PRECISION BOUNDS DIVERSITY: REWARD ENGINEERING FOR INSTRUCTION FOLLOWING VIA RLVR

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning with verifiable rewards (RLVR) has become a dominant paradigm for improving instruction following (IF), with recent work assuming that scaling constraint diversity is essential for generalization. We challenge this assumption and show that it is fundamentally misguided. By decoupling training data into hard-only, soft-only, and mixed subsets, we find that models trained exclusively on code-verifiable hard constraints outperform those trained on LLM-judged soft constraints on average, including on soft-constraint benchmarks, and match mixed training despite using fewer constraint types. Diagnosis reveals that LLM-based reward models suffer from critically low recall and a systematic inability to penalize violations, with reliability degrading sharply as constraint count increases. Controlled noise-injection experiments further confirm that reward precision, not constraint diversity, is the primary driver of RLVR performance. Motivated by these insights, we propose H1S, a data-composition strategy that retains all hard constraints while pruning soft constraints to at most one per instruction, confining LLM evaluation to its most reliable single-constraint regime. On five IF benchmarks, H1S achieves average gains of 11.3% (7B) and 6.7% (32B) over base models, surpassing mixed-constraint training by 5.5% at 7B while reducing per-step training latency by 58% and preserving general capabilities.

1 INTRODUCTION

Instruction Following (IF) has emerged as a critical evaluation axis for large language models, measuring whether a model can satisfy diverse user-specified constraints in its response (Zhou et al., 2023). These constraints are broadly categorized into two types: *hard constraints*, which admit deterministic code-based verification (e.g., word count, output format), and *soft constraints*, which require semantic judgment by an LLM-as-a-judge (e.g., tone, style, coherence) (Peng et al., 2025; Guo et al., 2025). Driven by the success of Reinforcement Learning with Verifiable Rewards (RLVR) (OpenAI, 2025; Team et al., 2026), the community has increasingly adopted RLVR for IF training. Given the inherent upper bound on the diversity of hard constraints, recent datasets have incorporated a growing proportion of soft constraints to broaden training coverage (Pyatkin et al., 2025; Guo et al., 2025; Peng et al., 2025). The prevailing assumption is straightforward: *more diverse constraints yield better generalization*.

In this work, we challenge this assumption. We construct controlled training sets containing exclusively hard constraints (*Hard-only*), exclusively soft constraints (*Soft-only*), and a mixture of both (*Mix*), and train models under identical RLVR settings. The results reveal a striking phenomenon: *Hard-only training outperforms Soft-only on average and matches Mix*, even when evaluated on soft-constraint benchmarks (§3.2). Moreover, the Soft-only model achieves 3.4% *higher* training rewards yet 2.7% *lower* test performance—a hallmark of reward hacking. This advantage persists across architectures (Qwen2.5, Qwen3, Llama3.2) and scales (3B–32B), indicating it is a property of the RLVR paradigm rather than a model-specific artifact.

To diagnose the root cause, we conduct a systematic reward reliability study (§4). Three key findings emerge. **First**, LLM-based judges exhibit substantially lower precision and, critically, much lower *recall* than code-based verifiers (e.g., Qwen3-32B: 76.5% precision, 30.6% recall vs. code verifier: 96.0%, 81.2%), revealing a systematic inability to penalize constraint violations.

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.

Second, this unreliability is sharply amplified by multi-constraint complexity: reward recall drops by 15.6% when a second soft constraint is introduced, exposing a compositional reasoning failure in the judges we evaluate. **Third**, through controlled noise-injection experiments, we decouple precision from diversity and demonstrate that *reward precision, not constraint diversity, is the primary driver of RLVR performance* (Figure 1). The apparent benefits of diversity are largely confounded by the noisy reward signal introduced by LLM judges.

