Extended Abstract

MaxRL with Imperfect Reward Signals on Countdown

Petru Cristian Budianu, Nicolas Bejar

Motivation. Reinforcement learning with verifiable rewards assumes a correct verifier, but real verifiers make mistakes. Maximum Likelihood RL (MaxRL) Tajwar et al. [2026] is especially exposed: its advantage up-weights rare successes by $\frac{N-K}{K}$, largest exactly when successes are scarce, so a single mislabeled success on a hard prompt receives the maximum possible weight and pulls the policy toward an incorrect solution. The original paper assumes a clean verifier and does not study noisy rewards. We ask: how fragile is MaxRL under imperfect supervision, and can simple modifications restore robustness without hurting clean performance?

Method. We study two complementary families of mitigations.

(1) Gradient- and diversity-side. We fine-tune Qwen2.5-0.5B on Countdown with the binary-reward MaxRL advantage, injecting symmetric label-flip noise during training only (evaluation is always clean). Because a full group of N=16 does not fit in memory, we compute the advantage over all 16 responses but run the gradient on a subselected 8 (6 successes + 2 failures). We test four mitigations: curriculum re-weighting (down-weighting low-K groups early), and three subselection strategies that pick the 8 responses by per-token log-prob—a log-prob noise filter (dropping the confidence-extreme samples as suspected noise), a flipped filter (dropping the most confident success, as anti-mode-collapse), and log-prob selection.

(2) Source-side. We introduce an RLAIF agreement gate, a second partially-independent LLM-judge channel combined with the noisy verifier by logical AND so that a rollout counts as a success only when both channels agree, driving the effective false-positive rate from p to $\approx p \cdot q$; we evaluate it on a tabular bandit, on an offline replay of real rollouts, and with a real `Qwen2.5-7B-Instruct` judge.

Implementation. Training runs on a single H100 with vLLM rollouts, group size 16, batch size 32, entropy coefficient 0.1, and 50 steps. We report pass@k on a held-out clean set (32 samples per prompt, unbiased estimator).

Results. Noise damages MaxRL asymmetrically: 10% noise drops vanilla pass@1 from 0.513 to 0.298 but pass@32 only from 0.780 to 0.740. Curriculum re-weighting is the only mitigation to improve both pass@1 and pass@32 under noise; the log-prob filter recovers pass@1 (0.470) but gives the weakest tail (0.700), while the flipped filter recovers the tail (0.780).

The offline analysis shows plain MaxRL routes up to 42% of its gradient weight to false positives; the RLAIF gate cuts this $\sim 7\times$ under an idealized judge, and with a real judge (measured false-positive rate $q_{\text{fp}} \approx 0.25$) it still removes most of the contamination ($\sim 2.5\times$).

Discussion. The mitigations that help do not detect noise; they preserve solution diversity. The one detection-based filter recovers pass@1 only by discarding genuine rare successes, which is why it costs the tail performance. We attribute the curriculum's benefit to its down-weighting of the low-K groups where noise distorts the advantage the most.

The complementary source-side gate instead cleans the success set before it enters the gradient, and its benefit is governed by the conditional independence of the second channel.

Conclusion. For MaxRL under verifier noise, robustness is primarily about protecting solution diversity rather than detecting corrupted labels: curriculum re-weighting is the most effective single end-to-end intervention we found, and a lightweight agreement-gated AI-feedback channel is a promising source-side complement, all while leaving the elegant MaxRL objective untouched.

MaxRL with Imperfect Reward Signals on Countdown

Petru Cristian Budianu
Department of Computer Science
Stanford University
cbudianu@stanford.edu

Nicolas Bejar
Department of Computer Science
Stanford University
nbejar@stanford.edu

Abstract

Reinforcement learning with verifiable rewards assumes a correct verifier, but real verifiers make mistakes. We study Maximum Likelihood RL (MaxRL) under reward-label noise on the Countdown arithmetic task with a 0.5B-parameter policy, focusing on the binary-reward advantage whose amplification of rare successes is both what makes MaxRL effective and what makes it sensitive to mislabeled rewards.

We find that noise damages MaxRL asymmetrically: it sharply degrades pass@1 (confident single-shot accuracy) while degrading pass@k at large k (whether a correct solution exists anywhere in the sampled set) much less. We evaluate two complementary families of mitigations. Gradient- and diversity-side interventions (a noise-aware curriculum and three sampling filters), trained end-to-end on real Countdown, help by preserving solution diversity rather than by detecting corrupted labels; indeed, the one method that explicitly attempts detection recovers pass@1 performance by discarding genuine rare successes, at the cost of the tail performance; curriculum re-weighting is the only intervention to improve both pass@1 and pass@32.

Separately, a source-side *RLAIF agreement gate* (a second, partially-independent LLM-judge reward channel combined with the noisy verifier via logical AND) provably reduces the effective false-positive rate from p to $\approx p\,q$ and, in an offline analysis of real rollouts, cuts misallocated gradient weight by up to $\sim 7\times$; a run with a real `Qwen2.5-7B-Instruct` judge confirms the effect survives a realistic judge error rate. All methods are implemented as an opt-in extension that leaves the base MaxRL objective unchanged.

1 Introduction

Reinforcement learning with verifiable rewards (RLVR) has become a standard recipe for improving the reasoning ability of large language models: a model samples candidate solutions, an automatic verifier labels each as correct or incorrect, and a policy-gradient update reinforces the correct ones. Maximum Likelihood RL (MaxRL) Tajwar et al. [2026] is a recent instance of this recipe that replaces the usual policy-gradient advantage with one derived from a maximum-likelihood objective, optimizing $J_{\mathrm{ML}} = \mathbb{E}_x[\log p_\theta(\text{success} \mid x)]$ rather than the expected reward $J_{\mathrm{RL}} = \mathbb{E}_x[p_\theta(x)]$ used by RLOO/GRPO. Its advantage up-weights a group's successful samples by a factor that grows as successes become rarer, which improves how efficiently the policy concentrates probability on correct solutions and, in particular, improves pass@k at test time. For binary-outcome problems this objective has a striking gradient identity,

$$\nabla_\theta J_{\mathrm{ML}}(x) \;=\; \sum_{k=1}^{\infty} \frac{1}{k}\, \nabla_\theta\, \mathrm{pass@}k(x),$$

which, with a finite group of G rollouts, is equivalent to a policy gradient that weights each successful trajectory by $1/S$, where S is the number of successes in the group.

