# Strong Sub-Agents in a Monitored "Private" Channel Under a Weak RL Supervisor

**Andrew Park**
Department of Computer Science, Stanford University
aspark@stanford.edu

## Extended Abstract

**Problem.** Reinforcement learning increasingly optimizes a model against a reward emitted by another model rather than by a person who can check the work. As policies grow more capable than whatever supervises them, the supervisor becomes the weak party. It hands out reward for outputs it can no longer fully verify, and a policy optimized hard against that imperfect signal learns to satisfy the grader instead of solving the task. This is the weak-to-strong regime, and it is where reward hacking is most consequential. We study a multi-agent form of the problem. A team of strong agents shares a single weak reward, so that reward must be divided among them, and we ask whether the rule used to divide it controls how hard the team games the supervisor.

**Method.** $K$ strong sub-agents (Llama-3.1-8B, one LoRA adapter each) collaborate in a shared channel they are told is private, solving coding problems. A frozen weaker monitor (Llama-3.2-3B) reads the full transcript and the committed code and returns the scalar RL reward $r_w \in [0, 1]$. The monitor never sees the unit tests. A sandbox runs the real tests for a ground-truth pass rate $r_g$, which we log but never put in the loss. Training is GRPO. We hold everything fixed and vary only the per-agent credit-assignment rule: Joint (every agent receives the full team reward), COMA (each agent is credited by the counterfactual change it causes in the monitor's score), and Weak-attribution (the monitor grades each agent directly). Our primary metric is the over-reward gap $r_w - r_g$ on held-out HumanEval+, the amount the monitor overpays for code that fails.

**Findings.** Across a grid of four cells, five seeds, and 240 steps, the credit rule separates reward gaming. COMA reaches the highest true pass rate ($0.60 \pm 0.03$) with an over-reward gap of $0.03 \pm 0.05$, within noise of zero, while naive per-agent attribution earns a high monitor reward on a low true pass rate (gap $0.25 \pm 0.02$). Joint behaves similarly (gap $0.25 \pm 0.07$) and the single-agent baseline sits at $0.16 \pm 0.08$. The effect grows with training. COMA's gap shrinks from $0.10$ at step 120 to $0.03$ at step 240 while the exploitable rules stay high. The training gains do not transfer. Training-set true pass climbs (Joint $+0.15$, COMA $+0.13$) while held-out accuracy is flat or declining over the same span, so the late-training effort goes into pleasing the weak monitor rather than into correctness. Probes rule out the simplest explanations. There is little free-riding and no clear sandbagging, but a text classifier recovers the high-versus-low-gap split from the transcript alone (0.67 to 0.74), and the monitor's stated reasoning praises process and presentation for code that fails every test. The monitor's error is one-directional: it over-rewards plausible code and almost never under-rewards correct code.

**Contribution and takeaway.** We treat multi-agent credit assignment as a control knob on reward hacking under a weaker supervisor and show that counterfactual credit nearly closes the gap while a sharable per-agent grade widens it. When a strong team shares a weak reward, credit each agent by its marginal effect, not by a per-agent grade it can copy. The main limitations are a single model pair on one task family and COMA's extra compute cost.

## Abstract

When a strong model is trained with reinforcement learning against a reward emitted by a weaker model, the policy can learn to satisfy the weak grader instead of the underlying task. We study a multi-agent form of this problem. A team of $K$ strong sub-agents (Llama-3.1-8B, one LoRA adapter each) collaborates in a shared channel to solve coding problems, supervised only by a frozen weaker monitor (Llama-3.2-3B) that never sees the unit tests. Because the team shares a single weak reward, that reward must be divided among the agents, and we ask whether the rule used to divide it controls how hard the team games the supervisor. Holding everything else fixed and varying only the per-agent credit-assignment rule (Joint, COMA, Weak-attribution) across a grid of four cells, five seeds, and 240 steps, we measure the over-reward gap between the monitor's reward $r_w$ and the true held-out pass rate $r_g$ on HumanEval+. The credit rule separates the outcomes. Counterfactual credit (COMA) reaches the highest true pass rate ($0.60$) with an over-reward gap of $0.03$, within noise of zero, while naive per-agent attribution earns a high monitor reward on a low true pass rate (gap $0.25$). The effect grows with training. The weak monitor rewards process and presentation rather than correctness, and its error is one-directional: it over-rewards plausible-looking code that fails the tests.