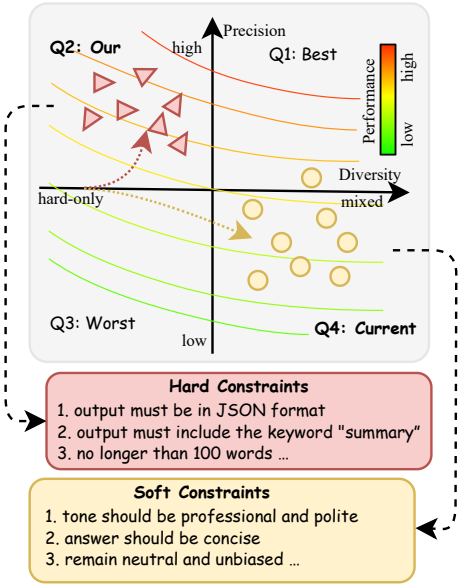


Figure 1: Reward precision vs. constraint diversity in RLVR training. High-precision rewards (code-based verifiers) achieve superior generalization even with limited diversity, whereas low-precision rewards (LLM judges) degrade performance despite broader coverage.

Motivated by these insights, we propose **Hard + 1 Soft (H1S)**, a simple yet effective data-composition strategy that prioritizes reward precision over constraint diversity (§5). H1S retains all hard constraints for their high-precision, code-verifiable supervision, while pruning soft constraints to at most one per instruction, confining LLM-based evaluation to the single-constraint regime where judge reliability is highest. This asymmetric design preserves the beneficial semantic guidance of soft constraints while eliminating the multi-constraint judge biases that enable reward hacking.

Extensive experiments validate H1S across multiple dimensions (§6). On five IF benchmarks, H1S achieves average gains of 11.3% (7B) and 6.7% (32B) over base models, surpassing the Mix variant by 5.5% despite using fewer constraint types. Critically, H1S preserves general capabilities (mathematics, knowledge, writing), and reduces per-step training latency by 58% compared to full mixed-constraint training. Ablation studies confirm that $k = 1$ soft constraint is the precision–diversity sweet spot: test performance follows an inverted-V curve, while training reward rises monotonically with k—a definitive signature of reward hacking beyond the optimal point.

Our contributions are summarized as follows:

1. **Empirical Discovery.** We reveal that hard-only RLVR training matches or exceeds diverse mixed training; the apparent benefits of constraint diversity are confounded by LLM reward noise (§3).
2. **Mechanistic Diagnosis.** We quantify reward reliability and identify two failure modes of LLM judges: low recall under multi-constraint settings and systematic inability to penalize violations, establishing reward precision as the primary driver of RLVR performance (§4).
3. **Simple and Effective Method.** We propose H1S, which confines LLM evaluation to the single-constraint regime, achieving superior IF performance with faster training while preserving general capabilities (§5, §6).

2 Preliminaries

2.1 Instruction Following with Verifiable Constraints

We formulate instruction following as generating a response $y \sim \pi_\theta(\cdot \mid x)$ that satisfies a set of atomic constraints $C(x) = \{c_1, \ldots, c_m\}$. Each constraint is evaluated by a binary verifier $V_c(x, y) \in \{0, 1\}$. Following prior work (Peng et al., 2025; Guo et al., 2025; Pyatkin et al., 2025), we partition $C(x)$ into two disjoint subsets based on the verification paradigm. The split is operational: a constraint is hard *iff* a rule-based verifier exists for it, and soft otherwise.

Hard constraints. A constraint is *hard* if it admits a deterministic executable verifier, $V_c(x, y) = V_{\text{rule}}(x, y, c) \in \{0, 1\}$, where V_{rule} is a deterministic rule-based program (e.g., a regular-expression

test, parser, or counter) that captures objective criteria such as format and length, at high precision and low cost.

Soft constraints. A constraint is *soft* if its satisfaction cannot be reduced to a deterministic predicate and must be inferred by an LLM-based semantic judge, $V_c(x, y) = \mathbf{1}[J_\theta(x, y, c) = \text{pass}]$, where J_θ is a language-model judge whose reliability must be estimated empirically. These capture subjective qualities (e.g., tone, coherence) at higher computational cost and lower reliability.