A central assumption behind RLVR, and behind MaxRL specifically, is that the verifier is correct. In practice, verifiers are imperfect: they accept some incorrect answers (false positives) and reject some correct ones (false negatives). This raises a natural question for MaxRL in particular, because the very mechanism that makes it effective, amplifying rare successes, is also a mechanism for amplifying mislabeled ones.

In this work we study how MaxRL behaves under controlled reward-label noise on the Countdown arithmetic task, using a small (0.5B-parameter) policy, and ask whether simple modifications to the training procedure can improve robustness without sacrificing clean performance. We find that label noise damages MaxRL asymmetrically across the pass@k curve, degrading pass@1 (confident single-shot accuracy) sharply while leaving pass@k at large k comparatively intact. The interventions that improve robustness do not detect or correct noise; they preserve solution diversity, and the one intervention that explicitly tries to detect corrupted labels from model confidence helps least at the tail.

We make three contributions. First, we **characterize** MaxRL's noise fragility under a symmetric reward-flip model, both through real single-H100 Countdown training (clean held-out pass@k) and, separately, by replaying real grouped rollouts and measuring the fraction of MaxRL gradient weight routed to false positives. Second, we evaluate **gradient- and diversity-side mitigations** (a curriculum that down-weights low-K groups and three log-prob subselection filters) and find that diversity preservation, not noise detection, is the operative mechanism for robustness. Third, we propose a **source-side RLAIF agreement gate**: a second, partially-independent reward channel from an off-the-shelf LLM judge, combined with the noisy verifier by logical AND, which provably reduces the effective false-positive rate from p to $\approx p\,q$. All methods are implemented as an opt-in extension that leaves the base MaxRL objective, and every default project component, untouched. We present these as preliminary findings from single-run experiments under tight compute constraints; we are explicit throughout about the resulting statistical limitations and about which patterns we believe are robust to them.

2 Related Work

Reinforcement learning with verifiable rewards. RLVR, popularized by the training of strong reasoning models such as DeepSeek-R1 DeepSeek-AI et al. [2025], optimizes a policy against a rule-based or reference-based verifier that returns a binary correctness signal, typically using a group-relative policy-gradient method such as GRPO or RLOO Ahmadian et al. [2024]. Binary, outcome-based rewards reduce the risk of reward hacking relative to learned reward models and have proven effective in mathematical reasoning and code generation. Our work takes this paradigm as its starting point and studies a specific algorithm within it, MaxRL, under imperfect rewards.

Maximum Likelihood RL. MaxRL Tajwar et al. [2026] observes that standard RL does not maximize the likelihood the policy implicitly assigns to correct rollouts, but only a lower-order approximation, and introduces a family of sample-based objectives that interpolate between standard RL and exact maximum likelihood as more sampling compute is allocated. The resulting advantage admits a simple unbiased policy-gradient estimator and, empirically, improves pass@k and test-time scaling efficiency relative to GRPO. A property emphasized by the authors is that MaxRL *preserves solution diversity*: where standard RL training tends to raise pass@1 while collapsing pass@k at large k (losing coverage on the hardest prompts), MaxRL maintains that coverage. This behavior comes from up-weighting rare successes, the higher-order terms in the objective that become important precisely when the success probability is small. We make two observations about this design in the noisy-reward setting. First, the rare-success amplification that the authors introduce deliberately is also, under reward-label noise, the mechanism that most amplifies a mislabeled success. Second, the diversity preservation that MaxRL is valued for is exactly the property our robustness interventions turn out to protect. We use the binary-reward form of the MaxRL advantage and study its sensitivity to reward-label noise, a setting the original work does not focus on; we note that "noise" in the MaxRL paper refers to gradient-estimator variance (addressed by baseline subtraction), which is distinct from the reward-label noise we study here.

Learning under imperfect or noisy verifiers. Recent work studies RLVR with unreliable verifiers via gradient corrections under a known or estimated noise channel Cai et al. [2026], Everitt et al.

[2017], Wang et al. [2020], or by relabeling suspected-noisy labels with the model's majority answer Yang et al. [2026], with mixed evidence on RLVR's noise tolerance Plesner et al. [2026], Zhu and Kang [2026]. We instead assume an *unknown* flip rate on a *binary success indicator* and test lightweight, noise-agnostic modifications plus a source-side gate, finding that confidence-based filtering—the approach most analogous to detection—fails at the tail.

RL from AI feedback. RLAIF Lee et al. [2024] replaces human preference labels with labels from an off-the-shelf LLM and matches RLHF on summarization and dialogue; the d-RLAIF variant uses continuous LLM scores directly. We repurpose RLAIF not as a preference source but as a *denoising channel*: a second, partially-independent label that, gated against the verifier, attacks the success-set corruption specific to MaxRL. We are not aware of prior work using RLAIF this way for verifier-based reasoning RL.

Preference and REINFORCE-style RL, and curriculum learning. RLOO Ahmadian et al. [2024] revisits leave-one-out REINFORCE baselines for LLM alignment and is our expected-reward reference point. IPO Azar et al. [2024] and RLHF Ouyang et al. [2022] optimize from preference data; we use an IPO checkpoint only as a warm start and noise-free comparison, not as part of the extension. Down-weighting unreliable examples early and re-introducing them as the model improves is a classic curriculum idea Bengio et al. [2009]; our curriculum specializes it to the low-K groups where reward noise most distorts the MaxRL advantage.

3 Method

3.1 Maximum Likelihood RL and the advantage transformation

We study Maximum Likelihood RL (MaxRL) Tajwar et al. [2026], a variant of policy-gradient fine-tuning that replaces the standard RLOO advantage with one derived from a maximum-likelihood objective. For a prompt with a group of N sampled responses, let K be the number of responses that receive reward $r_i = 1$ (a verified success), with the remaining $N - K$ receiving $r_i = 0$. Writing the group mean as $\bar{r} = K/N$, the MaxRL advantage is

$$A_i = \frac{r_i - \bar{r}}{\bar{r}} \implies A_i = \begin{cases} \dfrac{N-K}{K}, & \text{if } r_i = 1 \text{ (success)}, \\ -1, & \text{if } r_i = 0 \text{ (failure)}. \end{cases} \tag{1}$$

The successes in a group are up-weighted by $\frac{N-K}{K}$, which grows large precisely when successes are rare (K small). This amplification of rare successes is the property that makes MaxRL effective at concentrating probability mass on correct solutions, but it is also, as we show, the property that makes MaxRL sensitive to reward-label noise: a single corrupted label changes K, and the change matters most exactly when K is small and the $\frac{N-K}{K}$ factor is large.