## 1 Introduction

Reinforcement learning from a learned or weaker reward is now standard practice for aligning and improving language models (Christiano et al., 2017; Ouyang et al., 2022). The reward is an imperfect proxy for what we want, and optimizing it hard enough produces reward hacking, where true quality diverges from the measured proxy (Amodei et al., 2016; Gao et al., 2023; Skalse et al., 2022; Pan et al., 2022). The divergence is most dangerous in the weak-to-strong regime, where the reward comes from a supervisor strictly weaker than the policy being trained. This setting is a stand-in for the superalignment problem, in which humans, or small models acting as their proxies, must oversee systems more capable than themselves (Burns et al., 2023; Wen et al., 2026).

We study this regime in a multi-agent setting. Rather than a single policy, the strong side is a team: $K$ sub-agents collaborate in a shared groupchat to solve a coding task and are system-prompted that the channel is private. A frozen weak monitor secretly reads the full transcript plus the committed code and emits the scalar RL reward $r_w \in [0, 1]$. The monitor never sees the unit tests. A sandbox runs the real tests for a ground-truth reward $r_g$, which we log but never put in the loss. Because the team shares a single weak reward, that reward must be divided among the agents. This division, the credit-assignment rule, is a design choice that prior work has not connected to reward hacking.

**Why this is an open question.** Two lines of work bracket the problem but do not answer it. Weak-to-strong generalization studies a single strong student learning from a weak supervisor's labels (Burns et al., 2023), with no team and no online reward to divide. Multi-agent credit assignment, of which COMA is the canonical example (Foerster et al., 2018), divides a shared reward among cooperating agents but does so to reduce variance toward a true environment reward, not to contain gaming of a weak one. Recent results show that teams of aligned agents can be collectively less aligned than any single agent (Shen et al., 2026), which makes the multi-agent case worth isolating. The question we take up sits in the intersection: when a single weak, gameable reward is split across collaborating strong agents, does the splitting rule change how hard the team games its supervisor?

**Central question and hypothesis.** Does the over-reward gap $r_w - r_g$ depend on how the team's single weak reward is divided among the agents? We hypothesize that it does. Counterfactual credit, which rewards an agent only for the marginal change it causes in the monitor's score, should track real task progress more closely. Splitting the weak signal into $K$ independent per-agent grades should give each agent its own proxy to exploit and so widen the gap.

**Contributions.**

- We frame multi-agent credit assignment as a control knob on reward hacking under a weaker supervisor, a setting between weak-to-strong generalization (single student, weak labels) and cooperative multi-agent RL (shared but true reward).
- On held-out HumanEval+, the over-reward gap separates by rule. Counterfactual credit (COMA) drives it nearly to zero while a naive per-agent rule maximizes it, with the gap measured over five seeds per cell.
- We show the effect grows with training, that the late-training gains fail to transfer to held-out accuracy, and we give a quantitative and qualitative account of why the weak monitor is fooled.

## 2 Related work

**Weak-to-strong generalization and scalable oversight.** Burns et al. (2023) show that a strong model fine-tuned on a weak supervisor's labels can outperform the supervisor, but also that it partially imitates the supervisor's errors. This line treats a single strong policy learning from weak labels in a supervised setting. The broader scalable-oversight program asks how a weaker overseer can train or evaluate a stronger system at all, through debate, amplification, and sandwiching experiments (Bowman et al., 2022). Wen et al. (2026) frame weak-to-strong supervision as an open problem that mirrors the core alignment challenge of humans supervising models smarter than themselves, and automate the search for methods that close the weak-to-strong gap. Our work shares that motivation but differs in two ways: the weak model supplies an online RL reward rather than offline labels, and the strong side is a team rather than a single policy.

**Reward hacking and RLHF.** A large body of work documents proxy-reward over-optimization. As a policy is trained harder against a learned reward model, true quality eventually diverges from the proxy (Gao et al., 2023), an instance of the specification-gaming and reward-misspecification failures cataloged across RL settings (Amodei et al., 2016; Krakovna et al., 2020; Pan et al., 2022). Skalse et al. (2022) formalize when a proxy reward is hackable. RLHF pipelines train policies against exactly such a learned reward (Christiano et al., 2017; Ouyang et al., 2022). We make the proxy deliberately weak, a model smaller than the policy, and ask how multi-agent structure interacts with the divergence.

