SOFT-SVERL: SELF-VERIFIED REINFORCEMENT LEARNING WITH SOFT REWARDS

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement Learning from Verifiable Rewards (RLVR) has improved language models on tasks with automatic correctness checks, but many important tasks remain only partially verifiable. We introduce SOFT-RLVR, a framework that decomposes prompts into checklists of atomic requirements, verifies candidate responses item by item with an LLM, and trains on the resulting soft reward. These checklist rewards provide dense partial-credit supervision, but can also reward incomplete responses; we formalize this tradeoff and characterize when checklist verification yields a more reliable RL signal than holistic verification. We further introduce SOFT-SVERL, a self-verifying variant of SOFT-RLVR, and identify conditions under which self-verification improves RL outcomes. In controlled instruction-following experiments with rule-based ground truth, checklist-based SOFT-RLVR improves IFEVAL by up to 11.1 points using only learned verifier rewards, with verifier and checklist quality both affecting downstream gains.

1 INTRODUCTION

Reinforcement Learning from Verifiable Rewards (RLVR) can substantially improve language models when reliable reward signals are available, as demonstrated by the rapid progress in mathematics (Shao et al., 2024; DeepSeek-AI, 2025) and code generation (DeepSeek-AI, 2025; Wang et al., 2025), where symbolic verifiers and unit tests can check correctness with perfect accuracy.

However, many important applications do not admit exact verification. Complex instruction following is a central example: prompts often combine surface constraints with semantic requirements, such as preserving, transforming, or excluding information, that are difficult to grade with rule-based checkers or a single reference answer (Zhou et al., 2023). Responses may satisfy some requirements but not others, making binary rewards too sparse, while holistic judgments are noisy and obscure which requirements were met (Bai et al., 2022; Lee et al., 2023; Gunjal et al., 2025). This raises a central question: *how can we train language models with RL when correctness is partial, noisy, and difficult to specify?*

We address this challenge with SOFT-RLVR, a framework for reinforcement learning with learned checklist-based rewards. SOFT-RLVR exploits the approximately decomposable structure of instruction-following tasks by converting prompts into checklists of criteria, evaluating each item with a learned (LLM)verifier, and rewarding the fraction of satisfied items. This turns sparse pass/fail supervision into a graded signal that captures partial progress while remaining tied to explicit criteria.

Because learned verifiers are imperfect, checklist rewards introduce a tradeoff: averaging item-level judgments can reduce reward variance when verifier errors are weakly correlated, but partial credit can bias training toward incomplete responses. We formalize this tradeoff and derive conditions under which checklist verification yields lower RL-gradient error than holistic verification.

We further introduce SOFT-SVERL, a self-verifying variant of SOFT-RLVR where the policy model also serves as the verifier. This removes the need for a separate online reward model and can reduce the inference footprint of RL training. We identify conditions under which self-verification improves RL outcomes and stabilize training with verifier co-training and an anti-collapse penalty that reduces reward inflation.

We evaluate our approach in a controlled instruction-following setting based on IFEVAL-style verifiable constraints (Zhou et al., 2023). This setting provides rule-based ground-truth verification

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.

for evaluation and analysis, while training uses learned checklist-based rewards rather than the rule-based verifier. Our experiments show that checklist-based soft verification improves instruction following and quantify the impacts of verifier and checklist quality on downstream RL outcomes.

Our work makes the following key contributions: (i) SOFT-RLVR, which turns prompt-specific checklist criteria into dense learned rewards; (ii) an analysis of when checklist rewards improve the RL signal despite partial-credit bias; and (iii) SOFT-SVERL, a self-verifying variant stabilized by verifier co-training, replay, and anti-collapse regularization. On IFEVAL-style instruction following, learned checklist rewards improve Command R7B by up to 11.1 points.

2 PRELIMINARIES

Let $\pi_\theta(y \mid x)$ denote a language model policy parameterized by θ, where x is a prompt and y is a sampled completion. Given a reward function R that scores completions, the goal of policy optimization is to maximize expected reward under the policy. In reference-based settings, this reward may depend on a reference answer y^*, giving the objective

$$\arg\max_\theta \; \mathbb{E}_{(x,y^*)\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)}\big[R(y, y^*)\big], \tag{1}$$

where $\mathcal{D} = \{(x_i, y_i^*)\}$ is a dataset of prompt-reference pairs. In the settings we study, however, exact reference answers may be unavailable or insufficient, so the central question is how to construct a useful reward directly from the prompt and candidate response.

Reinforcement learning with verifiable rewards (RLVR). When outputs admit exact verification, as in mathematics or code, rewards can be computed by deterministic checkers, e.g., $R_{\text{correct}}(y, y^*) = \mathbf{1}[y \equiv y^*]$. These reliable rewards have enabled strong RLVR results in reasoning and code-generation settings (DeepSeek-AI, 2025; Shao et al., 2024; Kimi Team et al., 2025; Bercovich et al., 2025), but many instruction-following tasks do not admit a single exact check.

Holistic learned verifiers. For hard-to-verify tasks, an LLM verifier can provide a reward $R_{\text{ver}}(x, y; \phi) = \rho_\phi(x, y)$, where $\rho_\phi(x, y) \in \{0, 1\}$ is a pass/fail judgment for the whole response. This holistic decision can be noisy or too coarse when prompts contain multiple requirements; we therefore use checklist-based verification to judge individual requirements separately.

Checklist-based verification with learned verifiers. We instead decompose each prompt into a checklist of atomic criteria $C(x) = \{c_1, \ldots, c_K\}$. For each response y and criterion c_k, a learned verifier judges whether the criterion is satisfied: $\rho_\phi(x, y, c_k) \in \{0, 1\}$. The resulting soft reward is the fraction of satisfied criteria,

$$R_{\text{soft}}(x, y) = \frac{1}{K} \sum_{k=1}^{K} \rho_\phi(x, y, c_k). \tag{2}$$

This reward gives partial credit to responses that satisfy only some requirements, preserving learning signal when strict pass/fail rewards would be zero. The same relaxation can also bias training toward partially correct outputs, a tradeoff analyzed in Section 3.1.

In our experiments, checklists are generated offline by an LLM from the instruction x as ordered yes/no criteria, parsed into checklist items, and kept fixed during RL training; the prompt template and examples are in Appendix A.9.3 and Appendix A.9.4. The verifier is implemented as a generative LLM whose outputs are parsed into `Yes`/`No` decisions. When using multiple verifier samples, we compute $\hat{p}_k(x, y) = \frac{1}{J} \sum_{j=1}^{J} \text{parse}(z_{kj})$, with $z_{kj} \sim \pi_\phi(\cdot \mid x, y, c_k)$, and set $\rho_\phi(x, y, c_k) = \mathbf{1}[\hat{p}_k(x, y) \geq \tau]$. Unless otherwise stated, ρ_ϕ denotes this parsed and aggregated checklist-item judgment.

3 REINFORCEMENT LEARNING WITH CHECKLIST-BASED SOFT REWARDS

Given a prompt x, a checklist $C(x) = \{c_1, \ldots, c_K\}$, and a learned verifier ρ_ϕ, SOFT-RLVR trains the policy $\pi_\theta(y \mid x)$ to maximize the expected checklist reward:

$$\arg\max_\theta \mathbb{E}_{x\sim D_X,\, y\sim\pi_\theta(\cdot|x)} \left[r(x, y, C(x), \rho_\phi)\right]. \tag{3}$$

Here r is computed from item-level verifier judgments. In the simplest case, $r = R_{\text{soft}}$, the fraction of checklist items judged as satisfied. This gives the policy partial credit for satisfying some requirements even when the full response is not a strict success, so responses that would both receive zero reward under a pass/fail verifier can receive different rewards depending on how much of the instruction they satisfy.

This formulation is analogous to RLVR for code, where a generated program is evaluated against multiple unit tests. The key difference is that checklist items are evaluated by a learned LLM verifier rather than executable tests. This extends RLVR-style training beyond exactly verifiable domains, but introduces two coupled sources of error: item-level verifier noise and partial-credit bias. We next analyze when the benefit of decomposing the reward outweighs this additional bias.

3.1 Bias–variance tradeoff of checklist verification

Checklist verification changes the error profile of the reward. A holistic verifier produces one noisy judgment for the entire response, whereas a checklist verifier produces K item-level judgments and averages them. Averaging can reduce variance, especially when item-level errors are weakly correlated. However, the resulting soft reward relaxes the strict success target: a response that fails one requirement may still receive positive reward. Thus, checklist verification improves the training signal only when the variance reduction from decomposition is large enough to offset the bias introduced by partial credit.

Setup. Let $\mathbf{Y}^*(x, y) = [Y_1^*, \ldots, Y_K^*]^\top \in \{0, 1\}^K$ denote the latent checklist vector, where $Y_k^* = 1$ means that response y satisfies checklist item c_k. We define the strict success target and partial-credit target as

$$S^*(x, y) := \prod_{k=1}^{K} Y_k^*, \ \bar{S}^*(x, y) := \frac{1}{K} \sum_{k=1}^{K} Y_k^*. \tag{4}$$

The relaxation gap $\Delta^*(x, y) := \bar{S}^*(x, y) - S^*(x, y) \geq 0$ measures how much partial credit rewards incomplete responses beyond the strict success target. We take strict success as the ideal target because our controlled evaluation uses rule-based pass/fail instruction satisfaction; partial credit is a training relaxation that may improve optimization but can introduce bias relative to this target.

Let J denote the output of a holistic verifier, J_k the output of the checklist verifier for item k, and $\bar{J} := K^{-1} \sum_{k=1}^{K} J_k$ the averaged checklist judgment. We compare the ideal policy-gradient update g^*, the holistic-verifier update g_{single}, and the checklist-verifier update g_{chk}; their formal definitions are in Appendix A.10. Let p, q denote the true positive and true negative rates of the holistic verifier with respect to S^*, and let p', q' denote the corresponding item-level rates with respect to Y_k^*.

The following theorem gives sufficient conditions under which the checklist update has lower MSE relative to the ideal update than the holistic-verifier update. Appendix A.10 gives the bias–variance decomposition used in the proof: checklist averaging controls variance under conditional independence, while the theorem's conditions control the bias introduced by partial credit.

Theorem 1 (Sufficient conditions for gradient MSE improvement). *Suppose the holistic verifier satisfies $\mathbb{E}[J \mid S^*] = (1 - q) + \alpha S^*$ with $\alpha := p + q - 1$, each checklist verifier satisfies $\mathbb{E}[J_k \mid Y_k^*] = (1 - q') + \alpha' Y_k^*$ with $\alpha' := p' + q' - 1$, and $J_1, \ldots, J_K$ are conditionally independent given (x, y). Then*

$$\mathbb{E}\big[\|g_{\text{chk}} - g^*\|_2^2 \mid x, y\big] \leq \mathbb{E}\big[\|g_{\text{single}} - g^*\|_2^2 \mid x, y\big] \tag{5}$$

whenever either

$$S^*(x, y) = 1 \qquad \textit{and} \qquad p' \geq p, \tag{6}$$

or

$$S^*(x, y) = 0, \quad \textit{and} \quad \Big(1 - q' + \alpha' \Delta^*(x, y)\Big)^2 + \frac{1}{4K} \leq 1 - q. \tag{7}$$

The proof is in Appendix A.10.1, where the two cases are derived by separately bounding the bias and variance terms.

Takeaway. Checklist verification is most useful when decomposition makes the reward less noisy without substantially changing what the reward is trying to optimize. Larger checklists can reduce variance, but they cannot by themselves fix a poor decomposition or a verifier that systematically approves incomplete responses. Thus, checklist quality and verifier specificity are central: the checklist items must capture meaningful requirements, and the verifier must reject failures often enough that partial credit does not become reward inflation.