3.2 Reward-label noise model

We model an imperfect verifier as a binary label-flip channel applied to the rewards during training only; evaluation always uses the clean verifier. We treat the verifier output as a binary success indicator $b_i = \mathbf{1}[r_i \geq \tau]$ with threshold τ=0.5 (Countdown scores lie in $\{0.0, 0.1, 1.0\}$, so τ cleanly separates fully correct rollouts). After the true binary reward is computed, each label is independently flipped with probability p_{flip}, $\tilde{b}_i = b_i \oplus \text{Bernoulli}(p_{\text{flip}})$. In the symmetric setting used for our main experiments, a true success is recorded as a failure (a false negative) and a true failure as a success (a false positive) with the same probability p_{flip}. We report results primarily at $p_{\text{flip}} = 0.1$, with $p_{\text{flip}} = 0$ as the clean reference. The asymmetry that matters for MaxRL is the false positive: a $0 \to 1$ flip injects a spurious member into the conditional success set.

3.3 Group subselection

A full group of $N = 16$ responses does not fit in GPU memory for the forward/backward pass on the single H100 available to us. We therefore decouple advantage estimation from the gradient computation. The advantage in Eq. (1) is computed over the full group of 16 (which requires only the rewards and is memory-cheap), while the forward/backward pass is run on a subset of 8 responses

per group: we keep 6 successes and 2 failures when available, backfilling randomly from the other class if a group is short of either. This keeps the per-step memory footprint fixed while preserving the correct K/N statistics in the advantage. The choice of *which* 8 responses to keep is itself a design element we study (Section 3.4).

3.4 Gradient- and diversity-side mitigations

We first evaluate some methods that act on the policy gradient and the selection of responses. These techniques are split in two categories: curriculum reweighting and sub-selection strategies.

Curriculum reweighting. A corrupted label distorts $\bar{r} = K/N$ most when K is small, so we down-weight the contribution of low-K groups early in training and admit them gradually. Each group's loss contribution is scaled by $w = \min\big(1, (K/K_{\min})^{K_{\min}}\big)$, where $K_{\min}$ is relaxed from 4 toward 1 over the course of training: $K_{\min} = 4$ for the first 20 epochs and decreasing by 1 each 10 epochs. Groups with $K \geq K_{\min}$ receive full weight; groups with $K = 0$ receive zero weight. We experimented with a different variation law as well, however we kept this one due to better results.

Log-prob noise filter. Motivated by the hypothesis that corrupted labels can be detected from model confidence, this filter drops, within each group, the lowest per-token-log-prob success and the highest per-token-log-prob failure (the responses most likely to be a false positive and a false negative respectively) before computing the advantage (Figure 1, top).

Flipped filter. The flipped filter instead drops the *most confident* success in a group. It is not a noise-detection mechanism; it is an anti-mode-collapse intervention, intended to prevent the policy from collapsing onto a single high-probability solution and thereby to preserve solution diversity (Figure 1, middle).

Log-prob selection. Rather than randomly backfilling the $6{+}2$ subselection, this variant chooses the retained successes and failures by per-token log-prob rank. It changes which responses drive the gradient without changing the group-level advantage (Figure 1, bottom).

3.5 Source-side mitigation: the RLAIF agreement gate

The mitigations above act *after* corrupted labels enter the success set. We also study a source-side alternative that removes them first. Let the (noisy) verifier emit indicator $\tilde{b}^v$ with false-positive rate

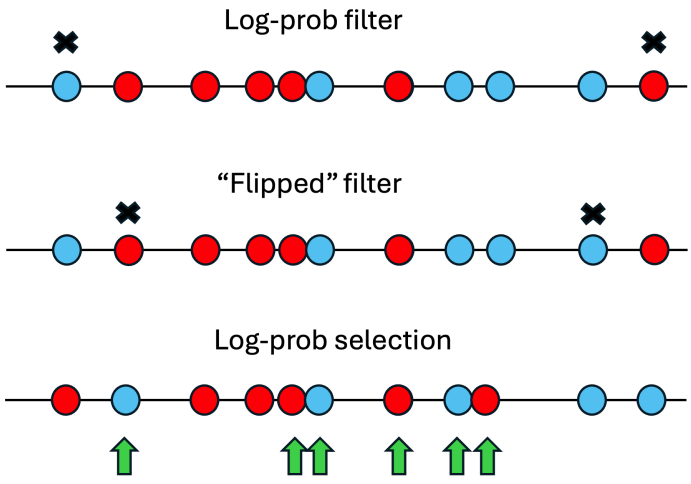

Figure 1: Three selection mechanisms; successes in blue, failures in red, ordered by log-probability (low to high).

p, and let an LLM judge emit a second indicator $\tilde{b}^j$ with error rate q. We define the gated success indicator by logical AND,

$$b_i^{\text{gate}} = \tilde{b}_i^v \wedge \tilde{b}_i^j.$$

A false positive survives the gate only if *both* channels err. Under the assumption that the two channels' errors are conditionally independent given the true label, the effective false-positive rate drops from p to $p\,q$; with general error correlation ρ we show the gate yields an effective rate

$$p_{\text{eff}} = \rho\,p + (1-\rho)\,p\,q,$$

which interpolates between the independent ideal ($\rho{=}0 \Rightarrow p\,q$) and a useless gate ($\rho{=}1 \Rightarrow p$). We also implement OR, majority-vote, and soft-average combiners and a real `VLLMJudge` that re-evaluates each candidate equation; for controlled sweeps we use a `SimulatedJudge` whose error rate and correlation with the verifier are explicit parameters (`extension/rlaif.py`). The agreement gate is deliberately conservative: it trades a small increase in false *negatives* for a large reduction in the false *positives* that MaxRL is uniquely sensitive to.

