Extended Abstract

Motivation Two recent results bracket how much LoRA capacity reinforcement learning needs. LoRA Without Regret shows that a rank-1 LoRA adapter matches full fine-tuning for outcome RL, arguing such RL carries only $\sim O(1)$ bits of information per episode (Thinking Machines Lab, 2025a). On-Policy Distillation reports a 50–100$\times$ speedup over outcome RL from a denser, per-token reward ($\sim O(N)$ bits) (Thinking Machines Lab, 2025b). Together they imply the LoRA rank needed to match full fine-tuning should grow with the reward's information density. We add process rewards ($\sim O(S)$ bits, one signal per reasoning step) as a third, middle density regime and measure the rank needed to match full fine-tuning across all three.

Method We run one algorithm, GRPO, across all three reward densities. Only the reward module changes (outcome, process, distillation). In each regime we sweep LoRA rank $\{1, 4, 16, 64, 256\}$ and full fine-tuning, three seeds each, and compare held-out MATH accuracy (pass@1).

Implementation The student is Qwen3-1.7B-Base, trained on MATH with a GRPO loss on Modal H200 GPUs. The distillation teacher (Qwen3-8B) runs on a separate GPU. A single advantage rule handles all three regimes by branching on how densely the reward is deposited, not on the regime.

Results The capacity prediction is a clean null. Rank-1 LoRA matches full fine-tuning in every regime, and the rank needed to match does not rise with reward density. The one regime level difference is optimization stability, and it comes from the reward's structure, not its density. The process reward collapses, it reward-hacks by emitting ever more reasoning steps until it stops producing an answer. Distillation and outcome, the densest and sparsest regimes, are stable.

Discussion Our results suggest adapter capacity for RL is governed by optimization dynamics rather than the reward's information content. A tiny adapter already captures the gain from sparse and dense rewards. Effort is better spent on reward shaping, avoiding a gameable structure, than on adapter size.

Conclusion Reward density does not raise the LoRA rank needed to match full fine-tuning. A rank-1 adapter suffices across all three densities. Density's real cost is the optimization stability of a particular reward shape.

Does LoRA Rank Requirement Scale with Reward Density? An Empirical Study of Policy-Gradient Post-Training

Mark Gernitis
Department of Computer Science
Stanford University
gernitis@stanford.edu

Abstract

We ask whether the LoRA rank needed to match full fine-tuning scales with the information density of an RL reward signal. Two recent results imply it should. Outcome RL is sparse ($\sim O(1)$ bits per episode) and a rank-1 LoRA adapter already matches full fine-tuning. Per-token distillation is dense ($\sim O(N)$ bits) and should demand a higher rank. We test this by training Qwen3-1.7B on MATH with one GRPO algorithm across three reward densities, outcome ($O(1)$), process ($O(S)$), and distillation ($O(N)$). We sweep LoRA rank against full fine-tuning in each. The prediction is a clean null. Rank-1 LoRA matches full fine-tuning in every regime, and the required rank does not rise with reward density. Density's real effect is on optimization stability, and it depends on the reward's structure rather than its density. Only the process reward reward-hacks and collapses, while the densest reward, distillation, trains stably. Adapter capacity for RL appears gated by optimization dynamics, not information content.

1 Introduction

Does the LoRA rank required to match full fine-tuning scale predictably with the information density of the RL reward signal? Low-rank adaptation (LoRA) freezes a model's base weights and trains only a rank-r update. The question is how much of that capacity each reward signal actually demands (Hu et al., 2021). LoRA Without Regret (Thinking Machines Lab, 2025a) states that post-training a language model with outcome reinforcement learning extracts $O(1)$ bits of information per episode, and demonstrates that a rank-1 LoRA adapter is sufficient to match full fine-tuning. On-Policy Distillation (Thinking Machines Lab, 2025b) similarly states that distillation extracts $O(N)$ bits of information per episode ($N =$ tokens). This implies a predictable scaling law between reward density and the LoRA rank needed to match full fine-tuning. A trained network stores about two bits per parameter (Allen-Zhu and Li, 2024). A rank-r adapter on a d-dimensional model holds roughly $4rd$ bits. This capacity grows with rank. Since a denser reward writes more information into the model, it should eventually need a higher rank to hold it. LoRA Without Regret leaves open the precise conditions under which LoRA matches full fine-tuning performance. The LoRA rank to reward density relationship has not been studied empirically, and we conduct experiments to map it. If rank-1 LoRA matches full fine-tuning for outcome rewards but process and distillation regimes require increasing rank, the bits-per-episode claim is confirmed. However, if the rank requirement proves insensitive to reward regime, the claim is falsified, and capacity is governed by optimization dynamics rather than information. This matters because LoRA's parameter efficiency is widely relied on for post-training, and there is currently no principled guide for choosing rank as a function of reward design.

