 to 0.95 lowers GLM4 9B's best reward from 0.905 to 0.817 and final reward from 0.884 to 0.764. Qwen3 8B's best reward falls from 0.900 to 0.829. Lowering the temperature from 1.00 to 0.70 also lowers both models' observed scores, particularly Qwen3 8B's (0.803 vs. 0.900 best). Reduced sampling diversity is one possible explanation, but these runs do not isolate the mechanism. Increasing the global batch size from 96 to 768 yields lower observed peak scores for both models (GLM4 9B: 0.888 vs. 0.905; Qwen3 8B: 0.845 vs. 0.900); the rounded best and final scores coincide in each larger-batch run. The small seed counts, including single runs for temperature and batch-size changes, limit conclusions about stability and convergence speed.

Table 3: Hyperparameters used for all experiments. MBPP experiments use GRPO; GPQA experiments use GSPO.

Hyperparameter	Value
Optimization	
Optimizer	Adam
Learning rate	1×10^{-6}
LR schedule	Constant
Weight decay	0.10
β_1, β_2	0.90, 0.98
GRPO / GSPO	
Clip range (ϵ_{low}, ϵ_{high})	0.20, 0.28
KL coefficient	0.00
Entropy coefficient	0.00
Rollouts per prompt	16
Prompts per rollout batch (recovered MBPP runs)	48
Global batch size	96
Sampling (training)	
Temperature	1.00
Top-p	1.00
Max response length	4096 (MBPP) / 8192 (GPQA)
Evaluation	
Samples per prompt	16
Temperature	0.70
Top-p	1.00
Max response length	8192
Eval interval	Every 10 or 20 rollout steps

Table 4: Sensitivity to sampling hyperparameters for baseline (no noise) training on MBPP. T denotes temperature, B the global batch size. All runs use 16 rollouts per prompt. The main setting rows are reproduced from Table 1.

Model	Hyperparameters	Seeds	Best	Final
GLM4 9B				
	top-p=1.00, T=1.00, B=96 (main)	3	0.905 ± 0.002	0.884 ± 0.006
	top-p=0.95, T=1.00, B=96	2	0.817 ± 0.005	0.764 ± 0.002
	top-p=1.00, T=0.70, B=96	1	0.895	0.881
	top-p=1.00, T=1.00, B=768	1	0.888	0.888
Qwen3 8B				
	top-p=1.00, T=1.00, B=96 (main)	4	0.900 ± 0.010	0.884 ± 0.013
	top-p=0.95, T=1.00, B=96	2	0.829 ± 0.025	0.827 ± 0.022
	top-p=1.00, T=0.70, B=96	1	0.803	0.769
	top-p=1.00, T=1.00, B=768	1	0.845	0.845

C Additional evaluation results

Expanded peak results. Table 5 reports the additional Qwen3 8B group-rollout runs on MBPP. Each seed contributes its highest recorded validation reward. These maxima include different training durations and evaluation frequencies, giving unequal opportunities for checkpoint selection. We therefore report a separate comparison at fixed evaluation steps in Appendix I. Means, sample standard deviations, seed counts and p-values are reproduced from the recorded seed scores; the statistical test is the two-sided independent equal-variance t-test used in Excel (`T.TEST(...,2,2)`). The comparisons share a baseline and are unadjusted for multiple testing. They characterize the observed peak scores rather than establish equivalent performance or a universal noise threshold.

Table 5: Expanded peak mean unit-test rewards for Qwen3 8B on MBPP with group-rollout noise. n is the number of seeds; Δ is the difference from the clean baseline in percentage points, with a Welch 95% confidence interval. Each run contributes its best recorded score, including later checkpoints where available.

Noise rate	n	Peak score (mean $\pm$ SD)	Δ (pp)	95% CI (pp)	p-value
0	4	0.9003 ± 0.0104	—	—	—
0.05	3	0.8879 ± 0.0220	-1.24	$[-5.9, +3.5]$	0.3601
0.10	6	0.8939 ± 0.0168	-0.63	$[-2.6, +1.4]$	0.5246
0.15	4	0.8922 ± 0.0102	-0.81	$[-2.6, +1.0]$	0.3085
0.20	4	0.8960 ± 0.0071	-0.42	$[-2.0, +1.2]$	0.5269
0.25	4	0.8837 ± 0.0051	-1.66	$[-3.2, -0.1]$	0.0284
0.30	4	0.8798 ± 0.0086	-2.05	$[-3.7, -0.4]$	0.0231
0.40	4	0.8168 ± 0.0348	-8.34	$[-13.7, -3.0]$	0.0037
0.50	4	0.5988 ± 0.1132	-30.14	$[-48.1, -12.2]$	0.0018

Full-program correctness. Table 6 reports the recorded best pass@k over evaluated checkpoints, using 16 generations per validation problem and requiring all three tests to pass. For c successful generations, the standard estimator is $1 - \binom{16-c}{k} / \binom{16}{k}$, averaged across problems. The available summaries cover fewer seeds than Table 5. These are peak metrics and remain subject to checkpoint selection; individual seed, checkpoint and generation records were not recovered, so a common-checkpoint pass@k comparison is unavailable. The aggregate table can be reproduced from the archived summary, but the contributing runs, selected checkpoints and per-problem estimates cannot be independently reconstructed. These results therefore provide descriptive supporting evidence, with weaker provenance than the expanded unit-test reward analysis.

Table 6: Peak full-program pass@k for Qwen3 8B on MBPP with group-rollout noise. Values are mean $\pm$ standard deviation across n seeds. An SD displayed as 0.000 is rounded. The metric-specific seed cohort is smaller than that of Table 5.