This analysis leads to two empirical predictions that we test in Section 6.3. First, checklist verification should help most when holistic verifier judgments are noisy. Second, checklist quality should matter: poorly decomposed criteria can increase the relaxation gap or make item-level judgments less reliable, reducing the benefit of decomposition. Appendix A.10 gives additional corollaries that make the checklist-size thresholds explicit and show that larger K alone cannot guarantee improvement when partial-credit bias is too large.

4 SELF-VERIFIED SOFT RLVR

The learned verifier ρ_ϕ introduced in Section 3 need not be separate from the generator. In self-verification, we set $\phi = \theta$, so the same model generates responses and judges checklist satisfaction. This removes the need for a separate online reward model and applies when no stronger external judge is available, but it also makes the reward function part of the policy update.

4.1 WHY SELF-VERIFICATION IS NON-TRIVIAL

The central failure mode is *always-yes collapse*: the verifier says Yes for almost every response–checklist pair, so $\hat{p}_k(x, y) \approx 1$ even when the requirement is not satisfied. Low-quality generations then receive inflated rewards, erasing the gradient signal needed for improvement. This collapse arises from two couplings. First, the verifier shares the generator's priors: responses that are easy to produce may also be the responses on which the verifier is overconfident, so calibration must hold on current-policy outputs, not only held-out data. Second, the shared objective no longer distinguishes better generation from a more permissive verifier. When $\phi = \theta$, reward can increase either because the policy improves or because the verifier assigns higher scores to responses the policy already produces.

Figure 5 in Appendix A.4 shows this pathology empirically: naive shared-parameter training drops IFEVAL by roughly 19 points over 400 steps, below the initial policy.

4.2 STABILIZING SELF-VERIFICATION

Stable self-verification requires addressing both failure modes above. Correlated priors make the verifier unreliable on the model-generated responses it must score, while gradient coupling allows the shared update to increase measured reward by making the verifier more permissive. SOFT-SVERL first reduces reward noise by aggregating multiple verifier samples for each response–checklist pair, then adds two stabilizing signals to the shared objective: verifier co-training, which anchors the verifier on labeled examples, and a partition-style Yes-rate penalty, which directly discourages reward inflation on failing items. Figure 4 in Appendix A.1 provides a schematic overview of SOFT-SVERL.

Repeated verification. For each response–checklist pair, we sample J verifier traces and compute the empirical Yes-rate $\hat{p}_k(x, y)$. This gives the generator a more stable item-level reward and provides confidence scores for replay labeling.

Verifier co-training. The shared model is also trained as a verifier on *gold* prompt–response–criterion triples (x, y, c) with labels $\ell \in \{0, 1\}$, plus *replay* examples sampled from π_θ and pseudo-labeled at the item level. Gold data anchors checklist satisfaction outside the current policy distribution; replay data moves this supervision onto reward-scored examples. Replay labels are conservative: items with $\hat{p}_k(x, y) \geq \tau_+$ become positives, items with $\hat{p}_k(x, y) \leq \tau_-$ become negatives, and intermediate items are ignored. The coefficient λ_v controls this term.

Partition-style penalty. Co-training alone does not close the reward-inflation channel, so SOFT-SVERL penalizes Yes-rates on mixed-vote items from failed responses: $\hat{p}_k(x, y) > 0$ and $s(x, y) < 1$. These permissive Yes traces are not used as hard negative labels; instead, the penalty directly opposes

updates that make the verifier more permissive. Appendix A.11 derives this term as a finite-sample approximation to the negative policy-sample term from the KL-regularized optimal policy. The coefficient λ_p controls its strength, and the penalty reuses verifier traces collected during reward scoring.

The resulting shared objective is

$$\mathcal{J}(\theta) = \mathcal{J}_{\text{gen}}(\theta) + \lambda_v \mathcal{J}_{\text{ver}}(\theta) - \lambda_p \mathcal{J}_{\text{part}}(\theta), \tag{8}$$

where the three terms train the generator, co-train the verifier, and penalize Yes-rates on the failing partition, respectively. Algorithm 1 in Appendix A.3 summarizes the resulting shared-parameter training loop.

5 Experiment Setup

Training data. We train on the RL split of the Llama-Nemotron-Post-Training Dataset (Bercovich et al., 2025), which contains 56,339 instruction-following prompts with explicit IFEval-style verifiable constraints (Zhou et al., 2023). Each constraint can be checked by a deterministic rule-based verifier, which we use as ground truth for evaluation and analysis. During learned-verifier training runs, however, rewards are computed from generated checklists and LLM verifier judgments rather than from the rule-based oracle. This controlled setting lets us measure learned checklist rewards against exact labels and test the predictions of Section 3.1.

Training details. All experiments start from Command R7B, further adapted with the math-and-code SFT data for a stronger baseline. We generate prompt checklists offline using DeepSeek-R1-05-28; the checklist-generation prompt is given in Appendix A.9.3. For external-verifier experiments, we use GPT-OSS-20B and GPT-OSS-120B as the main LLM verifiers, with additional verifier choices included in the verifier-quality ablations. We optimize the policy with GRPO (Shao et al., 2024); implementation details and hyperparameters are listed in Appendix A.2.

Evaluation and baselines. We evaluate instruction following on IFEval (Zhou et al., 2023), which contains 541 English prompts spanning 25 verifiable instruction types. As a sanity check for capability regression, we also evaluate on MATH500 (Hendrycks et al., 2021), AIME 2024, and AIME 2025. We compare against the initial Command R7B policy before RL training and, where computationally feasible, an oracle RLVR reference that uses the deterministic IFEval-style verifier on the same training data. This oracle removes both checklist-generation error and verifier noise, and serves as an upper reference for learned checklist rewards.

6 Results

We evaluate whether learned checklist rewards improve instruction following, how gains depend on verifier and checklist quality, and whether self-verification can be stabilized when the generator and verifier share parameters.

Table 1 summarizes the main external-verifier results. We train Command R7B with DeepSeek R1 checklists and LLM-based soft verification, then evaluate on IFEval and three out-of-domain math benchmarks. Unless mentioned otherwise, we use a single verifier pass ($J = 1$) and do not train the verifier.

Checklist-based soft verification substantially improves instruction following. IFEval increases from 73.89 to 84.20 with the GPT-OSS-20B verifier and to 85.00 with the GPT-OSS-120B verifier, corresponding to gains of +10.3 and +11.1 points. These gains use generated checklists and imperfect LLM verifier rewards, without access during training to the rule-based verifier or to the true constraint annotations associated with each prompt.

Despite training only on instruction-following prompts, RL training also improves math benchmark performance: MATH500 improves by 2.0–2.6 points and AIME 2024 improves by 3.7–4.3 points across verifier scales. AIME 2025 gains are modest, around one point, and should be interpreted cautiously given the small benchmark size of 30 problems. The absence of math regression is notable in itself, as targeted post-training can degrade out-of-domain capabilities; we provide an SFT comparison in Appendix A.8. The oracle rule-based verifier remains an upper reference for this controlled setting, but relies on deterministic ground-truth checklists unavailable in practice.

Table 1: **Results on instruction following and math reasoning.** The baseline is Command R7B before RL training. SOFT-RLVR rows use DeepSeek R1 checklists and GPT-OSS verifier rewards; OSS-20B/OSS-120B indicate the verifier. The rule-based verifier is an oracle reference. Bold marks the best learned-verifier result. Scores are percentages at step 500.

Reward	IFEVAL	MATH500	AIME24	AIME25
No RL	73.89	87.60	35.67	27.33
OSS-20B	84.20	89.60	39.33	**28.67**
OSS-120B	**85.00**	**90.20**	**40.00**	28.50
Oracle	88.92	87.80	36.83	28.00

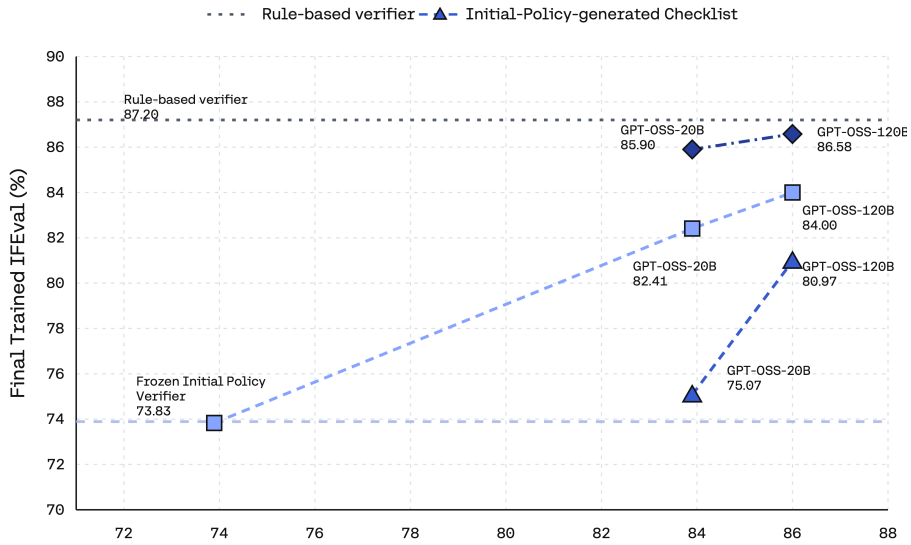


Figure 1: **Verifier and checklist quality on IFEVAL.** Verifier quality and checklist quality both affect downstream IFEVAL performance. Curves compare checklist sources, and horizontal reference lines mark the pre-training baseline and deterministic IFEVAL rule-based verifier reference.

Increasing the verifier scale from GPT-OSS-20B to GPT-OSS-120B gives consistent but modest additional gains: +0.8 on IFEVAL, +0.6 on MATH500, and +0.7 on AIME 2024, while AIME 2025 remains essentially unchanged. This suggests that the checklist-based reward formulation is robust to moderate variation in verifier quality, which we analyze more directly next.

6.1 DISENTANGLING VERIFIER AND CHECKLIST QUALITY

SOFT-RLVR depends on both the checklist used to decompose the prompt and the verifier used to judge each item. To disentangle these effects, we keep the RL pipeline fixed while varying checklist source and verifier model, including Initial Policy checklists, DeepSeek R1 checklists, metadata constraints as gold checklists, and an oracle rule-based verifier. All runs in this ablation are evaluated at step 200 to reduce compute.

Figure 1 shows that both checklist and verifier quality affect downstream RL performance. Holding the DeepSeek R1 checklist source fixed, the Initial Policy verifier yields essentially no improvement over baseline: the trained model reaches 73.83% IFEVAL compared with the 73.89% starting point. Replacing this weak verifier with GPT-OSS-20B raises IFEVAL to 82.41%, and GPT-OSS-120B raises it further to 84.00%. This comparison isolates the effect of verifier quality under a fixed generated checklist.