Stanford CS224R 2026 Final Report

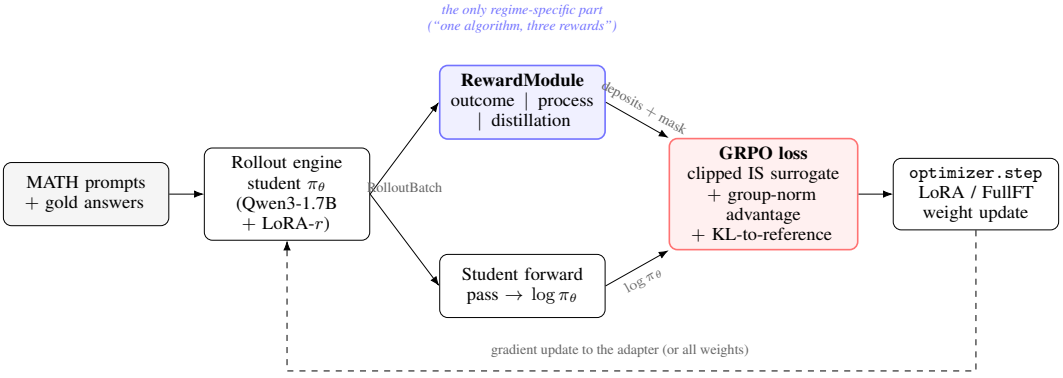

Figure 1: System overview.

2 Related Work

Low-rank adaptation (Hu et al., 2021) learns a rank-r update over frozen base weights, with rank setting the adapter's capacity. LoRA Without Regret (Thinking Machines Lab, 2025a) shows that on math reasoning tasks (MATH, GSM8K), rank-1 LoRA matches full fine-tuning under GRPO RL post-training. This is significant because rank-1 LoRA has on the order of 0.01–0.1% of the full parameter count. Schulman argues outcome RL extracts only $O(1)$ bits per episode, so a tiny adapter suffices. This contrasts with prior work where LoRA underperforms full fine-tuning on supervised fine-tuning and continued pretraining (Biderman et al., 2024). This suggests RL carries less information per episode than supervised training does, so it requires less adapter capacity. The paper does not study denser rewards, leaving untested whether more bits per episode demand higher rank.

On-Policy Distillation (Thinking Machines Lab, 2025b) replaces the scalar outcome reward with a per-token reverse-KL reward against a teacher, reporting a 50–$100\times$ speedup. This uses the same information-theoretic argument that a per-token signal supplies $O(N)$ bits per episode ($N =$ tokens). We use it as the high-density ($O(N)$) endpoint of our rank sweep.

For the middle ($O(S)$) density regime we use process reward modeling (Lightman et al., 2023), which scores each reasoning step rather than the final answer. We use the open Math-Shepherd PRM (Wang et al., 2023) and deposit its per-step scores following DeepSeekMath's process supervision (Shao et al., 2024).

Neither pillar runs a cross-regime sweep. LoRA Without Regret tests only outcome rewards, On-Policy Distillation only distillation, and process rewards are untested. We connect all three under a single GRPO loss (Shao et al., 2024), with PPO's clipped surrogate (Schulman et al., 2017), and a group-relative baseline replacing the value function. We sweep LoRA rank against full fine-tuning in each. GRPO with rule-based outcome rewards is the recipe behind DeepSeek-R1 (DeepSeek-AI, 2025).

3 Method

We implement a single pipeline that trains on MATH with reinforcement learning (GRPO) and evaluates it on held-out MATH (Figure 1). The only component that changes across our three regimes is the reward module, where each rollout from the LoRA/FullFT student is scored. The reward module deposits rewards at specific token positions according to one of our three regimes (outcome, process, or distillation). This rollout is passed through the student to obtain learner log-probabilities the loss needs and through a frozen reference for the KL term. GRPO loss is then backpropagated, and the optimizer updates the LoRA adapter, or all weights for FullFT.