Noise rate	n	pass@1	pass@4	pass@8	pass@16
0	2	0.814 ± 0.005	0.860 ± 0.017	0.875 ± 0.018	0.894 ± 0.008
0.10	2	0.841 ± 0.001	0.891 ± 0.004	0.907 ± 0.006	0.922 ± 0.000
0.15	3	0.816 ± 0.012	0.876 ± 0.011	0.895 ± 0.012	0.915 ± 0.006
0.20	2	0.827 ± 0.001	0.884 ± 0.001	0.911 ± 0.003	0.933 ± 0.000
0.25	2	0.818 ± 0.004	0.868 ± 0.000	0.882 ± 0.001	0.894 ± 0.008
0.30	2	0.810 ± 0.005	0.872 ± 0.013	0.898 ± 0.013	0.922 ± 0.016
0.40	2	0.756 ± 0.012	0.849 ± 0.005	0.876 ± 0.000	0.894 ± 0.008
0.50	2	0.551 ± 0.104	0.715 ± 0.025	0.744 ± 0.013	0.767 ± 0.000

Asymmetric noise. Table 7 reports all 25 settings of the recovered Qwen3 8B sweep. This implementation draws independent group-level decisions $Z_{\mathrm{FN}} \sim \mathrm{Bernoulli}(\mathrm{FNR})$ and $Z_{\mathrm{FP}} \sim \mathrm{Bernoulli}(\mathrm{FPR})$, applying $M'_{i,j} = M_{i,j}(1 - Z_{\mathrm{FN}}) + (1 - M_{i,j})Z_{\mathrm{FP}}$ to the original test outcomes. It differs from the single-decision corruption of the main symmetric design, even when FPR and FNR are equal; format penalties remain unchanged.

The grid is exploratory: all settings use seed 12 and provide no across-seed uncertainty. More fundamentally, without retained format penalties, let $a = \mathrm{FNR}$, $b = \mathrm{FPR}$ and $\hat{\mathbf{A}}$ denote the clean advantage vector. The corrupted vector equals $\hat{\mathbf{A}}$, $-\hat{\mathbf{A}}$ or $\mathbf{0}$ with probabilities $(1-a)(1-b)$, ab and $a(1-b) + (1-a)b$, respectively (Appendix F). Its full conditional distribution is invariant to exchanging a and b, so this design cannot identify different effects of false positives and false negatives on correctness learning in the absence of penalties. Retaining the uniform negative format penalty replaces the zero-advantage states of mixed groups with a formatting signal. Those two states have the same normalized advantage direction, and the swap invariance remains exact when the normalization stabilizer is zero; with a positive stabilizer, their magnitudes can differ. Thus, the penalty does not resolve the intended identification problem. Individual trajectories also vary with sampled groups and noise decisions. Moreover, exchanging rates does not hold verifier accuracy

fixed when the pass/fail prevalence differs from one half. The observed differences between swapped settings therefore do not establish a general precision–recall preference.

Table 7: Peak mean unit-test rewards in the asymmetric-noise sweep. Rows are false-negative rates and columns are false-positive rates. Runs use seed 12 and evaluate every 10 rollout steps through 259, except †, interrupted after step 229. ‡ averages peaks from two records sharing seed 12, as in the original summary.

FNR \ FPR	0.05	0.15	0.30	0.40	0.50
0.05	0.892	0.906	0.905	0.887	0.878
0.15	0.900	0.899	0.893	0.896	0.886
0.30	0.900	0.893	0.869	0.875	0.864
0.40	0.891	0.901	0.895	0.851	0.823†
0.50	0.896	0.860	0.888	0.843	0.781‡

D PROMPTS

We list the prompts used in our experiments. For MBPP and GPQA, the task description is wrapped using the model's chat template (`apply_chat_template` (Hugging Face, 2025a)). For the model-based verifier experiments, we use the following system and user prompts.

D.1 MODEL-BASED VERIFIER: UNIT TEST EVALUATION

Below we show the single-test version of the judge prompt. The archived configurations do not identify whether requests used single-test or batched judging.

The archived judge models are `Qwen3-4B` and `Qwen3-30B-A3B-Instruct-2507`, with temperature 0. The requested response is a JSON object containing a boolean `status`, as shown below. Recovered source validates the JSON, attempts to extract an embedded JSON object if needed, and assigns a false verdict if parsing still fails; request exceptions return reward 0. The configurations specify up to five attempts with a 5000-second timeout per attempt, but the exact active retry implementation is not archived per run.

Historical judge training code also differs in its reward conventions: an earlier version assigns 0 to missing code and floors formatted-code rewards at $0.5/T$, whereas a later version uses the -0.25 format penalty without that floor. The missing per-run source revisions prevent resolving this detail for each judge run. The controlled-noise derivation in Appendix F concerns the synthetic corruption rules and should not be assumed to describe these judge training rewards. Clean validation uses the executed-unit-test metric described in the main text.

> **System:** You are an expert code evaluator evaluating the quality of a model's response against a test case. Your task is to determine whether the GENERATED CODE satisfies the TEST CASE. I.e., whether the assert statement would pass. Answer with a JSON object with the key "status" and a boolean value.
>
> Instructions:
>
> - Carefully read the TEST CASE to understand what is required.
> - Evaluate the GENERATED CODE against this TEST CASE only (ignore anything outside it).
>
> **User:**
> GENERATED CODE:
> `{generated_code}`
>
> TEST CASE:
> `assert {function_name}({test_input}) == {test_output}`

D.2 GPQA: ANSWER EXTRACTION

For GPQA, the model is presented with the question and multiple-choice options via the chat template. The reward is computed by extracting the selected option letter from the model's response using a

series of regex patterns and comparing it against the correct answer. No model-based verifier is used for GPQA evaluation.

E BACKGROUND

In RLVR, a language model is trained using reinforcement learning, where a verifier determines the reward for each generated response—for example, executing unit tests for code or checking a multiple-choice answer. This section describes the training objective.

GRPO. We use GRPO as our training objective. Unlike the earlier Proximal Policy Optimization (PPO) (Schulman et al., 2017), GRPO omits the value network and instead estimates advantages by comparing rollouts within a group sampled for the same prompt.

Concretely, for each prompt x, GRPO samples a group of G responses $\{y_1, \ldots, y_G\} \sim \pi_{\theta_{\text{old}}}(\cdot \mid x)$ and computes a reward $r_i = R(x, y_i)$ for each response. The advantage for response y_i is computed by normalizing rewards within the group:

$$\hat{A}_i = \frac{r_i - \mu(\mathbf{r})}{\sigma(\mathbf{r}) + \epsilon} \tag{1}$$

where $\mu(\mathbf{r})$ and $\sigma(\mathbf{r})$ are the mean and standard deviation of the group rewards, and ϵ is a small constant for numerical stability.

