

RLAC: RL with Adversarial Critic for Free-Form Generation

2026-03-06

Paper Info

- **Authors:** Mian Wu, Gavin Zhang, Sewon Min, Sergey Levine, Aviral Kumar (UC Berkeley)
- **Link:** [arXiv](#)
- **Read date:** 2026-03-06

Summary

Scaling RL post-training to free-form generation (biography writing, code generation) is hard because evaluation requires checking many rubrics/criteria, and exhaustive verification is computationally prohibitive. RLAC frames this as an adversarial game: a critic LLM dynamically proposes the rubric the generator is most likely to fail, and an external validator checks only that rubric. The generator is rewarded when the critic can't find a failure; the critic is rewarded when it correctly identifies one. Both are trained jointly via DPO with binary validator feedback. This achieves strong results on factual text generation and code generation while using 5–97% fewer verification calls than exhaustive approaches, and avoids reward hacking that plagues static critics.

Key Ideas

1. **Exhaustive verification doesn't scale** — Free-form tasks have many evaluation criteria; checking all of them per sample is computationally prohibitive (e.g. 7.86M test executions for code).
2. **Static critics get exploited** — Fixed reward models or frozen critics are vulnerable to reward hacking as the generator improves. The generator learns to exploit blind spots.

3. **Adversarial min-max game** — The critic proposes the worst-case rubric (most likely failure mode); the validator checks it. Generator maximizes performance, critic maximizes failure detection. Both co-evolve.
4. **Prompt-specific, verifiable supervision** — Unlike generic reward models, the critic generates task-specific failure hypotheses that are verified against ground truth, maintaining signal fidelity.
5. **Massive efficiency gains** — By checking only the most informative rubric per sample, RLAC reduces verification calls by $5.7\times$ (text) to 97.5% (code) compared to exhaustive methods.

Method

- **Adversarial formulation:** Min-max game where generator maximizes expected reward and critic identifies worst-case rubrics. Reformulated as: generator wins if critic can't find a failing rubric; critic wins if it correctly pinpoints a failure.
- **Training:** Both generator and critic trained via DPO using binary feedback from external validator. Iterative process: sample outputs $\rightarrow$ critic proposes criteria $\rightarrow$ validator checks $\rightarrow$ update both policies.
- **External validators:** Ground-truth checkers (e.g. fact-checking APIs for biography, test case execution for code). These provide reliable but expensive binary signals.
- **Critic adaptation:** The critic co-evolves with the generator, continuously finding new failure modes as old ones are fixed. This prevents the reward hacking plateau seen with static critics.

Results

- **Factual text generation (Biography):**
 - Qwen3-8B: FactScore 0.889, surpassing FactTune-FS (0.867)
 - $5.7\times$ fewer verification calls than exhaustive checking
- **Code generation:**
 - Qwen2.5-Coder-7B: 56.6 average score across benchmarks
 - Outperformed AceCoder methods using only 9% of training data
 - 97.5% fewer test case executions (192k vs. 7.86M)
- **Ablations:**
 - Noisy validation destabilizes training below baseline
 - Static critics plateau; adversarial critics maintain detection effectiveness
 - Dynamic adaptation is key to preventing reward hacking

Strengths

- Elegant adversarial formulation that naturally focuses compute on the most informative evaluation criteria.
- Massive efficiency gains make RL practical for free-form tasks where exhaustive verification was infeasible.
- Co-evolving critic prevents reward hacking—a real and common failure mode with static reward models.
- Demonstrated across qualitatively different domains (factual text, code) with different verification structures.
- Clean experimental design with informative ablations (noisy validator, static vs. adversarial critic).

Weaknesses / Limitations

- Requires an external validator for each rubric—still needs ground-truth checking infrastructure, just less of it.
- DPO-based training for both generator and critic; unclear how it interacts with online RL methods like GRPO.
- Evaluated only on 7–8B models; scaling behavior to larger models unknown.
- The critic must be expressive enough to generate meaningful failure hypotheses; may struggle on tasks with subtle or hard-to-articulate failure modes.
- Adversarial dynamics can be unstable; paper reports stabilization but long-term training dynamics not deeply analyzed.

Relevance to My Work

- **Self-distilling reward models connection** — RLAC’s adversarial critic is conceptually related to the self-judge idea: both use an LLM to generate evaluation signals. The key difference is RLAC uses a separate critic model + external validator, while self-distilling reward models use the same model with asymmetric information. Could combine: use the policy itself as the adversarial critic, grounded by environment feedback.
- **Reward hacking mitigation** — The adversarial critic that co-evolves with the generator directly addresses reward hacking. Relevant to the reward hack detection angle of the self-distilling reward models idea.
- **Efficiency for RLVR** — The “check only the most informative criterion” principle could apply to RLVR settings with multiple verification signals (e.g. multiple test cases in code).
- **Dense rewards from sparse verification** — Similar motivation to the self-judge work: turn expensive/sparse verification into more informative training signals.

Key Takeaways

1. Free-form generation RL can be made practical by adversarially selecting which rubrics to verify, rather than checking all of them.
2. Co-evolving the critic with the generator is essential—static critics get exploited and lead to reward hacking plateaus.
3. The adversarial framework achieves 5–97% verification cost reduction while matching or exceeding exhaustive verification quality.
4. External ground-truth validators remain necessary but are used far more efficiently as targeted checks on critic-proposed failure modes.
5. Same research group (Aviral Kumar) as POPE—both address making RL post-training more practical, from different angles (exploration vs. evaluation).