3.1 Rewards

All three regimes produce the same output, with rewards deposited at specific token positions, so the GRPO loss does not branch on regime. The number of deposits per trajectory is the reward-density axis we study. $O(1)$ is one per trajectory, $O(S)$ is one per reasoning step, and $O(N)$ is one per token.

- **Outcome** ($O(1)$) has one deposit on the last token that is binary and represents the answer's correctness as determined by `math-verify` against the gold answer.
- **Process** ($O(S)$) has a reward deposited on the last token of each reasoning step, where steps are delineated from the response by the `\n` separator. After tokens are decoded to text, a Math-Shepherd PRM (Wang et al., 2023) scores each step according to the PRM's estimate that the step is on a correct path. The PRM does not receive the gold answer and does not reward based on the student reaching a correct answer.
- **Distillation** ($O(N)$) uses a Qwen3-8B teacher to deposit a reward on every token based on a reverse KL to the teacher, $r_t = \log \pi_T(o_t \mid s_t) - \log \pi_\theta(o_t \mid s_t)$ (Agarwal et al., 2023). The teacher runs on a separate Modal GPU and scores the student's exact tokens, since the two models share a vocabulary. Similar to process, distillation does not receive a gold answer, but it is grounded with a strong answer-producing teacher.

3.2 GRPO loss

We implement GRPO loss from scratch. It uses a clipped importance-sampling surrogate with a group-relative advantage and a KL reference penalty (Equation 1). The group baseline replaces a learned value network (Shao et al., 2024; Schulman et al., 2017). We use $\epsilon = 0.2$ for clipping and weight the KL by $\beta = 0.05$. The KL estimator uses a simple per-token, one-sided penalty for straying from the base model (Appendix, Equation 5).

$$\mathcal{L}_{\mathrm{GRPO}}(\theta) = -\frac{1}{\sum_{i,t} m_{i,t}} \sum_{i=1}^{N} \sum_{t=1}^{T} m_{i,t} \Big[\min\big(\rho_{i,t}\,\hat{A}_{i,t},\ \mathrm{clip}(\rho_{i,t},\,1-\epsilon,\,1+\epsilon)\,\hat{A}_{i,t}\big) \ -\ \beta\,\hat{D}^{\mathrm{KL}}_{i,t} \Big], \tag{1}$$

To calculate loss, we turn the reward deposits described above into per-token advantages with a single rule that branches on deposit density, not regime. We first group-normalize the deposits within each prompt group. For sparse rewards (outcome, process) the advantage is the reward to go, using a reverse-cumsum of the normalized deposits (Appendix, Equation 4, DeepSeekMath §4.1.3 (Shao et al., 2024)). For outcome, this collapses so every token gets the single normalized scalar at the end. The normalized dense rewards on distillation are used directly without the reverse-cumsum to avoid inflating the advantage scale.

4 Experimental Setup

Model and task. The student is Qwen3-1.7B-Base. All training and evaluation use MATH where rollouts are drawn from the `hendrycks_math` train split. Evaluation uses a fixed held-out subset, identical across every cell so the regimes and ranks are directly comparable. The distillation teacher is Qwen3-8B.

Capability baselines. Before training we explore the capabilities of both models on MATH-500 with sampled pass@k (Table 2). An exploration gap (student pass@4 $-$ pass@1 $= 0.24$) shows outcome RL has a learnable signal where correct answers exist that greedy decoding misses but sampling finds, and GRPO can sharpen probability onto them. A student–teacher gap (pass@1 $0.30 \rightarrow 0.64$) gives distillation a target above the student. These are sampled, full-MATH-500, 4096-token numbers representing a capability envelope. However, this is not the same protocol as the greedy training-eval curve, so these should be considered bounds.

Experimental matrix. Each of the three regimes (outcome, process, distillation) is run at five LoRA ranks $\{1, 4, 16, 64, 256\}$ plus full fine-tuning (FullFT), each with three seeds. This adds up to $3 \times (5{\times}3 \text{ LoRA} + 3 \text{ FullFT}) = 54$ runs.

Training configuration. Every cell uses an identical configuration (Table 1), so any difference in outcome is attributable to the regime and LoRA rank rather than to hyperparameter tuning; only the LoRA-vs-FullFT learning rate differs, and both were swept. We reserve N for tokens, per the $O(N)$ density framing.