The clipped GRPO objective is

$$\begin{aligned} J(\theta) &= \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|y_i|}\sum_{t=1}^{|y_i|}\ell_{i,t}(\theta)\right], \\ \ell_{i,t}(\theta) &= \min\Big(\rho_{i,t}\hat{A}_i, \text{clip}(\rho_{i,t}, 1-\epsilon_{\text{low}}, 1+\epsilon_{\text{high}})\hat{A}_i\Big), \end{aligned} \tag{2}$$

where the expectation is over training prompts and responses sampled from $\pi_{\theta_{\text{old}}}$, and $\rho_{i,t} = \pi_\theta(y_{i,t} \mid x, y_{i,<t})/\pi_{\theta_{\text{old}}}(y_{i,t} \mid x, y_{i,<t})$. The recovered configurations use per-response token averaging, clipping bounds $\epsilon_{\text{low}} = 0.20$ and $\epsilon_{\text{high}} = 0.28$, and zero KL and entropy coefficients.

GSPO. For the scientific reasoning experiments, we use Group Sequence Policy Optimization (GSPO) (Zheng et al., 2025) instead of GRPO with the aim of stabilizing training. GSPO retains group-normalized advantages but uses sequence-level importance ratios and clipping. We do not compare the optimizers on this task. We refer the reader to Zheng et al. (2025) for technical details.

F CONDITIONAL ADVANTAGE DISTRIBUTIONS UNDER GROUP-LEVEL NOISE

This section analyzes the two group-rollout corruption rules used in the controlled experiments. All distributions and expectations below condition on a fixed prompt, its sampled responses and clean test outcomes; the only randomness is the corruption decision. When discussing gradients, we also fix the current and sampling policies and treat advantages as detached, as in GRPO. These conditional identities do not assume that different training runs visit the same policies or responses.

F.1 CENTERING AND RETAINED FORMAT PENALTIES

Assume $G \geq 2$. Let $m_i \in \{0, 1\}$ indicate whether response i has a nonnegative, test-scored reward, and let $q_i \in [0, 1]$ be that reward when $m_i = 1$. Execution failures have $q_i = 0$; missing-code responses have $m_i = 0$ and retain the penalty $c = -1/4$. The value assigned to q_i when $m_i = 0$ is immaterial. Write

$$\mathbf{r} = \mathbf{m} \odot \mathbf{q} + c(\mathbf{1} - \mathbf{m}), \qquad H = I - \frac{1}{G}\mathbf{1}\mathbf{1}^\top, \qquad \mathbf{z} = H\mathbf{r}, \quad \mathbf{h} = H\mathbf{m}. \tag{3}$$

For a centered vector $\mathbf{v}$, define

$$s_d(\mathbf{v}) = \sqrt{\frac{\|\mathbf{v}\|_2^2}{d}}, \qquad F(\mathbf{v}) = \frac{\mathbf{v}}{s_d(\mathbf{v}) + \eta}, \qquad \mathbf{A} = F(\mathbf{z}), \tag{4}$$

where $\eta > 0$ is the normalization stabilizer, distinct from the clipping bounds. The identities hold for either population standard deviation ($d = G$) or sample standard deviation ($d = G - 1$). The recovered normalization source uses the sample convention and $\eta = 10^{-6}$; exact source revisions are unavailable per run. In the idealized $\eta = 0$ case, define $F(\mathbf{0}) = \mathbf{0}$.

Suppose corruption maps every eligible reward q_i to $\alpha q_i + \beta$ and leaves c unchanged. Centering gives the general identity

$$H\mathbf{r}' = \alpha\mathbf{z} + [\beta + c(\alpha - 1)]\mathbf{h}, \qquad \mathbf{A}' = F(\alpha\mathbf{z} + [\beta + c(\alpha - 1)]\mathbf{h}). \tag{5}$$

Each realized reward vector must be normalized separately: in general, $\mathbb{E}[\mathbf{A}'] \neq F(H\mathbb{E}[\mathbf{r}'])$.

F.2 Symmetric corruption with one shared decision

In the main sweep, $Z \sim \text{Bernoulli}(p)$ and eligible rewards become $q_i' = (1 - Z)q_i + Z(1 - q_i)$. Define the flipped advantage vector

$$\mathbf{A}_{\text{F}} = F(-\mathbf{z} + (1 - 2c)\mathbf{h}) = F(-\mathbf{z} + \tfrac{3}{2}\mathbf{h}). \tag{6}$$

The exact conditional law and moments, including penalties, are

$$\mathbf{A}' = \begin{cases} \mathbf{A}, & \text{with probability } 1 - p, \\ \mathbf{A}_{\text{F}}, & \text{with probability } p, \end{cases} \tag{7}$$

$$\mathbb{E}[\mathbf{A}'] = (1 - p)\mathbf{A} + p\mathbf{A}_{\text{F}}, \tag{8}$$

$$\text{Cov}(\mathbf{A}') = p(1 - p)(\mathbf{A}_{\text{F}} - \mathbf{A})(\mathbf{A}_{\text{F}} - \mathbf{A})^{\mathsf{T}}. \tag{9}$$

If all responses are test-scored, $\mathbf{h} = \mathbf{0}$ and $F(-\mathbf{z}) = -F(\mathbf{z})$, including for $\eta > 0$. Thus

$$\mathbf{A}' = (1 - 2Z)\mathbf{A}, \qquad \mathbb{E}[\mathbf{A}'] = (1 - 2p)\mathbf{A}, \qquad \text{Cov}(\mathbf{A}') = 4p(1 - p)\mathbf{A}\mathbf{A}^{\mathsf{T}}. \tag{10}$$

The corruption preserves the expected advantage direction for $p < 1/2$ and applies a single shared sign multiplier within a fixed group. This is a property of this particular corruption rule, not a general property of erroneous verifiers.

With mixed test-scored and penalized responses, full-group corruption does not generally negate the advantages. Even at $p = 1/2$, eligible rewards have conditional mean $1/2$, whereas penalized rewards remain $-1/4$. A format-related signal also survives normalization: if $k = \sum_i m_i$ satisfies $0 < k < G$, and $\bar{q}$ is the mean eligible score, then

$$\mathbf{h}^{\mathsf{T}}\mathbf{z} = k(1 - k/G)(\bar{q} - c) > 0, \qquad \mathbf{h}^{\mathsf{T}}(-\mathbf{z} + (1 - 2c)\mathbf{h}) = k(1 - k/G)(1 - \bar{q} - c) > 0. \tag{11}$$

Both realizations therefore favor test-scored responses over penalized responses on average within the group. If all responses are penalized, every advantage is zero. Consequently, 50% corruption does not remove all reward information from the implemented training setup.

