WRONG VOTES, RIGHT GRADIENTS: WHY MAJORITY-VOTE REWARDS TRACK VERIFIABLE RL

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning with self-generated labels, such as majority vote over a model's own samples, often recovers most of the gain of training with verifiable rewards. This is puzzling from the usual prompt-level view, in which majority vote only returns a model's modal answer and cannot add information about correctness. We argue that the relevant unit is the batch-level update. Because parameters are shared, a prompt's own gradient accounts for less than 1% of the change in that prompt's output at practical batch sizes, so a wrong label matters only to the extent that label errors share a direction across prompts. We decompose the majority-vote update into shrinkage and misdirection relative to the gold update, and show that misdirection vanishes with batch size whenever errors are incoherent across prompts, however often the labels are wrong. Measuring per-response gradients on five models, we find that majority-vote errors are as incoherent as a random-reward null, that the majority-vote update agrees with the gold update as closely as two independent gold updates agree with each other, and that its noise-corrected cosine with the gold update is 1 within error. Randomly flipping up to 30% of gold labels leaves the update direction intact, while the same fraction flipped by a shared rule destroys it, and rewards that favor a kind of response on every prompt, such as a learned reward model, fall measurably short. The equivalence holds at every checkpoint we examine but is local: gold-reward training still ends ahead on held-out majority accuracy. Error coherence, not error rate, determines when self-generated labels can stand in for verification.

1 INTRODUCTION

Reinforcement learning is increasingly used to post-train language models, and its most reliable form uses verifiable rewards: programs, proof checkers, or reference answers that decide whether an output is correct (Shao et al., 2024; Guo et al., 2025; Zheng et al., 2026). Verification also confines training to existing labeled data or to domains with an automatic oracle (Zhao et al., 2026a). Self-generated rewards remove this constraint by deriving feedback from the model's own outputs, such as agreement among sampled answers (Zuo et al., 2026; Shafayat et al., 2025; Huang et al., 2026d; Yu et al., 2025), self-certainty (Zhao et al., 2026b), or entropy (Prabhudesai et al., 2025; Agarwal et al., 2026). They often work surprisingly well: label-free training frequently recovers most of the improvement obtained with ground-truth rewards, and in some settings matches it (Zuo et al., 2026; Shafayat et al., 2025; Zhao et al., 2026b; Yu et al., 2026) (Table 10). Yet the same rewards can also improve and then collapse (Shafayat et al., 2025; Zhang et al., 2025c; Jin et al., 2026), or reinforce a model's existing preferences regardless of correctness (He et al., 2026). When do self-generated labels work, and why?

The standard account is prompt-level (He et al., 2026; Zhang et al., 2025a). For a fixed prompt, majority vote (MV) converges to the model's modal answer, which need not be correct; it can sharpen the answer distribution but cannot extract correctness information that the distribution does not already contain. Several results sit uneasily with this account. TTRL raises the maj@64 of Qwen2.5-Math-7B on MATH-500 from 66.4 to 85.2, and its post-training average accuracy exceeds the initial maj@64 (Zuo et al., 2026, Fig. 7). In our own runs, held-out problems whose majority answer is wrong acquire a correct majority at about three times the rate that resampling alone produces (Section 5). Both effects require information to move between problems, which a prompt-level account does not allow.

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.

We argue that the prompt is the wrong unit of analysis. Reinforcement learning sums reward-weighted gradients over a batch of diverse prompts and applies a single update to shared parameters. At practical batch sizes, a prompt's own gradient accounts for less than 1% of the change in that prompt's output, so a wrong label barely steers its own prompt; what matters is whether the errors of many wrong labels add up to a common direction. We make this precise by decomposing the MV update into a component along the gold update, which only rescales the step, and an orthogonal component, misdirection, which changes its direction (Section 4). Misdirection is governed by the coherence of label errors across prompts rather than by their rate. Incoherent errors vanish once the batch exceeds a critical size set by the coherence of the gold signal, however often the labels are wrong, whereas errors that share a direction leave a floor that no batch size removes. That independent noise averages out is familiar from supervised learning (Rolnick et al., 2017; Chatterjee, 2020); what is new here is to apply it to the difference between two reward sources, to measure that difference directly in LLM post-training, and to turn it into a testable criterion for when self-generated labels can replace verification.

We test this account with per-response gradient sketches, which let us evaluate any reward on the same samples. On five models from two families, the MV update agrees with the gold update as closely as two independent gold updates agree with each other, and after correcting for prompt-sampling noise its cosine with the gold update is 1 within error (Section 3). MV errors are as incoherent as a random-reward null, while gold gradients share a direction that dominates beyond a few dozen prompts. Controls behave as the account predicts. Randomly flipping up to 30% of gold labels leaves the update direction unchanged, whereas rewards that favor a kind of response on every prompt (frequent answers, confident answers, or those a learned reward model prefers) fall measurably short.

The equivalence is local. It holds at every checkpoint we examine, but it does not make the two training runs identical: gold training still ends ahead on held-out majority accuracy, and MV training mainly sharpens single-sample answers (Section 5). Our results therefore give a criterion rather than a guarantee. Self-generated labels can stand in for verification when a model's residual errors are sparse and specific to individual prompts, and they should fail where errors are shared, as with widely held misconceptions or prompts revisited until their errors accumulate.

2 PRELIMINARIES

Group-relative policy gradients. We train with GRPO (Shao et al., 2024). A training step draws B prompts $x_1, \dots, x_B$ and samples n responses $y_{i1}, \dots, y_{in}$ per prompt from the current policy π_θ. A reward $r_{ij} \in \{0, 1\}$ scores each response, and its advantage is the reward relative to its group, $A_{ij} = (r_{ij} - \bar{r}_i)/s_i$, where $\bar{r}_i$ is the group mean and s_i is either the group standard deviation or 1. At the sampling policy the importance ratios equal one and clipping is inactive, so the update direction is a sum of per-prompt gradients,

$$G(r) = \sum_{i=1}^{B} g_i(r), \qquad g_i(r) = \sum_{j=1}^{n} A_{ij}\, \nabla_\theta \log \pi_\theta(y_{ij} \mid x_i).$$

We call g_i the prompt gradient and G the batch update.

Gold and majority-vote rewards. Both rewards compare the final answer $a(y)$, taken from the last `\boxed{}`, with a target. The gold reward uses the reference answer, $r^{\text{gold}}_{ij} = \mathbf{1}[a(y_{ij}) = a^*_i]$. The majority-vote (MV) reward uses the most frequent answer in the group, $r^{\text{mv}}_{ij} = \mathbf{1}[a(y_{ij}) = \hat{a}_i]$, and needs no label (Zuo et al., 2026; Shafayat et al., 2025; Huang et al., 2026d). We write G_{mv} and G_{gold} for the two batch updates. When the majority answer is correct, $\hat{a}_i = a^*_i$, the two rewards coincide and so do the prompt gradients. The updates therefore differ only through prompts whose majority answer is wrong,

$$E = G_{\text{mv}} - G_{\text{gold}} = \sum_{i:\, \hat{a}_i \neq a^*_i} \big(g_i(r^{\text{mv}}) - g_i(r^{\text{gold}})\big),$$

which we call the MV error. A group in which every response receives the same *gold* reward has zero gold advantage, so no cosine with gold is defined on it; we use the groups that are mixed under gold (164 of 256 prompts on Qwen3-4B-Base). The filter is on gold alone and forced rather than chosen: MV is often still live on the excluded groups.

Models and training. Our main model is Qwen3-4B-Base (Yang et al., 2025), trained on DeepScaleR (Luo et al., 2025) with $B = 256$ prompts and $n = 16$ responses per step. We train it twice from the same initialization, once with gold and once with MV rewards. Both runs see the same prompts at every step, so they differ only in the reward. Neither uses a KL or entropy term. For gradient analyses at initialization we add Qwen3-1.7B-Base, Qwen2.5-7B, Llama-3.2-3B-Instruct, and Llama-3.1-8B-Instruct (Grattafiori et al., 2024). We evaluate on MATH-500 (Hendrycks et al., 2021; Lightman et al., 2024), including its 134 level-5 problems, with 64 samples per problem at temperature 1. Appendix A gives the full configuration.

Measuring updates. Comparing G_{mv} with G_{gold} requires both updates on the same responses. For every response we store the gradient of $\log \pi_\theta(y_{ij} \mid x_i)$ at the sampling policy, compressed by a count sketch (Charikar et al., 2002) to $K = 2^{16}$ dimensions. Because g_i is linear in the advantages, the update under any reward is a reweighting of the same per-response sketches, so every reward we compare is evaluated on identical samples. Cosines computed from sketches agree with exact fp32 gradients to within 0.004. Our analyses use mean-centered advantages ($s_i = 1$). Training divided by the group standard deviation; since s_i is a per-group scalar, recomputing under that weighting moves c_∞ by at most 0.010 on any arm (Table 7).

Gold agreement as the benchmark. A cosine of one is the wrong benchmark: most of a prompt gradient is specific to that prompt, so even two gold updates on disjoint prompts are far from parallel. We therefore split the prompt groups into disjoint halves $\mathcal{A}$ and $\mathcal{B}$ and compute

$$X = \cos\big(G^{\mathcal{A}}_{\text{mv}}, G^{\mathcal{B}}_{\text{gold}}\big), \qquad Y = \cos\big(G^{\mathcal{A}}_{\text{gold}}, G^{\mathcal{B}}_{\text{gold}}\big), \qquad Z = \cos\big(G^{\mathcal{A}}_{\text{mv}}, G^{\mathcal{B}}_{\text{mv}}\big).$$

Y, the agreement of gold with itself, is the ceiling, and MV is as close to gold as the data can show when $X \approx Y$; pairing different halves keeps MV and gold from sharing sampling noise. If each half-batch update is its expectation plus independent prompt-sampling noise, the noise enters X, Y, and Z only through their norms, so

$$c_\infty = \frac{X}{\sqrt{YZ}}$$

estimates the cosine between the expected MV and gold updates, which $\cos(G_{\text{mv}}, G_{\text{gold}})$ approaches as the batch grows. This is the classical correction for attenuation (Spearman, 1904); as a ratio of estimates, c_∞ can exceed 1 by sampling error, although the cosine it estimates cannot. We average over random splits, bootstrap over prompt groups, and report c_∞ only when $Z \geq 0.3$, below which it is unstable.