3.6 A statistical curriculum for the offline analysis

For the offline and toy analyses of the agreement gate (§5.3–5.6) we also use a statistical variant of the curriculum above, distinct from the K-based training curriculum of Section 3.4. The curriculum (`extension/curriculum.py`) maintains an EMA estimate $\hat{q}_x$ of each prompt's success rate and asks whether $\hat{q}_x$ is statistically distinguishable from the noise floor f (the false-positive rate the prompt would exhibit even if unsolvable). Modeling the floor as a binomial with effective sample size n_{eff}, its standard deviation is $\sigma = \sqrt{f(1-f)/n_{\text{eff}}}$, and we set a smooth gate

$$w_x = w_{\min} + (1 - w_{\min})\,\sigma\Big(\tfrac{\hat{q}_x - (f + \kappa\sigma)}{\max(\beta\sigma,\, w_{\text{width}})}\Big),$$

with $\sigma(\cdot)$ the logistic function. Crucially, at $p{=}0$ the floor and its spread vanish, the midpoint $f + \kappa\sigma \to 0$, and the gate becomes a no-op: every genuine success is kept. When RLAIF is active we feed the curriculum the *lowered* effective floor ($\approx p\,q$) so the two mechanisms compose rather than double-count.

3.7 Trainer integration

The noise model, RLAIF gate, and statistical curriculum are wired into `maxrl_trainer/maxrl.py` and `maxrl_update_worker.py` through a single `_apply_reward_pipeline` stage (verifier scores $\rightarrow$ flip noise $\rightarrow$ judge $\rightarrow$ combiner) plus optional per-prompt curriculum weights multiplied into the existing MaxRL weighting. Every knob is exposed as an argparse flag that defaults OFF; with defaults the pipeline is a literal identity and the trainer is byte-identical to base MaxRL, a property enforced by an explicit CPU integration test (`test_pipeline_noop_when_all_off`).

4 Experimental Setup

Real Countdown training. Our primary experiments fine-tune a Qwen2.5-0.5B policy, initialized from the default supervised fine-tuning (SFT) checkpoint, on the Countdown arithmetic task, in which the model must reach a target integer using a given set of numbers and basic arithmetic operations. The task reward is 1.0 for a correct, well-formed solution, 0.1 for a well-formed but incorrect solution, and 0.0 for a malformed output; for the MaxRL estimator we binarize this to $r = 1$ for a correct solution and $r = 0$ otherwise. Unless otherwise noted, we use group size $N = 16$, batch size 32, gradient accumulation 32 (one group per microbatch), entropy coefficient 0.1, a constant learning rate of $1\text{e}{-5}$, and 50 optimization steps, all on a single H100 GPU. Rollouts are generated with vLLM at sampling temperature 1.0. All runs start from the same SFT checkpoint. We evaluate on a held-out set of Countdown problems using the *clean* verifier, regardless of the training noise level: for each prompt we sample 32 responses at temperature 0.6 (top-p 0.95, top-k 20) and report pass@k computed with the standard unbiased estimator. We report pass@1, pass@16, and pass@32 as representative points on the pass@k curve; pass@1 measures confident single-shot accuracy while larger k measures whether a correct solution exists anywhere in the sampled set, i.e. solution diversity.

Offline and toy analyses of the agreement gate. To probe the source-side gate at scale without the cost of repeated end-to-end training, we use two further regimes. (1) A **controlled tabular bandit** (`experiments/toy_maxrl_noise.py`): a numpy contextual bandit with 48 prompts, 12 actions, group size G=8, trained for 140 steps, where per-prompt difficulty is set so that MaxRL and RLOO are cleanly separable and the noise effect is legible. This isolates the optimization dynamics with no confounds from a real LLM; the absolute pass@1 values are not comparable to the real-training numbers above and should be read only for relative trends. (2) An **offline signal-fidelity study** (`experiments/offline_signal_fidelity.py`) that replays *real* grouped Countdown rollouts from our SFT (`sft_full_e6`) and IPO (`ipo_e1`) evaluation runs (50 prompts $\times$ 16 samples), applies the flip model, and measures the MaxRL weighting directly, without training, so the diagnostic reflects the true success-rate distribution of our actual policy. Both sweep $p \in \{0, 0.001, \ldots, 0.2\}$; the offline study averages over 100 noise seeds. The LLM-judge error rate is fixed at q=0.1 unless noted.

On statistical strength. Because of compute constraints, the real-training results we report are from a single training run per configuration. We observe run-to-run variation on the order of ± 0.02 in pass@k for fixed configurations, and substantially larger variation in pass@1 at higher noise levels. We therefore treat differences within this band as inconclusive and frame our findings as a preliminary study; establishing the effects with multiple seeds and a denser noise sweep is left to future work.

5 Results

5.1 Quantitative evaluation : pass@k on real Countdown training

Figure 2 shows some initial simulation for RLOO, and MaxRL with no noise, and two noise levels. Upon this simulation we continued to test our mitigation strategies for the case of 10% noise. A subset of performance curves for different noise mitigation strategies is shown in Figure 3. Table 1 reports pass@k on the clean held-out evaluation set for each configuration, under no noise and under 10% symmetric training noise. All runs start from the same SFT checkpoint and are evaluated identically.

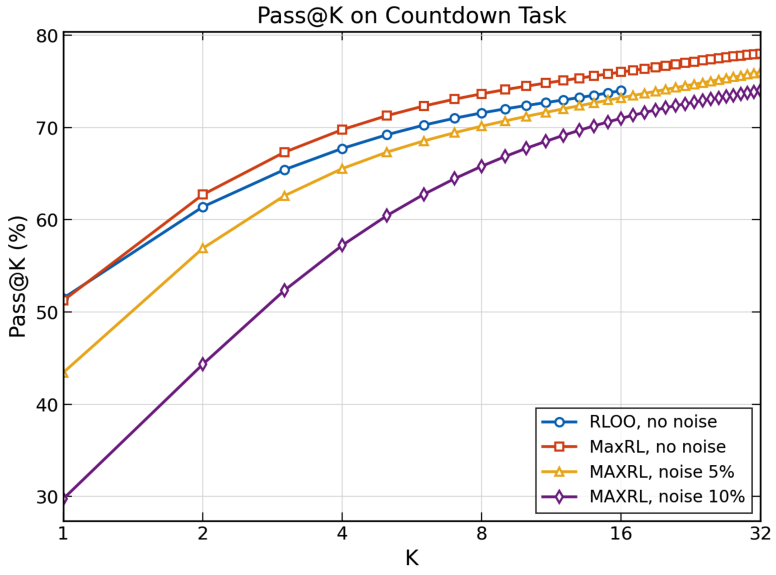

Figure 2: Pass@k on the Countdown task (clean held-out evaluation) for MaxRL under no noise and under 5% and 10% symmetric training noise. Single run per configuration.