F.3 Asymmetric corruption with two shared decisions

Let $a = \text{FNR}$ and $b = \text{FPR}$. The exploratory implementation draws independent $U \sim \text{Bernoulli}(a)$ and $V \sim \text{Bernoulli}(b)$, shared across the group, and applies both decisions to the original outcomes. Eligible rewards become

$$q_i' = (1 - U - V)q_i + V. \tag{12}$$

Applying Equation (5) yields the full conditional distribution:

(U, V)	Probability	$H\mathbf{r}'$	$\mathbf{A}'$
$(0, 0)$	$(1 - a)(1 - b)$	$\mathbf{z}$	$\mathbf{A}$
$(1, 0)$	$a(1 - b)$	$-c\mathbf{h}$	$\mathbf{A}_0 := F(-c\mathbf{h})$
$(0, 1)$	$(1 - a)b$	$(1 - c)\mathbf{h}$	$\mathbf{A}_1 := F((1 - c)\mathbf{h})$
$(1, 1)$	ab	$-\mathbf{z} + (1 - 2c)\mathbf{h}$	$\mathbf{A}_{\text{F}}$.

(13)

Its mean is the probability-weighted sum of these four vectors; its covariance is $\sum_j w_j \mathbf{A}_j \mathbf{A}_j^{\mathsf{T}} - \boldsymbol{\mu}\boldsymbol{\mu}^{\mathsf{T}}$, where w_j are the listed probabilities and $\boldsymbol{\mu}$ is their weighted mean.

Without penalties, $\mathbf{h} = \mathbf{0}$, so exactly one decision makes every reward identical and all advantages zero. In this case

$$\mathbf{A}' = \begin{cases} \mathbf{A}, & \text{with probability } (1-a)(1-b), \\ -\mathbf{A}, & \text{with probability } ab, \\ \mathbf{0}, & \text{with probability } a+b-2ab, \end{cases} \tag{14}$$

and

$$\mathbb{E}[\mathbf{A}'] = (1-a-b)\mathbf{A}, \qquad \mathrm{Cov}(\mathbf{A}') = [a(1-a)+b(1-b)]\mathbf{A}\mathbf{A}^{\mathsf{T}}. \tag{15}$$

The entire distribution is invariant to exchanging a and b. When $a = b = p$, its mean matches the one-decision rule, but it has zero-advantage probability $2p(1-p)$ and inversion probability p^2, and half the conditional covariance. Equal marginal error rates therefore do not make the two interventions equivalent.

Retained penalties do not resolve this identifiability limitation. In a mixed group, let $s = s_d(\mathbf{h}) > 0$. For $c = -1/4$,

$$\mathbf{A}_0 = \frac{\mathbf{h}}{s + 4\eta}, \qquad \mathbf{A}_1 = \frac{\mathbf{h}}{s + \frac{4}{5}\eta}. \tag{16}$$

These vectors have the same direction and coincide exactly when $\eta = 0$, in which case the entire distribution remains invariant under exchanging a and b, even with penalties. For $\eta > 0$, swapping the rates only swaps the probabilities of these two nearby vectors. In particular,

$$\mathbb{E}_{a,b}[\mathbf{A}'] - \mathbb{E}_{b,a}[\mathbf{A}'] = (a-b)(\mathbf{A}_0 - \mathbf{A}_1), \tag{17}$$

$$\mathbf{A}_0 - \mathbf{A}_1 = -\frac{\frac{16}{5}\eta\,\mathbf{h}}{(s+4\eta)(s+\frac{4}{5}\eta)}. \tag{18}$$

For sample normalization, $s^2 = k(G-k)/[G(G-1)]$, so with $G = 16$ every mixed group has $s \geq 1/4$. At the recovered stabilizer $\eta = 10^{-6}$, the relative difference $\|\mathbf{A}_0 - \mathbf{A}_1\|_2 / \|\mathbf{A}_1\|_2$ is at most 1.28×10^{-5} in exact arithmetic. This is an algebraic bound, not a measured effect. Individual training trajectories can still differ, but swapped-rate runs under this design do not isolate a substantive preference between false positives and false negatives in correctness learning.

F.4 Unclipped gradients, clipping, and optimizer updates

For a fixed group and current policy, the unclipped surrogate gradient is linear in the advantages:

$$g_\theta(\mathbf{A}) = \frac{1}{G}\sum_{i=1}^{G} \frac{A_i}{|y_i|} \sum_{t=1}^{|y_i|} \nabla_\theta \rho_{i,t}. \tag{19}$$

At $\theta = \theta_{\mathrm{old}}$, each ratio is one and $\nabla_\theta \rho_{i,t} = \nabla_\theta \log \pi_\theta(y_{i,t} \mid x, y_{i,<t})$. Without penalties, Equations (10) and (14) give

$$\mathbb{E}[g_\theta(\mathbf{A}')] = \begin{cases} (1-2p)g_\theta(\mathbf{A}), & \text{one decision}, \\ (1-a-b)g_\theta(\mathbf{A}), & \text{two decisions}. \end{cases} \tag{20}$$

With penalties, linearity still gives $\mathbb{E}[g_\theta(\mathbf{A}')] = g_\theta(\mathbb{E}[\mathbf{A}'])$, using the full mixtures above, rather than a universal scalar attenuation.

Clipping limits the sign-reversal interpretation. Let $L = 1 - \epsilon_{\mathrm{low}}$, $U = 1 + \epsilon_{\mathrm{high}}$, and $[v]_+ = \max(v, 0)$. For one token, the clipped surrogate in Equation (2) satisfies

$$\ell(\rho, A) = \rho A - [A]_+[\rho - U]_+ - [-A]_+[L - \rho]_+, \tag{21}$$

$$\ell(\rho, A) + \ell(\rho, -A) = -|A| D(\rho), \qquad D(\rho) = [\rho - U]_+ + [L - \rho]_+. \tag{22}$$

Thus, for groups without penalties, the exact conditional expected clipped surrogates are

$$\mathbb{E}[\ell(\rho, A')] = \begin{cases} (1-2p)\ell(\rho, A) - p|A|D(\rho), & \text{one decision}, \\ (1-a-b)\ell(\rho, A) - ab|A|D(\rho), & \text{two decisions}. \end{cases} \tag{23}$$

These identities apply token by token and hence to the full response-averaged objective. Within the clipping interval, $D(\rho) = 0$ and unclipped scaling holds; at $\theta = \theta_{\mathrm{old}}$ all ratios lie strictly inside this

interval. Outside it, an additional term penalizes distance from the interval. In particular, at $p = 1/2$ the expected one-decision surrogate is $-|A|D(\rho)/2$: its gradient is zero inside the interval but can be nonzero outside, pointing toward the interval as a function of ρ. Statements about derivatives exclude the clipping boundaries, where the objective is not differentiable. With retained penalties the exact expected clipped objective is instead the weighted sum of $\ell(\rho, A_j)$ over Equations (7) and (13); substituting only the mean advantage is generally invalid.