Holding the verifier fixed isolates checklist quality. DeepSeek R1 checklists outperform Initial Policy checklists at both verifier scales: 82.41% vs. 75.07% with GPT-OSS-20B, and 84.00% vs. 80.97%

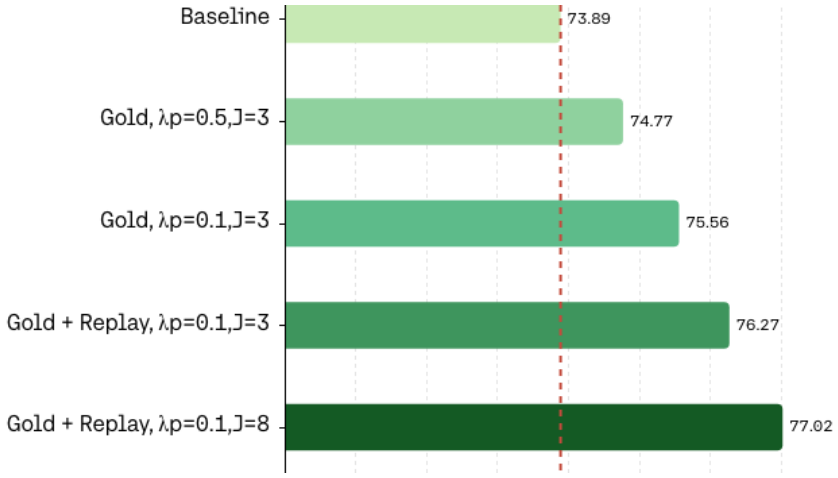


Figure 2: **Self-verification ablations on IFEVAL.** Self-verification benefits from the full suite of stabilization components. Setting the partition weight too high hampers performance, while adding replay data and multiple verifier votes further improves performance.

with GPT-OSS-120B. Using true training metadata constraints as checklists improves performance further, reaching 85.90% with GPT-OSS-20B and 86.58% with GPT-OSS-120B. These metadata checklists are not generally available, but they quantify headroom from better checklist generation. The remaining gap to the 87.20% rule-based oracle estimates residual verifier imperfection after checklist-generation error is removed. These results are consistent with prior work showing that the quality of evaluation criteria can affect downstream optimization performance (Gunjal et al., 2025).

6.2 SELF-VERIFICATION

We next evaluate SOFT-SVERL, where the generator and verifier share parameters. This removes the need for a separate online reward model, but introduces the collapse mode described in Section 4: the shared verifier can become more permissive on the generator's own samples. All variants in this section are trained for 100 steps.

Figure 2 shows that stabilization is essential for effective self-verification. For replay admission, we use thresholds $\tau_+ = 0.75$ and $\tau_- = 0.375$. With gold verifier-improvement examples, $J = 3$ verifier votes, and $\lambda_p = 0.5$, SOFT-SVERL reaches 74.77% IFEVAL; reducing the partition weight to $\lambda_p = 0.1$ raises this to 75.56%. This suggests that too large a partition penalty can overcorrect the verifier, suppressing positive judgments needed for useful reward.

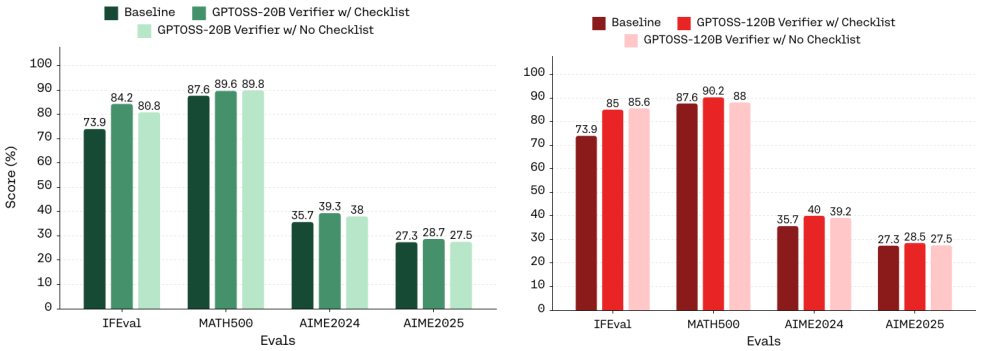
Adding on-policy replay gives a larger improvement. With the same $J = 3$ vote count, including tagged examples from the current run increases IFEVAL to 76.27%. This supports the role of $\mathcal{D}_{\text{Replay}}$: gold completions anchor checklist satisfaction, while on-policy examples expose the verifier to the current generator's actual failure modes. Increasing the vote count from $J = 3$ to $J = 8$ further improves the gold-plus-replay setting from 76.27% to 77.02%, consistent with reduced sampling noise in the empirical vote fraction $\hat{p}_{ik}$. We do not interpret this limited range as a monotonic scaling law. Overall, self-verification benefits from verifier co-training, on-policy replay, multiple verifier votes, and a tuned anti-collapse penalty.

6.3 CHECKLIST VS. HOLISTIC VERIFICATION

The analysis in Section 3.1 predicts that checklist verification should help when decomposition reduces verifier noise without introducing too much partial-credit bias. In particular, checklist rewards are expected to be most useful when item-level judgments are more sensitive than holistic judgments and when item-level specificity is not substantially worse. We test this prediction by comparing checklist-based verification against holistic verification, where the verifier evaluates the full response against the prompt in a single judgment.

Table 2: **Estimated holistic and checklist verifier quality on initial-policy completions.** Checklist estimates use IFEVAL constraints as item-level ground truth. "Bias condition" reports the fraction of samples for which the partial-credit bias is small enough for checklist verification to satisfy the sufficient condition in Proposition 4.

Verifier	p	p'	q	q'	$p'-p$	$q'-q$	α	α'	Bias cond. (%)
GPT-OSS-20B	0.477	0.873	0.662	0.717	+0.396	+0.055	0.139	0.590	81.5
GPT-OSS-120B	0.552	0.860	0.709	0.712	+0.308	+0.003	0.261	0.572	81.5



(a) GPT-OSS-20B verifier *(b) GPT-OSS-120B verifier*

Figure 3: **Checklist-based vs. holistic (non-checklist) verification for two verifier scales.** With the noisier GPT-OSS-20B verifier, checklist decomposition improves IFEVAL substantially. With the stronger GPT-OSS-120B verifier, the IFEVAL gap closes, though checklist-based training retains an edge on math benchmarks.

To estimate the verifier quantities in the theory, we generate completions from the initial policy and grade the same completions with each GPT-OSS verifier. We use the IFEVAL verifier to obtain holistic ground-truth labels for estimating single-judgment sensitivity and specificity, p and q. For checklist estimates, we use the IFEVAL constraints as checklist items and compare each LLM verifier's item-level decisions against the corresponding constraint labels, yielding p' and q'.

Table 2 shows that both checklist verifiers have much higher item-level sensitivity than holistic sensitivity ($p' > p$), while item-level specificity is similar to or slightly higher than holistic specificity ($q' \geq q$). This matches the theory's main prediction: checklist decomposition is most helpful when it improves sensitivity and reduces variance without substantially weakening specificity.

Figure 3 compares checklist-based and holistic training at two verifier scales. With GPT-OSS-20B, checklist training outperforms holistic training by 3.4 points on IFEVAL (84.20% vs. 80.80%), consistent with decomposition helping most when verifier judgments are noisy. With GPT-OSS-120B, holistic training matches checklist training on IFEVAL (85.58% vs. 85.00%), suggesting that stronger holistic judgments reduce the variance-reduction benefit of decomposition.

This result does not contradict the low estimated holistic sensitivity in Table 2, since those estimates measure agreement with rule-based IFEVAL labels. Rule-based checks capture operational constraints, while holistic LLM verifiers may also judge semantic adequacy. For example, consider the prompt: "List 2 science fiction books with their authors. End the response with Happy Reading!" The response "$*$ 1984 by George Orwell; $*$ Brave New World by Aldous Huxley; Happy Reading!" may satisfy the rule-based checks for two listed items and the required closing phrase, but a holistic verifier may reject it because these works are often classified more specifically as political fiction rather than as science fiction. Such disagreements count as false negatives relative to the rule-based label and lower the estimated holistic sensitivity, even though the holistic reward can still provide a useful training signal.

Checklist training also performs better on math benchmarks at both verifier scales. With GPT-OSS-120B, it reaches 90.20% on MATH500 versus 88.00% for holistic verification, and 40.00% on AIME

2024 versus 39.17%. One possible explanation is that graded partial credit provides more informative reward shaping than a binary holistic judgment, especially for tasks where partial progress matters.

7 Related Work

RL beyond exact verifiers. Recent work extends reinforcement learning beyond exactly verifiable domains by replacing formal checkers with weaker supervision signals, such as answer likelihood, reference-conditioned reasoning quality, or self-generated rewards (Zhou et al., 2025; Yu et al., 2025b; Chen et al., 2024; Tang et al., 2025; Zuo et al., 2025; Xu et al., 2025; Zhao et al., 2025). These methods reduce reliance on rule-based verifiers, but many still assume gold final answers, reference outcomes, or answer-extractable reasoning tasks. In contrast, Soft-RLVR targets instruction-following settings where no single answer string is available and correctness must be assessed through prompt-specific criteria.

Rubrics, checklists, and learned judges. Prior work has used rubrics, checklists, or learned judges to evaluate open-ended model outputs (Gunjal et al., 2025; Jia et al., 2025; Jayalath et al., 2025). The closest checklist-style prior is Gunjal et al. (2025), which constructs rubrics for non-verifiable tasks and scores outputs with an external judge. Our focus is complementary: we use checklist decomposition to construct dense, interpretable RL rewards and analyze when these rewards provide a reliable training signal despite verifier noise and partial-credit bias.

Self-verifying RL. Several methods study self-generated or co-evolving reward signals (Aggarwal et al., 2026; Lu et al., 2025). Soft-SVeRL differs by using checklist-conditioned soft rewards in a shared generator-verifier model, where self-verification can collapse by becoming overly permissive. We address this with repeated verifier voting, verifier co-training from gold and replay examples, and explicit anti-collapse regularization. Appendix A.7 gives a more detailed comparison of supervision assumptions and verification settings.

8 Conclusion

We showed that learned LLM verifiers producing graded, checklist-based rewards can serve as effective training rewards for instruction-following RL. Training Command R7B with checklist-based Soft-RLVR improves IFEval by up to 11.1 points without degrading out-of-distribution math performance. Our results show that checklist decomposition is most useful when verifier judgments are noisy, consistent with our MSE analysis, and that both verifier quality and checklist quality affect downstream gains. We also introduced Soft-SVeRL, a self-verifying variant that requires verifier co-training, replay, and anti-collapse regularization to avoid reward inflation.

Overall, checklist-based soft verification offers a path toward broadening RLVR beyond exactly verifiable domains by turning partially verifiable instructions into dense, actionable reward signals. Future work should extend this approach to more open-ended settings, larger model scales, and stronger checklist generation methods.

Limitations

Checklist decomposition assumes that prompt requirements can be expressed as discrete, atomic pass-fail items. This assumption is natural for instruction-following tasks with explicit constraints, but may not hold for more subjective or holistic tasks such as open-ended writing, dialogue quality, helpfulness, or style. Extending soft verification to such settings may require richer rubrics, non-binary criteria, or aggregation methods that account for interactions among criteria.

Our experiments are limited to one base model scale and one main training domain. Although we include out-of-distribution math benchmarks as a regression check, broader evaluation across model families, model sizes, longer training horizons, and less controlled task distributions is needed to establish generality.

For Soft-SVeRL, we assume access to gold-labeled verifier examples that anchor the shared verifier and reduce reward inflation. This supervision may not always be available, and future work should study weaker anchors, synthetic verifier data, or fully unsupervised stabilization methods.

AI USE STATEMENT

Generative AI tools were used as writing aids to improve the clarity and presentation of the paper and to help improve the quality of the plots. All reported values and plots were reviewed and verified by the authors to guard against hallucinations and other inaccuracies. The authors take full responsibility for the content of the paper.

REFERENCES

Pranjal Aggarwal, Marjan Ghazvininejad, Seungone Kim, Ilia Kulikov, Jack Lanchantin, Xian Li, Tianjian Li, Bo Liu, Graham Neubig, Anaelia Ovalle, Swarnadeep Saha, Sainbayar Sukhbaatar, Sean Welleck, Jason Weston, Chenxi Whitehouse, Adina Williams, Jing Xu, Ping Yu, Weizhe Yuan, Jingyu Zhang, and Wenting Zhao. Reasoning over mathematical objects: On-policy reward modeling and test time aggregation, 2026. URL https://arxiv.org/abs/2603.18886.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional AI: Harmlessness from AI feedback, 2022. URL https://arxiv.org/abs/2212.08073.