Effect of noise on the baseline. Figure 2 compares RLOO and MaxRL on clean data and shows how MaxRL degrades as the training-noise level increases from 0% to 5% to 10%. Comparing vanilla MaxRL across the two blocks of Table 1, 10% label noise reduces pass@1 sharply, from 0.513 to 0.298, while pass@32 falls only modestly, from 0.780 to 0.740. The damage from noise is

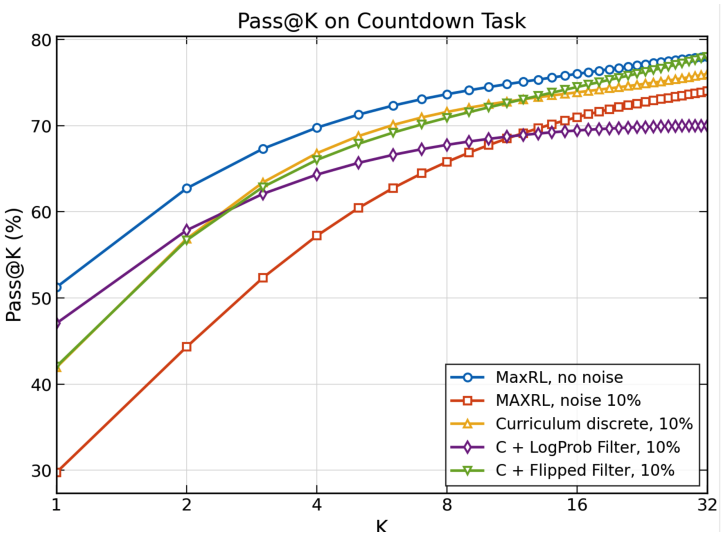

Figure 3: Pass@k on the Countdown task (clean held-out evaluation) for MaxRL under no noise and under 10% symmetric training noise, across mitigation strategies: discrete curriculum, curriculum + log-prob filter, and curriculum + flipped filter (all at 10% noise). The mitigations recover most of the gap between the noisy and clean MaxRL baselines. Single run per configuration.

Table 1: Pass@k on clean held-out data. Higher is better. Top block: no training noise. Bottom block: 10% symmetric training noise (evaluation is always clean). Bold marks the best value in each column within a block. Single run per configuration; differences within ± 0.02 are within run-to-run variation.

Configuration	Pass@1	Pass@16	Pass@32
No noise			
Vanilla MaxRL	0.513	0.760	0.780
Curriculum only	0.510	0.745	0.780
Log-prob selection + curriculum	0.518	**0.772**	**0.800**
Flipped filter + curriculum	**0.522**	0.734	0.760
10% *symmetric noise*			
Vanilla MaxRL	0.298	0.700	0.740
Curriculum only	0.419	0.730	0.760
Log-prob selection + curriculum	0.328	0.718	0.740
Log-prob noise filter + curriculum	**0.470**	0.685	0.700
Flipped filter + curriculum	0.421	**0.745**	**0.780**

thus concentrated at small k: the model's single most confident answer degrades substantially, but a correct solution still appears within the larger sampled set at close to the clean rate.

Effect of the mitigations under noise. Relative to the noisy vanilla baseline, the curriculum recovers pass@1 from 0.298 to 0.419 and the log-prob noise filter recovers it further to 0.470. At the tail, the flipped-filter-plus-curriculum configuration reaches pass@$32 = 0.780$, matching the clean vanilla baseline, while the log-prob noise filter's tail is the lowest in the block at 0.700. On clean data, log-prob selection gives the best tail performance (0.800 pass@32), exceeding clean vanilla. Figure 3 plots these mitigations against the clean and noisy vanilla baselines across k: the curriculum and flipped filter track the clean MaxRL curve closely, while the log-prob filter improves the low-k regime at the cost of the tail.

5.2 Qualitative analysis of the training runs

On a prompt whose responses contain no true successes ($K = 0$), the only way a "success" label can appear is through a false positive. The probability that in such a situation at least one of the N failures

is flipped is $1-(1-p_{\text{flip}})^N$, which for $N=16$ and $p_{\text{flip}}=0.1$ is approximately 0.81. Under Eq. (1), MaxRL then optimizes toward a wholly spurious target on a large proportion of zero-true-success prompts.

Noise attacks confident commitment far more than underlying capability. The central empirical pattern is the asymmetry between the variations of pass@1 and pass@k under noise. A natural interpretation is that pass@1 measures the policy's *confident commitment*, whether its single highest-probability answer is correct, while pass@k at large k measures whether the capability to produce a correct answer is preserved. Our finding is that label noise disrupts the former far more than the latter: corrupted rewards perturb which solution the policy commits to without erasing its ability to generate a correct solution among many samples. This reframes "robustness to reward noise" for MaxRL as the problem of protecting confident commitment (pass@1) and solution diversity (pass@k) at the same time, two objectives that, as our results show, are not always recovered by the same intervention.

Curriculum re-weighting helps both pass@1 and pass@k. The curriculum down-weights low-K groups early in training. This lets the model first reach a stable consensus on the easier problems, where successes are plentiful and genuine, without being pulled by the gradient noise that false positives inject into low-K groups. As training proceeds, the down-weighting is relaxed. Crucially, by this stage a given hard problem is no longer necessarily low-K: the model has learned to solve some of the harder problems with a degree of reliability, so their success counts have risen. The same problems that, early in training, would have had their advantage badly distorted by label noise (because one false positive dominates a small K) now have enough genuine successes that the noise has comparatively little effect. In this way the curriculum protects the early, formative phase of training from noise-dominated gradients while still eventually learning from the harder problems, helping performance at both ends of the pass@k curve.

Robustness comes from diversity preservation, not noise detection. None of the interventions that help under noise actually identifies which labels were corrupted. The curriculum down-weights low-K groups by their K value alone; the flipped filter drops the most confident success regardless of whether it is genuine. Both nonetheless improve noise robustness, and both do so by keeping the policy from collapsing onto a narrow set of solutions. We therefore attribute their benefit to diversity preservation rather than to noise correction, an important distinction, because it predicts that detection-based interventions should be comparatively ineffective.