These are conditional statements about raw surrogate gradients at a fixed policy. Reusing a corrupted batch makes later policies depend on its realized corruption. Adam additionally maps gradients and their history through nonlinear moment normalization, and weight decay contributes a separate update. Neither expected-gradient attenuation nor a single-group sign reversal therefore implies a correspondingly scaled or reversed parameter update, identical training dynamics, escape from local minima, or improved generalization. The preserved expected direction provides a direct explanation for why this specific resampled corruption can permit learning below 50% noise; it does not establish a practical verifier-accuracy threshold or a regularization benefit.

G In-depth discussion

G.1 Noise and flat minima: possible connections

As discussed in Section 6.2 and appendix F, group-wide reward complementation reverses normalized advantages when every response receives a test-based reward. Below 50% corruption, the expected unclipped contribution retains the clean direction. This algebraic property explains why the noise can preserve a useful learning signal without requiring a regularization or basin-escape mechanism.

Prior work studies connections between stochastic optimization, noise structure and flatness (Keskar et al., 2017; Le, 2018; Zhu et al., 2019), as well as deliberate perturbations and smoothed objectives (Foret et al., 2020; Chaudhari et al., 2017). Other analyses examine escape from local minima or saddle points under particular assumptions (Kleinberg et al., 2018; Jin et al., 2017). These results motivate hypotheses about our training dynamics, but their interventions and assumptions differ from randomly complementing rewards. We do not measure flatness, curvature, basin escape or improved generalization in the trained language models. The Ackley illustration uses additive reward noise and has no training/test distinction; it does not test the group-complementation mechanism or show that it prevents overfitting.

G.2 The results of Xu et al. (2025b)

Xu et al. (2025b) study false negatives in mathematical verification and introduce a lightweight model-based verifier that rechecks responses rejected by a rule-based verifier. Their reported improvements show the value of this intervention in their setting. They do not conflict with our observation that a coding verifier can have high recall while providing poor training rewards.

These are different comparisons: TinyV augments a verifier, whereas our judge experiment changes the model used for every judgment and evaluates each judge on its own trained policy's outputs. Our shared-decision asymmetric experiment also cannot isolate a general precision–recall trade-off (Appendix F). Consequently, our results neither refute the value of reducing false negatives nor establish that false positives are intrinsically more harmful. Comparing those effects would require comparable response distributions and an intervention that separates them.

H Detailed metrics for the model-based verifiers

We show in Figure 6 the reward for training and validation when using the model-based verifiers; these overlap with the results in Section 6.3. For training, the rewards are given by the model-based verifier, while the validation rewards come from running the actual unit tests. The accuracy, F1 score, precision, and recall are for the model-based verifier when comparing the pass/fail decisions with running the unit tests.

As discussed in Section 6.3, the 4B verifier has lower observed precision, accuracy and F1 score, with slightly higher recall. High recall alone does not ensure useful training rewards in these runs, but

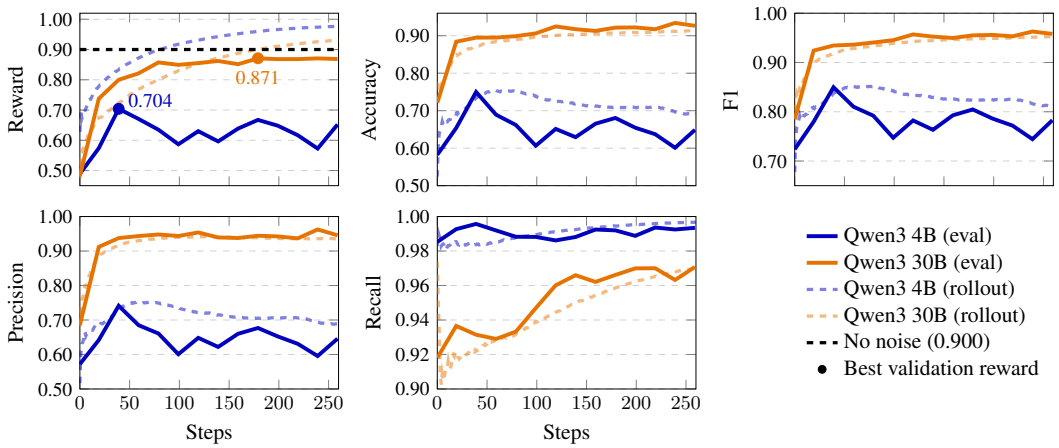

Figure 6: Training Qwen3 8B with model-based verifier rewards on MBPP. Solid lines show evaluation metrics; dashed lines show exponentially smoothed rollout metrics. **Top row, left to right:** validation reward (against ground-truth unit tests) with rollout reward from the model-based verifier, verifier accuracy, and verifier F1 score. **Bottom row:** verifier precision and recall. The Qwen3 4B verifier (blue) shows lower precision than the 30B verifier (orange) on their respective policy outputs, despite high recall. Their trained policies reach peak validation rewards of 0.704 and 0.871, respectively. Each verifier has one training run; this association does not isolate an effect of precision.

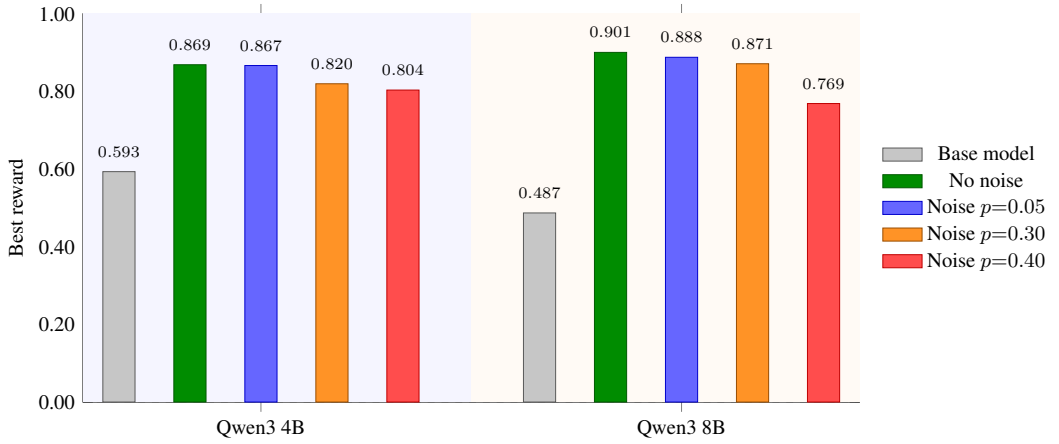

Figure 7: Best validation reward for Qwen3 4B and 8B under group rollout noise in the original model-size comparison. This figure retains the original 8B cohort, rather than the expanded cohort in Table 5; the ordering at p=0.40 changes with that expansion. The comparison does not establish a model-capacity effect.