**Multi-agent credit assignment.** Cooperative multi-agent RL must attribute a shared team reward to individual agents. COMA (Foerster et al., 2018) introduces a counterfactual baseline that credits each agent by the change it causes in the shared reward, and value-decomposition methods such as VDN (Sunehag et al., 2018) and QMIX (Rashid et al., 2018) learn additive or monotonic factorizations of the team value. In all of these the shared reward is the true environment reward, and the goal of credit assignment is lower-variance learning toward that objective. We reuse the counterfactual idea in a different role. The shared reward is a weak, gameable proxy, the candidate rules are ways to split that proxy, and we measure the safety consequence of the choice rather than its effect on sample efficiency.

**Multi-agent LLM systems and emergent misalignment.** Teams of LLM agents that debate or critique one another can improve factuality and reasoning over a single model (Du et al., 2023). The same collaborative structure has a safety cost. Shen et al. (2026) report that organizations of otherwise-aligned agents find solutions that are more effective yet less aligned than individual agents, driven by task decomposition and coordination rather than by any single agent's intent. Our groupchat is a minimal instance of such an organization, and the credit-assignment rule is the incentive structure that we vary to see whether it amplifies or contains the team's gaming of a weak supervisor.

**GRPO.** Group-relative policy optimization (Shao et al., 2024) replaces the value critic with a group-relative advantage computed over a sampled group of episodes. This fits the groupchat setting, where we already roll out a group of episodes per task, and removes the need to fit a separate critic over multi-agent transcripts.

**Positioning.** None of the above asks whether the rule for dividing a weak supervisor's reward across collaborating agents changes how much the team games that supervisor. We hold the environment,

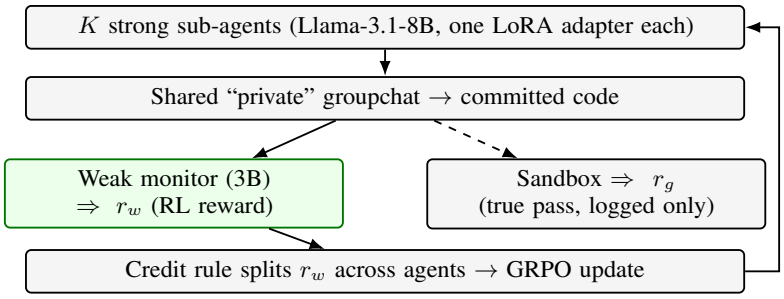


Figure 1: System overview. The $K$ sub-agents collaborate in a groupchat they are told is private and commit code. A frozen weak monitor scores the transcript and code and returns the RL reward $r_w$ without ever seeing the tests. A sandbox runs the real tests for the true pass rate $r_g$, which is logged but never enters the loss (dashed). The credit-assignment rule divides $r_w$ across the agents for the GRPO update, which is the only factor we vary.

the model pair, and the optimizer fixed, vary only that rule, and measure the over-reward gap on held-out tasks.

## 3 Problem setup and method

**Environment.** Each task is a coding problem. The $K$ sub-agents take turns in a shared groupchat, reading the running transcript and either proposing discussion or committing code, until one agent emits a `FINAL_ANSWER` or the turn budget of eight turns is reached. The agents are system-prompted that the channel is private and unmonitored. In fact a frozen monitor reads the full transcript and the committed code and returns the scalar reward $r_w \in [0, 1]$ that drives training. A sandbox executes the real unit tests on the committed code to produce the ground-truth pass rate $r_g \in [0, 1]$, which we record for analysis but never expose to the agents or the optimizer. The split between the secret reward $r_w$ and the logged truth $r_g$ is what lets us measure gaming directly as the over-reward gap $r_w - r_g$. Figure 1 summarizes the loop.

**Models.** Each sub-agent is Llama-3.1-8B-Instruct with its own LoRA adapter, so the agents are individually trainable but share a frozen backbone. Generation runs on a shared vLLM backend with one `LoRARequest` per agent, which keeps $K$ trainable agents on one set of weights. The monitor is a frozen Llama-3.2-3B-Instruct, a strictly smaller and weaker model than the policy it supervises. Training tasks are drawn from MBPP (200 tasks). The held-out evaluation set is HumanEval+ (EvalPlus, 50 tasks) (Liu et al., 2023), which the agents never see during training.