The log-prob noise filter helps pass@1 but degrades pass@32. The log-prob noise filter explicitly tries to *detect* corrupted labels from model confidence, dropping the lowest-log-prob success and highest-log-prob failure in each group as suspected noise. However, even in the noise-free case, this filter still activates and removes around 20% of groups' extreme samples, which, with no noise present, are by definition genuine. This is the key observation: among the samples it discards are real rare successes. The effect shows in the results: the filter improves pass@1 under noise (to 0.470) but yields the lowest tail performance in the noisy block (pass@$32 = 0.700$). Those genuine rare successes are exactly the diverse samples that the diversity-preserving interventions retain, so the filter trades tail diversity for a pass@1 gain. This negative result was among the most informative in our study: it is the intervention that most directly attempts noise *detection*, and its failure at the tail is what pointed us toward diversity preservation, rather than detection, as the operative mechanism for robustness.

The flipped filter helps in an unexpected way. After observing the pass@32 drop caused by the log-prob filter, we tried the opposite intervention: rather than removing the lowest-confidence success as suspected noise, the flipped filter removes the *most confident* success. This is clearly not a noise-removal mechanism; a high-confidence success is, if anything, the least likely to be a false positive. Its actual effect is anti-mode-collapse: by removing the single most reinforced success, it prevents the policy from concentrating all of its probability mass on one solution, which lifts pass@k close to the clean baseline even under noise.

Comparison. Among the techniques tried, curriculum re-weighting is the only one that improves performance at both pass@1 and pass@32. All filters were applied together with curriculum. The filters fall in two categories, the flipped filter helps the tail while the log-prob filter improves pass@1. The log-prob *selection* does not bring major performance improvements for the noisy case. Finally,

all techniques tested perform similarly without noise, so the differences under noise reflect genuine robustness effects.

5.3 Toy bandit: MaxRL degrades faster than RLOO

The tabular bandit isolates the optimization dynamics. Figure 4 (and Table 2) show toy pass@1 as the flip rate increases. RLOO degrades gracefully ($0.891 \to 0.696$, a 22% relative drop), whereas MaxRL collapses ($0.665 \to 0.253$, a 62% relative drop), roughly $2.8\times$ the relative degradation. The gap is even sharper on the hard-prompt subset (Figure 5), where MaxRL pass@1 falls from 0.514 to 0.071: noise destroys exactly the rare-success signal MaxRL relies on. In this setting the statistical curriculum alone is a mixed result: by down-weighting prompts near the noise floor it also under-trains genuinely hard-but-real prompts, leaving toy pass@1 slightly below MaxRL at low p (Table 2), in contrast to the real-training K-based curriculum of Section 5.1, which is the strongest single mitigation.

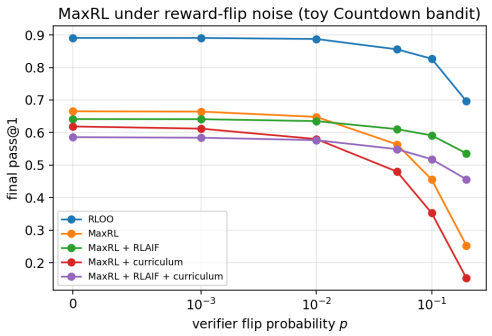

Figure 4: Toy pass@1 vs. flip rate p. MaxRL collapses far faster than RLOO; RLAIF recovers most of the loss.

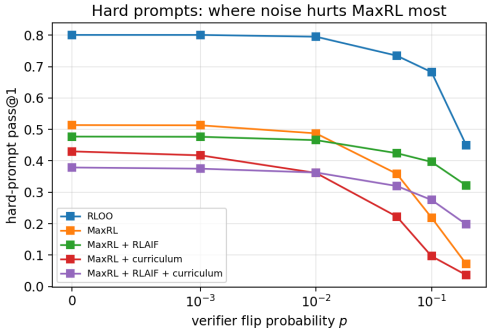

Figure 5: Hard-prompt pass@1 vs. p. The fragility is concentrated on the low-success prompts MaxRL is designed to exploit.

Table 2: Toy bandit pass@1 across flip rates p. RLAIF is the dominant mitigation in this setting; the statistical curriculum alone slightly under-trains. Absolute values are not comparable to the real-training numbers in Table 1.

Method	p=0	0.001	0.01	0.05	0.1	0.2
RLOO (expected-reward)	0.891	0.891	0.888	0.856	0.827	0.696
MaxRL	0.665	0.664	0.648	0.564	0.455	0.253
MaxRL + RLAIF gate	0.641	0.641	0.635	0.610	0.591	**0.536**
MaxRL + curriculum	0.619	0.612	0.580	0.480	0.353	0.153
MaxRL + RLAIF + curriculum	0.586	0.584	0.577	0.549	0.518	0.456

5.4 Where the gradient mass goes: real rollouts

The offline study makes the mechanism concrete. Figure 6 reports the fraction of total MaxRL gradient weight that lands on false positives as a function of p, averaged over 100 seeds on real IPO rollouts. Plain MaxRL routes 4.5% of its weight to false positives at only p=0.01, 30.8% at p=0.1, and 42.1% at p=0.2. The agreement gate caps this at 0.4%, 4.0%, and 9.7% respectively, an $\approx 7\times$ reduction that tracks the predicted $p \to p^2$ scaling. The companion spurious-success-prompt metric (Figure 7) tells the same story: at p=0.1, 80% of MaxRL's previously-zero-success prompts acquire a spurious success, versus 9% under the gate. Results on the SFT rollouts are nearly identical, confirming the effect is a property of the estimator, not of a particular checkpoint.

5.5 Independence, not just a second opinion

The gate's value depends on error *independence*. Figure 8 sweeps the verifier–judge error correlation ρ at fixed p=0.1, q=0.1: recovered pass@1 falls monotonically from 0.591 at ρ=0 to 0.455 at ρ=1,

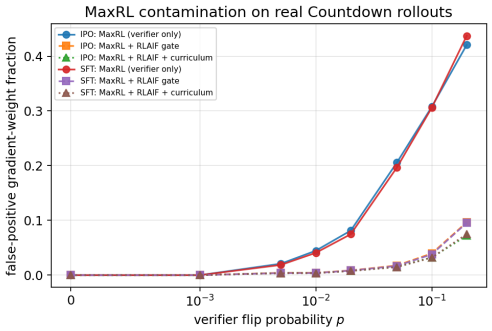

Figure 6: Fraction of MaxRL gradient weight routed to false positives on *real* IPO rollouts. The RLAIF gate cuts misallocated weight $\sim 7\times$.