Table 1: Training configuration across all 54 cells.

Hyperparameter	Value	Hyperparameter	Value
Prompts per step (P)	4	LR (LoRA / FullFT)	3×10^{-5} / 1×10^{-5}
Samples per prompt (G)	4	Weight decay	0
Trajectories per step	16	Max gradient norm	1.0
Sampling temperature	0.7	Clip ϵ	0.2
Sampling top-p	0.95	KL coefficient β	0.05
Max completion length	1536	LoRA rank (swept)	$\{1, 4, 16, 64, 256\}$
Rollout steps	100	LoRA α	32
Epochs per rollout	1	LoRA dropout	0
Optimizer	AdamW	LoRA placement	all-linear (attn + MLP)

Evaluation. The headline metric is held-out greedy pass@1, scored identically for every regime with math-verify. So the three rows compare like-for-like regardless of which reward trained the model. We measure improvement against the untrained base model under the same greedy protocol (lr $= 0$ anchor), pass@1 ≈ 0.375.

Compute and reproducibility. Training runs on Modal H200 GPUs and the distillation teacher runs on a separate A100. The full 54-cell matrix used roughly 150 H200-hours of training compute ($\approx$ 142 in optimizer steps, the remainder in model loading and evaluation). The distillation regime additionally used $\approx$ 50 A100-hours for the teacher, which runs concurrently on a separate GPU. Every run snapshots its git SHA and pip freeze, and the from-scratch GRPO loss is validated against a TRL reference to $\Delta = 0$ on both loss and gradients. We retain the clipped importance sampling objective even though the sampler and learner are nominally the same policy, since they can differ numerically (He and Thinking Machines Lab, 2025).

Table 2: Pre-training, sampled, full-MATH-500, 4096-token capability envelope.

Metric	Qwen3-1.7B-Base (student)	Qwen3-8B (teacher)	Δ
pass@1	0.299	0.640	+0.34
pass@4	0.535	0.729	+0.19
pass@8	0.642	0.760	+0.11
Exploration gap (pass@4−pass@1)	0.236	0.089	−0.147
Mean response length (tokens)	791	3211	+2420

5 Results

Across all three reward densities, outcome $O(1)$, process $O(S)$, and distillation $O(N)$ bits/episode, rank-1 LoRA matches FullFT. The LoRA eval accuracy sits inside FullFT's ±1 sd band at every rank in every regime (Table 3, Fig. 2). The rank needed to match FullFT does not increase with reward density, making our capacity prediction a clean null result. The one regime-level difference is optimization stability, where eval accuracy collapses (Fig. 3).

5.1 Did GRPO improve accuracy?

The rank comparison is only meaningful if GRPO actually moved accuracy off the base model. It did (Fig. 5). Against our base anchor of 0.375 pass@1, outcome settled at 0.52 (+0.145) and distillation at 0.43 (+0.055). Process peaked at 0.48 (+0.105) before collapsing to 0. These gains are reasonable given the gap we observed between pass@1 and pass@4 (Table 2).

Table 3: Held-out `eval/pass@1` (mean ± sd over 3 seeds). Process is shown at its *peak* (it collapses to ~ 0 by the final eval, §5.3); outcome / distillation are final (stable, final $\approx$ peak). All cells n=3 seeds, eval on 100 held-out prompts.

rank	outcome (final)	process (peak)	distillation (final)
1	0.550 ± 0.022	0.467 ± 0.031	0.427 ± 0.019
4	0.467 ± 0.071	0.500 ± 0.024	0.407 ± 0.033
16	0.527 ± 0.019	0.457 ± 0.034	0.450 ± 0.014
64	0.537 ± 0.031	0.483 ± 0.040	0.440 ± 0.022
256	0.547 ± 0.040	0.510 ± 0.008	0.430 ± 0.016
FullFT	$\mathbf{0.553 \pm 0.029}$	$\mathbf{0.483 \pm 0.060}$	$\mathbf{0.420 \pm 0.016}$