Akhiad Bercovich, Itay Levy, Izik Golan, Mohammad Dabbah, Ran El-Yaniv, Omri Puny, Ido Galil, Zach Moshe, Tomer Ronen, Najeeb Nabwani, et al. Llama-nemotron: Efficient reasoning models, 2025. URL https://arxiv.org/abs/2505.00949.

Haolin Chen, Yihao Feng, Zuxin Liu, Weiran Yao, Akshara Prabhakar, Shelby Heinecke, Ricky Ho, Phil Mui, Silvio Savarese, Caiming Xiong, and Huan Wang. Language models are hidden reasoners: Unlocking latent reasoning capabilities via self-rewarding, 2024. URL https://arxiv.org/abs/2411.04282.

DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean Hendryx. Rubrics as rewards: Reinforcement learning beyond verifiable domains, 2025. URL https://arxiv.org/abs/2507.17746.

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, 2021. URL https://arxiv.org/abs/2103.03874.

Dulhan Jayalath, Shashwat Goel, Thomas Foster, Parag Jain, Suchin Gururangan, Cheng Zhang, Anirudh Goyal, and Alan Schelten. Compute as teacher: Turning inference compute into reference-free supervision, 2025. URL https://arxiv.org/abs/2509.14234.

Ruipeng Jia, Yunyi Yang, Yongbo Gai, Kai Luo, Shihao Huang, Jianhe Lin, Xiaoxi Jiang, and Guanjun Jiang. Writing-zero: Bridge the gap between non-verifiable tasks and verifiable rewards, 2025. URL https://arxiv.org/abs/2506.00103.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, et al. Kimi k1.5: Scaling reinforcement learning with LLMs, 2025. URL https://arxiv.org/abs/2501.12599.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, and Sushant Prakash. RLAIF: Scaling reinforcement learning from human feedback with AI feedback, 2023. URL https://arxiv.org/abs/2309.00267.

Songshuo Lu, Hua Wang, Zhi Chen, and Yaohua Tang. Urpo: A unified reward & policy optimization framework for large language models, 2025. URL https://arxiv.org/abs/2507.17515.

John Schulman and Thinking Machines Lab. LoRA without regret. *Thinking Machines Lab: Connectionism*, 2025. doi: 10.64434/tml.20250929. https://thinkingmachines.ai/blog/lora/.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Hao Peng, Lei Zhang, Y. K. Li, Zhen Huang, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Yunhao Tang, Sid Wang, Lovish Madaan, and Rémi Munos. Beyond verifiable rewards: Scaling reinforcement learning for language models to unverifiable data, 2025. URL https://arxiv.org/abs/2503.19618.

Yinjie Wang, Ling Yang, Ye Tian, Ke Shen, and Mengdi Wang. Co-evolving llm coder and unit tester via reinforcement learning, 2025. URL https://arxiv.org/abs/2506.03136.

Fangzhi Xu, Hang Yan, Chang Ma, Haiteng Zhao, Qiushi Sun, Kanzhi Cheng, Junxian He, Jun Liu, and Zhiyong Wu. Genius: A generalizable and purely unsupervised self-training framework for advanced reasoning. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13153–13167, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.644. URL https://aclanthology.org/2025.acl-long.644/.

Joanna Yoo, Kuba Perlin, Siddhartha Rao Kamalakara, and João G. M. Araújo. Scalable Training of Language Models using JAX pjit and TPUv4, 2022. URL https://arxiv.org/abs/2204.06514.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale, 2025a. URL https://arxiv.org/abs/2503.14476.

Tianyu Yu, Bo Ji, Shouli Wang, Shu Yao, Zefan Wang, Ganqu Cui, Lifan Yuan, Ning Ding, Yuan Yao, Zhiyuan Liu, Maosong Sun, and Tat-Seng Chua. RLPR: Extrapolating RLVR to general domains without verifiers, 2025b. URL https://arxiv.org/abs/2506.18254.

Xuandong Zhao, Zhewei Kang, Aosong Feng, Sergey Levine, and Dawn Song. Learning to reason without external rewards, 2025. URL https://arxiv.org/abs/2505.19590.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models, 2023. URL https://arxiv.org/abs/2311.07911.

Xiangxin Zhou, Zichen Liu, Anya Sims, Haonan Wang, Tianyu Pang, Chongxuan Li, Liang Wang, Min Lin, and Chao Du. Reinforcing general reasoning without verifiers, 2025. URL https://arxiv.org/abs/2505.21493.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang, Xinwei Long, Ermo Hua, Biqing Qi, Youbang Sun, Zhiyuan Ma, Lifan Yuan, Ning Ding, and Bowen Zhou. Ttrl: Test-time reinforcement learning, 2025. URL https://arxiv.org/abs/2504.16084.

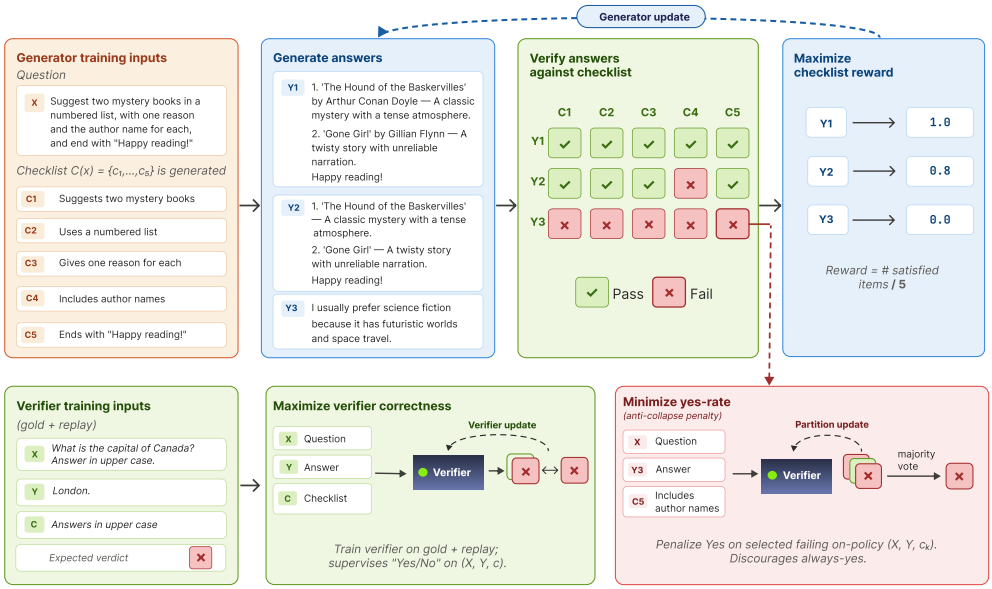


Figure 4: **Overview of SOFT-SVERL.** A prompt is decomposed into checklist criteria, candidate answers are scored item by item, and the resulting soft rewards update the generator. Because the generator also serves as the verifier, SOFT-SVERL adds verifier-side stabilization: labeled examples improve item-level judgments, while failing checklist contexts discourage inflated Yes predictions.

A APPENDIX

A.1 SOFT-SVERL OVERVIEW

Figure 4 illustrates the SOFT-SVERL training loop. A prompt is decomposed into checklist criteria, candidate answers are scored item by item, and the resulting soft rewards update the generator. Because the generator also serves as the verifier, SOFT-SVERL adds verifier-side stabilization: labeled examples improve item-level judgments, while failing checklist contexts discourage inflated Yes predictions.

A.2 TRAINING HYPERPARAMETERS

All runs use Group Relative Policy Optimization (Shao et al., 2024) on a FAX-based (Yoo et al., 2022) asynchronous RL training framework. We sweep learning rates and find that 5×10^{-6} provides the best balance of stability and performance across our runs. Unless stated otherwise, we train for 500 steps with a batch size of 256, group size 8, and maximum sequence length of 16,384 tokens. We use asymmetric clipping bounds from DAPO (Yu et al., 2025a), with $\epsilon_{\text{low}} = 0.1$ and $\epsilon_{\text{high}} = 0.2$, and no KL regularization. For external verifiers, we use a single pass ($J = 1$) to generate the decision. For verifier models that support adaptive thinking, we use the highest thinking-budget setting, up to 32k tokens.

A.3 SOFT-SVERL TRAINING ALGORITHM

Algorithm 1 summarizes the SOFT-SVERL shared generator-verifier training loop, including checklist reward computation, verifier replay labeling, verifier co-training, and the partition-style anti-collapse penalty.

Algorithm 1 Soft-SVeRL

Inputs: Input distribution $\mathcal{D}_X$; gold verifier buffer $\mathcal{D}_{\text{Gold}}$; replay verifier buffer $\mathcal{D}_{\text{Replay}}$; checklist generator $\mathcal{C}(\cdot)$; candidate count G; verifier vote count J; thresholds τ_+, τ_-; partial-credit scale β; weights λ_v, λ_p
Model: Shared model θ used for generation π_θ and verification ρ_θ

for $t = 1, \ldots, T$ **do**
  Sample $x \sim \mathcal{D}_X$ and checklist $\mathcal{C}(x) = \{c_k\}_{k=1}^{K}$
  Sample candidates $y_1, \ldots, y_G \sim \pi_\theta(\cdot \mid x)$
  for all $(i, k, j) \in [G] \times [K] \times [J]$ **do**
    Sample $z_{ikj} \sim \rho_\theta(\cdot \mid x, y_i, c_k)$ and set $d_{ikj} \leftarrow \text{parse}(z_{ikj})$
  end for
  Compute $\hat{p}_{ik} \leftarrow \frac{1}{J} \sum_{j=1}^{J} d_{ikj}$ for all (i, k)
  Set $\hat{\ell}_{ik} \leftarrow \mathbf{1}[\hat{p}_{ik} \geq \tau_+]$ for all (i, k) ▷ pass label
  for $i = 1, \ldots, G$ **do**
    Set $s_i \leftarrow \frac{1}{K} \sum_{k=1}^{K} \hat{\ell}_{ik}$ ▷ soft checklist score
    Set $R_i \leftarrow \mathbf{1}[s_i = 1] + \beta s_i \, \mathbf{1}[s_i < 1]$ ▷ generator reward
  end for
  for all $(i, k) \in [G] \times [K]$ **do**
    if $\hat{p}_{ik} \geq \tau_+$ **then**
      Insert $(x, y_i, c_k, \ell = 1)$ into $\mathcal{D}_{\text{Replay}}$
    else if $\hat{p}_{ik} \leq \tau_-$ **then**
      Insert $(x, y_i, c_k, \ell = 0)$ into $\mathcal{D}_{\text{Replay}}$
    end if
  end for
  Select subset $\mathcal{S}(x) \subseteq [G] \times [K]$ where $\mathcal{S}(x) = \{(i, k) : \hat{p}_{ik} > 0 \text{ and } s_i < 1\}$
  Sample verifier tuples $(x, y, c, \ell) \sim \mathcal{D}_{\text{Gold}} \cup \mathcal{D}_{\text{Replay}}$
  for $g = 1, \ldots, G$ **do**
    Sample $z_g \sim \rho_\theta(\cdot \mid x, y, c)$
    Set $R_g^{\text{ver}} \leftarrow \mathbf{1}[\text{parse}(z_g) = \ell]$ ▷ checklist verifier reward
  end for
  for all $(i, k) \in \mathcal{S}(x)$ **do**
    for $j = 1, \ldots, J$ **do**
      Set $R_{ikj}^{\text{part}} \leftarrow d_{ikj}$ ▷ Yes-rate penalty
    end for
  end for
  Update θ to maximize $\mathcal{J}(\theta) = \mathcal{J}_{\text{gen}}(\theta) + \lambda_v \mathcal{J}_{\text{ver}}(\theta) - \lambda_p \mathcal{J}_{\text{part}}(\theta)$ ▷ shared update
end for

A.4 Naive Self-Verification Training

We evaluate a naive self-verification baseline in which the generator and verifier share parameters and are optimized jointly using the verifier's own checklist reward, without verifier co-training, replay supervision, or an anti-collapse penalty. This setting isolates the failure mode discussed in Section 4: because the same model both generates responses and assigns rewards, training can increase measured reward by making the verifier more permissive rather than by improving task performance. Figure 5 shows the empirical signature of this pathology.