this comparison does not isolate the relative effects of false positives and false negatives. Each judge is measured on outputs from the policy trained with that judge. Precision depends on the prevalence of correct responses, so the measured precision difference can reflect both verifier behavior and differences between the trained policies. The 4B verifier's F1 score across evaluation checkpoints is approximately 0.72–0.85; F1 values from other tasks, such as those of He et al. (2025), are not directly comparable because the data and error distributions differ.

I Convergence Rate

Figure 8 compares the training curves of the clean baseline and group rollout noise at p=0.1. We extend this comparison across noise levels in Figure 9 and report fixed-step summaries in Table 8,

using all 37 expanded Qwen3 8B runs. The mean over evaluation steps 239 and 259 is 0.890 without noise and 0.884 at p=0.10. It falls to 0.870 at p=0.30, 0.806 at p=0.40, and 0.355 at p=0.50, with substantial variation across seeds at the highest rate. Thus the small observed mean differences at moderate noise also occur without selecting each run's peak. The intervals still permit larger losses: at p=0.20, the late-training difference is approximately -0.6 percentage points with a 95% interval of $[-2.9, +1.8]$. These finite-budget curves establish neither noninferiority within one point nor asymptotic convergence; some runs improve after step 259, and the expanded peak results include later observations where available.

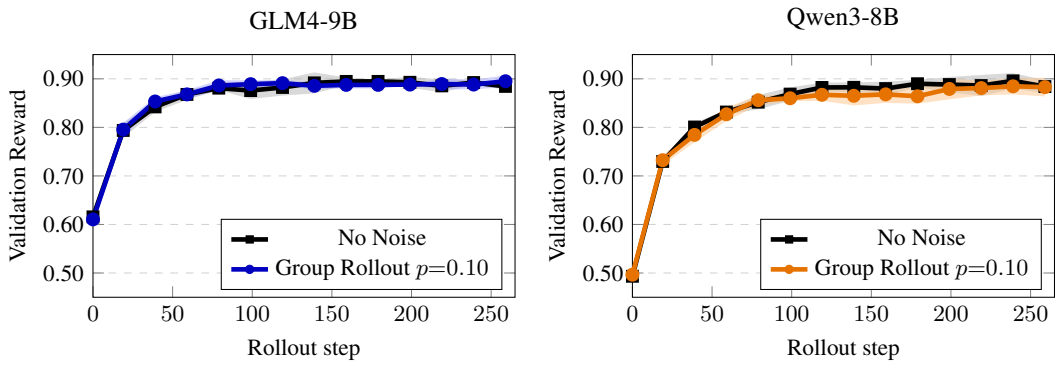

Figure 8: Recovered training curves for group rollout noise at p=0.1 through the common rollout step 259. Shaded regions indicate ± 1 sample standard deviation across seeds.

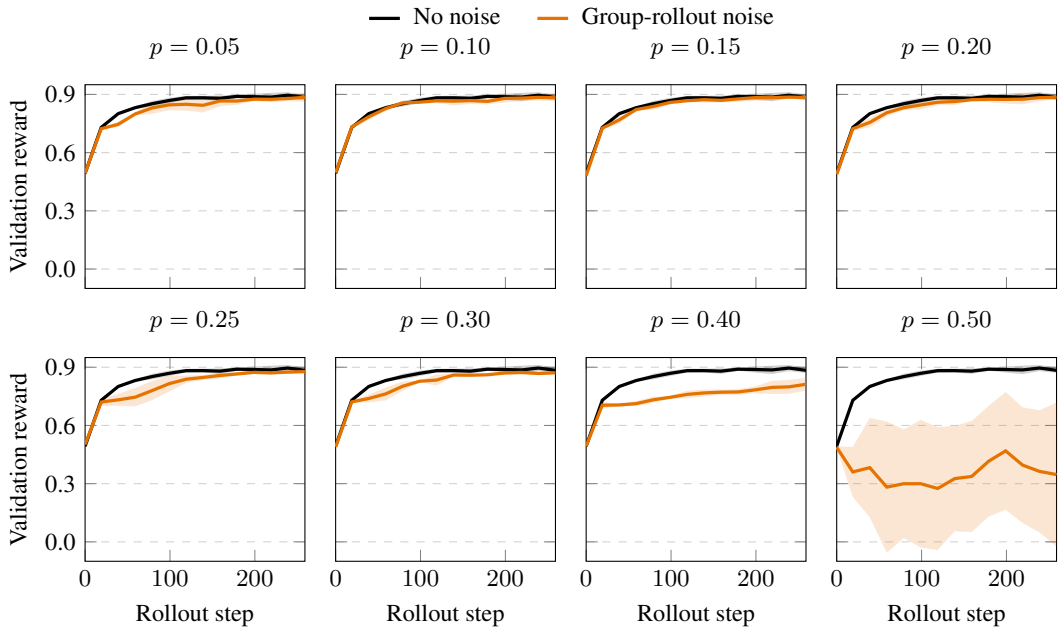

Figure 9: Qwen3 8B validation trajectories for the expanded seed cohort in Table 5, compared at common rollout steps. Each panel compares one group-rollout noise rate with the clean baseline. Bands show sample standard deviations where at least two seeds are available. Some resumed runs have no early evaluations, so contributing seed counts increase along a curve; all 37 runs have the last two evaluations.

Table 8: Qwen3 8B mean unit-test rewards for the expanded cohort at fixed evaluation steps. The late-training metric averages steps 239 and 259 within each seed before aggregation; ± denotes sample SD across seeds. Δ late is its difference from the clean baseline with a Welch 95% confidence interval; at p=0.50 the normal-theory interval extends beyond the attainable range because of the large variation across seeds. The best column uses available evaluations through 259 and still has unequal checkpoint-selection opportunities.