3 Gradient-Level Comparison of Majority Vote and Gold

We first ask how much the batch update changes when gold labels are replaced by MV. At initialization on all five models, it changes no more than prompt sampling already does; Section 5 follows training.

MV agrees with gold as well as gold agrees with itself. Two gold updates on disjoint halves of the prompt groups agree only to a cosine of 0.37 to 0.84 (Table 1), because most of each prompt gradient is specific to its prompt. This is the ceiling for any label source, and MV reaches it: its cross-half cosine with gold is within 0.02 of the ceiling on four of the six arms and slightly above it on Qwen2.5-7B. The largest gap, on Llama-3.1-8B-Instruct (0.399 against 0.438), comes from MV's own noise (its split-half agreement is 0.362); after correcting for it, c_∞ is 1.013. MV updates are as consistent with each other as gold updates are (Table 12).

The expected MV update points in the gold direction. The ceiling depends on how many prompts each half contains; Figure 1 and the last column of Table 1 remove this dependence. The solid curves show the cosine between summed MV and gold gradients over random subsets of B prompt groups. For a single prompt the two gradients coincide only when the majority is right, and the average cosine is 0.5 to 0.6. As B grows the cosine rises toward its asymptote c_∞, which is 1 within its 95% interval on every model (point estimates 1.00 to 1.01), across two model families, base and instruction-tuned models, and 1.7B to 8B parameters. In expectation, the MV and gold updates point in the same direction; the finite-batch gap averages out, and Section 4 shows why.

Table 1: MV is as close to gold as gold is to itself. We split the prompt groups into two disjoint halves and compare summed gradients across the halves. *Gold vs. gold* is the ceiling: how well two independent gold updates agree. *MV vs. gold* pairs the MV update on one half with the gold update on the other; MV is indistinguishable from gold when the two columns match. c_∞ removes the dependence on the number of prompt groups N by estimating the cosine between the expected MV and gold updates; $c_\infty = 1$ means they point in the same direction. Table 12 gives the full results.

		Split-half cosine		Expected update
Model	N	Gold vs. gold (ceiling)	MV vs. gold	c_∞ (95% CI)
Qwen3-4B-Base[†]	331	0.647	0.647	1.006 (.979, 1.023)
Qwen3-4B-Base	164	0.606	0.600	0.999 (.965, 1.043)
Qwen3-1.7B-Base	636	0.842	0.834	0.998 (.988, 1.006)
Qwen2.5-7B	151	0.799	0.818	1.010 (.968, 1.021)
Llama-3.2-3B-Instruct	247	0.371	0.360	0.997 (.921, 1.072)
Llama-3.1-8B-Instruct	208	0.438	0.399	1.013 (.940, 1.073)

[†]Pools two batches of 256 prompts; the second was generated after one update. The row below is the on-policy batch alone. Columns and c_∞ use the same 200 partitions, so $c_\infty = X/\sqrt{YZ}$ checks row by row.

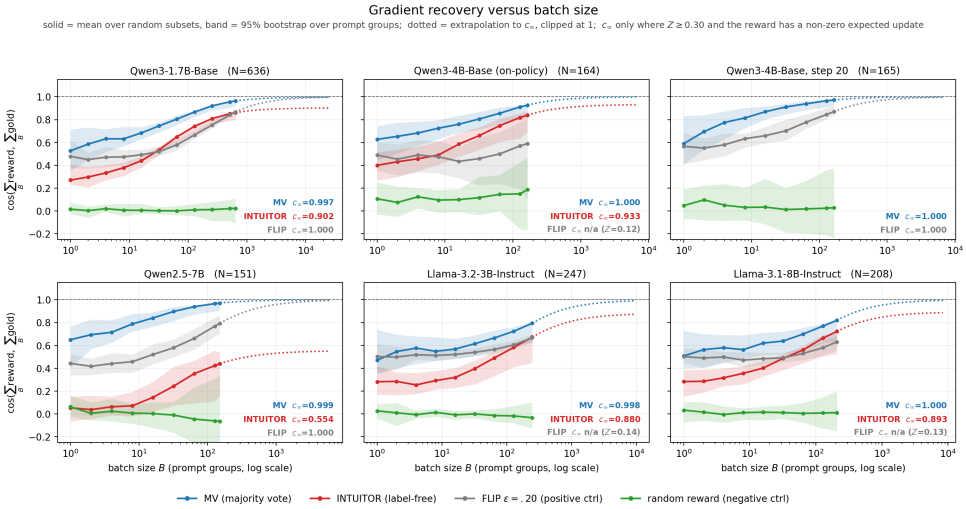

Figure 1: Cosine between each reward's summed gradient and the summed gold gradient, against the number of prompt groups B. Lines are means over random subsets; bands are 95% bootstrap intervals over prompt groups; dotted lines extrapolate to c_∞. The flips (gray) and the random reward (green) are averaged over 12 labellings, since a single frozen draw converges to its own overlap with gold rather than its expectation. c_∞ is clipped at 1 and shown only where $Z \geq 0.30$ and the expected update is non-zero; panels below that threshold are marked. INTUITOR (red) is the raw reward (Table 15 gives the length-controlled variant).

Each prompt moves with the batch. Why does a wrong label cost so little? Because it barely steers its own prompt. On Qwen3-1.7B-Base, a small step along a prompt's own gradient changes that prompt's output only 1.7 to 1.9 times as much as a step along another prompt's gradient (Table 18). Summed over a batch, the other prompts dominate: at $B = 64$ to 128, a prompt's own gradient accounts for 0.09–0.38% of the change in its output on the Qwen3 models, and 0.83% on Llama-3.2-3B-Instruct at $B = 64$ (Table 19).

The batch also sets the direction of that change. On the 76 prompts in the first Qwen3-4B-Base training batch whose majority answer is wrong, the MV batch update still increases the expected gold reward on 82% of them, to first order. The prompt's own mislabeled term contributes $+0.007$ to the cosine between the update and the prompt's gold gradient; the other prompts contribute $+0.075$, almost exactly as much as under gold labels ($+0.074$). A wrong label is outvoted by the rest of the batch, so whether the MV update is right depends on how its errors add up across prompts.

4 WHY LABEL ERRORS CANCEL

Section 3 showed that the MV and gold updates point in the same direction in expectation. We now explain why, by separating two ways a label source can differ from gold and deriving when each survives a large batch.

4.1 SHRINKAGE AND MISDIRECTION

Write the MV batch update relative to the gold update as

$$G_{\mathrm{mv}} = (1-\beta)\, G_{\mathrm{gold}} + E_{\perp}, \qquad E_{\perp} \perp G_{\mathrm{gold}}.$$

The coefficient β measures *shrinkage*, the part of the gold update that MV does not deliver, and $E_{\perp}$ measures *misdirection*. With $R_{\perp} = \|E_{\perp}\|/\|G_{\mathrm{gold}}\|$,

$$\cos\big(G_{\mathrm{mv}}, G_{\mathrm{gold}}\big) = \frac{1-\beta}{\sqrt{(1-\beta)^2 + R_{\perp}^2}}, \qquad \frac{\|G_{\mathrm{mv}}\|}{\|G_{\mathrm{gold}}\|} = \sqrt{(1-\beta)^2 + R_{\perp}^2}.$$

Only misdirection changes the direction of the update: under SGD, shrinkage acts like a smaller learning rate, and under an adaptive optimizer a roughly constant shrinkage is largely absorbed by the second-moment normalization. The question is why $R_{\perp}$ is small, and the answer lies in how vectors add.

Proposition 1 (Growth of a sum). *For vectors $v_1, \dots, v_B$, let $s^2 = \frac{1}{B}\sum_i \|v_i\|^2$ and define the mean pairwise alignment $\bar{\rho} = \sum_{i \neq j} \langle v_i, v_j \rangle \big/ \big((B-1)\sum_i \|v_i\|^2\big)$. Then*

$$\Big\| \sum_{i=1}^{B} v_i \Big\|^2 = B\, s^2 \left(1 + (B-1)\bar{\rho}\right),$$

and the local growth exponent is $\alpha(B) = \frac{d \log \| \sum_i v_i \|}{d \log B} = \frac{1}{2} + \frac{1}{2} \cdot \frac{B\bar{\rho}}{1+(B-1)\bar{\rho}}$.

Incoherent vectors ($\bar{\rho} = 0$) add like a random walk, growing as $B^{1/2}$, while vectors with a shared component grow as B beyond the crossover $B^* \approx 1/\bar{\rho}$; a fitted exponent of $1/2$ therefore means no shared direction, and $\alpha \to 1$ a dominant one. For prompt gradients, B^* is the gradient noise scale of McCandlish et al. (2018): writing $g_i = \mu + \xi_i$ with independent noise ξ_i gives $\bar{\rho} = \|\mu\|^2/(\|\mu\|^2 + \mathbb{E}\|\xi\|^2)$, so $B^* - 1 = \mathbb{E}\|\xi\|^2/\|\mu\|^2$. We apply the same accounting to the error between two rewards.

Proposition 2 (Misdirection vanishes for incoherent errors). *Let a fraction ε of the B prompts carry a wrong majority label, and for each such prompt let e_i be the component of $g_i(r^{\mathrm{mv}}) - g_i(r^{\mathrm{gold}})$ orthogonal to G_{gold}. Let s_e and ρ_e be the root-mean-square norm and mean pairwise alignment of the e_i, and s_g and ρ_g those of the gold prompt gradients. Then*

$$R_{\perp}^2 = \varepsilon\, \frac{s_e^2}{s_g^2} \cdot \frac{1 + (\varepsilon B - 1)\rho_e}{1 + (B-1)\rho_g}.$$

If $\rho_e = 0$, then $R_{\perp} \to 0$ once $B \gg 1/\rho_g$, for any error rate ε. If $\rho_e > 0$, misdirection has a floor, $R_{\perp} \to \varepsilon\,(s_e/s_g)\sqrt{\rho_e/\rho_g}$, that no batch size removes.

The error rate ε enters only as a prefactor; whether misdirection survives a large batch depends on the coherence ρ_e. Incoherent errors grow as $\sqrt{\varepsilon B}$ while the gold signal grows as B, so their ratio decays as $1/\sqrt{B\rho_g}$ however often the labels are wrong, whereas coherent errors grow as fast as the signal and leave a floor (proofs in Appendix B). For random label noise, the extreme case of incoherent error, the decomposition is exact in expectation.