A.5 Dataset and Evaluation Details

The RL split of the Llama-Nemotron-Post-Training Dataset (Bercovich et al., 2025) contains 56,339 instruction-following prompts. Each prompt includes one or more explicit constraints drawn from the 25 verifiable instruction types defined by IFEval (Zhou et al., 2023), such as formatting requirements, keyword inclusion or exclusion, length limits, and structural specifications. Each constraint in this dataset can be checked by a deterministic rule-based verifier.

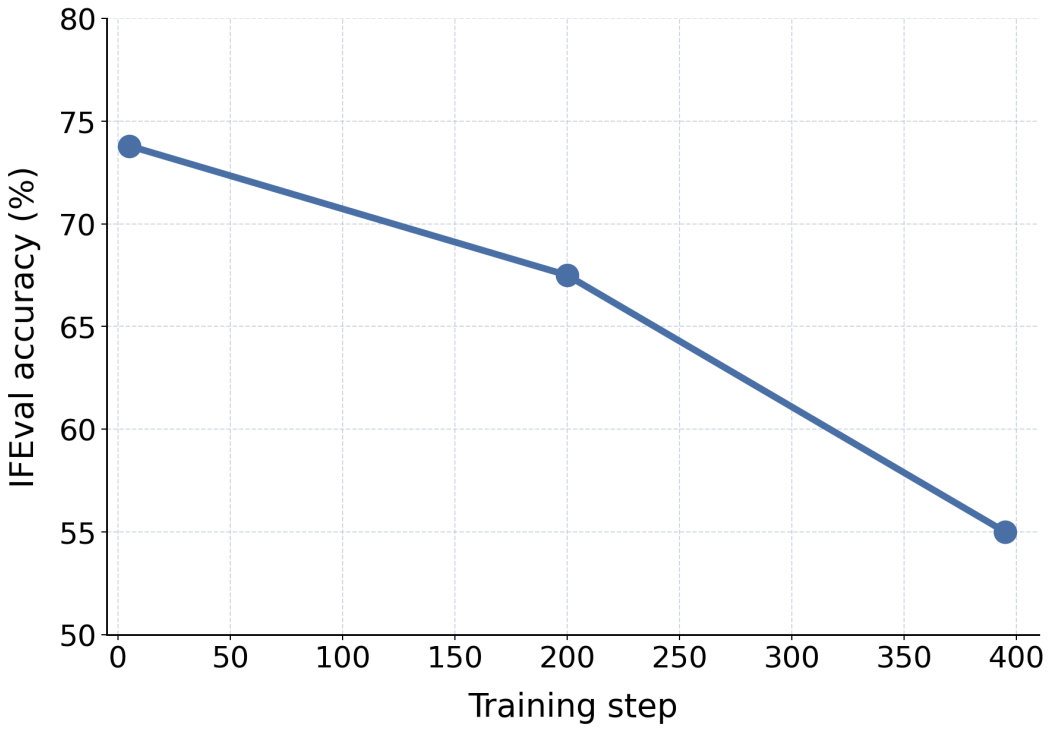


Figure 5: **Naive self-verification training.** The generator and verifier share parameters and are optimized jointly using the verifier's own reward signal. IFEVAL drops from 73.9% at initialization to 55.1% by step 400, consistent with always-yes collapse: measured reward improves while task accuracy degrades.

For evaluation, IFEVAL covers 25 verifiable instruction types, including length, keyword, formatting, casing, punctuation, and structural constraints. The released English benchmark contains 541 prompts with one or more instructions per prompt.

A.6 Additional Experimental Details

The Llama-Nemotron-Post-Training RL split contains prompts with one or more explicit constraints drawn from the 25 verifiable instruction types defined by IFEVAL, including formatting requirements, keyword inclusion or exclusion, length limits, casing, punctuation, and structural specifications. Because the constraint universe matches IFEVAL, training with the oracle verifier uses the same family of rule-based checks as evaluation, eliminating verifier train–test mismatch.

This controlled setup separates reward-learning effects from evaluation ambiguity. The deterministic verifier lets us evaluate learned checklist rewards against exact rule-based labels, measure verifier agreement with ground-truth constraints, and test the empirical predictions of the MSE analysis in Section 3.1.

For external-verifier experiments, GPT-OSS-20B and GPT-OSS-120B provide a practical tradeoff between instruction-following quality and verification throughput, which is important because reward computation requires scoring multiple checklist items per sampled response. The verifier prompt is shown in Appendix A.9.2. For verifier models that support adaptive thinking, we use the maximum thinking-budget setting.

We optimize the policy with Group Relative Policy Optimization (GRPO) (Shao et al., 2024) using an asynchronous RL training framework. Following DAPO (Yu et al., 2025a), we use asymmetric clipping bounds. For external verifiers, we use a single verifier pass ($J = 1$). We sweep learning

rates and report the best-performing run for each setting, with all other hyperparameters listed in Appendix A.2.

A.7 Additional Related Work

RL beyond exact verifiers. Recent work extends reinforcement learning beyond standard verifiable domains by replacing formal verification with weaker supervision signals. One line of work removes explicit rule-based verifiers but still relies on gold final answers or reference outcomes, for example by rewarding answer likelihood or reference-conditioned reasoning quality (Zhou et al., 2025; Yu et al., 2025b; Chen et al., 2024; Tang et al., 2025). These methods are important contrasts, but they do not address our target setting because they still assume access to a correct target answer or reference outcome. In contrast, SOFT-RLVR targets tasks where no single answer string is available and correctness must instead be assessed through prompt-specific criteria.

Self-generated and label-light rewards. A second line of work reduces labeled supervision through intrinsic or self-generated signals, including TTRL, Genius, and Learning to Reason without External Rewards (Zuo et al., 2025; Xu et al., 2025; Zhao et al., 2025). TTRL is especially relevant because it trains from unlabeled prompts and self-generated rewards rather than gold answers. However, its reward is based on majority-vote pseudo-labels on answer-extractable reasoning benchmarks. Current evidence does not show that the same mechanism extends to open-ended or checklist-style evaluation, where responses can satisfy some requirements but not others. These methods are therefore closer to SOFT-SVERL in being label-light than in addressing the same verification problem.

Rubrics, checklists, and learned judges. Prior work has used rubrics, checklists, or learned judges to evaluate open-ended model outputs (Gunjal et al., 2025; Jia et al., 2025; Jayalath et al., 2025). Gunjal et al. (2025) is the closest checklist-style prior: it constructs rubrics for non-verifiable tasks and scores outputs with an external judge. Our focus is different. We use checklist decomposition to construct dense, interpretable RL rewards and analyze when these rewards provide a reliable training signal. This extends the principle behind RLVR beyond domains with hand-written verifiers: the model is still evaluated against explicit criteria, but the checker is an LLM verifier rather than a symbolic rule or unit test.

More broadly, rubric-based approaches are closest to what we call a rubric-bound setting, where the main challenge is specifying useful criteria for open-ended tasks. SOFT-RLVR instead targets a verifier-bound setting: even when evaluation criteria can be written as checklist items, the central problem is how to obtain stable and useful reward from an imperfect verifier. This distinction becomes especially important for SOFT-SVERL, where the generator and verifier share parameters and the model can inflate its own rewards by becoming overly permissive.

Self-verifying RL. Reasoning over Mathematical Objects (Aggarwal et al., 2026) is close to our work on the verifier side because it studies strong LM judges, on-policy judge training, and hard-to-verify mathematical-object equivalence. However, its setting remains centered on externally trained judges for mathematically structured outputs. URPO (Lu et al., 2025) is also relevant because the generator and evaluator co-evolve within one training loop, though it remains grounded in preference data and verifiable reasoning tasks. In contrast, SOFT-SVERL combines checklist-conditioned soft rewards, repeated verifier voting, verifier co-training from gold and replay supervision, and explicit anti-collapse regularization for self-verification.

A.8 SFT vs. RL

A natural alternative to RL with learned verifiers is supervised fine-tuning (SFT) on high-quality outputs from a stronger model. We compare RL training with a 20B verifier against SFT on outputs generated by GPT-OSS-120B in the high reasoning setting, giving SFT an advantage in both model scale and generation quality.

Figure 6 shows that RL with the 20B verifier outperforms SFT with the 120B verifier on IFEval (84.20 vs. 81.93, +2.3 points) despite using a 6$\times$ smaller verifier model. The gap is more pronounced on math benchmarks. SFT degrades MATH500 from 87.60 to 83.60 ($-$4.0 points) and AIME 2024 from 35.67 to 31.83 ($-$3.8 points), while RL improves both (89.60 and 39.33, respectively).

Table 3: Supervision assumptions of closely related RL methods. "Gold" denotes reliance on a known final answer or reference outcome; "Judge/RM" denotes use of a fixed LLM judge or separately trained reward model. Partial entries indicate that the property applies only to part of the method or evaluation setting.

Method	Gold	Judge/RM	Open tasks	Rubric	Self-verif.
Answer-likelihood RL (Zhou et al., 2025; Yu et al., 2025b)	Yes	No	Partial	No	No
TTRL / unlabeled self-labeling (Zuo et al., 2025)	No	No	No evidence	No	Partial
INTUITOR / GENIUS (Zhao et al., 2025; Xu et al., 2025)	No	No	Limited	No	Partial
Writing-Zero (Jia et al., 2025)	No	Yes	Yes	No	No
URPO (Lu et al., 2025)	Mixed	No	Yes	No	Partial
Compute as Teacher (Jayalath et al., 2025)	No	Yes	Yes	Yes	No
Rubrics as Rewards (Gunjal et al., 2025)	Yes	Yes	Yes	Yes	No
Reasoning over Mathematical Objects (Aggarwal et al., 2026)	Task-spec.	Yes	Partial	No	No
Soft-SVeRL (ours)	**No**	**No**	**Yes**	**Yes**	**Yes**

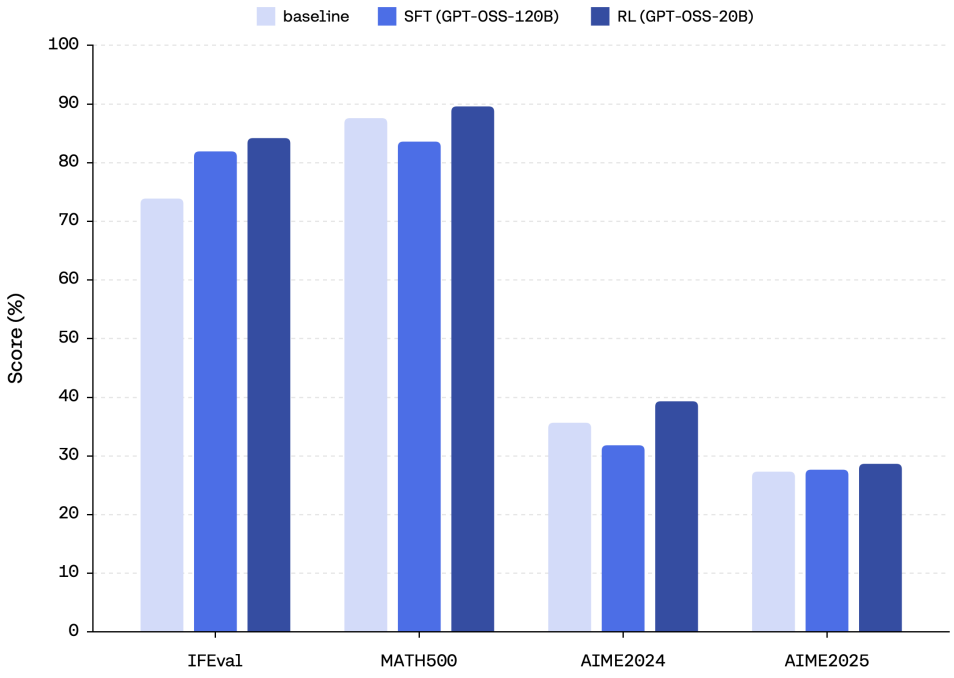


Figure 6: SFT on GPT-OSS-120B generated outputs vs. RL with GPT-OSS-20B verifier rewards. RL outperforms SFT on all benchmarks despite using a 6× smaller model, and avoids the math degradation that SFT exhibits.