Noise rate	n	Best through 259	At step 259	Mean of 239, 259	Δ late (pp) [95% CI]
0	4	0.900 ± 0.010	0.884 ± 0.013	0.890 ± 0.014	—
0.05	3	0.888 ± 0.022	0.885 ± 0.023	0.882 ± 0.018	$-0.8\ [-4.3, +2.7]$
0.10	6	0.891 ± 0.015	0.883 ± 0.019	0.884 ± 0.018	$-0.6\ [-3.0, +1.8]$
0.15	4	0.891 ± 0.009	0.884 ± 0.009	0.885 ± 0.008	$-0.5\ [-2.7, +1.7]$
0.20	4	0.893 ± 0.012	0.885 ± 0.007	0.884 ± 0.012	$-0.6\ [-2.9, +1.8]$
0.25	4	0.884 ± 0.005	0.878 ± 0.010	0.876 ± 0.007	$-1.4\ [-3.5, +0.8]$
0.30	4	0.880 ± 0.009	0.871 ± 0.006	0.870 ± 0.004	$-2.0\ [-4.2, +0.2]$
0.40	4	0.817 ± 0.035	0.812 ± 0.032	0.806 ± 0.031	$-8.4\ [-13.1, -3.8]$
0.50	4	0.599 ± 0.113	0.347 ± 0.371	0.355 ± 0.339	$-53.5\ [-107.3, +0.3]$

J Model size

Figure 7 compares Qwen3 4B and Qwen3 8B under group rollout noise in the original model-size cohort. At p=0.05, both models' recorded peak scores are close to their clean baselines. The original comparison gives 0.769 for 8B and 0.804 for 4B at p=0.40, but the expanded 8B peak mean is approximately 0.817 (Table 5), reversing that ordering. The cohorts and checkpoint-selection opportunities differ, and the 4B results lack a corresponding expanded analysis. These descriptive observations do not support a conclusion that either size is intrinsically more robust to noise.

K Response length

Figure 10 shows the median response length of Qwen3 8B and GLM4 9B during training with clean rewards and with group rollout noise at p=0.1. Both conditions show decreasing median lengths with qualitatively similar trajectories. This comparison does not establish that noise has no effect on response length or on the rate of change.

Why does the response length decrease over time? One of the key observations in the DeepSeek R1 paper was that the model tends to generate longer responses during training, which they link to the model's ability to solve harder problems more accurately (DeepSeek-AI et al., 2025, Figure 1 (b)). Specifically, the authors write the following.

> As shown in Figure 1(b), DeepSeek-R1-Zero exhibits a steady increase in thinking time throughout training, driven solely by intrinsic adaptation rather than external modifications. Leveraging long CoT, the model progressively refines its reasoning, generating hundreds to thousands of tokens to explore and improve its problem-solving strategies.
>
> The increase in thinking time fosters the autonomous development of sophisticated behaviors. Specifically, DeepSeek-R1-Zero increasingly exhibits advanced reasoning strategies such as reflective reasoning and systematic exploration of alternative solutions. (DeepSeek-AI et al., 2025, p. 5)

However, Liu et al. (2025) noted that output length does not imply better downstream performance. In qualitative inspection of early rollouts, we observe long, sometimes repetitive reasoning chains and responses that exhaust the token budget before producing an answer. Paragraphs often start with expressions such as "Alternatively," "But wait," and "Wait." These observations do not establish why the models produce these patterns.

Shorter responses may help the models produce complete answers within the token budget. Missing-code penalties remain uncorrupted, so mixed groups retain an incentive to produce an extracted code

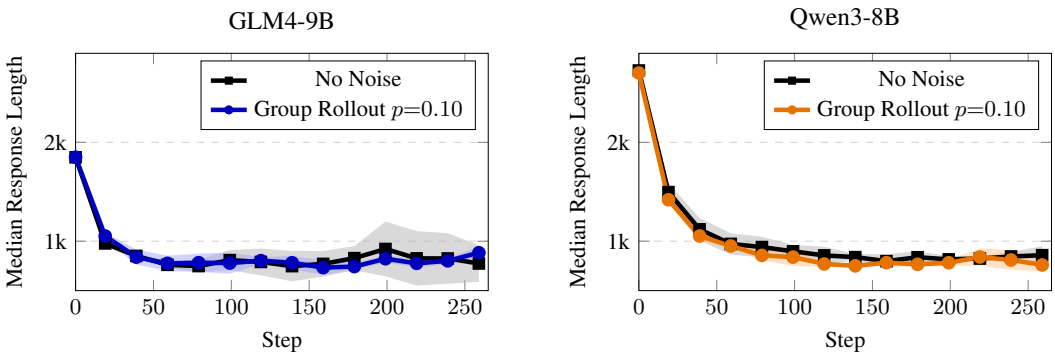

Figure 10: Median response length over training steps for group rollout noise at p=0.1 compared to the no-noise baseline. Shaded regions indicate ± 1 standard deviation across seeds.

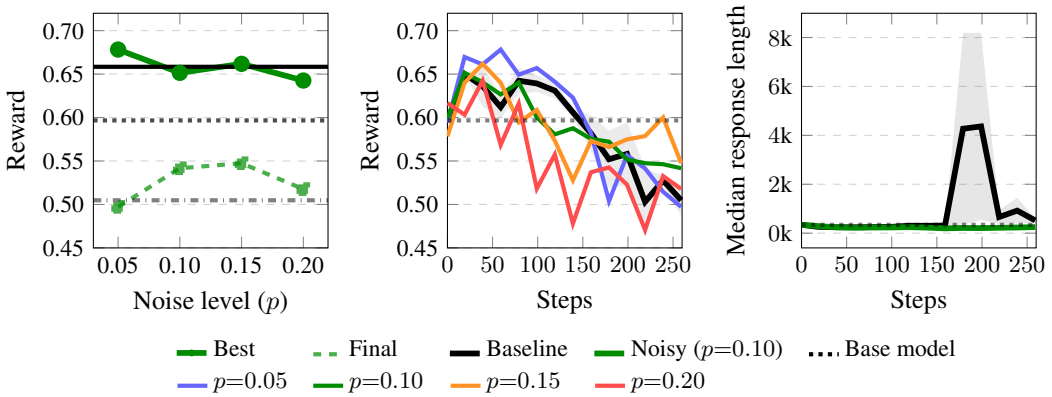

Figure 11: Llama 3.1 8B with group rollout noise on MBPP. **Left:** best and final validation reward vs. noise level. **Center:** training curves by noise level. **Right:** median response length comparison.

block even when correctness rewards are highly noisy (Appendix F). We do not quantify truncation rates, format compliance or correctness conditional on a complete answer. Consequently, these results do not separate improvements in presenting an answer from improvements in solving the task.