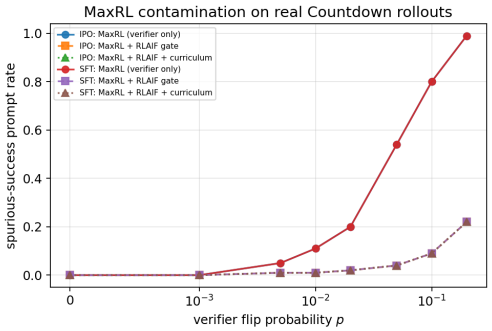

Figure 7: Fraction of truly-unsolved prompts that acquire a spurious success. MaxRL is dominated by false positives well before $p{=}0.1$.

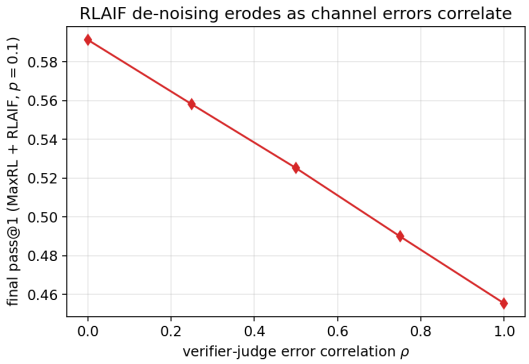

Figure 8: Recovered MaxRL pass@1 vs. verifier–judge error correlation ρ (toy, $p{=}0.1$). The gate's benefit erodes smoothly as the two channels' errors align.

matching the $p_{\text{eff}} = \rho p + (1-\rho)pq$ prediction. A judge that fails on the same inputs as the verifier provides no denoising, an important caveat for deploying RLAIF gates with judges drawn from the same model family as the policy.

5.6 From a simulated judge to a real one

The results above fix the judge error at $q{=}0.1$. To check whether the gate survives a *real* judge, we ran an off-the-shelf `Qwen2.5-7B-Instruct` over every recorded rollout (1,600 responses across the SFT and IPO sets) and graded each candidate equation with a short YES/NO prompt (`experiments/offline_real_judge.py`, one H100 on Modal). Because the Countdown verifier is exact on these clean rollouts, the clean label is ground truth, so we can measure the judge's error directly rather than assume it. The judge accepts $q_{\text{fp}}{=}25.5\%$ of truly-incorrect IPO answers (24.7% for SFT) and wrongly rejects only $q_{\text{fn}}{\approx}6\%$ of correct ones, so the real false-positive rate is roughly $2.5\times$ the $q{=}0.1$ used in the controlled sweep. Even at that higher error, the agreement gate still removes most of the contamination (Figure 9): at $p{=}0.1$ it cuts false-positive gradient weight from 30.8% to 12.4% on IPO ($30.6\% \to 11.9\%$ on SFT), and the spurious-success prompt rate from 80% to 31%. The reduction is about $2.5\times$ rather than the $\sim 7\times$ of the idealized $q{=}0.1$ simulation, exactly as the $p\,q$ scaling predicts once the judge's true q is plugged in. The qualitative claim holds with a real judge; the quantitative benefit is set by the judge's measured accuracy.

6 Discussion

Our results draw a clear line between *magnitude* robustness and *support* robustness. Expected-reward RL is robust to label noise in the usual averaging sense: a few flipped labels perturb a mean. MaxRL

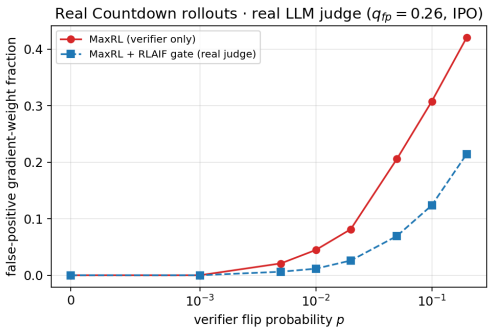

Figure 9: False-positive gradient weight with a *real* `Qwen2.5-7B-Instruct` judge ($q_{\text{fp}} \approx 0.25$) on IPO rollouts. The gate still removes most contamination.

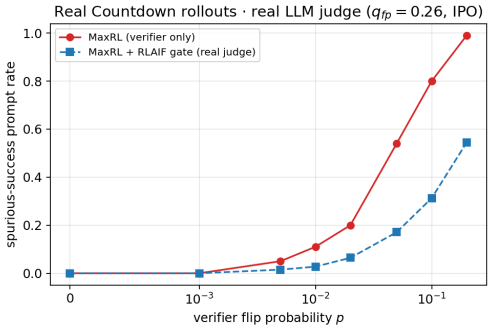

Figure 10: Spurious-success prompt rate under the real judge. Truly-unsolved prompts that acquire a fake success drop from 80% to 31% at p=0.1.

trades that robustness for sample efficiency by conditioning on the success set, and the same $1/S$ weighting that makes rare successes informative makes spurious successes maximally harmful. Two routes follow. The gradient- and diversity-side mitigations do not detect noise; they keep the policy from collapsing onto a narrow set of solutions, which is why curriculum re-weighting and the flipped filter recover pass@k while the detection-based log-prob filter sacrifices it. The source-side agreement gate instead cleans the set MaxRL conditions on before it enters the gradient, and its effectiveness is governed by a single interpretable quantity: the conditional independence of the second channel. We also report the honest negative finding that the detection-based log-prob filter and the statistical curriculum each trade one end of the pass@k curve for the other, and the correlation ablation showing the RLAIF gate's reliance on channel independence.

A further direction concerns the *scheduling* of the filters. Our curriculum re-weighting is applied on a schedule (the threshold $K_{\min}$ relaxes from 4 to 1 over training), but the log-prob and flipped filters were applied at full strength throughout. The mechanism we propose for the curriculum (that the early, noise-fragile phase of training benefits from suppressing unreliable gradient terms, with that suppression relaxed as the policy becomes competent) suggests the same logic should apply to the filters: their optimal strength is likely time-dependent rather than constant. A filter that is aggressive early and relaxed later (or the reverse) may outperform a fixed one. We did not explore filter scheduling, and identify it as a natural extension, alongside placing the real LLM judge inside the end-to-end training loop rather than only in the offline analysis.