The math degradation under SFT is consistent with the well-documented pattern of capability regression during supervised post-training: fine-tuning on domain-specific data can overwrite previously learned capabilities. RL avoids this because policy-gradient updates adjust the model toward higher-reward behavior without forcing it to match specific output distributions, preserving useful capabilities that are not penalized by the reward. This is consistent with recent evidence that RL training produces

sparse, low-rank updates to model parameters, requiring far less capacity than supervised fine-tuning (Schulman & Thinking Machines Lab, 2025).

A.9 Checklist Generation and Verification Details

A.9.1 Generated checklist statistics

Figure 7 summarizes the number of checklist items produced by the LLM checklist generator across the training prompts. Most generated checklists contain between five and eight items, with a long tail of prompts receiving more detailed decompositions.

A.9.2 Verifier prompt.

The following template is used to query the verifier model for each checklist item. The verifier receives the original instruction, the candidate response, and a single checklist question, and outputs a binary `yes`/`no` judgment.

A.9.3 Checklist-generation prompt.

The following template is used to generate checklist items from a prompt. The checklist generator produces an ordered list of yes/no questions that collectively cover the requirements of the instruction.

A.9.4 Checklist examples and failure modes

Table 4 shows representative checklist outputs. In general, useful checklist items are atomic, necessary, and directly verifiable from the instruction and candidate response. Common failure modes include under-decomposition, over-decomposition, spurious criteria, and missing constraints.

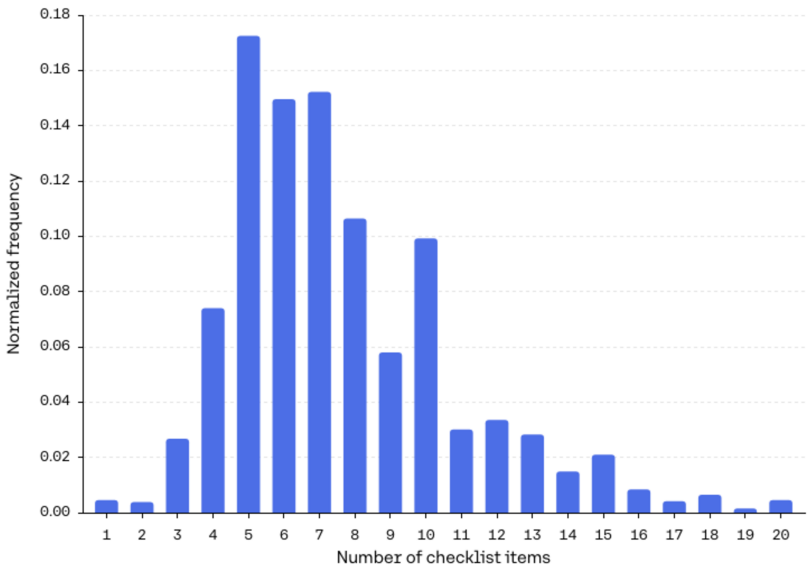


Figure 7: **Distribution of LLM-generated checklist lengths.** The x-axis gives the number of checklist items generated for a prompt, and the y-axis reports the normalized frequency.

```
Here is an instruction given to an AI Assistant:

<START OF INSTRUCTION>
{instruction}
<END OF INSTRUCTION>

Here is an AI Assistant's response to the instruction:

<START OF RESPONSE>
{response}
<END OF RESPONSE>

Please answer the following question about the AI Assistant's response
to the instruction:

<START OF QUESTION>
{checklist_question}
<END OF QUESTION>

Output "yes" or "no" and nothing else.
```

Figure 8: **Verifier prompt template.** The verifier receives the instruction, response, and checklist item, then returns a binary judgment.

```
Here is an instruction given to an AI assistant:

<START OF INSTRUCTION>
{prompt}
<END OF INSTRUCTION>

Please write an exhaustive checklist of yes/no questions that could be
used to evaluate whether the AI assistant's response to the question is
completely correct. The answer to each question should be yes if and only
if the AI assistant's response is correct. Be sure to consider edge cases
if applicable. The checklist should not include questions which are not
absolutely essential for a completely correct response even if the answer
"yes" would likely imply a better response. The ordering of the checklist
should be such that the questions are ordered from most important to least
important. Output the checklist as a markdown numbered list. DO NOT output
anything else besides the numbered list.
```

Figure 9: **Checklist-generation prompt template.** The checklist generator decomposes an instruction into ordered yes/no criteria.

Instruction	Good checklist items	Failure mode examples
Write exactly three bullet points about Paris. Do not use the word "travel".	• Does the response contain exactly three bullet points? • Is each bullet point about Paris? • Does the response avoid the word "travel"?	• Is the response engaging? *(spurious)* • Does the response mention Paris and use three bullet points? *(under-decomposed)*

Table 4: Illustrative checklist items and checklist-generation failure modes.

A.10 Additional Analysis of Checklist Reward MSE

This section gives the formal gradient definitions, bias–variance decomposition, and additional corollaries used in Section 3.1. The main text states sufficient conditions for checklist verification to reduce gradient MSE relative to holistic verification; here we provide the derivation and make the dependence on checklist size explicit.

We use the notation from the main text: $\mathbf{Y}^*(x,y)$ denotes the latent checklist vector, $S^*(x,y)$ denotes strict success, $\bar{S}^*(x,y)$ denotes partial credit, and $\Delta^*(x,y) = \bar{S}^*(x,y) - S^*(x,y)$ denotes the relaxation gap. Let $s_\theta(x,y) := \nabla_\theta \log \pi_\theta(y \mid x)$ be the score function and let $b(x)$ be a baseline. We define the ideal, holistic-verifier, and checklist-verifier updates as

$$\begin{aligned} g^*(x,y) &:= \big(S^*(x,y) - b(x)\big)s_\theta(x,y), \\ g_{\text{single}} &:= (J - b(x))s_\theta(x,y), \\ g_{\text{chk}} &:= (\bar{J} - b(x))s_\theta(x,y). \end{aligned}$$

The proof of Theorem 1 follows a bias–variance decomposition of the conditional gradient MSE. The variance terms capture verifier noise, while the bias terms capture the mismatch between the verifier-induced update and the update from the strict-success target.

Proposition 1 (Checklist variance under conditional independence). *Under the setup of Section 3.1, if $J_1, \ldots, J_K$ are conditionally independent given (x,y), then*

$$\mathrm{Var}(\bar{J} \mid x,y) = \frac{1}{K^2}\sum_{k=1}^{K} \mu_k(x,y)\big(1-\mu_k(x,y)\big) \le \frac{1}{4K},$$

where $\mu_k(x,y) := (1-q') + \alpha' Y_k^*(x,y)$, and therefore

$$\mathbb{E}\big[\|g_{\text{chk}} - \mathbb{E}[g_{\text{chk}} \mid x,y]\|_2^2 \,\big|\, x,y\big] \le \frac{1}{4K}\|s_\theta(x,y)\|_2^2.$$

Proof. Under conditional independence, $\mathrm{Cov}(J_k, J_\ell \mid x,y) = 0$ for all $k \neq \ell$. The general variance formula from Proposition 3 reduces to

$$\mathrm{Var}(\bar{J} \mid x,y) = \frac{1}{K^2}\sum_{k=1}^{K} \mu_k(x,y)\big(1-\mu_k(x,y)\big).$$

Since $\mu_k(x,y) \in [0,1]$, we have $\mu_k(x,y)(1-\mu_k(x,y)) \le 1/4$ for each k, so

$$\mathrm{Var}(\bar{J} \mid x,y) \le \frac{1}{K^2} \cdot K \cdot \frac{1}{4} = \frac{1}{4K}.$$

Multiplying by $\|s_\theta(x,y)\|_2^2$ gives

$$\mathbb{E}\big[\|g_{\text{chk}} - \mathbb{E}[g_{\text{chk}} \mid x,y]\|_2^2 \,\big|\, x,y\big] \le \frac{1}{4K}\|s_\theta(x,y)\|_2^2.$$

□

Proposition 2 (Conditional bias of single-verifier and checklist estimators). *Under the setup of Section 3.1, conditioning on (x,y),*

$$\mathbb{E}[g_{\text{single}} \mid x,y] - g^*(x,y) = \Big((1-q) + (\alpha-1)S^*(x,y)\Big)s_\theta(x,y),$$

and

$$\mathbb{E}[g_{\text{chk}} \mid x,y] - g^*(x,y) = \Big((1-q') + \alpha'\Delta^*(x,y) + (\alpha'-1)S^*(x,y)\Big)s_\theta(x,y).$$

Equivalently,

$$\|\mathbb{E}[g_{\text{single}} \mid x,y] - g^*(x,y)\|_2^2 = \Big((1-q) + (\alpha-1)S^*(x,y)\Big)^2\|s_\theta(x,y)\|_2^2,$$

and

$$\|\mathbb{E}[g_{\text{chk}} \mid x,y] - g^*(x,y)\|_2^2 = \Big((1-q') + \alpha'\Delta^*(x,y) + (\alpha'-1)S^*(x,y)\Big)^2\|s_\theta(x,y)\|_2^2.$$

Proof. Fix (x, y); then $s_\theta(x, y)$ and $b(x)$ are deterministic. For the single verifier,

$$\mathbb{E}[g_{\text{single}} \mid x, y] = \big(\mathbb{E}[J \mid x, y] - b(x)\big) s_\theta(x, y).$$

Using

$$\mathbb{E}[J \mid x, y] = (1 - q) + \alpha S^*(x, y)$$

and subtracting

$$g^*(x, y) = \big(S^*(x, y) - b(x)\big) s_\theta(x, y)$$

gives

$$\begin{aligned}\mathbb{E}[g_{\text{single}} \mid x, y] - g^*(x, y) &= \Big((1 - q) + \alpha S^*(x, y) - S^*(x, y)\Big) \\ &\quad \cdot s_\theta(x, y) \\ &= \Big((1 - q) + (\alpha - 1) S^*(x, y)\Big) s_\theta(x, y).\end{aligned}$$

For the checklist estimator, linearity of expectation gives

$$\mathbb{E}[\bar{J} \mid x, y] = \frac{1}{K} \sum_{k=1}^{K} \big((1 - q') + \alpha' Y_k^*(x, y)\big) = (1 - q') + \alpha' \bar{S}^*(x, y).$$

Subtracting $g^*(x, y)$ and using $\bar{S}^*(x, y) = S^*(x, y) + \Delta^*(x, y)$ yields

$$\begin{aligned}\mathbb{E}[g_{\text{chk}} \mid x, y] - g^*(x, y) &= \Big((1 - q') + \alpha' \bar{S}^*(x, y) - S^*(x, y)\Big) s_\theta(x, y) \\ &= \Big((1 - q') + \alpha' \Delta^*(x, y) + (\alpha' - 1) S^*(x, y)\Big) s_\theta(x, y).\end{aligned}$$