Corollary 1 (Independent label flips only shrink the update). *Suppose each gold reward is flipped independently with probability $\varepsilon < 1/2$, and advantages are mean-centered. Then the expected update given the responses is $(1-2\varepsilon)\, G_{\mathrm{gold}}$: the flips shrink the update by $\beta = 2\varepsilon$ and add zero-mean noise that is independent across prompts. The disattenuated cosine c_{∞} therefore equals 1 for every $\varepsilon < 1/2$.*

These results predict that MV errors are incoherent, unlike gold gradients; that random flips preserve the update direction at any rate below one half; and that rewards whose errors recur across prompts fall short, however accurate they are.

Table 2: MV errors are as incoherent as a random reward. Mean pairwise cosine between prompt gradients ($\times 10^{-3}$); a positive value means the gradients share a direction. Gold gradients share one, while the MV error E stays at the level of a Gaussian random reward (null). Qwen3-4B-Base along the MV run, split by whether the base model's majority answer is right or wrong. Because this majority is computed on separate samples, it can differ from the majority within a 16-response training group, so E is not exactly zero on right-majority prompts. Checkpoints use different prompt draws; Table 13 adds the MV gradients.

	Right majority			Wrong majority		
Checkpoint	Gold	MV error E	Null	Gold	MV error E	Null
Base	22.41	0.39	0.30	6.55	0.23	0.69
Step 10	6.52	0.01	0.36	0.82	0.15	0.50
Step 20	27.99	0.14	−0.15	13.44	0.55	0.66

The null is a Gaussian random reward on the same responses, taken from the gradients themselves rather than their difference with gold: a reward with zero expected update would otherwise inherit gold's coherence through $-G_{\text{gold}}$ instead of providing a floor.

4.2 MV ERRORS ARE INCOHERENT

Table 2 reports the mean pairwise cosine between prompt gradients of Qwen3-4B-Base, with a Gaussian random reward as the null. At initialization, gold gradients share a direction (0.022 on prompts with a correct majority, 0.0066 on prompts with a wrong one), while the MV error E is roughly 30 to 60 times less aligned (0.0004 and 0.0002), of the same order as the null. A single sketched cosine is accurate only to about 0.004, but each entry averages over thousands of prompt pairs, and the null, computed on the same sketches, absorbs any bias of the sketch.

The growth of batch sums gives an independent estimate (Table 11). The error's exponent stays at $1/2$ at every scale, while the gold exponent rises from 0.58 at small batches to 0.87 at $B \approx 150$, as Proposition 1 predicts for vectors with a shared component. The two estimates agree: inverting Proposition 1 at $B \approx 145$ gives $\bar{\rho} \approx 0.020$ for gold, against a measured 0.022, and a crossover $B^* \approx 50$, close to the $B^* = 37$ from a separate decomposition (Table 19). Practical batch sizes are well above this crossover.

4.3 ERROR STRUCTURE, NOT ERROR RATE

Table 3 compares rewards whose errors differ in rate and structure, all on the same Qwen3-1.7B-Base responses. Independent flips isolate the error rate: as Corollary 1 predicts, c_∞ stays at 1 for flip rates up to 30%, so a high error rate alone does not misdirect the update.

Holding the rate fixed and changing only the structure separates the two. Flipping the *same* fraction by a rule shared across prompts, the longest correct and shortest incorrect responses, leaves the rate unchanged but makes the errors coherent (ρ_e rises from the null, 0.2–1.4×10^{-3}, to 42–48×10^{-3}, above the gold gradients' own 7.9). At a matched 20% error rate random flips give $c_\infty = 1.011$ and shared-rule flips 0.287; at 31% the shared-rule update points away from gold (-0.129). The gap widens with ε while the random-flip column does not move; Figure 2 shows the same contrast on all six arms.

The rewards that fall short favor a kind of response on every prompt. In expectation, random vote rewards each answer in proportion to its frequency, favoring frequent answers everywhere rather than only where the majority is wrong; self-certainty vote and INTUITOR favor confident answers; and an off-the-shelf reward model favors the responses it has learned to prefer, including confident ones (Leng et al., 2025). Each preference is a direction shared across prompts, and each interval lies below 1 (random vote 0.961, self-certainty vote 0.975, length-controlled INTUITOR 0.969, reward model 0.982). A pure length reward, whose error has the same direction on every prompt, points away from gold ($c_\infty = -0.415$), and a reward with no signal stays near zero at every batch size (Figure 1). MV is the exception among the votes: it differs from gold only where the majority is wrong, and there its errors are specific to the prompt (Section 4.2). Length-controlled INTUITOR stays at 0.94–0.97 on the Qwen models but falls to 0.76–0.79 on the instruction-tuned Llama models (Table 15), although model family and instruction tuning are confounded here.

Table 3: Error structure, not error rate, sets the update direction. Disattenuated cosine c_∞ between the expected update of each reward and the expected gold update, all on the same Qwen3-1.7B-Base responses at initialization ($N = 636$ prompt groups). A value of 1 means the reward moves the model in the gold direction; an interval below 1 means its errors add a systematic deviation. Wrong is the percentage of responses whose reward differs from gold; ρ_e is the mean pairwise alignment of the per-prompt errors ($\times 10^{-3}$). Random vote rewards a randomly chosen response's answer; self-certainty vote the answer with the highest aggregated self-certainty.

Reward	Wrong (%)	ρ_e ($\times 10^{-3}$)	c_∞ (95% CI)
Independent noise: gold labels flipped at random with probability ε			
$\varepsilon = 0.05$	5.5	1.4	1.000 (.979, 1.019)
$\varepsilon = 0.10$	9.8	0.8	1.003 (.977, 1.026)
$\varepsilon = 0.20$	20.4	1.2	1.011 (.969, 1.059)
$\varepsilon = 0.30$	30.8	0.2	1.036[‡]
Shared-rule noise: the same fraction flipped by response length			
$\varepsilon = 0.10$	9.7	48.3	0.773 (.746, .797)
$\varepsilon = 0.20$	20.4	42.8	0.287 (.249, .323)
$\varepsilon = 0.30$	31.4	42.0	−0.129 (−.160, −.084)
Votes over sampled answers			
Majority vote	7.5	0.3	0.997 (.989, 1.006)
Self-certainty vote	23.3	1.6	0.975 (.955, .984)
Random vote	18.9	6.3	0.961 (.942, .989)
Rewards that favor a kind of response on every prompt			
Reward model (GRM-Llama3.2-3B)	–	54.3	0.982 (.974, .991)
INTUITOR, length-controlled	–	41.6	0.969 (.955, .971)
INTUITOR	–	48.5	0.902
Response length	–	14.2	−0.415 (−.458, −.399)

[‡] $Z = 0.27$ here, below the threshold for an interval. An error rate is given only for $\{0, 1\}$ rewards; the four below are continuous scores, so disagreement with gold is undefined for them. The ρ_e null here is 1.1×10^{-3}: the independent flips scatter around it (0.2–1.4) and all keep c_∞ at 1, while the rewards that fall short sit an order of magnitude higher. ρ_e separates the regimes by scale, not by a threshold.

4.4 Do the errors stay incoherent across steps?

Everything so far holds at fixed parameters. Extending it to a run needs the per-step errors to be mutually incoherent too: to first order the gap between the two runs' parameters is $\eta \sum_{t \le T} E_t$, which grows as $\sqrt{T}$ while the E_t share no direction and as T once they do. We sketch the MV run at steps $t = 3, \ldots, 11$ on each step's own on-policy rollouts and score both rewards on identical samples.

The per-step errors are nearly, but not exactly, orthogonal. Over the thirty-six pairs $t \neq s$, $\cos(E_t, E_s) = 0.030$, against 0.001 for a random reward and 0.400 for the gold gradients on the same steps (Table 17); a sketched cosine resolves 0.004, so the error carries a small shared component, an order of magnitude below the signal's but above the floor. It shows in the accumulation: $\| \sum_{t \le T} E_t \|$ grows with exponent 0.594 against 0.762 for gold, and randomising the sign of each E_t (preserving norms, destroying direction) gives a $\sqrt{T}$ reference of 0.540 [0.522, 0.576]. The observed exponent lies above it ($p = 0.005$): the error accumulates slightly faster than a random walk, while staying far short of the signal's coherence.

The horizon is short, though the estimate is stable over it and the test strengthens steadily as it grows: $p = 0.064$ over $t = 3, \ldots, 7$, then 0.026, 0.013, 0.009 and 0.005 as each further step is added, with the exponent moving only between 0.594 and 0.598. Gradients at different t sit at different parameters, making these first-order comparisons.

5 Training Dynamics

Sections 3 and 4 compare updates at fixed parameters, but training changes the model and with it the errors that MV makes. Following the gold and MV runs of Qwen3-4B-Base, which see identical prompts at every step, we ask whether the property behind the equivalence survives training, and whether equivalence at each step makes the two runs end in the same place. Answer parsing degrades slowly rather than failing: the parseable fraction falls from 0.84 at step 8 to 0.75 at step 20 while the majority answer's accuracy stays flat, and what moves monotonically is the concentration of the vote (Appendix A). We therefore report step 20 throughout.

Table 4: The MV update stays close to the gold update on the same batch. Qwen3-4B-Base along the MV run. Writing $G_{\text{mv}} = (1-\beta)G_{\text{gold}} + E_{\perp}$ (Section 4), shrinkage β is the fraction of the gold step that MV does not deliver (negative: MV moves further along the gold direction than gold does), and misdirection $R_{\perp} = \|E_{\perp}\|/\|G_{\text{gold}}\|$ is the size of the part pointing elsewhere. Cosine is $\cos(G_{\text{mv}}, G_{\text{gold}})$ on the same batch, error/signal is $\|\sum E\|/\|\sum G_{\text{gold}}\|$, and ceiling is the split-half gold vs. gold cosine, which is anomalously low at step 10.

Checkpoint	Cosine	Shrinkage β	Misdirection $R_{\perp}$	Error/signal	Ceiling
Base	0.927	−0.055	0.428	0.432	0.647[†]
Step 10	0.810	0.143	0.621	0.637	0.265
Step 20	0.972	0.020	0.239	0.240	0.658

[†]The pooled arm of Table 1.

