

When the Verifier Labels Only Some Samples: A Frozen, a Recalibrated and a Scratch Judge Head in GRPO

DC-Jev-RL project, Part III-B

October 2026

Abstract

When a verifier can label only some of a GRPO run’s samples, a judge can fill in rewards for the rest. We study this on MATH with Qwen3.5-4B-Base, using Part II’s setup: the verifier labels 2 of the 8 completions of each group, and the other 6 are dropped (`rlvr_sub`), rewarded with Kev-4B’s frozen judgment (`sub_kev`), rewarded with Kev-4B after its pointer head is refitted online on the labels collected so far (`sub_kev_recal`), or rewarded with a head of the same shape trained from scratch on the same features and labels (`sub_probe`); two seeds each, 200 steps. The frozen judge ranks the unlabelled completions well (AUROC 0.91) but is too generous, and both of its policies learned to repeat themselves after the answer, which the judge pays for; against dropping the completions they end -2.3 $[-4.4, -0.1]$ points on MATH-500. Refitting the head fixes its calibration (expected calibration error 0.10 against 0.19), but one of its two policies still repeats itself. The head trained from scratch ends $+4.1$ $[1.6, 6.6]$ points against dropping on MATH-500 and $+12.2$ on AIME24, above the full-label baseline too. Both of its policies switched to opening their solutions with an informal draft in plain text, a switch that in this project’s other GRPO runs marks the best runs whatever their reward, so two seeds cannot tell how much of the gain is the probe’s. Code, logs, every training completion and every number in this paper are in the repository.

1 Introduction

RLVR assumes that every sampled solution can be checked. When checking is expensive, for example with human grading, long test suites or slow simulators, a training run may afford labels for only some of its samples. A judge can fill in the rest. A judge that was right on yesterday’s samples may be wrong on today’s, though, because the policy it judges keeps changing.

This paper studies that setting with GRPO on MATH, using Part II’s setup, where the verifier labels 2 of the 8 completions in each group. We compare four ways to use the other 6: drop them, reward them with Kev-4B’s frozen judgment, reward them with Kev-4B after its head is refitted online on the labels collected so far, or reward them with a head of the same shape trained from scratch on the same labels. Because the verifier is in fact available, we also measure how good each fill-in reward was on the completions it scored, step by step.

2 Related work

Semi-supervised reward learning and active labelling reduce the number of human labels in RLHF [4]. Reward models trained once and then frozen are exploited as the policy drifts [1]. Re-fitting the reward model to fresh labels during training is the usual remedy in RLHF, but it costs labels. Recalibrating a classifier’s output layer on new data is a standard fix for distribution shift [2]. Here it is applied online to a judge’s pointer head. GRPO and our update follow Part II [3, 5].

3 Setting: a verifier that labels only some completions

Part II gives the verifier’s verdict to every completion. Here only a fraction $f = 0.25$ of each group’s completions is labelled, two of eight, chosen at random. The labelled share models a verifier that is costly to run, such as human grading or slow tests. The question is how to use the unlabelled 75%:

- **rlvr_sub**: leave them out. The advantages come from the two labelled completions of each group, and the others get no gradient.
- **sub_kev**: reward them with Kev-4B’s P_c (“Is the final answer of the proposed solution correct?”), as released and frozen.
- **sub_kev_recal**: reward them with P_c from a copy of Kev-4B’s pointer head that is refitted online on all labelled completions seen so far.
- **sub_probe**: reward them with a head of the same shape, on the same features, trained from scratch on the same labels. This separates what Kev-4B’s released head contributes from what any small head fitted to these labels would.

Labelled completions always get the verifier’s reward. The upper reference is Part II’s **rlvr**, where every completion is labelled. Everything else is Part II’s setup: the same model, data, learning rate, batch, number of steps and evaluation (§2 of Part II). For logging only, the verifier also checks every unlabelled completion, which measures how good each fill-in reward was on the completions it was used for.