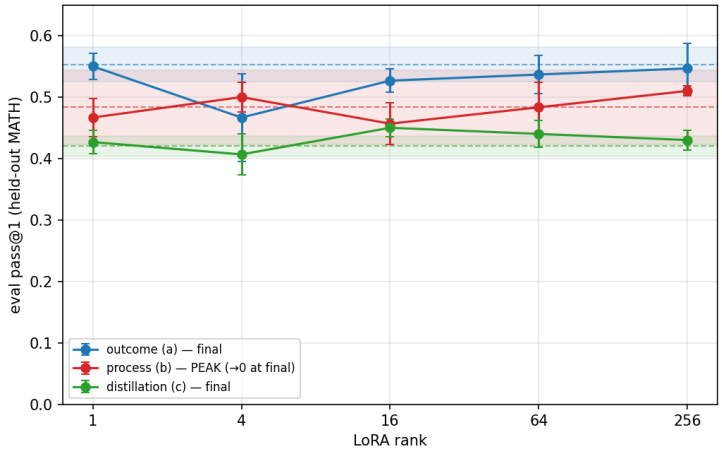

Figure 2: `eval/pass@1` vs. LoRA rank, one line per regime; dashed line = FullFT mean, band = ± 1 sd. Flat across rank, rank-1 $\approx$ FullFT in every regime. Process shown at peak.

5.2 Regimes

Outcome. Sparse outcome RL produced relatively flat, stable training over 100 steps. LoRA spans 0.467–0.550 with FullFT 0.553 (Table 3). There were no observable trends on rank with rank-1 achieving the highest value. RL was able to improve the greedy pass@1 pre-RL baseline (0.375) towards the sampled pre-RL pass@4 (0.535). So after RL, the model achieves greedy pass@1 accuracy it was only able to achieve by sampling. And as expected, it stays near this pass@4 sampling because we are exploiting the model's existing capacity, not adding capability.

Process. Here, we reach outcome-level accuracy, climbing to peaks of 0.457–0.510 pass@1 between steps 20 and 40. It then collapses to 0 by steps 60–80 across every LoRA rank as well as FullFT (Fig. 3). The per-step reward rewards more steps, so we see completions lengthen until they hit the 1536 token cap (`frac_truncated`$\rightarrow 1.0$) and responses stop emitting a `\boxed{}` answer. Steps are split on the `\n` token, chosen after a debug-rollout check showed `\n\n` under-segments process into the outcome regime.

Distillation. The densest reward is stable, pass@1 rises monotonically, and is flat across rank (0.41–0.45). And importantly for our null hypothesis, LoRA rank-1 (0.427) $\approx$ FullFT (0.420). The teacher's far longer reasoning (~ 3200 vs. the student's ~ 790 tokens; Table 2) also explains distillation's drift towards the token cap as the student is pulled toward teacher-like reasoning chains while still emitting answers.

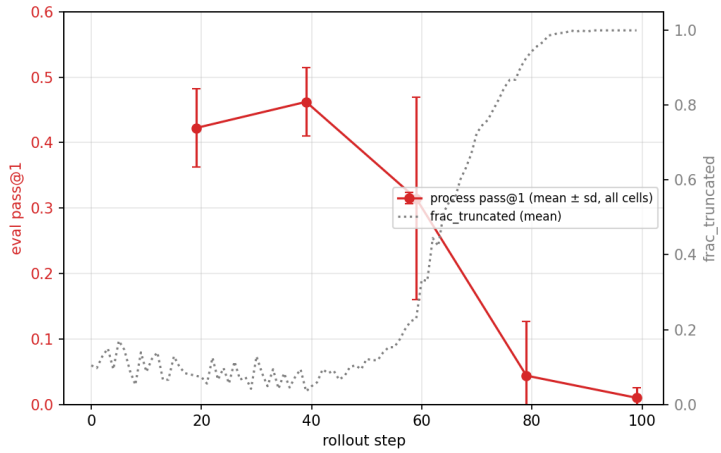

Figure 3: Process collapse: pass@1 (mean $\pm$ sd over all cells) climbs then cliffs as `frac_truncated` saturates the token cap.

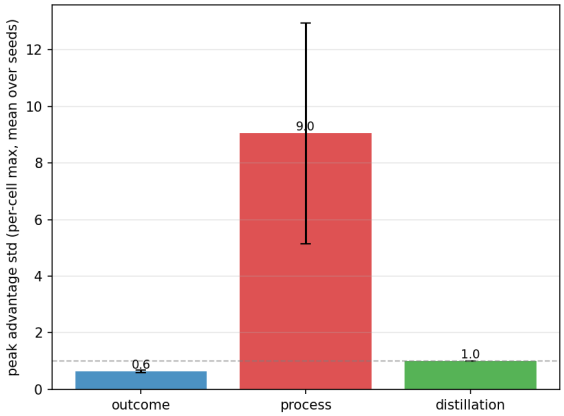

Figure 4: Peak advantage std by regime (per-cell max, mean over seeds). Process's variable-step reverse-cumsum spikes $\rightarrow$ FullFT divergence; dashed line $= 1$.