2.2 GRPO WITH STRICT BINARY REWARD

We adopt Group Relative Policy Optimization (GRPO) (Shao et al., 2024). For each instruction x, the policy samples G responses $\{y_i\}_{i=1}^{G}$ and optimizes:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{\substack{x\sim\mathcal{D} \\ \{y_i\}\sim\pi_{\theta_{\text{old}}}}} \left[\frac{1}{G}\sum_{i=1}^{G} \frac{1}{|y_i|}\sum_{t=1}^{|y_i|} \Big(\min\big(\rho_{i,t}\hat{A}_i,\ \text{clip}(\rho_{i,t}, 1-\varepsilon, 1+\varepsilon)\hat{A}_i\big) - \beta\, \mathbb{D}_{\text{KL}}^{(i,t)}\left[\pi_\theta \,\|\, \pi_{\text{ref}}\right] \Big) \right], \tag{1}$$

where $\rho_{i,t} = \pi_\theta(y_{i,t} \mid x, y_{i,<t}) / \pi_{\theta_{\text{old}}}(y_{i,t} \mid x, y_{i,<t})$. The token-level advantage $\hat{A}_i$ is derived from group-normalized rewards: $\hat{A}_i = (r_i - \mu)/(\sigma + \epsilon)$, where μ and σ are the mean and standard deviation of $\{r_i\}_{i=1}^{G}$.

Unlike dense reward formulations, we employ a *strict binary reward* that grants credit only when all constraints are satisfied:

$$R(x, y) = \prod_{c\in C(x)} V_c(x, y) = \mathbf{1}\big[\forall c \in C(x),\ V_c(x, y) = 1\big]. \tag{2}$$

This ensures the policy learns complete constraint satisfaction rather than partial compliance.

Table 1: Results on five benchmarks. ♣ and ♠ denote hard-only and mixed constraints (including soft constraints) benchmarks, respectively. Results show that *Hard-only* models outperform *Soft-only* models on average across the evaluated architectures and benchmarks ($p < 0.05$ on a pooled test), with parentheses indicating scores on the `soft-only` subset for CFBench and FollowBench.

Model	♣IFEval	♣Multi-IF	♣IFBench	♠CFBench	♠FollowBench	**Average**
Qwen2.5-7B-inst	72.46	51.05	28.91	44.0 (47)	61.40 (65.46)	$51.56_{\pm 0.24}$
w/ *Hard-only*	**80.78**	**58.89**	**31.63**	49.3 (54)	68.96 (76.90)	$\mathbf{57.85}_{\pm 0.11}$
w/ *Soft-only*	77.82	54.44	27.89	47.6 (52)	68.73 (76.12)	$55.18_{\pm 0.27}$
w/ *Mix*	78.37	58.25	29.59	**51.2** (**55**)	**69.07** (**77.50**)	$57.26_{\pm 0.05}$
Qwen2.5-32B-inst	81.70	64.45	33.67	57.7 (60)	73.06 (80.32)	$61.98_{\pm 0.21}$
w/ *Hard-only*	**84.10**	68.59	35.71	60.5 (**62**)	**75.12** (**82.69**)	$64.70_{\pm 0.25}$
w/ *Soft-only*	82.99	66.04	30.95	58.6 (61)	74.19 (82.34)	$62.43_{\pm 0.13}$
w/ *Mix*	83.55	**68.87**	**37.75**	**60.7** (**62**)	74.99 (81.90)	$\mathbf{65.03}_{\pm 0.13}$
Qwen3-8B	85.77	70.36	24.48	55.4 (56)	67.28 (74.83)	$60.58_{\pm 0.09}$
w/ *Hard-only*	**88.54**	**73.37**	25.17	57.3 (58)	**67.79** (**74.96**)	$\mathbf{62.37}_{\pm 0.06}$
w/ *Soft-only*	86.32	70.97	**27.21**	56.2 (58)	67.21 (73.97)	$61.54_{\pm 0.03}$
w/ *Mix*	86.51	72.73	25.85	**57.6** (**60**)	67.70 (74.58)	$61.96_{\pm 0.03}$
Llama3.2-3B-inst	74.12	40.99	20.74	17.1 (22)	50.40 (55.15)	$40.65_{\pm 0.14}$
w/ *Hard-only*	78.00	**53.00**	**25.85**	22.0 (26)	53.65 (58.12)	$46.50_{\pm 0.09}$
w/ *Soft-only*	74.86	44.75	24.82	22.0 (26)	**54.11** (**59.20**)	$44.11_{\pm 0.22}$
w/ *Mix*	**79.30**	51.31	24.48	**25.2** (**29**)	53.61 (57.77)	$\mathbf{46.74}_{\pm 0.21}$