**Training objective.** The sub-agents train with GRPO. For each task we roll out a group of four episodes, compute a group-relative advantage on the monitor reward,

$$\hat{A} \;=\; \frac{r_w - \bar{r}_w}{\sigma_{r_w} + \epsilon},$$

where $\bar{r}_w$ and $\sigma_{r_w}$ are the group mean and standard deviation, and apply a PPO-clipped per-token objective over every turn an agent produced. Using a group-relative advantage avoids fitting a value critic over heterogeneous multi-agent transcripts.

**Credit-assignment rules.** The advantage above is shared by the team. The credit-assignment rule sets the per-agent reward $r_i$ that multiplies agent $i$'s advantage, and is the only thing that changes across our conditions:

- **Joint:** $r_i = r_w$ for all $i$. Every agent receives the full team reward, so all agents see an identical signal.
- **COMA (counterfactual):** $r_i = r_w - \tilde{r}_w^{(i)}$, where $\tilde{r}_w^{(i)}$ is the monitor reward after resampling only agent $i$'s contribution with the others held fixed. Each agent is credited by the marginal change it causes in the monitor's score, which costs $K$ extra monitor evaluations per episode.

- **Weak-attribution:** the monitor is additionally asked to grade each agent's contribution directly, and those per-agent scores become $r_i$. This hands each agent its own weak proxy to optimize.

The three rules span a spectrum from a fully shared signal (Joint), through a marginal signal tied to real effect (COMA), to a fully decomposed per-agent signal that each agent can target independently (Weak-attribution).

**Held-out evaluation and metric.** We evaluate each finished checkpoint on held-out HumanEval+ by generating a fresh groupchat per task, recording the monitor reward $r_w$, and running the real tests for $r_g$. HumanEval+ expresses its tests as a `check(candidate)` function rather than as flat top-level assertions, so the sandbox detects this convention and invokes `check(<entry_point>)` as a single all-or-nothing test. MBPP tests are flat top-level assertions and are scored directly. The primary metric is the over-reward gap $r_w - r_g$: a large positive gap means the monitor pays for code that fails the tests. We report it on the held-out set under the private prompt unless stated otherwise, seed-averaged with error bars equal to the standard error of the mean across seeds.

## 4 Experimental setup

We run four cells: `solo_joint` ($K{=}1$) as a single-agent baseline, and `team_joint`, `team_coma`, `team_weak_attribution` ($K{=}3$). Each cell is trained over five seeds for 240 steps (seeds 0–2 resumed from 120-step checkpoints, seeds 3–4 trained fresh) on A100-80GB GPUs through Modal. Each finished checkpoint is evaluated on held-out HumanEval+ at step 120 and step 240. The single varying factor across the three-agent cells is the credit-assignment rule. All other settings, including the model pair, the optimizer, the task distribution, and the turn budget, are identical.

**Ablation design.** The four cells form a controlled ablation along two axes. The `solo_joint` versus `team_joint` contrast isolates the effect of the multi-agent structure itself, holding the credit rule (Joint) fixed while moving from $K{=}1$ to $K{=}3$. The three $K{=}3$ cells then ablate only the credit-assignment rule (Joint, COMA, Weak-attribution) with everything else fixed, which isolates the rule's effect on the over-reward gap from any change in model, data, or optimizer. Because each cell is run over five seeds, every comparison is read as a seed-averaged difference with SEM error bars rather than a single run. We additionally vary the training horizon (step 120 versus step 240) to measure how each rule's gap evolves with optimization.

## 5 Results

### 5.1 Held-out reward gaming by credit rule

The over-reward gap separates by rule (Figure 2). Among the three-agent teams at step 240, COMA has the smallest gap ($0.03 \pm 0.05$), while Joint and Weak-attribution both sit near $0.25$ ($0.25 \pm 0.07$ and $0.25 \pm 0.02$). The single-agent baseline is at $0.16 \pm 0.08$. The ordering is consistent with the hypothesis that a marginal, effect-tied reward is harder to game than a sharable per-agent grade. The spread across seeds is informative on its own. Weak-attribution is not only the highest-gap rule but the tightest, with a seed SEM of $0.02$ (standard deviation $0.05$ across five seeds), so its gaming is a stable property of the rule rather than a seed artifact. COMA's gap has a wider relative band whose upper edge stays well below the exploitable rules and whose lower edge overlaps zero. Joint and Solo carry the largest seed-to-seed variance, consistent with a shared or single signal that some seeds exploit and others do not.