The errors stay incoherent. MV errors stay at the random-reward null at steps 10 and 20, while gold gradients stay coherent (Table 2); at step 20 the error's growth exponent is still $1/2$ and the gold exponent reaches 0.90 (Table 11). The error keeps shrinking relative to the signal: $\|\sum E\|/\|\sum G_{\text{gold}}\|$ falls from 0.43 at initialization to 0.24 at step 20, where $c_{\infty} = 1.000$ [0.972, 1.030]. By the decomposition in Table 4, MV at step 20 delivers 98% of the gold component and is rotated from the gold update by $14°$. Step 10 is an outlier in every measure, including gold's agreement with itself. Holding the prompts fixed confirms this: on the identical 256 step-21 prompts, shrinkage is flat ($\beta = 0.019 \to 0.020$) while misdirection halves ($R_{\perp} = 0.480 \to 0.239$), so the whole rise in batch cosine, $0.898 \to 0.972$, comes from misdirection (Table 6). The step-10 excursion is not specific to one prompt draw: an independently seeded MV run gives the same depressed gold–gold ceiling at step 10 (0.276 against 0.248), with c_{∞} unreportable in both ($Z = 0.27$ and 0.25). We do not have an explanation for it.

The runs track each other on the training distribution. Evaluated on both runs, the MV reward rises almost as fast in the gold run as in the run that optimizes it (Figure 4, Appendix C), because the two rewards coincide wherever the majority is correct. Training accuracy is indistinguishable through step 15 (prompt-paired difference $+0.002$ [-0.001, $+0.006$] over 3,840 pairs). The runs differ in the weight placed on wrong majorities: the gap between MV reward and accuracy grows from 0.09 to 0.15 in the MV run but stays at 0.09 to 0.10 in the gold run.

MV repairs wrong majorities through other prompts. Of the 134 level-5 MATH-500 problems, 66 have a wrong majority under the base model; by step 20 of the MV run, 13 of them (19.7%) have a correct one (Table 14), about three times the flip rate between two independent halves of the samples. Since the run does not train on these problems, the repair comes from the shared update. Breakage runs at 4.4%, below the resampling rate. The gains are largest at small k, where maj@1 and maj@4 rise by 17 to 19 points, but maj@16 and maj@64 rise significantly too ($+0.100$ and $+0.082$). MV training changes the answer the model commits to on a single sample more than it changes the vote. The evaluation problems are all but disjoint from training: against the 29k DeepScaleR problems, one level-5 problem appears verbatim and two more exceed a 0.80 token-overlap threshold (Appendix A). Dropping all three leaves the repair rate inside its interval.

Local equivalence does not make the trajectories identical. On held-out problems, gold training ends ahead. At step 10 on the level-5 problems, averaging over two seeds per reward, gold leads MV in maj@1 (0.344 vs. 0.320) and maj@64 (0.590 vs. 0.541), though not in pass@64 (0.855 vs. 0.862); on all of MATH-500 at step 20, maj@64 rises from 0.747 to 0.772 with MV and to 0.845 with gold. Doubling the MV batch to 512 prompts does not close the gap (step-10 maj@64 0.545 vs. 0.582, paired 95% CI $[-0.007, +0.086]$), as expected when 256 prompts is already far above B^*, so the gap must come from something that per-step misdirection does not capture. Labels must also come from the current policy: labels from the frozen base model are nearly as accurate as on-policy MV but deliver far less of the gold update (Appendix C). Paired over the 500 problems, MV's gain over base is significant at every k but collapses with it, from $+0.237$ [$+0.223$, $+0.252$] at maj@1 to $+0.026$ [$+0.005$, $+0.047$] at maj@64, against $+0.262$ and $+0.099$ for gold. At step 10 gold's advantage is weaker than the spread between seeds (Table 8).

The property behind the equivalence thus persists through training, yet the runs separate on held-out majority accuracy: equivalence at each step bounds the error an update adds, not how errors accumulate (Section 7).

6 Related Work

Self-generated rewards. Models can improve from their own outputs, by fine-tuning on self-consistent answers (Huang et al., 2023; Wang et al., 2023) or by judging their own responses (Yuan et al., 2024; Li et al., 2026c); see Tao et al. (2024) and Tan et al. (2024) for surveys. In RL for reasoning, the reward can come from majority vote (Zuo et al., 2026; Shafayat et al., 2025), from confidence or entropy (Kang et al., 2025; Zhao et al., 2026b; Huang et al., 2026a; Prabhudesai et al., 2025; Agarwal et al., 2026; Zhang et al., 2025b), or from stabilized variants (Zhou et al., 2025; Zhang et al., 2025d; Yu et al., 2026). Self-play systems also generate the tasks themselves (Huang et al., 2026d; Yu et al., 2025; Chen et al., 2025a; Huang et al., 2026b;f; Yang et al., 2026; Zhao et al., 2026a; Liu et al., 2026; 2025; Yue et al., 2026; Shi et al., 2026; Huang et al., 2026c; He et al., 2025; Li et al., 2026a;b), sometimes with checks on the generated data (Lai et al., 2026; Shen et al., 2026; Liang et al., 2026). Rather than propose a new reward, we explain when such rewards can replace verification.

Label-free reinforcement learning. When label-free training works remains unsettled (Rahman et al., 2026; He et al., 2026). It can collapse (Shafayat et al., 2025; Zhang et al., 2025c; Jin et al., 2026), reinforce the model's own preferences (He et al., 2026; Arazo et al., 2020), or lose the correct answer when the consensus is wrong (Lin et al., 2026), which has motivated complementary signals and verifiers (Yan et al., 2026; Pan et al., 2026; Wang et al., 2025a; Liao et al., 2026). Gains can also come from spurious rewards or contamination (Shao et al., 2025; Chen et al., 2025b; Wu et al., 2025), and RL may mostly sharpen existing behavior unless exploration is broadened (Yue et al., 2025; Huang et al., 2026e). Existing explanations are prompt-level (He et al., 2026; Zhang et al., 2025a). Ours is batch-level: in the spirit of label-noise robustness (Rolnick et al., 2017) and shared gradient directions (Chatterjee, 2020; McCandlish et al., 2018; Wei et al., 2026), it asks whether the error between two reward sources has a shared component.

7 Conclusion and Limitations

We have argued that the batch-level update, not the individual prompt, is the right unit for understanding self-generated rewards. A fixed policy can have a wrong modal answer on a given problem, and MV will return it; but reinforcement learning updates shared parameters from many diverse prompts, and a wrong label matters only when errors across a batch share a direction. In our measurements they do not: a pretrained model's residual errors are sparse and prompt-specific, stay incoherent across prompts at every checkpoint we measure, and cancel in aggregate, while structure shared across prompts is reinforced.

The same view clarifies what verification adds. With any reward, GRPO improves only behavior that can be expressed as a coherent change to one shared policy, so the advantage of verifiable over self-generated labels should be largest for errors that recur across many prompts: in domains with widely shared misconceptions (Lin et al., 2022; Prelec et al., 2017), or when the same prompts are revisited until their errors accumulate. For label-free methods, the criterion suggests that making errors incoherent, by labelling with the current policy (Appendix C) and drawing fresh and diverse prompts, matters as much as making them rare.

Limitations. Our training runs use one model, Qwen3-4B-Base, and Qwen models benefit even from rewards unrelated to correctness (Shao et al., 2025); only the gradient-level results cover five models from two families. The runs are short and unregularized, and beyond step 15 the MV run's entropy rises sharply, a regime we do not analyze. We have two runs per reward and measure cosines of raw gradients, whereas Adam's preconditioner can reweight directions. Most importantly, per-step equivalence does not establish equivalence of trajectories, and over the nine steps we measure the errors already accumulate slightly faster than a random walk (Section 4.4).

AI USE STATEMENT

We used generative AI systems as research assistants. Claude Code (Anthropic) was used to implement and debug experiment and analysis software and to manage cluster jobs. Claude (Anthropic) was used to draft and edit portions of the manuscript. The language models studied in this work generated the training rollouts and evaluation completions described in the main text and appendix. The authors ran the experiments, reviewed the AI-assisted code, analyses, and text, and take responsibility for all reported methods, claims, and artifacts.

REPRODUCIBILITY STATEMENT

Appendix A lists the training configuration, prompt format, evaluation protocol, and gradient-sketch procedure, and Appendix B gives all proofs. We will release, under an open licence: the gradient-sketching code and the count-sketch implementation; the reward-reweighting, split-half and disattenuation estimators that produce every number in this paper; the scripts that generate each table and figure; and the launcher and configuration files for every training and evaluation run, including the majority-vote advantage estimator. We will also release the per-response gradient sketches, the gold and majority-vote label sets, the full text of every generation used in the analyses rather than extracted answers alone, and the dated experiment log recording each run's configuration, failures, and diagnosed causes, so that both the reported analyses and the path taken to them can be reproduced.

REFERENCES

Shivam Agarwal, Zimin Zhang, Lifan Yuan, Jiawei Han, and Hao Peng. The unreasonable effectiveness of entropy minimization in llm reasoning. *Advances in Neural Information Processing Systems*, 38:107150–107180, 2026.

Eric Arazo, Diego Ortego, Paul Albert, Noel E. O'Connor, and Kevin McGuinness. Pseudo-labeling and confirmation bias in deep semi-supervised learning. In *International Joint Conference on Neural Networks*, 2020.

Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. In *International Colloquium on Automata, Languages, and Programming*, 2002.

Satrajit Chatterjee. Coherent gradients: An approach to understanding generalization in gradient descent-based optimization. In *International Conference on Learning Representations*, 2020.

Lili Chen, Mihir Prabhudesai, Katerina Fragkiadaki, Hao Liu, and Deepak Pathak. Self-questioning language models. *arXiv preprint arXiv:2508.03682*, 2025a.

Peter Chen et al. Exploration vs. exploitation: Rethinking RLVR through clipping, entropy, and spurious reward. *arXiv preprint arXiv:2512.16912*, 2025b.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Bingxiang He, Yuxin Zuo, Zeyuan Liu, Shangziqi Zhao, Zixuan Fu, Junlin Yang, Cheng Qian, Kaiyan Zhang, Yuchen Fan, Ganqu Cui, Xiusi Chen, Youbang Sun, Xingtai Lv, Xuekai Zhu, Li Sheng, Ran Li, Huan-ang Gao, Yuchen Zhang, Lifan Yuan, Bowen Zhou, Zhiyuan Liu, and Ning Ding. How far can unsupervised RLVR scale LLM training? In *International Conference on Learning Representations*, 2026.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.211. URL https://aclanthology.org/2024.acl-long.211/.

Yicheng He, Chengsong Huang, Zongxia Li, Jiaxin Huang, and Yonghui Yang. Visplay: Self-evolving vision-language models from images. *arXiv preprint arXiv:2511.15661*, 2025.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2021.