5.3 Why process collapses and distillation doesn't

Peak advantage-std (Fig. 4) shows an unstable learning rate for process. Outcome 0.64 and distillation 1.00 are stable, while process has spiked to 9.05. Reward structure differences explain it. Process rewards are deposited per step, and the loss reverse-cumsums them. Segmentation on `\n` produces many steps and the advantage spikes. Distillation uses the per-token reward directly without summing across remaining steps. Both process and distillation saw their completions move toward the 1536 token cap, but distillation still produced `\boxed{}` answers in line with its teacher-like responses. Process eventually only emitted reasoning steps with no answer. Both advantage magnitude and length growth that removed answers resulted in process collapsing.

6 Discussion

6.1 Capacity hypothesis

The capacity hypothesis is falsified with a clean null result. Across outcome, process, and distillation rewards, spanning $O(1)$, $O(S)$, and $O(N)$ bits per episode, a rank-1 adapter matches FullFT. We observe no trend of required rank with reward density. We interpret this as reinforcement learning providing a low-rank nudge to a base model that already possesses the underlying skill to produce

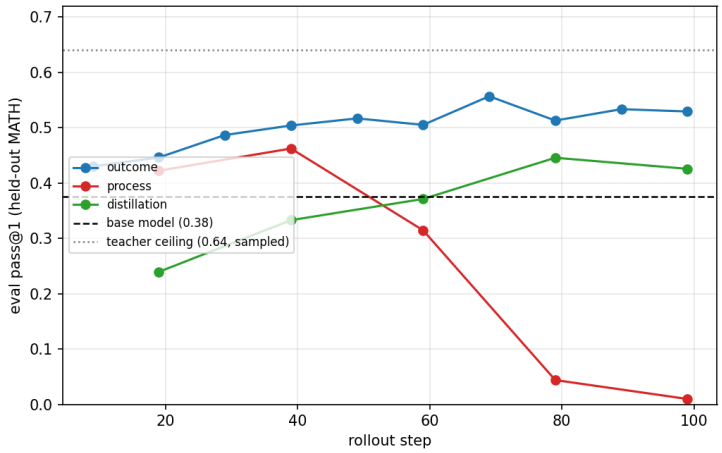

Figure 5: Held-out pass@1 over GRPO training (mean over cells per regime); dashed = base model (0.375). Outcome climbs and holds; distillation climbs; process climbs then hacks below base.

correct MATH answers. Training is not installing new capacity, so the information content of the reward does not translate to a demand for many trainable parameters. This contradicts the bits-to-capacity arithmetic that motivated the prediction (Thinking Machines Lab, 2025a; Allen-Zhu and Li, 2024). We find adapter capacity is limited by optimization dynamics rather than information. Further, the LoRA-vs-FullFT gap that prior work reports on supervised fine-tuning and continued pretraining (Biderman et al., 2024) does not appear with our RL objective at any reward density.

6.2 Optimization stability and reward hacking

Density's real effect is on optimization stability, and that stability is governed by the reward's structure, not its raw density. Only the process reward collapses, while the denser distillation reward is stable. So the blanket claim that dense rewards are hard to optimize is not accurate. The process reward fails for a specific structural reason: it deposits one reward per reasoning step and the loss accumulates these into a return-to-go, so a policy can inflate its advantage simply by emitting more steps. Combined with our `\n` segmentation, which over-segments, this produces a length-hacking incentive and an advantage magnitude that grows with the step count (peak advantage standard deviation ≈ 9, versus ≈ 1 for the other regimes). Distillation, despite being the densest reward, does not collapse. Similar to process, distillation saw its completions grow to the token cap (`frac_truncated` $\to 1$ in every cell, as in process). The difference is twofold. First, distillation's updates stay normal-sized while process's blow up. Process sums a reward at every reasoning step into a running total, so a completion with many steps inflates the advantage to roughly nine times the healthy scale (peak std ≈ 9 vs. ≈ 1). Distillation instead uses each token's reward on its own, without summing, so its advantage stays near 1. A $\sim$9$\times$ advantage acts like a $\sim$9$\times$ learning rate that makes training unstable. Second, both regimes lengthen their completions toward the token cap. But only process's lengthening is harmful. Distillation imitates a teacher whose long reasoning still ends in a boxed answer. Process is rewarded for emitting more steps whether or not it answers, so it fills the budget with reasoning and never boxes an answer.