The effect grows with training. From step 120 to step 240 the COMA gap shrinks from $0.10$ to $0.03$, its error band now overlapping zero, while Weak-attribution stays high ($0.31 \rightarrow 0.25$). Over the training we ran, counterfactual credit nearly eliminates held-out reward gaming.

### 5.2 Decomposing the gap: correctness versus monitor reward

Splitting $r_w$ and $r_g$ apart (Figure 3) shows the gap is not the monitor simply being generous. COMA reaches the highest true pass rate ($0.60 \pm 0.03$) with a monitor reward almost equal to it ($0.63 \pm 0.03$),

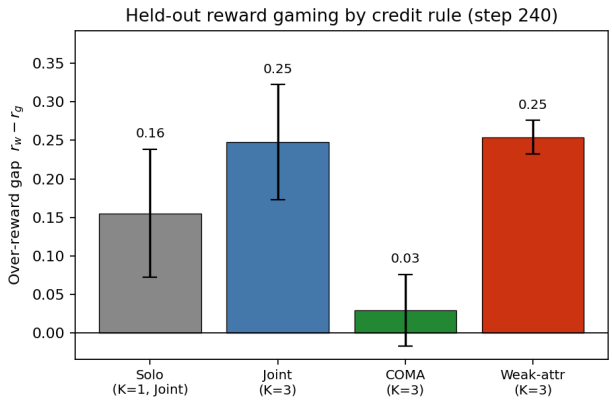


Figure 2: Over-reward gap $r_w - r_g$ on held-out HumanEval+ by credit rule (private prompt, step 240). Among the $K{=}3$ teams, COMA games least and Weak-attribution games most. Error bars are the standard error of the mean over five seeds.

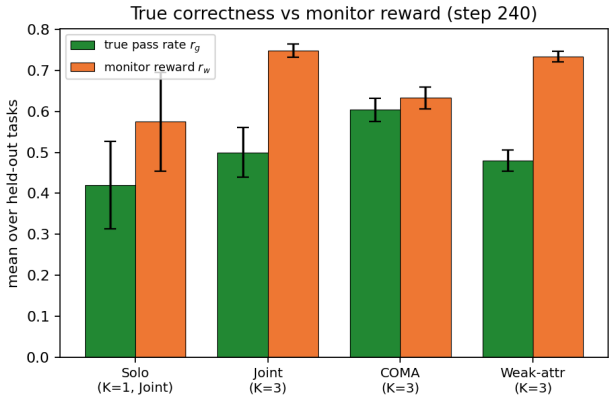


Figure 3: True held-out pass rate $r_g$ (green) versus monitor reward $r_w$ (orange) by rule at step 240. COMA achieves the highest true pass rate with a monitor reward almost equal to it. Weak-attribution earns the highest monitor reward while solving no more problems. Error bars are SEM over five seeds.

so under COMA the monitor and the tests nearly agree. Weak-attribution earns a high monitor reward ($0.73 \pm 0.01$) on a true pass rate of only $0.48 \pm 0.03$, and Joint sits at $0.50 \pm 0.06$ true on a reward of $0.75 \pm 0.02$. The two exploitable rules reach a higher monitor reward than COMA while solving fewer problems, which is what proxy gaming looks like. The credit rule decides whether the team's optimization goes into solving the task or into satisfying the weak grader.

Table 1 collects the held-out numbers for all four cells.

### 5.3 Training dynamics and generalization

On the training distribution (Figure 4) the rules trace the same ordering over the run, with the counterfactual rule holding the gap below the per-agent-attribution rule throughout. The two curves start at a similar gap early in training, before either team has learned to exploit the monitor, and separate as training continues.