4 Online recalibration

Kev-4B’s pointer head turns the backbone’s hidden states into option logits z . Its $p(\text{yes})$ is $\text{softmax}(z/T)_{\text{yes}}$ with the released temperature T . **sub_kev_recal** keeps the backbone and Kev-4B’s adapter frozen and refits a float32 copy θ of the head after every step. The loss is the binary cross-entropy of $p(\text{yes})$ against the verifier’s labels, plus $\alpha\|\theta - \theta_0\|^2$, an anchor to the released head θ_0 ($\alpha = 10^{-3}$). The fit uses 50 Adam steps (learning rate 10^{-4} , batches of 256) over a buffer of the last 8192 labelled completions. The features (the head’s inputs) are computed once, when Kev-4B scores a completion, so the refit costs no backbone pass. The released head on the backbone is never changed. **sub_probe** uses the same procedure from a fresh initialisation of the head’s projections, with $T = 1$ and no anchor.

The reason to expect recalibration to matter is that the policy’s completions drift during training. They get longer or shorter and change style, so a judge calibrated on its own training distribution can lose calibration on them. The labelled completions measure that drift directly, and online refitting is the cheapest way to follow it.

5 Results

Each condition ran with 2 seeds of 200 steps; Part II’s **rlvr**, which labels every completion, is the reference with the full label budget (3 seeds). Table 1 gives the final policies’ accuracies, Table 2 those of every run, and Table 3 the paired differences to **rlvr_sub**, the condition that simply drops the unlabelled completions. Table 4 shows how good each fill-in reward was on the completions it scored, Table 5 what each final policy writes, and Figure 1 the training curves. The intervals resample problems and do not include the variation between seeds, which in this project’s GRPO runs can be larger than the differences between conditions (Part II, §8).

Dropping three quarters of the labels costs a little. With only the two labelled completions of each group, **rlvr_sub** reaches 81.8% on MATH-500 (seeds 80.6 and 83.0%). Labelling all eight (**rlvr**) gives +0.7

condition	seeds	MATH-500	GSM8K	AMC23	AIME24	AIME25
base model	–	76.5	87.9	50.6	10.0	10.9
rlvr	3	82.5 $\pm$ 0.6	93.0 $\pm$ 0.3	60.9 $\pm$ 2.4	21.5 $\pm$ 0.4	16.7 $\pm$ 2.1
rlvr_sub	2	81.8 $\pm$ 1.7	92.4 $\pm$ 0.4	60.8 $\pm$ 1.1	18.3 $\pm$ 1.8	15.4 $\pm$ 0.3
sub_kev	2	79.5 $\pm$ 1.3	91.2 $\pm$ 1.3	56.1 $\pm$ 3.5	15.9 $\pm$ 1.6	14.5 $\pm$ 1.6
sub_kev_recal	2	81.2 $\pm$ 2.5	92.3 $\pm$ 0.9	59.4 $\pm$ 1.8	21.1 $\pm$ 4.6	14.7 $\pm$ 3.4
sub_probe	2	85.9 $\pm$ 1.8	93.7 $\pm$ 0.5	67.7 $\pm$ 2.3	30.5 $\pm$ 0.1	23.9 $\pm$ 1.3

Table 1: Final policies: accuracy in percent, seed mean $\pm$ standard deviation over seeds. MATH-500 and GSM8K: greedy. AMC23, AIME24 and AIME25: mean over 16 samples per problem at temperature 0.6. rlvr: every completion labelled (Part II); the other conditions label 2 of the 8 completions of each group.

condition	seed	MATH-500	GSM8K	AMC23	AIME24	AIME25
rlvr	1	82.8	93.0	63.6	21.0	18.8
	2	83.0	93.3	59.8	21.9	16.9
	3	81.8	92.6	59.2	21.7	14.6
rlvr_sub	1	80.6	92.6	61.6	19.6	15.6
	2	83.0	92.1	60.0	17.1	15.2
sub_kev	1	78.6	90.3	53.6	14.8	13.3
	2	80.4	92.1	58.6	17.1	15.6
sub_kev_recal	1	79.4	91.7	58.1	17.9	12.3
	2	83.0	92.9	60.6	24.4	17.1
sub_probe	1	87.2	93.3	69.4	30.6	22.9
	2	84.6	94.1	66.1	30.4	24.8

Table 2: The final policy of every run: accuracy in percent, as in Table 1.

$[-1.1, 2.7]$ points more, and $+3.2 [1.0, 5.7]$ on AIME24. A fill-in reward has this much to win back, as long as it does not teach the policy something else.