7 Conclusion

We studied Maximum Likelihood RL under reward-label noise on the Countdown task and found that noise damages MaxRL asymmetrically: it sharply degrades pass@1 (confident single-shot accuracy); the pass@k at large k (the underlying ability to produce a correct solution) is affected much less.

Among the mitigation techniques we trained end-to-end, curriculum re-weighting was the only one to improve both pass@1 and pass@k under noise, which we attribute to its down-weighting of exactly the low-K groups where label noise most distorts MaxRL's advantage; the interventions that improved robustness did so by preserving solution diversity rather than by detecting corrupted labels, and the one intervention that explicitly attempts detection recovered pass@1, but did so by discarding genuine rare successes, giving it the weakest tail performance.

Complementing these, a lightweight, agreement-gated AI-feedback channel cleans the success set at its source, reducing misallocated gradient weight by up to $\sim 7\times$ offline and $\sim 2.5\times$ with a real judge. These findings are preliminary: the real-training results rest on single runs per configuration, but the qualitative patterns were consistent, and they suggest that robustness to verifier noise in MaxRL is primarily a matter of protecting solution diversity. Beyond the contributions promised in our proposal (noise characterization, curriculum re-weighting, sampling filters, and the RLAIF channel), we added two analyses: a *real-rollout* offline diagnostic that measures gradient contamination directly, and

a real-LLM-judge run that replaces the simulated judge's assumed error rate with a measured one. Establishing these effects with multiple seeds, a denser noise sweep, scheduled filters, and a live training run that places the real judge inside the MaxRL loop are natural directions for future work.

8 Team Contributions

- **Cristian Budianu:** MaxRL implementation and the real single-H100 Countdown training; the symmetric reward-flip noise model and group subselection; the gradient- and diversity-side mitigations (curriculum re-weighting and the sampling filters) and the analysis of the clean held-out pass@k results. Cristian thanks Fahim Tajwar for confirming the idea of studying MaxRL with imperfect rewards.
- **Nicolas Bejar:** SFT/IPO/RLOO pipelines and the vLLM pass@k harness; Modal/Ray training orchestration; design and implementation of the RLAIF agreement-gate channel (simulated and vLLM judges, channel combiners, joint-noise denoising); the offline signal-fidelity study and the real `Qwen2.5-7B-Instruct` judge experiment and figures; trainer integration and the CPU test suite.

References

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting REINFORCE-style optimization for learning from human feedback in LLMs. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024. arXiv:2402.14740.

Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Rémi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. A general theoretical paradigm to understand learning from human preferences. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2024. arXiv:2310.12036.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *International Conference on Machine Learning (ICML)*, 2009.

Xin-Qiang Cai, Wei Wang, Feng Liu, Tongliang Liu, Gang Niu, and Masashi Sugiyama. Reinforcement learning with verifiable yet noisy rewards under imperfect verifiers, 2026. arXiv:2510.00915.

DeepSeek-AI, Daya Guo, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, Sept 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL `http://dx.doi.org/10.1038/s41586-025-09422-z`.

Tom Everitt, Victoria Krakovna, Laurent Orseau, Marcus Hutter, and Shane Legg. Reinforcement learning with a corrupted reward channel. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2017. arXiv:1705.08417.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, and Sushant Prakash. RLAIF vs. RLHF: Scaling reinforcement learning from human feedback with AI feedback. In *International Conference on Machine Learning (ICML)*, 2024. arXiv:2309.00267.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022.

Andreas Plesner, Francisco Guzmán, and Anish Athalye. An imperfect verifier is good enough: Learning with noisy rewards, 2026. arXiv:2604.07666.

Fahim Tajwar, Guanning Zeng, Yueer Zhou, Yuda Song, Daman Arora, Yiding Jiang, Jeff Schneider, Ruslan Salakhutdinov, Haiwen Feng, and Andrea Zanette. Maximum likelihood reinforcement learning, 2026. arXiv:2602.02710.

Jingkang Wang, Yang Liu, and Bo Li. Reinforcement learning with perturbed rewards. *AAAI Conference on Artificial Intelligence*, 2020. arXiv:1810.01032.

Shenzhi Yang, Guangcheng Zhu, Bowen Song, Sharon Li, Haobo Wang, Xing Zheng, Yingfan Ma, Zhongqi Chen, Weiqiang Wang, and Gang Chen. Can llms learn to reason robustly under noisy supervision?, 2026. arXiv:2604.03993.

Yuxuan Zhu and Daniel Kang. Noisy data is destructive to reinforcement learning with verifiable rewards, 2026. arXiv:2603.16140.

A Implementation Details

The extension is a numpy-only package with `torch`/`vllm`/`ray` imported lazily, so every test runs on CPU. The noise RNG is reseeded per step as `seed` $\cdot\, 1{,}000{,}003 +$ `step` for reproducibility. The statistical curriculum uses an EMA decay of 0.9 and an effective sample size $n_{\text{eff}} = G/(1-\text{decay})$ to reflect the EMA window. A Modal launcher (`train_maxrl_noise_modal.sh`) threads all extension knobs through the existing `modal_train.py maxrl` entrypoint via argparse REMAINDER; with default env-vars it launches vanilla MaxRL.

B Reproducibility

All offline experiments are deterministic given a seed and run from the repository root:

```
python maxrl_trainer/extension/experiments/toy_maxrl_noise.py
python maxrl_trainer/extension/experiments/offline_signal_fidelity.py
python maxrl_trainer/extension/tests/test_extension.py
python maxrl_trainer/extension/tests/test_trainer_integration.py
```

The two experiment scripts regenerate every offline figure and the JSON result files used in this report; the two test scripts (17 checks total) validate the modules and the trainer no-op guarantee.

The real-judge experiment requires a GPU and is launched on Modal:

```
modal run maxrl_trainer/extension/experiments/real_judge_modal.py
          -judge-model Qwen/Qwen2.5-7B-Instruct
```

This is the run reported in §5.6 (one H100, $1{,}600$ judgements, a few minutes). Its plumbing and metrics are CPU-checkable without a GPU via the stub backend (`offline_real_judge.py -judge-backend stub`).