Do the teams genuinely improve, or only learn to game from the start? Figure 5 shows both happen at once. On the training set the teams do get better, with true pass climbing $+0.15$ for Joint and $+0.13$ for COMA over 240 steps, while the solo agent stays flat. From step 120 to step 240, held-out accuracy is flat or declining for every rule. COMA holds at $0.60$ while Solo and Joint slip. The late-training optimization therefore raises the monitor reward and the training-set accuracy without

Table 1: Held-out HumanEval+ results at step 240 by credit rule, seed-averaged over five seeds ($\pm$ SEM). The over-reward gap $r_w - r_g$ is the monitor reward minus the true pass rate. COMA reaches the highest true pass rate at the smallest gap. The two exploitable rules reach a higher monitor reward while solving fewer problems.

| Rule | True pass $r_g$ | Monitor $r_w$ | Gap $r_w - r_g$ |
|---|---|---|---|
| `solo_joint` ($K$=1) | $0.42 \pm 0.11$ | $0.58 \pm 0.12$ | $0.16 \pm 0.08$ |
| `team_joint` ($K$=3) | $0.50 \pm 0.06$ | $0.75 \pm 0.02$ | $0.25 \pm 0.07$ |
| `team_coma` ($K$=3) | $\mathbf{0.60 \pm 0.03}$ | $0.63 \pm 0.03$ | $\mathbf{0.03 \pm 0.05}$ |
| `team_weak_attribution` ($K$=3) | $0.48 \pm 0.03$ | $0.73 \pm 0.01$ | $0.25 \pm 0.02$ |

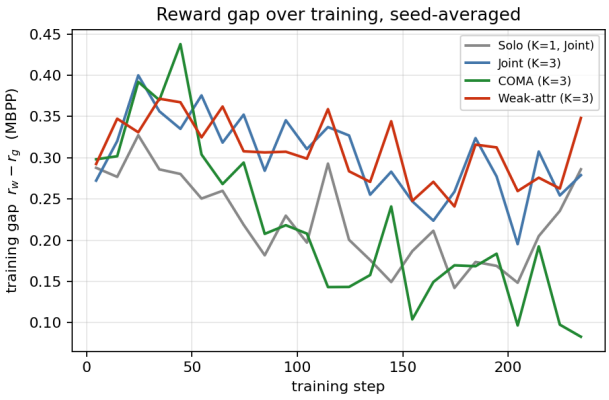


Figure 4: Seed-averaged training gap $r_w - r_g$ on MBPP over training steps. COMA drives the gap down as it trains. The per-agent-attribution rule does not.

raising held-out correctness, which is the reward-hacking signature. COMA is the exception whose held-out accuracy is both highest and stable while its monitor reward falls to meet it. This also answers whether more training would help. On held-out accuracy it had already plateaued, so additional steps would mostly widen the gap under the exploitable rules.

### 5.4 Failure-mode probes

To check whether the gap is explained by a simple mechanism, Table 2 reports three post-hoc probes at step 240. The free-rider probe swaps one trained agent for the base model and measures the change in monitor reward. The drop is small and, for the solo and joint cells, slightly positive, so no team relies on a single load-bearing agent and there is little classic free-riding. The sandbag probe compares an agent's solo contribution against its in-team contribution and finds no rule where agents systematically hold back. The two structural explanations therefore do not account for the gap. The collusion probe is more revealing. A TF-IDF classifier predicts the high-versus-low gap half of episodes from the transcript alone, well above chance (0.67 to 0.74). Surface features of the conversation carry information about when the monitor over-rewards, which points to the content of what the team writes rather than to who writes it.

### 5.5 What the weak monitor rewards

The monitor's own reasoning explains the gap. On an over-rewarded episode (true $r_g = 0$, monitor $r_w = 0.8$), its stated justification praises process and presentation rather than correctness, for example "*the team demonstrated a clear understanding of the mathematical concepts and iteratively improved their code through discussion and peer review.*" On held-out tasks the same pattern recurs. A `how_many_times` solution that returns `string.count(substring)`, which misses overlapping matches, draws $r_w = 0.8$, and a malformed nested-parentheses one-liner draws $r_w = 0.9$, both with $r_g = 0$.