The frozen judge makes things worse. Rewarding the unlabelled completions with Kev-4B’s P_c as released (sub_kev) ends below dropping them, in both seeds (78.6 and 80.4% on MATH-500): $-2.3 [-4.4, -0.1]$ points on MATH-500, $-4.7 [-8.8, -0.3]$ on AMC23 and $-2.4 [-5.3, 0.1]$ on AIME24. Yet the judge kept ranking the unlabelled completions well: its AUROC for the withheld labels was 0.91 over the run (Table 4). It was too generous, though. Its mean P_c on the unlabelled completions was 0.65 in the first 20 steps, when 47.6% of them were correct, and 0.79 in the last 20, when 63.3% were. Which completions are labelled is random, so a constant offset raises the expected reward of every behaviour by the same amount, which the group baseline removes in expectation. A judge that pays more for some behaviour than its correctness warrants changes what the policy learns. Both policies came to repeat themselves: 48.6% and 24.4% of the lines of their MATH-500 solutions repeat an earlier line, against 1.3% and 1.1% for rlvr_sub, and 69.0% of seed 1’s solutions run into the 2048-token limit (Table 5). After the first answer seed 1 restates its conclusion again and again, often until the length limit, and seed 2 solves the problem again. Neither changes the answer the verifier reads, so the labelled completions do not punish it, and on the unlabelled ones the judge pays for it (§5.2).

minus rlv_r_sub	MATH-500	GSM8K	AMC23	AIME24	AIME25
rlvr	+0.7 [-1.1, 2.7]	+0.6 [-0.2, 1.4]	+0.1 [-3.2, 3.3]	+3.2 [1.0, 5.7]	+1.3 [-0.7, 3.3]
sub_kev	-2.3 [-4.4, -0.1]	-1.2 [-2.1, -0.2]	-4.7 [-8.8, -0.3]	-2.4 [-5.3, 0.1]	-0.9 [-4.2, 2.2]
sub_kev_recal	-0.6 [-2.8, 1.6]	+0.0 [-0.9, 0.8]	-1.4 [-5.5, 3.2]	+2.8 [0.8, 5.2]	-0.7 [-3.1, 1.4]
sub_probe	+4.1 [1.6, 6.6]	+1.3 [0.4, 2.3]	+7.0 [2.7, 11.2]	+12.2 [5.2, 20.3]	+8.4 [4.0, 13.5]

Table 3: Paired differences to rlv_r_sub, in percentage points: per problem, each condition’s seed mean, averaged over problems, with a 95% bootstrap interval over problems.

condition	seeds	fill-in AUROC			frozen Kev-4B		mean	
		steps 0–19	first half	second half	AUROC	ECE	fill-in	label
sub_kev	2	0.89	0.92	0.90	0.91	0.19	0.75	0.60
sub_kev_recal	2	0.92	0.94	0.92	0.90	0.10	0.63	0.62
sub_probe	2	0.93	0.95	0.92	0.91	0.11	0.66	0.65

Table 4: The fill-in rewards on the unlabelled completions, against the verifier’s withheld labels (per step, mean over steps and seeds): AUROC in the first 20 steps and in each half of the run, the frozen Kev-4B’s AUROC on the same completions, expected calibration error, and the mean fill-in reward against the share that is correct.

Recalibration fixes the calibration, not the preference. Refitting Kev-4B’s head on the labels collected so far (sub_kev_recal) works as intended on the numbers it was fitted for. Its calibration error on the unlabelled completions is 0.10 against 0.19 for the frozen head, its mean fill-in reward 0.63 matches the 61.6% that are correct, and it ranks them slightly better than the frozen head does on the same completions (AUROC 0.93 against 0.90). Seed 1 still learned to repeat itself (22.1% of its lines) and ends at 79.4% on MATH-500; seed 2 did not (0.8%) and ends at 83.0%. Over both seeds the condition is not better than dropping the unlabelled completions (−0.6 [−2.8, 1.6] on MATH-500).

The head trained from scratch: both seeds switched to drafting first. The probe (sub_probe) ranks and calibrates about as well as the recalibrated head (AUROC 0.93, calibration error 0.11), and neither of its policies repeats itself (0.2% and 0.4% of their lines). Both end far above every other run of this paper: 87.2 and 84.6% on MATH-500, 30.6 and 30.4% on AIME24. Over both seeds that is +4.1 [1.6, 6.6] points above rlv_r_sub on MATH-500 and +12.2 [5.2, 20.3] on AIME24, and above the full-label rlv_r as well (82.5% on MATH-500). Both policies write differently from the others: they work the problem in plain text first, and in 50% and 72% of these drafts then write a formatted solution (§5.1).