Taking squared norms gives the stated formulas. □

Proposition 3 (Conditional variance of single-verifier and checklist estimators). *Under the setup of Section 3.1, conditioning on (x, y),*

$$\mathrm{Cov}(g_{\text{single}} \mid x, y) = \mathrm{Var}(J \mid x, y)\, s_\theta(x, y) s_\theta(x, y)^\top,$$

and

$$\mathrm{Cov}(g_{\text{chk}} \mid x, y) = \mathrm{Var}(\bar{J} \mid x, y)\, s_\theta(x, y) s_\theta(x, y)^\top,$$

where

$$\mathrm{Var}(J \mid x, y) = \mu_{\text{single}}(x, y)\big(1 - \mu_{\text{single}}(x, y)\big), \qquad \mu_{\text{single}}(x, y) := (1 - q) + \alpha S^*(x, y),$$

and

$$\mathrm{Var}(\bar{J} \mid x, y) = \frac{1}{K^2} \left(\sum_{k=1}^{K} \mu_k(x, y)\big(1 - \mu_k(x, y)\big) + 2 \sum_{1 \le k < \ell \le K} \mathrm{Cov}(J_k, J_\ell \mid x, y) \right),$$

with

$$\mu_k(x, y) := (1 - q') + \alpha' Y_k^*(x, y).$$

Equivalently,

$$\mathbb{E}\big[\|g_{\text{single}} - \mathbb{E}[g_{\text{single}} \mid x, y]\|_2^2 \,\big|\, x, y\big] = \|s_\theta(x, y)\|_2^2\, \mu_{\text{single}}(x, y)\big(1 - \mu_{\text{single}}(x, y)\big),$$

and

$$\mathbb{E}\big[\|g_{\text{chk}} - \mathbb{E}[g_{\text{chk}} \mid x, y]\|_2^2 \,\big|\, x, y\big] = \|s_\theta(x, y)\|_2^2\, \mathrm{Var}(\bar{J} \mid x, y).$$

Under the additional assumption that $J_1, \ldots, J_K$ are conditionally independent given (x, y),

$$\mathrm{Var}(\bar{J} \mid x, y) = \frac{1}{K^2} \sum_{k=1}^{K} \mu_k(x, y)\big(1 - \mu_k(x, y)\big) \le \frac{1}{4K}.$$

Proof. Condition on (x, y); then $s_\theta(x, y)$ and $b(x)$ are deterministic and all randomness comes from verifier outputs. Since

$$g_{\text{single}} = (J - b(x))s_\theta(x, y),$$

we have

$$g_{\text{single}} - \mathbb{E}[g_{\text{single}} \mid x, y] = \big(J - \mathbb{E}[J \mid x, y]\big)s_\theta(x, y).$$

Therefore,

$$\text{Cov}(g_{\text{single}} \mid x, y) = \text{Var}(J \mid x, y)\, s_\theta(x, y)s_\theta(x, y)^\top.$$

Since $J \in \{0, 1\}$ is Bernoulli with mean

$$\mu_{\text{single}}(x, y) = (1 - q) + \alpha S^*(x, y),$$

its variance is

$$\mu_{\text{single}}(x, y)\big(1 - \mu_{\text{single}}(x, y)\big).$$

The same argument applies to

$$g_{\text{chk}} = (\bar{J} - b(x))s_\theta(x, y),$$

giving

$$\text{Cov}(g_{\text{chk}} \mid x, y) = \text{Var}(\bar{J} \mid x, y)\, s_\theta(x, y)s_\theta(x, y)^\top.$$

Expanding the variance of the average,

$$\text{Var}(\bar{J} \mid x, y) = \frac{1}{K^2}\left(\sum_{k=1}^{K} \text{Var}(J_k \mid x, y) + 2 \sum_{1 \le k < \ell \le K} \text{Cov}(J_k, J_\ell \mid x, y)\right).$$

Since each J_k is Bernoulli with mean

$$\mu_k(x, y) = (1 - q') + \alpha' Y_k^*(x, y),$$

we have

$$\text{Var}(J_k \mid x, y) = \mu_k(x, y)\big(1 - \mu_k(x, y)\big).$$

The norm-squared identities follow from

$$\mathbb{E}\big[\|(Z - \mathbb{E}[Z])s\|_2^2\big] = \|s\|_2^2 \text{Var}(Z).$$

Under conditional independence, the cross-covariance terms vanish, and $\mu_k(1 - \mu_k) \le 1/4$ gives the $1/(4K)$ bound. □

Proposition 4 (Sufficient conditions for gradient bias reduction). *Under the setup of Section 3.1,*

$$\|\mathbb{E}[g_{\text{chk}} \mid x, y] - g^*(x, y)\|_2^2 \le \|\mathbb{E}[g_{\text{single}} \mid x, y] - g^*(x, y)\|_2^2$$

whenever either

$$S^*(x, y) = 1 \qquad \textit{and} \qquad p' \ge p,$$

or

$$S^*(x, y) = 0 \qquad \textit{and} \qquad \alpha' \Delta^*(x, y) \le q' - q.$$

Proof. By Proposition 2,

$$\mathbb{E}[g_{\text{single}} \mid x, y] - g^*(x, y) = \Big((1 - q) + (\alpha - 1)S^*(x, y)\Big)s_\theta(x, y),$$

and

$$\mathbb{E}[g_{\text{chk}} \mid x, y] - g^*(x, y) = \Big((1 - q') + \alpha' \Delta^*(x, y) + (\alpha' - 1)S^*(x, y)\Big)s_\theta(x, y).$$

Case (i): $S^*(x, y) = 1$. When $S^*(x, y) = 1$, all checklist items are satisfied, so $\bar{S}^*(x, y) = 1$ and $\Delta^*(x, y) = 0$. The bias expressions reduce to

$$\|\mathbb{E}[g_{\text{single}} \mid x, y] - g^*(x, y)\|_2^2 = (1 - p)^2 \|s_\theta(x, y)\|_2^2,$$

and

$$\|\mathbb{E}[g_{\text{chk}} \mid x, y] - g^*(x, y)\|_2^2 = (1 - p')^2 \|s_\theta(x, y)\|_2^2.$$

Since $p' \geq p$ implies $(1-p')^2 \leq (1-p)^2$, the checklist squared bias is no larger.

Case (ii): $S^*(x,y) = 0$**.** When $S^*(x,y) = 0$, the bias expressions reduce to

$$\|\mathbb{E}[g_{\text{single}} \mid x,y] - g^*(x,y)\|_2^2 = (1-q)^2 \|s_\theta(x,y)\|_2^2,$$

and

$$\|\mathbb{E}[g_{\text{chk}} \mid x,y] - g^*(x,y)\|_2^2 = \big(1 - q' + \alpha'\Delta^*(x,y)\big)^2 \|s_\theta(x,y)\|_2^2.$$

If $\alpha'\Delta^*(x,y) \leq q' - q$, then

$$1 - q' + \alpha'\Delta^*(x,y) \leq 1 - q,$$

so the checklist squared bias is no larger. □

A.10.1 Proof of Theorem 1

Proposition 5 (Bias–variance decomposition of gradient MSE). *Under the setup of Section 3.1, conditioning on* (x,y)*,*

$$\mathbb{E}\big[\|g_{\text{single}} - g^*\|_2^2 \,\big|\, x,y\big] = \|\mathbb{E}[g_{\text{single}} \mid x,y] - g^*(x,y)\|_2^2 + \mathbb{E}\big[\|g_{\text{single}} - \mathbb{E}[g_{\text{single}} \mid x,y]\|_2^2 \,\big|\, x,y\big],$$

and

$$\mathbb{E}\big[\|g_{\text{chk}} - g^*\|_2^2 \,\big|\, x,y\big] = \|\mathbb{E}[g_{\text{chk}} \mid x,y] - g^*(x,y)\|_2^2 + \mathbb{E}\big[\|g_{\text{chk}} - \mathbb{E}[g_{\text{chk}} \mid x,y]\|_2^2 \,\big|\, x,y\big].$$

Proof. For any random vector G and deterministic vector a,

$$\mathbb{E}\big[\|G - a\|_2^2\big] = \|\mathbb{E}[G] - a\|_2^2 + \mathbb{E}\big[\|G - \mathbb{E}[G]\|_2^2\big].$$

Apply this identity conditionally on (x,y) with $G = g_{\text{single}}$ and $a = g^*(x,y)$, and then with $G = g_{\text{chk}}$ and $a = g^*(x,y)$. □

Proof of Theorem 1. We use the bias–variance decomposition in Proposition 5 and analyze the two cases separately.

Case (i): $S^*(x,y) = 1$**.** When $S^*(x,y) = 1$, the relaxation gap $\Delta^*(x,y) = 0$. By Proposition 2, the squared biases are

$$\|\mathbb{E}[g_{\text{single}} \mid x,y] - g^*(x,y)\|_2^2 = (1-p)^2 \|s_\theta(x,y)\|_2^2,$$

and

$$\|\mathbb{E}[g_{\text{chk}} \mid x,y] - g^*(x,y)\|_2^2 = (1-p')^2 \|s_\theta(x,y)\|_2^2.$$

For the variance terms, when $S^*(x,y) = 1$, the conditional means are $\mu_{\text{single}}(x,y) = p$ and $\mu_k(x,y) = p'$ for all k. Thus,

$$\mathbb{E}\big[\|g_{\text{single}} - \mathbb{E}[g_{\text{single}} \mid x,y]\|_2^2 \,\big|\, x,y\big] = p(1-p)\|s_\theta(x,y)\|_2^2,$$

while conditional independence gives

$$\mathbb{E}\big[\|g_{\text{chk}} - \mathbb{E}[g_{\text{chk}} \mid x,y]\|_2^2 \,\big|\, x,y\big] = \frac{p'(1-p')}{K} \|s_\theta(x,y)\|_2^2.$$

Therefore,

$$\mathbb{E}\big[\|g_{\text{chk}} - g^*\|_2^2 \,\big|\, x,y\big] = \left[(1-p')^2 + \frac{p'(1-p')}{K}\right] \|s_\theta(x,y)\|_2^2.$$

Since $K \geq 1$,

$$(1-p')^2 + \frac{p'(1-p')}{K} \leq (1-p')^2 + p'(1-p') = 1 - p'.$$

If $p' \geq p$, then $1 - p' \leq 1 - p$. Also,

$$(1-p)^2 + p(1-p) = 1 - p.$$

Hence

$$\mathbb{E}\big[\|g_{\text{chk}} - g^*\|_2^2 \,\big|\, x,y\big] \leq (1-p)\|s_\theta(x,y)\|_2^2 = \mathbb{E}\big[\|g_{\text{single}} - g^*\|_2^2 \,\big|\, x,y\big].$$

Case (ii): $S^*(x,y) = 0$**.** When $S^*(x,y) = 0$, Proposition 2 gives

$$\|\mathbb{E}[g_{\text{single}} \mid x, y] - g^*(x,y)\|_2^2 = (1-q)^2 \|s_\theta(x,y)\|_2^2,$$

and

$$\|\mathbb{E}[g_{\text{chk}} \mid x, y] - g^*(x,y)\|_2^2 = \big(1 - q' + \alpha' \Delta^*(x,y)\big)^2 \|s_\theta(x,y)\|_2^2.$$

The single-verifier variance term is

$$(1-q)q \, \|s_\theta(x,y)\|_2^2,$$

and the checklist variance term is bounded by

$$\frac{1}{4K} \|s_\theta(x,y)\|_2^2$$

by Proposition 1. Thus,

$$\mathbb{E}\big[\|g_{\text{single}} - g^*\|_2^2 \,\big|\, x, y\big] = \big[(1-q)^2 + (1-q)q\big] \|s_\theta(x,y)\|_2^2 = (1-q)\|s_\theta(x,y)\|_2^2.$$

A sufficient condition for the checklist MSE to be no larger is

$$\big(1 - q' + \alpha' \Delta^*(x,y)\big)^2 + \frac{1}{4K} \le 1 - q,$$

which is exactly the condition in the theorem. □

A.10.2 CHECKLIST-SIZE IMPLICATIONS

Corollary 1 (Bias condition implies MSE condition for large K). *In the regime $S^*(x,y) = 0$, let*

$$\textit{(A)} \quad \alpha' \Delta^*(x,y) \le q' - q \qquad \textit{and} \qquad \textit{(B)} \quad \big(1 - q' + \alpha' \Delta^*(x,y)\big)^2 + \frac{1}{4K} \le 1 - q.$$

Condition (A) is the bias-only sufficient condition of Proposition 4; condition (B) is the sufficient condition for full MSE improvement in Theorem 1. If (A) holds and

$$K \ge \frac{1}{4(1-q)q},$$

then (B) holds, and therefore checklist verification achieves lower conditional MSE than the single-verifier estimator.