3 PILOT STUDY A: HARD VS. SOFT CONSTRAINT

In this section, we systematically investigate the impact of constraint type on RLVR training, presenting an interesting phenomenon where training on hard-only constraints performs better than training on diverse constraints.

3.1 Experimental Setup

We train on *VerInstruct* (Peng et al., 2025) (22k instances), partitioned into *hard-only*, *soft-only*, and *mixed* subsets. Hard constraints are verified via a re-implemented code verifier for enhanced robustness; soft constraints use Qwen3-32B as the reward model. Evaluation covers five IF benchmarks: IFEval (Zhou et al., 2023), Multi-IF (He et al., 2024), and IFBench (Pyatkin et al., 2025) (hard-focused); FollowBench (Jiang et al., 2024) and CFBench (Zhang et al., 2025) (mixed). To assess hard-to-soft transfer, we also report results on a `soft-only` subset filtered from mixed benchmarks. All models use Qwen2.5-7B-Instruct (Qwen, 2025) as the base and are trained with GRPO (Shao et al., 2024). We report Instruction Satisfaction Rate (ISR) averaged over 5 runs ($\pm$ std). See Appendix §F for full details.

3.2 Results

Performance Overview. We train on *Hard-only*, *Soft-only*, and *Mix* datasets—using code verifiers for hard constraints and an LLM judge for soft ones—and evaluate on five benchmarks. As shown in Table 1, *Hard-only* training outperforms *Soft-only* on average and matches the *Mix* variants, including on soft IF benchmarks (e.g., CFBench, FollowBench): *Hard-only* 7B improves over *Soft-only* by 2.9% ($p < 0.03$) on in-distribution IFEval and by 2.0% ($p < 0.05$) on out-of-distribution CFBench (including its soft-only subset), challenging the notion that constraint diversity is essential for broad generalization. This advantage persists across architectures (Qwen2.5 (Qwen, 2025), Qwen3 (Yang et al., 2025), Llama3.2 (Dubey et al., 2024)) and scales (3B–32B), indicating it is a property of the RLVR paradigm rather than a model-specific artifact. See Appendix §F.1 for training-data and reward-model study.

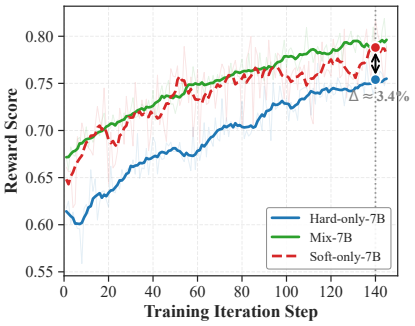


Figure 2: The training curves under hard-only, soft-only and mixed constraint train datasets.

Training Dynamics. Figure 2 reveals a critical discrepancy: compared to hard-only, the soft-only model achieves 3.4% higher training rewards but 2.7% lower test performance. This suggests soft-only training exploits LLM reward biases to artificially inflate reward scores rather than genuinely mastering constraint adherence, whereas rewards derived from code-based verifiers offer a more faithful metric of actual capability.

Key Takeaways

- **Hard-only sufficiency.** Training exclusively on hard constraints matches or exceeds mixed/soft variants while using fewer data types, challenging the assumption that constraint diversity is essential for generalization.
- **Reward reliability gap.** The inverse relationship between training rewards and test performance reveals that LLM-based rewards are prone to being hacked, whereas code-based verifiers offer a more faithful optimization signal.