Evaluation noise. Across the 11 runs in this comparison, the base model’s MATH-500 accuracy at step 0 ranges from 75.2% to 78.2%, and a run’s accuracy changes by 1.5 points on average between its evaluations at steps 150 and 200. The final weights of each run are evaluated twice on MATH-500, both times greedily: at step 200 with the periodic evaluations, and in the final evaluation. The two results differ by 1.0 points on average and by up to 2.2 (11 runs).

condition	seed	tokens	truncated	repeated lines		several answers	draft opening	Kev-4B P_c	
				share	after answer			right	wrong
rlvr	1	541	2%	1%	0%	2%	1%	0.84	0.60
	2	555	4%	1%	0%	6%	0%	0.85	0.59
	3	909	8%	28%	90%	95%	1%	0.88	0.64
rlvr_sub	1	630	6%	1%	0%	3%	1%	0.84	0.54
	2	588	3%	1%	0%	3%	2%	0.83	0.56
sub_kev	1	1723	69%	49%	49%	99%	1%	0.91	0.72
	2	1072	12%	24%	85%	97%	2%	0.91	0.71
sub_kev_recal	1	904	3%	22%	85%	98%	7%	0.90	0.71
	2	538	3%	1%	0%	11%	1%	0.85	0.62
sub_probe	1	546	6%	0%	0%	75%	90%	0.90	0.61
	2	694	6%	0%	0%	71%	90%	0.91	0.66

Table 5: What each final policy writes on MATH-500 (greedy): completion tokens, share truncated at 2048 tokens, share of lines (of at least 20 characters) that repeat an earlier line of the same completion, share of completions that repeat such a line after their first boxed answer, share with more than one boxed answer (the verifier reads the last one), share that open with an informal draft, and Kev-4B’s P_c for right and for wrong answers.

5.1 Drafting first

Part II found that some GRPO runs switch, abruptly, to a style in which the completion opens with an informal draft in plain text, without LaTeX or markdown, gives the answer, and often then writes the formatted solution; and that the runs that switch are the best ones. We measure the style by the share of completions that open with such a draft (no math delimiter, markdown heading or bold in the first 300 characters; Part II, §8.3, defined after reading the completions). The base model opens 11% to 12% of its MATH-500 completions this way.

Only the probe’s policies switched. Every run but one cut the share of drafts in its first 50 steps (Table 6). The exception is sub_probe seed 1, which drafted in 28% of its MATH-500 completions at step 50 and in 91% at step 100. In seed 2 of sub_probe the share fell to 5% at step 50 and then rose, from 9% at step 100 to 90% at step 150. In the six other runs of the label-budget conditions, and in the three of rlvr, it was at most 7% in the final evaluation. Across the 26 GRPO runs of this project, the 6 that switched, in 5 different conditions, are the 6 best on AIME24 (30.4 to 34.6%, every other run at most 24.4%; Part II, §8.3). The gain of sub_probe over rlvr_sub comes with this switch, and with two seeds we cannot separate the effect of the probe from that of the switch.

What the fill-in rewards paid for drafts. Within a group, GRPO favours the completions with the higher reward, so a style spreads when its completions are paid more than the others of their group. Table 7 compares the drafts with the other completions of the same group by the verifier’s verdict, and Table 8 by the fill-in reward among unlabelled completions with the same verdict. In steps 0–49 drafts were less often right than the other completions of their group in all 11 runs (between −15.3 and −2.4 points), which is why their share fell. In both sub_probe runs, from step 50 on, drafts were as often right as the others or more (+1.5 and +3.8 points in seed 1, +1.4 and +0.6 in seed 2), and in the windows in which drafts spread the probe paid right drafts more than the other right completions (+2.1 points in seed 1 over steps 50–99, +5.3 and +1.9 in seed 2 over steps 50–149). In the other fill-in runs the picture is mixed. In sub_kev_recal seed 2, drafts were more often right than the others in the second half of the run (+1.6 and +4.8 points), the recalibrated head