Proof. Assume condition (A), i.e.,

$$\alpha' \Delta^*(x,y) \le q' - q.$$

Then

$$1 - q' + \alpha' \Delta^*(x,y) \le 1 - q,$$

so

$$\big(1 - q' + \alpha' \Delta^*(x,y)\big)^2 \le (1-q)^2.$$

It therefore suffices to show

$$(1-q)^2 + \frac{1}{4K} \le 1 - q,$$

which is equivalent to

$$\frac{1}{4K} \le (1-q)q.$$

This holds whenever

$$K \ge \frac{1}{4(1-q)q}.$$

Hence condition (B) holds, and Theorem 1 gives the MSE inequality. □

Corollary 2 (Checklist-size thresholds). *On incorrect outputs* $(S^*(x,y)=0)$*, let*

$$A(x,y) := 1 - q' + \alpha' \Delta^*(x,y).$$

If $A(x,y) \le 1-q$ *and*

$$K \ge \frac{1}{4A(x,y)\big(1-A(x,y)\big)},$$

then

$$\mathbb{E}\big[\|g_{\text{chk}} - g^*\|_2^2 \,\big|\, x,y\big] \le \mathbb{E}\big[\|g_{\text{single}} - g^*\|_2^2 \,\big|\, x,y\big].$$

On correct outputs $(S^*(x,y)=1)$*, if* $(1-p')^2 < 1-p$ *and*

$$K \ge \frac{p'(1-p')}{(1-p)-(1-p')^2},$$

then

$$\mathbb{E}\big[\|g_{\text{chk}} - g^*\|_2^2 \,\big|\, x,y\big] \le \mathbb{E}\big[\|g_{\text{single}} - g^*\|_2^2 \,\big|\, x,y\big].$$

Proof.

Incorrect outputs. Since $A(x,y) \le 1-q$, it suffices to show

$$A(x,y)^2 + \frac{1}{4K} \le A(x,y).$$

This is equivalent to

$$\frac{1}{4K} \le A(x,y)\big(1-A(x,y)\big),$$

which is exactly the assumed lower bound on K. Under this condition,

$$A(x,y)^2 + \frac{1}{4K} \le A(x,y) \le 1-q,$$

so the incorrect-output condition in Theorem 1 holds.

Correct outputs. From the proof of Theorem 1, the checklist MSE normalized by $\|s_\theta(x,y)\|_2^2$ is

$$(1-p')^2 + \frac{p'(1-p')}{K},$$

and the single-verifier MSE normalized by $\|s_\theta(x,y)\|_2^2$ is

$$(1-p)^2 + p(1-p) = 1-p.$$

The checklist MSE is no larger whenever

$$(1-p')^2 + \frac{p'(1-p')}{K} \le 1-p.$$

Since $(1-p')^2 < 1-p$ by assumption, rearranging gives

$$K \ge \frac{p'(1-p')}{(1-p)-(1-p')^2},$$

which is the stated threshold. □

Corollary 3 (Necessity of a bias-side condition). *Larger* K *reduces variance but does not affect the structural bias term, so no condition on* K *alone can guarantee checklist MSE improvement.*

On incorrect outputs, if

$$\big(1 - q' + \alpha' \Delta^*(x,y)\big)^2 > 1-q,$$

then

$$\mathbb{E}\big[\|g_{\text{chk}} - g^*\|_2^2 \,\big|\, x,y\big] > \mathbb{E}\big[\|g_{\text{single}} - g^*\|_2^2 \,\big|\, x,y\big]$$

for all finite K*.*

Similarly, on correct outputs, if

$$(1-p')^2 > 1-p,$$

then the checklist estimator has larger conditional MSE than the single-verifier estimator for all finite K*.*

Proof.

Incorrect outputs. The checklist MSE is at least its squared bias:

$$\mathbb{E}\big[\|g_{\mathrm{chk}} - g^*\|_2^2 \,\big|\, x, y\big] \geq \big(1 - q' + \alpha'\Delta^*(x,y)\big)^2 \|s_\theta(x,y)\|_2^2.$$

The single-verifier MSE on incorrect outputs is

$$(1-q)\|s_\theta(x,y)\|_2^2.$$

Therefore, if

$$\big(1 - q' + \alpha'\Delta^*(x,y)\big)^2 > 1 - q,$$

then the checklist MSE is strictly larger for all finite K.

Correct outputs. The checklist MSE on correct outputs is

$$\left[(1-p')^2 + \frac{p'(1-p')}{K}\right] \|s_\theta(x,y)\|_2^2,$$

while the single-verifier MSE is

$$(1-p)\|s_\theta(x,y)\|_2^2.$$

If $(1-p')^2 > 1-p$, then

$$(1-p')^2 + \frac{p'(1-p')}{K} > 1 - p$$

for all finite K, so the checklist MSE is strictly larger. □

Together with Corollary 1, Corollary 3 gives the main checklist-size message: once the bias-side condition is controlled, sufficiently large K can make checklist verification lower-MSE than holistic verification; but increasing K alone cannot overcome a large partial-credit bias.

A.11 Motivation for the Partition-Style Penalty

This section gives a mathematical motivation for the partition-style Yes-rate penalty used in SOFT-SVERL. The result assumes the standard Gibbs form of the KL-regularized optimal policy and focuses on the reward gradient induced by a checklist-decomposed verifier reward.

Proposition 6 (Checklist-decomposed reward gradient). *Let x be a prompt, y a response, and $C(x) = \{c_1, \ldots, c_K\}$ a checklist. Let $\rho_\phi(z \mid x, y, c_k)$ be the verifier trace distribution and let $d(z) \in \{0,1\}$ denote the parsed Yes/No decision. Define the verifier's item-level Yes-rate as*

$$r_\phi(x, y, c_k) := \mathbb{E}_{z \sim \rho_\phi(\cdot \mid x, y, c_k)}[d(z)]. \tag{9}$$

The checklist reward is

$$R_\phi(x,y) := \frac{1}{K}\sum_{k=1}^{K} r_\phi(x, y, c_k). \tag{10}$$

For a fixed verifier reward R_ϕ, assume the KL-regularized optimal generator has the Gibbs form

$$\pi_{\theta^*(\phi)}(y \mid x) = \frac{1}{Z_\phi(x)}\pi_{\mathrm{ref}}(y \mid x)\exp\left(\frac{1}{\beta}R_\phi(x,y)\right), \tag{11}$$

where

$$Z_\phi(x) = \sum_{y'} \pi_{\mathrm{ref}}(y' \mid x)\exp\left(\frac{1}{\beta}R_\phi(x,y')\right). \tag{12}$$

Let $\mathcal{D}^+$ be a distribution of target-good prompt-response pairs, and define

$$\mathcal{L}_{\mathrm{gen}}(\phi) := \mathbb{E}_{(x,y)\sim\mathcal{D}^+}\left[\log \pi_{\theta^*(\phi)}(y \mid x)\right]. \tag{13}$$

Then

$$\nabla_\phi \mathcal{L}_{\mathrm{gen}}(\phi) = \frac{1}{\beta K}\sum_{k=1}^{K}\Big(\mathbb{E}_{(x,y)\sim\mathcal{D}^+}[\nabla_\phi r_\phi(x,y,c_k)] - \mathbb{E}_x \mathbb{E}_{y' \sim \pi_{\theta^*(\phi)}(\cdot \mid x)}[\nabla_\phi r_\phi(x, y', c_k)]\Big). \tag{14}$$

Proof. Using Eq. 11,

$$\log \pi_{\theta^*(\phi)}(y \mid x) = \log \pi_{\text{ref}}(y \mid x) + \frac{1}{\beta} R_\phi(x, y) - \log Z_\phi(x).$$

Since π_{ref} does not depend on ϕ,

$$\nabla_\phi \log \pi_{\theta^*(\phi)}(y \mid x) = \frac{1}{\beta} \nabla_\phi R_\phi(x, y) - \nabla_\phi \log Z_\phi(x).$$

From Eq. 12,

$$\nabla_\phi Z_\phi(x) = \sum_{y'} \pi_{\text{ref}}(y' \mid x) \exp\left(\frac{1}{\beta} R_\phi(x, y')\right) \frac{1}{\beta} \nabla_\phi R_\phi(x, y').$$

Dividing by $Z_\phi(x)$ gives

$$\nabla_\phi \log Z_\phi(x) = \frac{1}{\beta} \mathbb{E}_{y' \sim \pi_{\theta^*(\phi)}(\cdot|x)} [\nabla_\phi R_\phi(x, y')].$$

Therefore,

$$\nabla_\phi \mathcal{L}_{\text{gen}}(\phi) = \frac{1}{\beta} \left(\mathbb{E}_{(x,y)\sim\mathcal{D}^+} [\nabla_\phi R_\phi(x, y)] - \mathbb{E}_x \mathbb{E}_{y' \sim \pi_{\theta^*(\phi)}(\cdot|x)} [\nabla_\phi R_\phi(x, y')] \right).$$

Substituting

$$R_\phi(x, y) = \frac{1}{K} \sum_{k=1}^{K} r_\phi(x, y, c_k)$$

and exchanging the finite sum with the expectations yields Eq. 14. □

The second term in Eq. 14 comes from differentiating the partition function $Z_\phi(x)$. It is an item-level policy-sample correction: under the reward-induced generator distribution, it pushes down reward assigned to generated responses. The partition-style penalty in SOFT-SVERL is a selected approximation to this term. The exact correction would require an expectation over prompts, policy samples, and all checklist items:

$$\frac{1}{K} \sum_{k=1}^{K} \mathbb{E}_x \mathbb{E}_{y' \sim \pi_{\theta^*(\phi)}(\cdot|x)} [\nabla_\phi r_\phi(x, y', c_k)]. \tag{15}$$

In practice, SOFT-SVERL replaces $\pi_{\theta^*(\phi)}$ with the current generator π_θ and makes the expectation tractable by selecting a subset $\mathcal{S}$ of on-policy response-item contexts. This gives the partition objective

$$\mathcal{J}_{\text{part}}(\phi) := \mathbb{E}_{(x,y,c)\in\mathcal{S}} \left[r_\phi(x, y, c) \right]. \tag{16}$$

The gradient $\nabla_\phi \mathcal{J}_{\text{part}}(\phi)$ is therefore a selected finite-sample approximation to the policy-sample expectation in Eq. 15. Since SOFT-SVERL subtracts $\mathcal{J}_{\text{part}}$ in the shared objective, this term has the same sign as the negative policy-sample term in Eq. 14. Thus, the partition loss can be viewed as a tractable, selected approximation to the log-partition correction. The subset $\mathcal{S}$ matters because applying this correction to all on-policy items could also suppress genuine successes; SOFT-SVERL applies it only to selected contexts where Yes-rate inflation is the relevant failure mode.

The first term in Eq. 14 increases item-level reward on target-good responses. This term is written as if we had access to a distribution of target-good full responses. In SOFT-SVERL, we do not assume such completions are available for every prompt. Instead, we generate responses from the initial or current policy and label them at the checklist-item level, producing gold and replay tuples (x, y, c, ℓ). We approximate the positive part of the term with items labeled $\ell = 1$, and use items labeled $\ell = 0$ to train the verifier for item-level correctness.