L Llama results

We include a subset of experiments with Llama 3.1 8B as a check beyond the Qwen/GLM model families. Figure 11 shows its group rollout noise sweep and training curves. The clean baseline reaches a best validation reward of 0.658 under our protocol. The noisy runs' peaks are nonmonotonic across the tested rates, ranging from 0.643 at p=0.20 to 0.678 at p=0.05. Final scores are substantially below the peaks in both clean and noisy conditions. Each noisy setting has one seed, so these observations do not establish a beneficial effect of noise, equivalent performance or general robustness across model families.

M Cross-study evaluation comparisons

Cai et al. (2025) report mathematical reasoning results under clean and noisy verifier rewards, including moderate asymmetric noise, and compare reward-correction methods. Their results are relevant to our study. Different outcomes across studies can reflect different error structures, training procedures and evaluation protocols; they do not by themselves establish a contradiction.

Table 9 collects previously tabulated comparisons with the Qwen2.5-Math and DeepSeek-R1 technical reports (Yang et al., 2024; DeepSeek-AI et al., 2025). The ranges run from the reported base-model

Table 9: Previously tabulated pass@1 comparisons for Qwen2.5-Math-1.5B, Qwen2.5-Math-7B and DeepSeek-R1-Distill-Qwen-1.5B. References are the corresponding technical reports (Yang et al., 2024; DeepSeek-AI et al., 2025); ranges span the base and clean-reward post-training scores in Figure 2 of Cai et al. (2025). Evaluation protocols and benchmark variants have not been matched, so these numbers do not establish inconsistent evaluation or negative training effects.

	Qwen2.5-Math-1.5B		Qwen2.5-Math-7B		DeepSeek-R1-Distill-Qwen-1.5B	
	Reference	Range	Reference	Range	Reference	Range
Math	69.4	[48, 70]	75.1	[52, 80]	83.9	[60, 78]
MinervaMath	29.4	[6, 18]	34.6	[10, 25]		
Olympiad Bench	31.3	[24, 32]	38.2	[26, 38]		
AIME 2024	3.3	[7, 16]	13.3	[12, 30]	28.9	[10, 20]
AMC 2023	45.0	[34, 50]	62.5	[45, 63]		

scores to the scores after training with clean rewards; they are not uncertainty intervals. Among the 12 populated comparisons, five reference scores exceed the upper endpoint, including one difference of only 0.2 percentage points at the displayed precision.

We have not matched prompts, decoding settings, response-length budgets, answer extraction or benchmark variants across these sources. The table therefore cannot diagnose invalid evaluation or show that training worsens a model relative to its own baseline. Our Qwen3 baseline also differs from its technical report under our protocol, as noted above. A valid comparison of training effects uses a common protocol within each study; the cross-study discrepancies are a comparability limitation rather than grounds for excluding prior findings.

N Ackley function optimization

N.1 Setup

This illustration uses additive perturbations of scalar rewards, not the whole-group outcome complementation analyzed in Appendix F. The available archive does not recover the numerical policy standard deviation, group size, learning rate, initialization radius, or the distribution and across-response dependence of the additive noise. The original executable implementation is also unavailable in the archive. The equations below describe the recorded update construction but are insufficient to reproduce the plotted trajectories exactly.

The "policy" is a diagonal Gaussian with fixed standard deviation:

$$\pi(\cdot \mid \mu) = \mathcal{N}(\mu, \sigma^2 I), \qquad \mu \in \mathbb{R}^2.$$

μ is the only learnable parameter, optimized by Adam (Kingma & Ba, 2015).

N.2 Forward pass (one iteration)

N.2.1 Sampling

Samples are generated with μ *detached* from the computation graph:

$$s_i = \mu_{\text{detach}} + \sigma\, \varepsilon_i, \qquad \varepsilon_i \sim \mathcal{N}(0, I), \quad i = 1, \ldots, G.$$

Because μ is detached, the samples s_i are constants with no gradient connection to μ.

N.2.2 Rewards and advantages

The Ackley cost is evaluated on these constant samples and converted to group-relative advantages:

$$r_i = -\operatorname{ackley}(s_i) + \eta_i, \qquad A_i = \frac{r_i - \bar{r}}{\operatorname{std}(r) + 10^{-8}}.$$

Here η_i denotes the additive reward perturbation; its dependence across responses is not specified in the available records. A perturbation shared identically by every response would cancel during group centering. There are no gradients through the rewards or advantages.

N.2.3 LOG-PROBABILITIES (WHERE GRADIENTS ENTER)

A distribution $\mathcal{N}(\mu, \sigma^2 I)$ is constructed with the *live* μ, and log-probabilities are evaluated on the constant samples:

$$\log \pi(s_i \mid \mu) = \sum_{d=1}^{2} \left[-\frac{(s_{i,d} - \mu_d)^2}{2\sigma^2} - \log \sigma - \tfrac{1}{2} \log 2\pi \right].$$

The above expression is the only point where μ enters the computation graph.

N.2.4 POLICY LOSS

The loss is the standard REINFORCE objective with group-relative advantages:

$$L = -\frac{1}{G} \sum_{i=1}^{G} A_i \cdot \log \pi(s_i \mid \mu).$$

Differentiating the log-probability with respect to μ_d:

$$\frac{\partial \log \pi(s_i \mid \mu)}{\partial \mu_d} = \frac{s_{i,d} - \mu_d}{\sigma^2}.$$

Since $s_{i,d} - \mu_d = \sigma\, \varepsilon_{i,d}$ (the detached and live μ values share the same value at evaluation time):

$$\frac{\partial \log \pi(s_i \mid \mu)}{\partial \mu_d} = \frac{\varepsilon_{i,d}}{\sigma}.$$

N.3 GRADIENT

$$\nabla_\mu L = -\frac{1}{G\,\sigma} \sum_{i=1}^{G} A_i\, \varepsilon_i$$

Each step pushes μ toward samples that received above-average rewards and away from below-average ones, weighted by the normalized noise directions. Adam applies adaptive learning rates on top of this raw gradient.