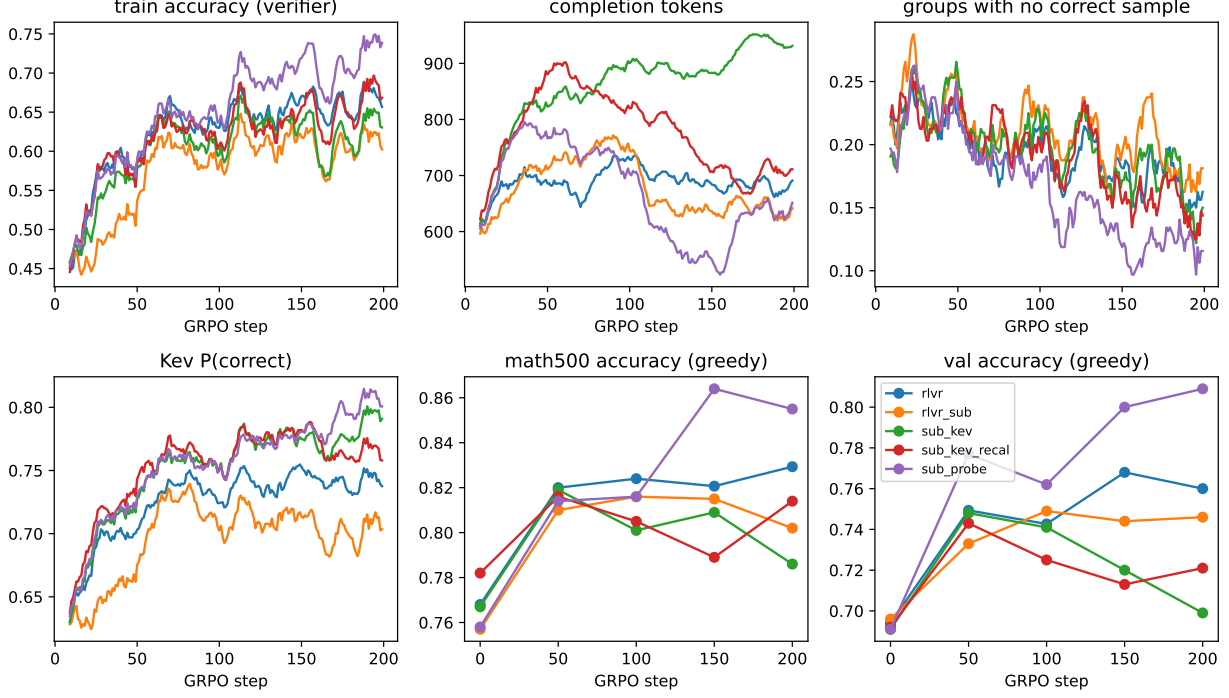


Figure 1: Training curves (seed mean): verifier accuracy of all training completions (labelled or not), completion tokens, share of groups with no correct completion and Kev-4B’s P_c on the training completions (10-step moving averages), and greedy accuracy on MATH-500 and the validation set every 50 steps.

paid right drafts less (−2.5 and −3.2), and drafts did not spread. In `sub_kev_recal` seed 1 the head paid right drafts more over steps 100–199 (+3.6 and +3.6), and drafts did not spread either. A few windows of a few runs, read after the fact, do not tell what made the probe’s runs switch, and in Part III-C runs switched with the verifier’s reward alone (Part II, §8.3).

5.2 Does the judge pay for repetition?

After the GRPO runs we tested whether Kev-4B pays for what the policies of `sub_kev` and `sub_kev_recal` write after their first answer, with the probe of Part II (`dcjev/11m/repeat_probe.py`; Table 9). `sub_kev`’s seed 1 restates its conclusion again and again; `sub_kev_recal`’s seed 1 solves the problem again in nearly the same words, like the repeating policies of Part II. We cut the final MATH-500 solutions of seed 1 that repeat lines after their first answer, as if the policy had stopped there. Solutions that repeat before their first answer, or whose first answer differs from their last, were left out (66 and 44). No edit changes the answer the verifier reads.