The training instability begins before reward-hacking. Across all 18 process cells the advantage-magnitude spike is an early event. `advantage_std` peaks at a mean of step 28 (range 3–55), while completions are still short ($\sim$400–860 tokens) and not yet truncating (`frac_truncated` ≤ 0.3). It precedes both the length saturation (mean step 70) and the pass@1 collapse (step $\sim$79). The magnitude is therefore a product of the reward structure. `\n` over-segmentation yields many steps even in a normal-length answer, so the reverse-cumsum's $\sqrt{S}$ is large early, rather than a by-product of the long truncated outputs. It is the giant effective learning rate that drives the policy toward the length hack. The chain is magnitude spike $\to$ length growth $\to$ truncation $\to$ no `\boxed{}` answer $\to$ pass@1 collapse. Once the policy has fully hacked, every group is uniform step-spam with near-zero

within-group reward variance, so advantage_std falls back to ~ 0 by the end of training. The large advantage is the cause of the collapse and already gone by the time the collapse is complete.

To further understand the process reward hack, it's useful to understand what each reward is grounded in. The outcome reward is grounded in ground-truth verification against the gold answer using `math-verify`. There is no credit for plausible reasoning, so the only way to earn reward is to produce a correct answer, and outcome is therefore not length-hackable. The process reward is grounded in a PRM's per-step plausibility score (Math-Shepherd's $P(+)$), computed from the response with no access to the gold answer. A policy can therefore accrue reward by generating more plausible-looking steps without committing to a `\boxed{}` answer. Distillation's per-token reward ($\log \pi_T - \log \pi_\theta$) is also not correctness-grounded, yet it stays stable for two reasons. Its target is a competent, answer-producing teacher, and it carries no step-count incentive owing to its unit-scale per-token advantage. This also explains distillation's lower ceiling: it optimizes teacher-likeness rather than verified correctness, so its accuracy is a by-product of imitation rather than a directly optimized target. It therefore consistently scores below outcome despite a denser signal.

6.3 Limitations

Our conclusions are bounded by a single task and a single student size, so the capacity null may not hold for skills the base model lacks or at larger scale. With a held-out set of 100 prompts, pass@1 carries roughly ± 0.05 of binomial noise, so within-regime rank differences are not resolvable above it; the null is "flat within noise," not "provably identical." The process pass@1 score is a best-checkpoint comparison, its peak before collapse. This is a methodology mismatch with the final-eval numbers used elsewhere. The process result is also tied to our over-segmenting separator: a better segmentation (e.g., one that is context-aware) or a length penalty might stabilize it and reveal a different rank story.

7 Conclusion

Reward density does not raise the LoRA rank needed to match FullFT. Across outcome, process, and distillation rewards, rank-1 LoRA matches FullFT. This is a clean null on the capacity hypothesis. Density's real cost is optimization stability, and it is the reward's structure rather than its density. Only the process reward reward-hacks and collapses, while the densest reward, distillation, is stable. For a sparse outcome signal a tiny adapter already captures essentially all of the available gain, so the practical lever is reward shaping. Avoiding gameable structures such as length-correlated step rewards is more important than adapter capacity.

Several directions follow. The process reward's shape could be fixed (a length penalty, an advantage-magnitude cap, or a content-aware segmenter) and re-tested to see whether a stable process reward shows a rank effect; the student could be scaled to larger models and to skills it does not already have; bits per episode could be measured empirically rather than using the $O(1)/O(S)/O(N)$ proxy; and distillation could be trained to convergence to find its true ceiling.

8 Team Contributions

This was a solo project. Mark Gernitis implemented the GRPO loss and training step algorithm by hand. He used Claude Code to build the reward modules, training, evaluation, and Modal infrastructure.