Chengsong Huang, Langlin Huang, Jixuan Leng, Jiacheng Liu, and Jiaxin Huang. Efficient test-time scaling via self-calibration. In *International Conference on Learning Representations*, 2026a.

Chengsong Huang, Haolin Liu, Tong Zheng, Runpeng Dai, Langlin Huang, Jinyuan Li, Zongxia Li, Zhepei Wei, Yu Meng, and Jiaxin Huang. G-zero: Self-play for open-ended generation from zero data. *arXiv preprint arXiv:2605.09959*, 2026b.

Chengsong Huang, Zifeng Wang, Rujun Han, Jun Yan, Yanfei Chen, Zoey CuiZhu, Ke Jiang, Peng Xia, Han Yu, Yufan Zhuang, et al. Envharness: Awakening static worlds for agent learning. *arXiv preprint arXiv:2608.19880*, 2026c.

Chengsong Huang et al. R-zero: Self-evolving reasoning llm from zero data. In *International Conference on Learning Representations*, 2026d. arXiv:2508.05004.

Jiaxin Huang, Shixiang Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. Large language models can self-improve. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 1051–1068, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.67. URL https://aclanthology.org/2023.emnlp-main.67/.

Langlin Huang, Chengsong Huang, Jinyuan Li, Donghong Cai, Yuyi Yang, and Jiaxin Huang. Nonsense helps: Prompt space perturbation broadens reasoning exploration. In *Findings of the Association for Computational Linguistics: EMNLP 2026*, 2026e.

Siyuan Huang, Pengyu Cheng, Haotian Liu, Tao Chen, Yihao Liu, Jingwei Ni, Shijie Zhou, Ziyi Yang, Gangwei Jiang, Mengyu Zhou, et al. Skill self-play: Pushing the frontier of llm capability with co-evolving skills. *arXiv preprint arXiv:2607.22529*, 2026f.

Xiaolong Jin et al. When do intrinsic rewards work for code reasoning? A comprehensive study. *arXiv preprint arXiv:2606.20881*, 2026.

Zhewei Kang, Xuandong Zhao, and Dawn Song. Scalable best-of-N selection for large language models via self-certainty. *arXiv preprint*, 2025.

Yuhang Lai, Jiazhan Feng, Yee Whye Teh, and Ning Miao. Verifier-backed hard problem generation for mathematical reasoning. *arXiv preprint arXiv:2605.06660*, 2026.

Jixuan Leng, Chengsong Huang, Banghua Zhu, and Jiaxin Huang. Taming overconfidence in LLMs: Reward calibration in RLHF. In *International Conference on Learning Representations*, 2025.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *Advances in neural information processing systems*, 35:3843–3857, 2022.

Zongxia Li, Hongyang Du, Chengsong Huang, Xiyang Wu, Lantao Yu, Yicheng He, Jing Xie, Xiaomin Wu, Zhichao Liu, Jiarui Zhang, et al. Mm-zero: Self-evolving multi-model vision language models from zero data. *arXiv preprint arXiv:2603.09206*, 2026a.

Zongxia Li, Hongyang Du, Chengsong Huang, Lantao Yu, Zhichao Liu, Xiyang Wu, Yicheng He, Jing Xie, Xiaomin Wu, Jiarui Zhang, and Fuxiao Liu. Self-evolving vision language models for structured visual understanding. In *Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing*, 2026b.

Zongxia Li, Wenhao Yu, Chengsong Huang, Zhenwen Liang, Rui Liu, Fuxiao Liu, Jingxi Che, Dian Yu, Jordan Boyd-Graber, Haitao Mi, and Dong Yu. Self-rewarding vision-language model via reasoning decomposition. In *International Conference on Learning Representations*, 2026c.

Hao Liang, Meiyi Qiang, Yuying Li, Zefeng He, Xiaochen Ma, Ruichuan An, Yongzhen Guo, Zhengzhou Zhu, Bin Cui, and Wentao Zhang. Mathdebugger: Detecting and diagnosing errors in synthetic mathematical data, 2026. URL https://arxiv.org/abs/2502.19058.

Ruotong Liao, Nikolai Röhrich, Xiaohan Wang, Yuhui Zhang, Yasaman Samadzadeh, Volker Tresp, and Serena Yeung-Levy. Tool verification for test-time reinforcement learning. *arXiv preprint arXiv:2603.02203*, 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. In *International Conference on Learning Representations*, 2024.

Hongxiang Lin, Zhirui Kuai, Erpeng Xue, and Lei Wang. Detecting and mitigating the correct-answer extinction window in test-time reinforcement learning with majority voting. *arXiv preprint arXiv:2605.19444*, 2026.

Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, 2022.

Bo Liu, Chuanyang Jin, Seungone Kim, Weizhe Yuan, Wenting Zhao, Ilia Kulikov, Xian Li, Sainbayar Sukhbaatar, Jack Lanchantin, and Jason Weston. Spice: Self-play in corpus environments improves reasoning. *arXiv preprint arXiv:2510.24684*, 2025.

Bo Liu, Simon Yu, Zichen Liu, Leon Guertler, Penghui Qi, Daniel Balcells, Mickel Liu, Cheston Tan, Weiyan Shi, Min Lin, et al. Spiral: Self-play on zero-sum games incentivizes reasoning via multi-agent multi-turn reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pp. 35407–35434, 2026.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Li Erran Li, Raluca Ada Popa, and Ion Stoica. DeepScaleR: Surpassing O1-Preview with a 1.5B model by scaling RL. Notion blog, 2025.

Sam McCandlish, Jared Kaplan, Dario Amodei, and OpenAI Dota Team. An empirical model of large-batch training. *arXiv preprint arXiv:1812.06162*, 2018.

Teng Pan, Yuchen Yan, Zixuan Wang, et al. CoVerRL: Breaking the consensus trap in label-free reasoning via generator-verifier co-evolution. *arXiv preprint arXiv:2603.17775*, 2026.

Mihir Prabhudesai, Lili Chen, Alex Ippoliti, Katerina Fragkiadaki, Hao Liu, and Deepak Pathak. Maximizing confidence alone improves reasoning. *arXiv preprint arXiv:2505.22660*, 2025.

Dražen Prelec, H. Sebastian Seung, and John McCoy. A solution to the single-question crowd wisdom problem. *Nature*, 541:532–535, 2017.

Salman Rahman, Jingyan Shen, Anna Mordvina, Hamid Palangi, Saadia Gabriel, and Pavel Izmailov. When can LLMs learn to reason with weak supervision? *arXiv preprint arXiv:2604.18574*, 2026.

David Rolnick, Andreas Veit, Serge Belongie, and Nir Shavit. Deep learning is robust to massive label noise. *arXiv preprint arXiv:1705.10694*, 2017.

Sheikh Shafayat, Fahim Tajwar, Ruslan Salakhutdinov, Jeff Schneider, and Andrea Zanette. Can large reasoning models self-train? *arXiv preprint arXiv:2505.21444*, 2025.

Rulin Shao et al. Spurious rewards: Rethinking training signals in RLVR. *arXiv preprint arXiv:2506.10947*, 2025.

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

Chengyu Shen, Zhen Hao Wong, Runming He, Hao Liang, Meiyi Qiang, Zimo Meng, Zhengyang Zhao, Bohan Zeng, Zhengzhou Zhu, Bin Cui, et al. Let's verify math questions step by step. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, pp. 2770–2781, 2026.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. HybridFlow: A flexible and efficient RLHF framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, 2025.

Yucheng Shi, Zhenwen Liang, Kishan Panaganti, Dian Yu, Wenhao Yu, and Haitao Mi. Learning to build the environment: Self-evolving reasoning rl via verifiable environment synthesis. *arXiv preprint arXiv:2605.14392*, 2026.

Charles Spearman. The proof and measurement of association between two things. *The American Journal of Psychology*, 15(1):72–101, 1904.

Zhen Tan, Dawei Li, Song Wang, Alimohammad Beigi, Bohan Jiang, Amrita Bhattacharjee, Mansooreh Karami, Jundong Li, Lu Cheng, and Huan Liu. Large language models for data annotation and synthesis: A survey. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pp. 930–957, 2024.

Zhengwei Tao, Ting-En Lin, Xiancai Chen, Hangyu Li, Yuchuan Wu, Yongbin Li, Zhi Jin, Fei Huang, Dacheng Tao, and Jingren Zhou. A survey on self-evolution of large language models. *arXiv preprint arXiv:2404.14387*, 2024.

W. Wang, Y. Wang, K. Chen, and H. Huang. Beyond majority voting: Towards fine-grained and more reliable reward signal for test-time reinforcement learning. *arXiv preprint arXiv:2512.15146*, 2025a.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023.

Zengzhi Wang, Fan Zhou, Xuefeng Li, and Pengfei Liu. Octothinker: Mid-training incentivizes reinforcement learning scaling. *arXiv preprint arXiv:2506.20512*, 2025b.

Zhepei Wei, Xinyu Zhu, Wei-Lin Chen, Chengsong Huang, Jiaxin Huang, and Yu Meng. RELEX: You only need minimal RLVR training: Extrapolating LLMs via rank-1 trajectories. In *Advances in Neural Information Processing Systems*, 2026.

Mingqi Wu et al. Reasoning or memorization: Unreliable results of reinforcement learning due to data contamination. *arXiv preprint*, 2025.

Dong Yan, Jian Liang, Yanbo Wang, Shuo Lu, Ran He, and Tieniu Tan. What if consensus lies? selective-complementary reinforcement learning at test time. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, 2026.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Chengyi Yang, Zhishang Xiang, Yunbo Tang, Zongpei Teng, Chengsong Huang, Fei Long, Yuhan Liu, and Jinsong Su. TTCS: Test-time curriculum synthesis for self-evolving. In *Advances in Neural Information Processing Systems*, 2026.

Wenhao Yu, Zhenwen Liang, Chengsong Huang, Kishan Panaganti, Tianqing Fang, Haitao Mi, and Dong Yu. Guided self-evolving llms with minimal human supervision. *arXiv preprint arXiv:2512.02472*, 2025.

Zhaoning Yu, Zhaolun Su, Leitian Tao, Haozhu Wang, Aashu Singh, Hanchao Yu, Jianyu Wang, Hongyang Gao, Weizhe Yuan, Jason Weston, Ping Yu, and Jing Xu. RESTRAIN: From spurious votes to signals—self-training RL with self-penalization. In *International Conference on Learning Representations*, 2026.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-rewarding language models. In *International Conference on Machine Learning*, 2024.