4 Pilot Study B: Diagnosing the Reward Strategy

We now diagnose the underlying causes of the performance gap observed in Pilot Study A §3, focusing on reward reliability and the trade-off between precision and constraint diversity.

4.1 Quantifying Reward Reliability

Setup. Following Zeng et al. (2026), we sample responses from Qwen2.5-7B-Instruct and obtain human expert annotations as ground-truth labels (see Appendix §G). We evaluate both code-based and LLM-based reward strategies against these gold labels. For hard constraints, code verifiers are evaluated alongside LLM judges; for soft constraints, only LLM judges apply.

Results. Detailed evaluation results are presented in Table 2. For hard constraints, code-based verifiers serve as a high-precision ground truth, achieving 96.0% agreement with human judgments in reward precision. In contrast, even the strongest model we evaluate (Gemini-2.5-Pro) lags by 10% in precision and 15.5% in recall, and open-source models perform significantly worse: Qwen3-32B shows a 19.5% precision drop and a catastrophic 50.6% recall drop compared to code verifiers. Similar conclusions are drawn from the soft constraint evaluation, suggesting that **none of the OSS judges we evaluate matches a code verifier closely enough to serve as a multi-constraint RLVR reward model.**

Table 2: Reward reliability evaluation results. **Prec.** denotes reward precision; **Rec.** measures the proportion of actually non-compliant responses that are correctly identified

Reward Strategy	Hard Constraint		Soft Constraint	
	Prec.	**Rec.**	**Prec.**	**Rec.**
Code Verifier	**96.0**	**81.2**	–	–
Gemini-2.5-pro	86.0$_{\downarrow 10}$	65.7$_{\downarrow 15.5}$	**86.3**	**63.8**
Qwen3-235B-inst	73.2	23.6	75.6	63.8
Qwen3-32B	76.5	30.6$_{\downarrow 50.6}$	74.5	20.9$_{\downarrow 42.9}$
QwQ-32B	80.7	44.1	78.2	33.1
Qwen-2.5-32B	71.5	9.2	71.8	12.3

Besides, we report the reward recall (**Rec**) to quantify the proportion of actually non-compliant responses that are correctly identified. Critically, we find **recall is lower than precision for every LLM judge we evaluate, with gaps ranging from 11.8 to 62.3%** (Table 2). This indicates a systematic failure mode: *LLMs struggle to penalize violations*, especially under multi-constraint settings, creating a noisy reward signal that enables policy exploitation (reward hacking) rather than genuine learning.

Constraint Quantity Study. To quantify the LLM reward hacking in multi-constraint IF, we counted the total number of constraints (including both hard and soft) for each instance and used Qwen3-32B from Table 2 as a representative judge. Figure 3 reveals a pronounced downward trend in reward precision and recall as constraint density increases. **The model exhibits decent performance under a single constraint, but adding a second constraint** ($1 \rightarrow 2$) **causes a substantial drop in recall and a smaller drop in precision.** For example, in soft constraint evaluations, **Prec** decreases by 2.5%, whereas **Rec** decreases by 15.6% when adding a second constraint. This degradation suggests that LLM-derived rewards suffer severely from multi-constraint complexity, indicating that current OSS LLMs are not yet robust to such complex scenarios.

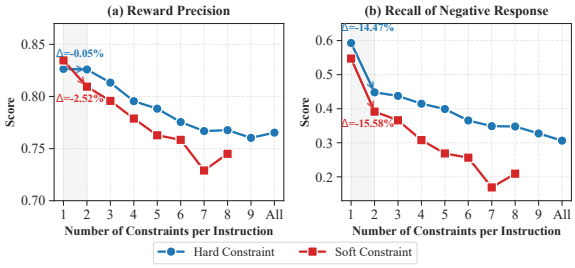


Figure 3: The reward reliability of LLM-judge under increasing constraints, showing clear degradation in reward Precision (**Prec.**) and Recall (**Rec.**) as constraint diversity grows.