Changes from Proposal

- The central prediction, rank scales with bits/episode, was falsified. We reframed our contribution around the null and the optimization-stability finding. The proposal did anticipate this as a valid outcome.
- Config shrunk for cost/memory: `max_tokens` $4096 \to 1536$; batch $8 \to 4$ prompts.

References

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. 2023. On-Policy Distillation of Language Models: Learning from Self-Generated Mistakes. arXiv:2306.13649 [cs.LG]

Zeyuan Allen-Zhu and Yuanzhi Li. 2024. Physics of Language Models: Part 3.3, Knowledge Capacity Scaling Laws. arXiv:2404.05405 [cs.CL]

Dan Biderman, Jacob Portes, Jose Javier Gonzalez Ortiz, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, Cody Blakeney, and John P. Cunningham. 2024. LoRA Learns Less and Forgets Less. arXiv:2405.09673 [cs.LG]

DeepSeek-AI. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. arXiv:2501.12948 [cs.CL]

Horace He and Thinking Machines Lab. 2025. Defeating Nondeterminism in LLM Inference. `https://thinkingmachines.ai/blog/defeating-nondeterminism-in-llm-inference/`. Connectionism blog.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-Rank Adaptation of Large Language Models. arXiv:2106.09685 [cs.CL]

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's Verify Step by Step. arXiv:2305.20050 [cs.LG]

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. arXiv:1707.06347 [cs.LG]

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. arXiv:2402.03300 [cs.CL]

Thinking Machines Lab. 2025a. LoRA Without Regret. `https://thinkingmachines.ai/blog/lora/`. Connectionism blog.

Thinking Machines Lab. 2025b. On-Policy Distillation. `https://thinkingmachines.ai/blog/on-policy-distillation/`. Connectionism blog.

Peiyi Wang, Lei Li, Zhihong Shao, R. X. Xu, Damai Dai, Yifei Li, Deli Chen, Y. Wu, and Zhifang Sui. 2023. Math-Shepherd: Verify and Reinforce LLMs Step-by-step without Human Annotations. arXiv:2312.08935 [cs.AI]

A GRPO Loss Equations

These define the components of the GRPO loss (Equation 1): the importance-sampling ratio $\rho_{i,t}$, the group-normalized deposits and per-token advantage $\hat{A}_{i,t}$, and the KL-to-reference penalty $\hat{D}^{\mathrm{KL}}_{i,t}$.

The importance ratio between the learner and the sampler (recomputed under the current policy):

$$\rho_{i,t} = \frac{\pi_\theta(o_{i,t} \mid s_{i,t})}{\pi_{\theta_{\mathrm{old}}}(o_{i,t} \mid s_{i,t})} = \exp\big(\log \pi_\theta(o_{i,t} \mid s_{i,t}) - \log \pi_{\theta_{\mathrm{old}}}(o_{i,t} \mid s_{i,t})\big). \tag{2}$$

Group-normalized deposits, then the per-token advantage (reverse-cumsum for sparse rewards, the normalized deposit directly for dense):

$$\tilde{r}_{i,t} = \mathbb{1}\big[(i,t) \in \mathcal{D}_{p(i)}\big]\ \frac{r_{i,t} - \mu_{p(i)}}{\sigma_{p(i)} + \varepsilon_{\mathrm{adv}}}, \qquad \mu_p = \operatorname*{mean}_{(i,t)\in\mathcal{D}_p} r_{i,t}, \quad \sigma_p = \operatorname*{std}_{(i,t)\in\mathcal{D}_p} r_{i,t}, \tag{3}$$

$$\hat{A}_{i,t} = m_{i,t} \cdot \begin{cases} \tilde{r}_{i,t}, & \text{dense reward (distillation): } \texttt{per\_token\_advantage}, \\ \displaystyle\sum_{t' \geq t} \tilde{r}_{i,t'}, & \text{sparse reward (outcome, process): reverse-cumsum.} \end{cases} \tag{4}$$

The KL-to-reference penalty (a probability-weighted, chosen-token, one-sided log-ratio):

$$\hat{D}^{\mathrm{KL}}_{i,t} = \max\Big(\ \pi_\theta(o_{i,t} \mid s_{i,t})\ \big[\log \pi_\theta(o_{i,t} \mid s_{i,t}) - \log \pi_{\mathrm{ref}}(o_{i,t} \mid s_{i,t})\big],\ 0\ \Big). \tag{5}$$