Yang Yue et al. Does reinforcement learning really incentivize reasoning capacity in LLMs beyond the base model? *arXiv preprint arXiv:2504.13837*, 2025.

Zhenrui Yue, Kartikeya Upasani, Xianjun Yang, Suyu Ge, Shaoliang Nie, Yuning Mao, Zhe Liu, and Dong Wang. Dr. zero: Self-evolving search agents without training data. *arXiv preprint arXiv:2601.07055*, 2026.

Kongcheng Zhang, Qi Yao, Shunyu Liu, Yingjie Wang, Baisheng Lai, Jieping Ye, Mingli Song, and Dacheng Tao. Consistent paths lead to truth: Self-rewarding reinforcement learning for LLM reasoning. *arXiv preprint arXiv:2506.08745*, 2025a.

Qingyang Zhang, Haitao Wu, Changqing Zhang, Peilin Zhao, and Yatao Bian. Right question is already half the answer: Fully unsupervised LLM reasoning incentivization. *arXiv preprint arXiv:2504.05812*, 2025b.

Yanzhi Zhang, Zhaoxi Zhang, Haoxiang Guan, Yilin Cheng, Yitong Duan, Chen Wang, Yue Wang, Shuxin Zheng, and Jiyan He. No free lunch: Rethinking internal feedback for LLM reasoning. *arXiv preprint arXiv:2506.17219*, 2025c.

Zizhuo Zhang, Jianing Zhu, Xinmu Ge, Zihua Zhao, Zhanke Zhou, Xuan Li, Xiao Feng, Jiangchao Yao, and Bo Han. Co-rewarding: Stable self-supervised RL for eliciting reasoning in large language models. *arXiv preprint arXiv:2508.00410*, 2025d.

Andrew Zhao, Yiran Wu, Tong Wu, Quentin Xu, Yang Yue, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. Absolute zero: Reinforced self-play reasoning with zero data. *Advances in Neural Information Processing Systems*, 38:105816–105879, 2026a.

Xuandong Zhao, Zhewei Kang, Aosong Feng, Sergey Levine, and Dawn Song. Learning to reason without external rewards. In *International Conference on Learning Representations*, volume 2026, pp. 2548–2581, 2026b.

Tong Zheng, Hongming Zhang, Wenhao Yu, Xiaoyang Wang, Runpeng Dai, Rui Liu, Huiwen Bao, Chengsong Huang, Heng Huang, and Dong Yu. Parallel-R1: Towards parallel thinking via reinforcement learning. In *International Conference on Learning Representations*, 2026.

Yujun Zhou, Zhenwen Liang, Haolin Liu, Wenhao Yu, Kishan Panaganti, Linfeng Song, Dian Yu, Xiangliang Zhang, Haitao Mi, and Dong Yu. Evolving language models without labels: Majority drives selection, novelty promotes variation. *arXiv preprint arXiv:2509.15194*, 2025.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Xinwei Long, Ermo Hua, Biqing Qi, et al. Ttrl: Test-time reinforcement learning. *Advances in Neural Information Processing Systems*, 38:131459–131483, 2026.

A Experimental Details

Training. Table 5 lists the configuration shared by the gold and MV runs. The two runs start from the same checkpoint and use the same data order, so every step trains on the same prompts. Each step processes 256 prompts in two optimizer updates of 128 prompts. The training loss uses std-normalized advantages for both rewards.

Table 5: Training configuration shared by the gold and MV runs.

Setting	Value
Model and data	
Base model	Qwen3-4B-Base
Training data	DeepScaleR (29k problems)
Framework	verl 0.8.0 (Sheng et al., 2025)
Training steps	26
Batch	
Prompts per step (B)	256
Responses per prompt (n)	16
Mini-batch	128 prompts (2 updates per step)
PPO epochs	1
Optimization	
Learning rate	1×10^{-6}, constant
Clip range	$\epsilon_{\text{low}} = 0.2$, $\epsilon_{\text{high}} = 0.28$
Loss aggregation	token mean
Advantage	group mean-centered, divided by group std
KL / entropy coefficient	0 / 0
Gradient clipping	1.0
Weight decay	0.01
Sampling	
Temperature, top-p, top-k	1.0, 1.0, 40
Max response length	4096 tokens

Prompt format and answer extraction. Prompts use the Qwen chat template with the system prompt "Please reason step by step, and put your final answer within `\boxed{}`." Thinking mode is disabled, so the response begins with an empty thinking block. The answer of a response is the content of the last `\boxed{}` in its final 400 characters. A response whose answer cannot be parsed takes no part in the vote and receives reward 0; ties are broken toward the answer that appears first in the group. The training reward and the reward applied to the stored sketches use the same rule.

Evaluation. We evaluate on MATH-500 and on its 134 level-5 problems, drawing 64 samples per problem at temperature 1.0 and top-p 1.0 with a 4096-token limit. maj@k is the accuracy of the most frequent parsable answer among k samples.

Table 6: Same-prompt replication (Section 5). Base and step 20 evaluated on the *identical* 256 step-21 prompts, each scored on its own on-policy rollouts. Shrinkage is unchanged while misdirection halves, and the two reproduce the measured batch cosine. Steps 5 and 15 are absent because the run does not checkpoint there.

	Base	Step 20
Mixed-outcome prompts (of 256)	125	165
wrong- / right-majority	41 / 84	81 / 84
$\cos(G_{\text{mv}}, G_{\text{gold}})$	0.898	**0.972**
$\|G_{\text{mv}}\| / \|G_{\text{gold}}\|$	1.092	1.009
Shrinkage β	$+0.019$	$+0.020$
Misdirection $R_\perp$	0.480	**0.239**
$(1-\beta)/\sqrt{(1-\beta)^2 + R_\perp^2}$	0.898	0.972

Table 7: Std-normalized advantages give the same answer. c_∞ between the MV and gold updates with advantages mean-centered ($s_i = 1$) and with s_i set to the group standard deviation, the weighting the training loss uses, computed from the same stored sketches. Step 10 is omitted: its own split-half agreement is $Z \approx 0.25$, below the threshold at which c_∞ is reported.

Model	N	Mean-centered	Std-normalized
Qwen3-1.7B-Base	636	0.997 (.989, 1.006)	0.994 (.982, 1.008)
Qwen3-4B-Base (on-policy)	164	0.999 (.965, 1.043)	1.000 (.968, 1.034)
Qwen3-4B-Base step 20	165	1.002 (.981, 1.015)	1.002 (.985, 1.013)
Qwen2.5-7B	151	0.998 (.968, 1.027)	0.996 (.962, 1.030)
Llama-3.2-3B-Instruct	247	0.996 (.929, 1.078)	0.998 (.911, 1.096)
Llama-3.1-8B-Instruct	208	1.007 (.944, 1.079)	1.017 (.929, 1.119)

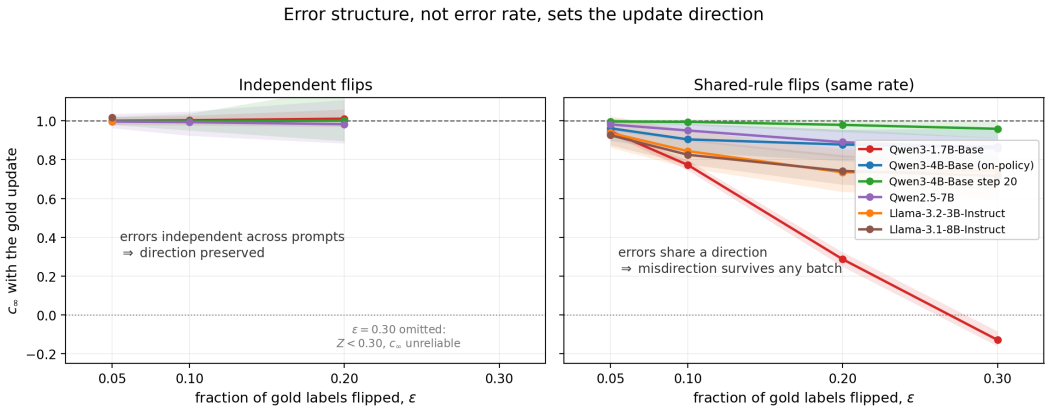

Figure 2: Error structure, not error rate, sets the update direction (Section 4). Both panels corrupt the same fraction ε of gold labels on the same responses and differ only in how the flips are chosen: independently per response (left), or by a rule shared across prompts, the longest correct and shortest incorrect responses (right). Independent flips leave c_∞ at 1 on every model; shared-rule flips fall monotonically on every model, and on Qwen3-1.7B-Base the update turns away from gold entirely by $\varepsilon = 0.30$. Bands are 95% bootstrap intervals. We plot c_∞ rather than the raw cosine against batch size because the two schemes differ in their own split-half agreement Z and so are attenuated by different amounts; the raw curve can rank the shared-rule flip above the independent one, which reflects its lower sampling noise rather than better alignment. Points with $Z < 0.30$ are omitted.

Table 8: Run-to-run variation is comparable to the gold–MV gap (Section 5). Two MV runs and two gold runs, trained from different seeds, all evaluated at step 10 on the same 134 level-5 problems with 64 samples each. maj@k for $k < 64$ averages over random k-subsets drawn without replacement; at $k = 64$ it uses the whole pool. Answers are clustered and graded exactly as the evaluation harness does, including its treatment of ties, so these numbers are directly comparable with those quoted in Section 5.