The part after the first answer raises the released Kev-4B’s P_c , `sub_kev`’s fill-in reward, by +0.031 [0.025, 0.038] on right answers and by +0.049 [0.032, 0.067] on wrong ones. On `sub_kev_recal`’s solutions it raises it by +0.059 [0.053, 0.065] and +0.110 [0.096, 0.124], for 96% of the wrong ones. Repetition as such is worth little: written into the base policy’s solutions, it raises Kev-4B’s P_c by +0.012 on right answers and changes it by −0.002 on wrong ones. So on the unlabelled completions the fill-in reward paid for what these policies learned to write after their first answer, most of all on wrong answers, to which the verifier would have given nothing. The probe scores with the released head. The head of `sub_kev_recal` was refitted during training, and its own preference at the end of the run is not measured here. The AR judge, which no condition used, rates `sub_kev_recal`’s solutions higher with the part after the first answer (+0.034 on right

condition	seed	MATH-500	AIME24	share of drafts, step					final
				0	50	100	150	200	
rlvr	1	82.8	21.0	12%	1%	0%	1%	1%	1%
	2	83.0	21.9	12%	0%	0%	0%	0%	0%
	3	81.8	21.7	12%	1%	1%	0%	1%	1%
rlvr_sub	1	80.6	19.6	11%	3%	1%	1%	1%	1%
	2	83.0	17.1	12%	3%	2%	1%	2%	2%
sub_kev	1	78.6	14.8	11%	2%	1%	1%	2%	1%
	2	80.4	17.1	12%	2%	1%	1%	1%	2%
sub_kev_recal	1	79.4	17.9	12%	2%	1%	2%	6%	7%
	2	83.0	24.4	12%	1%	2%	3%	1%	1%
sub_probe	1	87.2	30.6	12%	28%	91%	98%	91%	90%
	2	84.6	30.4	12%	5%	9%	90%	89%	90%

Table 6: The share of MATH-500 completions that open with an informal draft, at each periodic evaluation (greedy, every 50 steps) and in the final evaluation, per run, with the final accuracies.

answers) but `sub_kev`’s endless restatements lower (-0.018 on right answers, -0.074 on wrong ones).

6 Discussion

A good ranking is not enough. The frozen Kev-4B ranked the unlabelled completions well (AUROC 0.91), and rewarding them with it still ended below dropping them. Two things it pays for beyond correctness matter differently. Its generosity, a P_c well above the share of correct completions, raises every completion’s expected reward by about the same amount and is removed by the group baseline. Its preference for solutions that go on after their answer is not: such text leaves the verifier’s label unchanged, so the labelled completions never punish it, and the judge pays for it on the unlabelled ones (§5.2). Both policies of `sub_kev` learned to repeat themselves. A fill-in reward has to be judged by what it pays for among completions with the same verdict, which an AUROC for the verdict does not measure.

Recalibration fixes the numbers, not the preference. Refitting the head on the labels so far made the fill-in reward calibrated (ECE 0.10 against 0.19), and one of its two policies still learned to repeat itself. A plausible reason, which these runs do not test directly: the refitted head starts from the released one and is anchored to it, so it keeps the released head’s preferences wherever the labels say nothing, and repetition after the answer, which leaves the label unchanged, is such a place. A head trained from scratch has no preference to keep, and neither of its policies repeated itself.

The probe’s gain comes with a change of style. `sub_probe` ends +4.1 points above `rlvr_sub` on MATH-500 and +12.2 on AIME24, and above the full-label `rlvr` as well. Both of its policies switched to drafting first, which no other run of this paper did; in this project’s other GRPO runs the same switch marks the best runs whatever their reward, 2 of them trained with the verifier’s reward alone (Part II, §8.3). So with two seeds per condition we cannot say how much of the gain the probe is responsible for. If the probe makes the switch more likely, which two of two runs suggest but cannot show, a scratch head trained on a quarter of the labels would be a better fill-in reward than the judge it is built from, for a reason that has little to do with how well it predicts the labels. More seeds of `sub_probe` and `rlvr_sub`, and runs that force or forbid the draft opening, would tell.

condition	seed	steps 0–49		steps 50–99		steps 100–149		steps 150–199	
		drafts	Δ right	drafts	Δ right	drafts	Δ right	drafts	Δ right
rlvr	1	7%	−8.5	2%	+4.2	1%	−3.9	2%	+1.8
	2	5%	−15.3	1%	−9.3	0%	−5.0	0%	−2.1
	3	6%	−14.3	1%	−0.7	1%	+3.3	1%	+2.6
rlvr_sub	1	13%	−8.3	3%	−7.2	2%	−5.1	1%	−5.5
	2	11%	−13.7	4%	−3.3	2%	−10.0	4%	+0.2
sub_kev	1	10%	−8.9	4%	−5.9	2%	−5.7	3%	+2.8
	2	12%	−11.0	6%	−3.6	1%	−8.8	1%	−5.0
sub_kev_recal	1	8%	−8.9	3%	−2.9	2%	−4.0	5%	+0.0
	2	11%	−5.7	2%	−0.3	3%	+1.6	2%	+4.8
sub_probe	1	14%	−2.4	52%	+1.5	94%	+3.8	96%	+1.6
	2	10%	−8.3	3%	+1.4	56%	+0.6	88%	+2.8