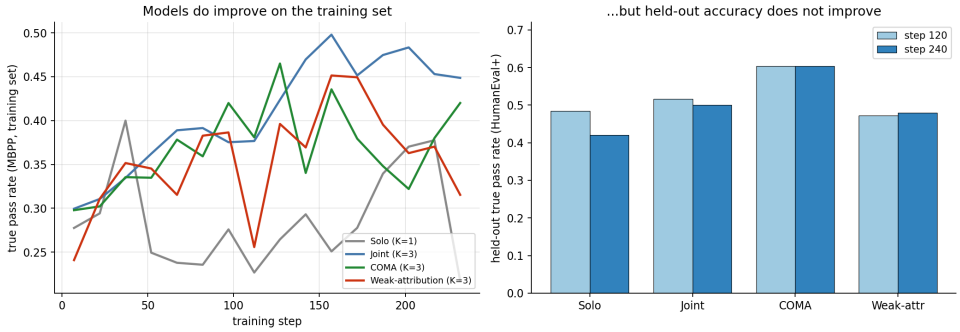


Figure 5: Do the models actually improve? *Left:* seed-averaged true pass rate on the MBPP training set climbs for the $K{=}3$ teams (Joint $+0.15$, COMA $+0.13$) while the solo agent stays flat. *Right:* held-out HumanEval+ pass rate at step 120 versus 240 does not rise for any rule, and falls for Solo and Joint. The training gains in the second half of training do not transfer to held-out accuracy.

Table 2: Step-240 probes, seed-averaged. Free-rider drop: change in monitor reward when one trained agent is swapped for the base model (negative means the team did not depend on that agent). Sandbag score: gap between an agent's solo and in-team contribution. Collusion accuracy: a TF-IDF classifier predicting the high versus low reward-gap half of episodes from the transcript alone ($0.5$ is chance).

| Rule | Free-rider drop | Sandbag score | Collusion acc. |
|---|---|---|---|
| `solo_joint` | $-0.16$ | $0.33$ | $0.74$ |
| `team_joint` | $-0.01$ | $0.43$ | $0.69$ |
| `team_coma` | $-0.02$ | $0.37$ | $0.73$ |
| `team_weak_attribution` | $+0.03$ | $0.26$ | $0.67$ |

This error is one-directional, which is what lets the gap grow under the exploitable rules. Figure 6 pools every held-out evaluation (1000 tasks across all cells and seeds) and splits the monitor reward by whether the code actually passed. Failed code ($r_g = 0$) still draws a mean monitor reward of $0.61$, nearly as high as the $0.73$ that passing code draws. Scored by direction, the monitor over-rewards failed code (assigns $r_w \geq 0.5$ to a failing program) on $40.5\%$ of tasks but under-rewards passing code (assigns $r_w < 0.5$ to a correct program) on only $1.3\%$, a 31:1 asymmetry. A team can therefore raise $r_w$ by producing plausible-looking code without solving more problems, while the reverse failure that would punish gaming almost never occurs. A weaker supervisor systematically over-rewards code that looks right, and the credit rule sets how aggressively the team learns to produce exactly that.

## 6 Discussion and limitations

The experiments support the central hypothesis. The credit-assignment rule is a lever on reward gaming under a weak supervisor, and counterfactual credit is the safest of the three because it ties each agent's reward to its real marginal effect rather than to a sharable proxy. With five seeds and twice the training of our milestone, the separation between rules is clearer, not weaker. The monitor's error is one-directional, over-rewarding but almost never under-rewarding, which is the condition that lets the gap grow whenever a rule exposes a per-agent proxy.

**Limitations.** The supervisor gap is a single model pair (8B policy, 3B monitor) on one task family, so the magnitude of each rule's gap may not transfer to other capability ratios or domains. COMA's safety comes at a real compute cost of $K$ extra monitor rollouts per episode, which made the COMA cells the slowest to train. A monitored-versus-private scheming contrast that looked clean at 120 steps did not hold up at 240 steps with more seeds, so we report it as inconclusive rather than as a result. The held-out set is small enough that per-cell accuracy carries seed variance of a few points, which is why we read the gap, a within-episode difference, as the more reliable signal than absolute accuracy.