Run	maj@1	maj@16	maj@64	pass@64
MV, seed A	0.325	0.525	0.545	0.858
MV, seed B	0.314	0.505	0.537	0.866
Gold, seed A	0.348	0.550	0.585	0.843
Gold, seed B	0.339	0.557	0.595	0.866
Within-reward spread (MV)	0.011	0.019	0.007	0.007
Within-reward spread (gold)	0.008	0.007	0.010	0.022
Gold − MV, mean of seeds	+0.024	+0.039	+0.049	−0.007

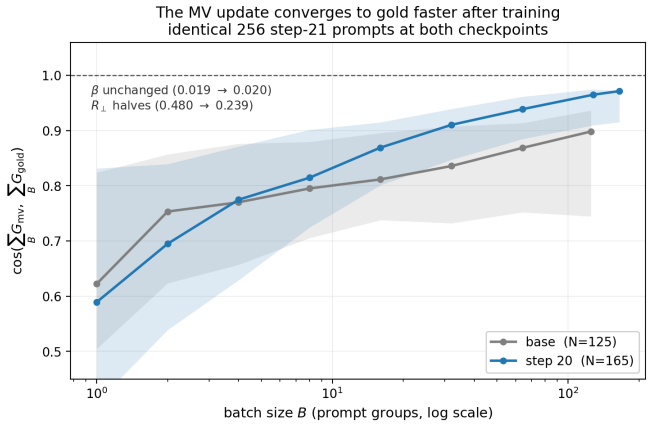

Figure 3: The same-prompt comparison of Table 6, as a function of batch size. Both checkpoints are evaluated on the identical 256 step-21 prompts, each on its own on-policy rollouts, so the difference between the curves cannot be a difference in prompt composition. Bands are 95% bootstrap intervals over prompt groups and overlap at every B; the figure illustrates the decomposition rather than adding evidence for it. The curves cross near $B = 4$: per prompt the step-20 update is no closer to gold than the base update is, and the gain is entirely a batch effect, which is what β unchanged and $R_\perp$ halved means.

Parsing over training. Table 9 tracks answer parsing along the gold run. Parsing improves over the first eight steps and then declines slowly; it does not break down. The accuracy of the majority answer is flat across the whole run, so the decline in parse rate is not accompanied by a decline in label quality. The vote does become steadily more concentrated, which is the same sharpening that Appendix C identifies as the reason on-policy labels deliver more of the gold update than frozen ones. The independently seeded MV run of Table 8, which checkpoints through step 12, parses at 0.87 at step 10 and 0.86 at step 12.

Table 9: Answer parsing along the gold run of Qwen3-4B-Base, over the 4,096 responses generated at each step. *Parse rate* is the fraction of responses with an extractable answer, *majority share* the fraction of parsed responses holding the group's plurality answer, and *maj. correct* the fraction of groups whose plurality answer matches the reference.

Step	1	8	10	15	20	26
Parse rate	0.668	0.843	0.840	0.811	0.748	0.682
Majority share	0.382	0.408	0.420	0.465	0.511	0.548
Maj. correct	0.344	0.348	0.398	0.352	0.316	0.367

Evaluation–training overlap. We checked all 500 MATH-500 problems against the 29,000 DeepScaleR training problems under three tests: equality of whitespace- and LaTeX-normalised strings, substring containment in either direction, and token-set Jaccard overlap at a 0.80 threshold. One problem matches exactly and 22 exceed the overlap threshold; restricted to the 134 level-5 problems used in Section 5, the counts are one exact and two near matches. The detector was validated by planting verbatim and reformatted copies of training problems into the evaluation set, all of which it recovered. Excluding the three flagged level-5 problems changes the wrong-to-right majority repair rate by less than its bootstrap interval.

Gradient sketches. For each response we compute the gradient of $\log \pi_\theta(y \mid x)$, summed over response tokens, at the policy that generated it, and compress it with a count sketch to $K = 2^{16}$ dimensions (fixed hash seed). The group gradient under a reward is $\sum_j (r_{ij} - \bar{r}_i)\, s_{ij}$, where s_{ij} is the sketch of response j. A checkpoint from step t is sketched on the responses the run generated at step $t + 1$, so the sketches are on-policy. For Qwen3-4B-Base each checkpoint yields 256×16 responses. Qwen3-1.7B-Base is sketched on 1536 prompts and the two Llama models on 512 fresh

DeepScaleR prompts, with 16 responses each. Qwen2.5-7B is sketched on 256 prompts with 16 responses each. The pooled Qwen3-4B arm of 512 prompts combines two batches of 256, the second generated after one update, so half of it is off-policy; the on-policy row of Table 1 is the first batch alone, and every result we draw conclusions from is reported for that row as well. Cosines from sketches agree with exact fp32 gradients to within 0.0039.

Groups with no gold signal. Section 2 restricts the comparisons to groups that are mixed under the gold reward, because a group whose responses all receive the same gold reward contributes the zero vector to G_{gold} and admits no cosine with it. MV is frequently still live on the excluded groups, so the restriction is not neutral for the MV side. Repeating the Qwen3-4B-Base comparison over all 256 prompts rather than the 164 mixed ones leaves the disattenuated cosine unchanged ($c_\infty = 0.985$, 95% CI $[0.940, 1.073]$, against 0.999 $[0.965, 1.043]$) while raising the norm ratio $\|G_{\text{mv}}\|/\|G_{\text{gold}}\|$ from 1.14 to 1.62 and the misdirection $R_\perp$ from 0.43 to 0.75. Including the gold-silent groups therefore makes the MV update larger without turning it: the extra mass has no gold counterpart to align with, which is what a group carrying MV signal and no gold signal must contribute.

Split-half estimates. X, Y, and Z are averaged over random partitions of the prompt groups into two halves. Confidence intervals come from a bootstrap over prompt groups in which the groups are partitioned first and then resampled within each half. Point estimates average X, Y, and Z over 200 random partitions. Intervals come from 250 bootstrap resamples of the prompt groups, with 12 partitions drawn inside each resample.

B Proofs

Proposition 1. Expanding the squared norm into diagonal and off-diagonal terms gives $\|\sum_i v_i\|^2 = \sum_i \|v_i\|^2 + \sum_{i\neq j}\langle v_i, v_j\rangle = Bs^2 + \bar{\rho}\,(B-1)\,Bs^2$. Treating s and $\bar{\rho}$ as constant in B and differentiating $\frac{1}{2}\log\big(Bs^2(1+(B-1)\bar{\rho})\big)$ with respect to $\log B$ gives $\alpha(B)$.

Proposition 2. Prompts whose majority label is correct contribute nothing to $G_{\text{mv}} - G_{\text{gold}}$, so $E_\perp = \sum_{i\in\mathcal{W}} e_i$, where $\mathcal{W}$ is the set of εB prompts with a wrong majority label. Applying Proposition 1 to the εB vectors e_i and to the B gold prompt gradients gives $\|E_\perp\|^2 = \varepsilon B\, s_e^2\,(1+(\varepsilon B-1)\rho_e)$ and $\|G_{\text{gold}}\|^2 = B\, s_g^2\,(1+(B-1)\rho_g)$, and the ratio is $R_\perp^2$. For $\rho_e = 0$ the ratio is $\varepsilon(s_e^2/s_g^2)/(1+(B-1)\rho_g) \to 0$. For $\rho_e > 0$ and large B it tends to $\varepsilon^2(s_e^2/s_g^2)(\rho_e/\rho_g)$.

Corollary 1. Write the flipped reward as $\tilde{r}_{ij} = r_{ij} + (1-2r_{ij})\,\zeta_{ij}$ with independent $\zeta_{ij} \sim \text{Bernoulli}(\varepsilon)$. Then $\mathbb{E}[\tilde{r}_{ij}] = (1-2\varepsilon)\,r_{ij} + \varepsilon$. Mean-centering is linear and removes the constant, so $\mathbb{E}[\tilde{A}_{ij}] = (1-2\varepsilon)A_{ij}$, and since g_i is linear in the advantages, $\mathbb{E}[g_i(\tilde{r})] = (1-2\varepsilon)\,g_i(r)$. Summing over prompts gives $\mathbb{E}[G(\tilde{r})] = (1-2\varepsilon)\,G_{\text{gold}}$. The deviations $g_i(\tilde{r}) - \mathbb{E}[g_i(\tilde{r})]$ depend only on the flips of prompt i, so they have zero mean and are independent across prompts. The expected update is parallel to the expected gold update, so $c_\infty = 1$.

The disattenuated cosine. Let the half-batch updates be $G_k^{\mathcal{H}} = m_k + \xi_k^{\mathcal{H}}$ for $k \in \{\text{mv}, \text{gold}\}$ and halves $\mathcal{H} \in \{\mathcal{A}, \mathcal{B}\}$, where m_k is the expected update and the noise from sampling prompts is independent across halves, with the same distribution in both. Because X, Y, and Z pair different halves, the inner products in their numerators concentrate on $\langle m_{\text{mv}}, m_{\text{gold}}\rangle$, $\|m_{\text{gold}}\|^2$, and $\|m_{\text{mv}}\|^2$, while the noise inflates only the norms in the denominators. The norm factors cancel in $X/\sqrt{YZ}$, which leaves $\langle m_{\text{mv}}, m_{\text{gold}}\rangle/(\|m_{\text{mv}}\|\,\|m_{\text{gold}}\|)$.

C Additional Results

Labels must come from the current policy. The labeller's accuracy is again not what matters. At step 20 we relabel the MV run's own responses with the majority answer of the frozen base model on the same prompts. These labels are nearly as accurate as the on-policy majority (a correct majority on 33% vs. 35% of prompts), yet they deliver only 63% of the gold component of the update ($\beta = 0.37$), against 98% for on-policy MV, and add more misdirection ($R_\perp = 0.29$ vs. 0.24). Stale

labels shrink the update far more than current ones, in line with the practice of recomputing majority labels from the current policy (Zuo et al., 2026; Shafayat et al., 2025).

Table 10: Label-free methods recover most of the gain of ground-truth RL. Recovery is (method − base)/(GT-RL − base), computed from benchmark-averaged scores reported within the same paper; values are means over settings, with per-setting results in Table 16. For MV, only online variants (labels from the current policy) are included.

Method	Source	Settings	Range	Mean recovery
Online MV				
TTRL[†]	Zuo et al. (2026)	2	109–111%	110%
Plurality-vote GRPO	Zhao et al. (2026b)	2	69–98%	83%
SRT	Shafayat et al. (2025)	4	55–100%	81%
Modified label-free rewards				
RESTRAIN	Yu et al. (2026)	2	81–97%	89%
INTUITOR	Zhao et al. (2026b)	6	76–111%	96%

[†]TTRL trains on evaluation questions, whereas its ground-truth reference (Oat-Zero) trains on MATH-train; this row therefore overstates recovery relative to the others.

Table 11: Errors grow like a random walk while the gold signal grows coherently. Local growth exponents $\alpha(B)$ of summed gradients between successive batch sizes, Qwen3-4B-Base. By Proposition 1, $\alpha = 1/2$ indicates incoherent vectors and $\alpha \to 1$ a dominant shared direction.