Table 7: Training completions that open with an informal draft, per 50-step window: their share, and Δ right, the mean over the groups that hold both kinds of the share of drafts the verifier accepts minus that of the other completions of the group, in percentage points.

condition	seed	steps 0–49		steps 50–99		steps 100–149		steps 150–199	
		right	wrong	right	wrong	right	wrong	right	wrong
sub_kev	1	−2.5	−1.6	+0.5	−0.7	+0.0	−3.2	+0.0	−2.5
	2	+2.2	−1.4	−1.1	−2.2	−1.8	−3.5	−2.4	−1.3
sub_kev_recal	1	−5.4	−10.4	−0.5	+5.1	+3.6	−4.6	+3.6	+6.3
	2	−2.9	−6.0	−0.3	−0.5	−2.5	−4.6	−3.2	−1.3
sub_probe	1	+0.5	−7.0	+2.1	−1.5	−0.4	+10.3	+0.3	+1.4
	2	−1.0	−6.7	+5.3	−7.6	+1.9	+3.3	−0.9	+2.1

Table 8: The fill-in reward of unlabelled training completions that open with an informal draft minus that of the other unlabelled completions of the same group with the same verdict (right or wrong), per 50-step window, mean over the groups that hold both, in percentage points; − where no group did.

Limitations. One model (4B), one task, one label fraction (a quarter), 200 steps, two seeds per condition; the recalibration’s hyperparameters (anchor, buffer, steps) were set once and not tuned. The intervals resample problems and not seeds. The style measures were defined after we had read the final completions. The repetition probe scores with the released head, not with the refitted heads as they were at the end of training. Other experiments shared the GPUs during training, so we report no training times.

Reproducibility

Code, prompts, raw logs (every GRPO step, with the fill-in rewards’ agreement with the withheld labels, and every evaluation) and every training completion of the GRPO runs are in the repository (`experiments/part3b_labels/`). The tables and every number in the text are generated from them by `dcjev/analysis/part2.py` and `dcjev/analysis/paper_numbers.py`.

	solutions	Kev-4B P_c		Kev-4B P_g		AR P_a	
	right / wrong	right	wrong	right	wrong	right	wrong
<i>base policy's solutions, repetition added</i>							
last two paragraphs twice more	2436 / 1537	+0.012	−0.002	+0.007	−0.006	+0.037	+0.012
written twice	2436 / 1564	+0.012	−0.003	+0.006	−0.004	+0.029	+0.007
<i>repeating final solutions of GRPO runs, as written against cut after the first answer</i>							
sub_kev, seed 1	351 / 66	+0.031	+0.049	+0.016	+0.031	−0.018	−0.074
sub_kev_recal, seed 1	318 / 81	+0.059	+0.110	+0.030	+0.095	+0.034	+0.034

Table 9: What repetition is worth to the judges (the probe of Part II): $P(\text{with the repetition})$ minus $P(\text{without})$, on the solutions the verifier accepts (right) and rejects (wrong), with the number of each. Top: the base policy’s solutions of Part II’s judge study, with repetition added. Bottom: the final MATH-500 solutions of seed 1 that repeat lines, as written against cut after their first answer. The released Kev-4B’s P_c and P_g and the AR judge’s P_a ; intervals in the text resample problems.

References

- [1] Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, 2023.
- [2] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, 2017.
- [3] Zihe Liu, Jiashun Liu, Yancheng He, Weixun Wang, Jiaheng Liu, Ling Pan, Xinyu Hu, Shaopan Xiong, Ju Huang, Jian Hu, et al. Tricks or traps? A deep dive into RL for LLM reasoning. In *International Conference on Learning Representations*, 2026. arXiv:2508.08221.
- [4] 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. In *Advances in Neural Information Processing Systems*, 2022.
- [5] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.