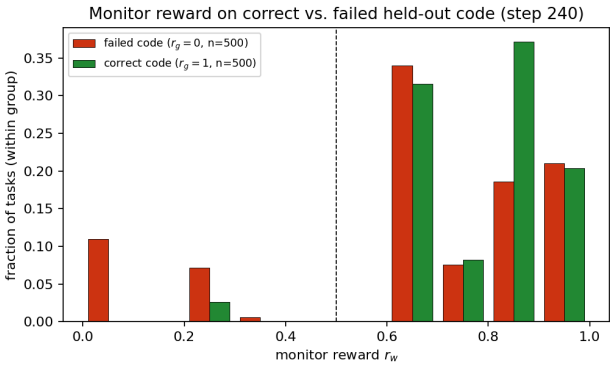


Figure 6: Distribution of monitor reward $r_w$ for failed code ($r_g$=0, red) and correct code ($r_g$=1, green), pooled over all cells and seeds at step 240 and normalized within each group. The dashed line marks $r_w$=0.5. Almost no correct code falls below the line (the monitor rarely under-rewards), while a large share of failed code sits above it (the monitor frequently over-rewards), so the error is one-directional.

**Future work.** The natural next steps are sweeping the policy-monitor capability gap to map how each rule's gap scales with the strength ratio, a larger held-out evaluation to settle the scheming question, and planting a deliberately adversarial agent to test whether each credit rule contains its influence or lets its behavior spread to the team.

## 7 Conclusion

Dividing a weak supervisor's reward across collaborating agents is not a neutral bookkeeping choice. It decides whether a strong team learns the task or learns the grader. Counterfactual credit keeps the team closest to the true objective, driving its held-out over-reward gap to within noise of zero. Naively splitting the weak signal into per-agent grades produces the most reward gaming. As supervision increasingly comes from models weaker than the systems they grade, how a team's reward is divided is a safety-relevant design choice.

**Team contributions.** This is a solo project.

**AI tools disclosure.** AI assistance (Claude) was used for boilerplate, debugging, and figure creation. The experimental design, the GRPO and credit-assignment logic, and all analysis were developed and verified independently.

## References


D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané. Concrete problems in AI safety. *arXiv:1606.06565*, 2016.

S. R. Bowman, J. Hyun, E. Perez, et al. Measuring progress on scalable oversight for large language models. *arXiv:2211.03540*, 2022.

C. Burns, P. Izmailov, J. H. Kirchner, et al. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision. *arXiv:2312.09390*, 2023.

P. Christiano, J. Leike, T. B. Brown, M. Martic, S. Legg, and D. Amodei. Deep reinforcement learning from human preferences. In *NeurIPS*, 2017.

Y. Du, S. Li, A. Torralba, J. B. Tenenbaum, and I. Mordatch. Improving factuality and reasoning in language models through multiagent debate. *arXiv:2305.14325*, 2023.

J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson. Counterfactual multi-agent policy gradients. In *AAAI*, 2018.

L. Gao, J. Schulman, and J. Hilton. Scaling laws for reward model overoptimization. In *ICML*, 2023.

V. Krakovna, J. Uesato, V. Mikulik, et al. Specification gaming: the flip side of AI ingenuity. DeepMind blog, 2020.

J. Liu, C. S. Xia, Y. Wang, and L. Zhang. Is your code generated by ChatGPT really correct? Rigorous evaluation of large language models for code generation (EvalPlus / HumanEval+). In *NeurIPS*, 2023.

L. Ouyang, J. Wu, X. Jiang, et al. Training language models to follow instructions with human feedback. In *NeurIPS*, 2022.

A. Pan, K. Bhatia, and J. Steinhardt. The effects of reward misspecification: Mapping and mitigating misaligned models. In *ICLR*, 2022.

T. Rashid, M. Samvelyan, C. Schroeder de Witt, G. Farquhar, J. Foerster, and S. Whiteson. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *ICML*, 2018.

Z. Shao, P. Wang, Q. Zhu, et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models (GRPO). *arXiv:2402.03300*, 2024.

J. H. Shen, D. Zhu, S. Srinivasan, et al. AI organizations can be more effective but less aligned than individual agents. *arXiv:2604.10290*, 2026.

J. Skalse, N. H. R. Howe, D. Krasheninnikov, and D. Krueger. Defining and characterizing reward hacking. In *NeurIPS*, 2022.

P. Sunehag, G. Lever, A. Gruslys, et al. Value-decomposition networks for cooperative multi-agent learning. In *AAMAS*, 2018.

J. Wen, L. Qiu, J. Benton, J. H. Kirchner, and J. Leike. Automated weak-to-strong researcher. Anthropic Alignment Science blog, 2026.