		Growth exponent $\alpha(B)$		
Checkpoint	B range	MV error E	Gold	Gap
Base	$8 \to 16$	0.524	0.581	0.057
	$32 \to 64$	0.500	0.720	0.220
	$64 \to 128$	0.524	0.813	0.289
	$128 \to 164$	0.509	0.874	0.365
Step 20	$16 \to 32$	0.506	0.684	0.179
	$64 \to 128$	0.503	0.835	0.333
	$128 \to 165$	0.486	0.901	0.415

Table 12: Full split-half results behind Table 1. Δ is the 95% interval of (MV vs. gold) minus (gold vs. gold); an interval containing zero means MV cannot be distinguished from gold. MV vs. MV is the agreement of MV with itself across halves.

Model	N	Gold vs. gold	MV vs. gold	MV vs. MV	Δ (95% CI)
Qwen3-4B-Base[†]	331	0.647	0.647	0.646	(−.014, +.014)
Qwen3-1.7B-Base	636	0.842	0.834	0.831	(−.016, −.003)
Qwen2.5-7B	151	0.799	0.818	0.843	(+.010, +.044)
Llama-3.2-3B-Instruct	247	0.371	0.360	0.352	(−.022, +.010)
Llama-3.1-8B-Instruct	208	0.438	0.399	0.362	(−.050, −.016)

[†]Pools two batches of 256 prompts; the second was generated after one update.

Table 13: Full version of Table 2, adding the MV gradients themselves. Mean pairwise cosine between prompt gradients ($\times 10^{-3}$), Qwen3-4B-Base along the MV run.

	Right majority				Wrong majority			
Checkpoint	Gold	MV	MV error E	Null	Gold	MV	MV error E	Null
Base	22.41	20.06	0.39	0.30	6.55	9.26	0.23	0.69
Step 10	6.52	6.52	0.01	0.36	0.82	2.79	0.15	0.50
Step 20	27.99	26.94	0.14	−0.15	13.44	14.71	0.55	0.66

Table 14: MV training repairs wrong majorities on held-out problems. 134 level-5 MATH-500 problems, base model against step 20 of the MV run (66 wrong-majority and 68 right-majority problems at base); majorities use all 64 samples. (a) maj@k and its change, with a 95% bootstrap interval over problems. (b) Problems whose majority answer changes; the resampling null is the flip rate between two disjoint halves of 32 samples at step 20. *The interval excludes zero in (a) and the resampling null in (b).

(a) Majority-vote accuracy

	Base	Step 20	Change (95% CI)
maj@1	0.185	0.374	+0.188 (+.158, +.218)*
maj@4	0.292	0.466	+0.174 (+.145, +.206)*
maj@16	0.443	0.543	+0.100 (+.066, +.136)*
maj@64	0.503	0.585	+0.082 (+.030, +.138)*

(b) Majority-answer flips, base → step 20

	Observed	Rate, % (95% CI)	Resampling null
Wrong → right	13/66	19.7 (10.6, 28.8)*	4/55 (7.3%)
Right → wrong	3/68	4.4 (0.0, 9.3)	10/79 (12.7%)

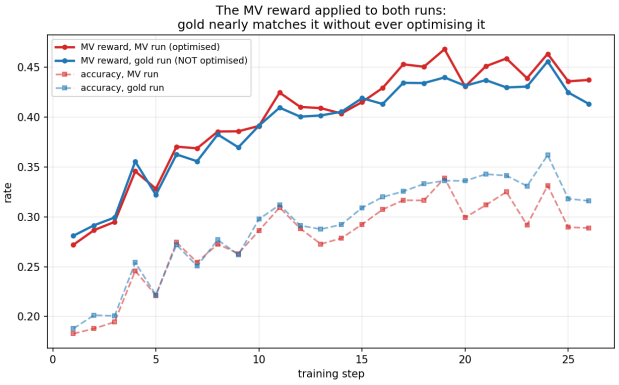

Figure 4: MV reward (solid) and accuracy (dashed) on the training prompts of the gold and MV runs of Qwen3-4B-Base. The gold run nearly matches the MV run on the MV reward without ever optimizing it.

Table 15: INTUITOR's alignment with gold depends on the model. Disattenuated cosine c_∞ between the INTUITOR and gold updates at initialization, raw and after controlling for response length.

Model	Raw	Length-controlled (95% CI)
Qwen3-4B-Base†	0.990	0.973 (.934, 1.029)
Qwen3-4B-Base (on-policy)	0.950	0.941 (.871, .993)
Qwen3-1.7B-Base	0.902	0.969 (.955, .971)
Qwen2.5-7B	0.536	0.959 (.926, 1.017)
Llama-3.2-3B-Instruct	0.855	0.786 (.727, .862)
Llama-3.1-8B-Instruct	0.883	0.756 (.738, .825)

†The pooled arm of Table 1.

Table 16: Per-setting numbers behind Table 10. Scores are the average over the listed benchmarks (pass@1, %), including GSM8K (Cobbe et al., 2021). Recovery = (Method − Base)/(GT-RL − Base). SRT values are read off Shafayat et al. (2025), Fig. 3 (final step), and are approximate.

Method	Model	Train set	Benchmarks averaged	Base	GT-RL	Method	Recovery
Zuo et al. (2026) — GT-RL = Oat-Zero (same backbone), greedy decoding							
TTRL	Qwen2.5-Math-1.5B	test set	AIME24, AMC, MATH-500	28.5	49.1	51.0	109%
TTRL	Qwen2.5-Math-7B	test set	AIME24, AMC, MATH-500	35.3	62.0	65.0	111%
Zhao et al. (2026b) — GT-RL = GRPO with gold answers, greedy decoding							
Plurality-vote GRPO	Qwen2.5-3B	MATH	GSM8K, MATH-500	60.9	73.1	72.8	98%
Plurality-vote GRPO	Llama-3.2-3B-Inst	MATH	GSM8K, MATH-500	56.2	60.4	59.1	69%
INTUITOR	Qwen2.5-1.5B	MATH	GSM8K, MATH-500	4.6	65.4	62.1	95%
INTUITOR	Qwen2.5-3B	MATH	GSM8K, MATH-500	60.9	73.1	70.2	76%
INTUITOR	Qwen2.5-7B	MATH	GSM8K, MATH-500	59.5	79.0	81.2	111%
INTUITOR	Qwen2.5-14B	MATH	GSM8K, MATH-500	71.3	83.8	84.7	107%
INTUITOR	Llama-3.2-3B-Inst	MATH	GSM8K, MATH-500	56.2	60.4	60.0	90%
INTUITOR	OLMo-2-7B-SFT	MATH	GSM8K, MATH-500	49.7	54.2	54.1	98%
Shafayat et al. (2025) — GT-RL = RLOO with gold answers							
SRT	Llama-3.1-8B-Inst	Big-Math	MATH-500	52.7	59.0	58.5	92%
SRT	Qwen2.5-Math-7B	MATH-12K	MATH-500	43	78	78.5	100%
SRT	Qwen3-14B-Base	DAPO	MATH-500	52	86.5	71	55%
SRT	DeepSeek-Math-7B-Inst	MATH-12K	MATH-500	43.0	47.5	46.4	76%
Yu et al. (2026) — GT-RL = GRPO with gold labels; 16 samples, T=0.6							
RESTRAIN	Qwen3-4B-Base	DAPO-14k	6 benchmarks*	38.0	51.4	51.0	97%
RESTRAIN	OctoThinker-8B	DAPO-14k	6 benchmarks*	19.2	39.6	35.8	81%

*MATH-500, AIME25, OlympiadBench (He et al., 2024), Minerva (Lewkowycz et al., 2022), MMLU-STEM, GPQA-Diamond. OctoThinker is from Wang et al. (2025b). On Qwen3-4B, RESTRAIN exceeds GT-RL on MMLU-STEM and GPQA-Diamond; on the four math benchmarks alone its recovery is about 80%.

Table 17: The per-step MV errors do not share a direction (Section 4.4). Qwen3-4B-Base along the MV run, each checkpoint sketched on its own on-policy rollouts. *Coherence* is the mean cosine over the ten step pairs $t \neq s$; the random reward is evaluated through the same pipeline and its expected update is the zero vector, so it gives the floor. *Exponent* is the log–log slope of $\|\sum_{t\le T} v_t\|$ in T. The sign-flip reference randomises the sign of each E_t, preserving all norms while destroying any consistent direction, and so is the $\sqrt{T}$ null for these data.

	Coherence $\cos(v_t, v_s)$	Growth exponent
MV error E_t	0.030	0.594
Gold gradient	0.400	0.762
Random reward (floor)	0.001	–
Sign-flip reference ($\sqrt{T}$ null)	–	0.540 (.522, .576)

A single sketched cosine resolves 0.004. Steps $t = 3, \ldots, 11$ (every checkpoint the run saved); $N = 163$–174 mixed-gold groups per step. The observed exponent exceeds the sign-flip reference with $p = 0.005$.

Table 18: A prompt's own gradient barely outweighs another prompt's. Change in the output distribution of prompt q after a small step along q's own gradient (d_{own}) or along another prompt's gradient (d_{other}), measured on identical tokens (Qwen3-1.7B-Base, 8 prompts, fp64). A fully tabular model would give own/other $\to \infty$; a fully shared one would give 1.

Step size	d_{own}	d_{other}	Own/other
10^{-7}	7.18×10^{-6}	3.70×10^{-6}	1.94
10^{-8}	7.23×10^{-8}	4.19×10^{-8}	1.72

Table 19: A single prompt's contribution shrinks with batch size. *Own share*: $d_{\text{own}}/d_{\text{batch}}$, the fraction of the change in a prompt's output caused by its own gradient within a batch of size B (step size 10^{-7}). *Scaling*, computed over every prompt group from the gradient sketches: the KL-share slope is the log–log slope of the KL change from swapping one prompt against B (i.i.d. theory: -2); writing $g_q = \mu + \xi_q$, the shared energy is $\|\mu\|^2$ as a fraction of the total, and B^* is the batch size at which the shared component overtakes the prompt-specific one.

	Own share (%)				Scaling		
Model	B=8	B=16	B=64	B=128	KL-share slope	Shared energy	B^*
Qwen3-1.7B-Base	9.81	3.37	0.38	0.13	-1.37	2.27%	43
Qwen3-4B-Base	3.36	0.86	0.18	0.09	-1.31	2.63%	37
Llama-3.2-3B-Instruct	29.74	25.69	0.83	–	-1.15	1.49%	66

For Llama-3.2-3B-Instruct the share barely falls from $B = 8$ to 16 because the small-batch gradient is noisy (d_{batch} moves only from 1.63×10^{-5} to 1.89×10^{-5}).