Key Takeaways

- **LLM judges lack reward precision.** Most exhibit a substantial gap compared to code-based verifiers, rendering the cost-effective OSS judges we evaluate unreliable as direct reward models for multi-constraint RLVR.
- **Multi-constraint complexity exposes failure modes in LLM judges.** Reward recall drops sharply—and precision more modestly—from single to multi constraints, revealing a systematic inability to penalize violations, which enables reward hacking during policy training.

4.2 Decoupling Precision and Diversity

Impact of Reward Precision. We simulate varying levels of reward noise during hard-only training by injecting random false positives: with probability p, the reward is forcibly set to 1 regardless of actual compliance. As shown in Figure 4, test performance declines monoton-

ically as p increases beyond 10%, confirming reward precision as a critical factor. Interestingly, when $p \leq 10\%$, the model shows no degradation, and even improves slightly on IFEval. We attribute this robustness to two factors: (1) low-frequency noise preserves relative rankings essential for GRPO's group-based optimization, and (2) it encourages exploration beyond known solution paths (Yue et al., 2025).

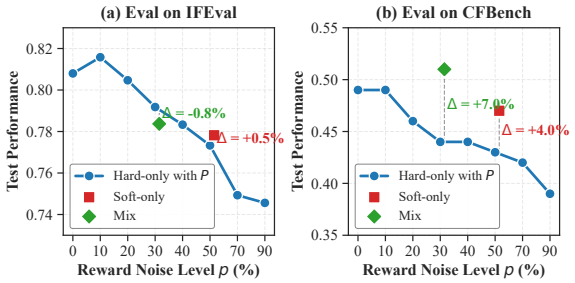


Figure 4: Impact of reward precision by injecting random noise. It shows reward noise is the critical factor of test performance.

To compare fairly across training regimes, we estimate the effective noise level of *Soft-only* and *Mix* training by matching their recall gap to the code verifier (Table 2): $p^* := \mathrm{Rec}_{\mathrm{code}} - \mathrm{Rec}_{\mathrm{judge}}$, i.e., the fraction of non-compliant responses that an LLM judge fails to penalize relative to a code verifier—precisely the failure mode that random false-positive injection at rate p emulates. At matched p^*, all three regimes achieve nearly identical in-distribution performance ($\Delta \leq 0.5\%$ on IFEval), demonstrating that *reward precision, not data diversity, drives ID generalization*. Furthermore, when hard-only training is subjected to high noise ($p = 50\%$), its OOD performance on CFBench also degrades significantly, confirming that the OOD advantage of hard-only training stems from its superior reward reliability rather than an inherent limitation of the data type.

Impact of Constraint Diversity. We aim to isolate the effect of constraint diversity on model performance, independent of reward precision. To this end, we progressively scale up the training data, which inherently increases the number of constraints per instruction while keeping the reward setting fixed. As shown in Figure 5, however, we do not observe a consistent improvement in test performance as the number of constraints increases: performance improves noticeably up to a low average constraint count, but further increases lead to a slight decline. Once a minimal threshold of diversity is reached, adding more constraints yields negligible additional benefit under a controlled reward.

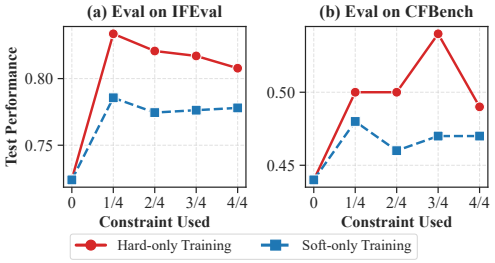


Figure 5: Impact of constraint diversity with using limited constraints. No consistent improvement is observed as constraints increase.

Key Takeaways

- **Confounded benefits.** The apparent gains from constraint diversity are largely overstated, as they are entangled with the noisy rewards introduced by the LLM judge.
- **Precision over diversity.** Reward precision, not constraint diversity, is the primary driver of RLVR performance; improving precision is more effective than scaling constraints.

5 METHODS

We propose **Hard + 1 Soft** (H1S), a simple data-composition strategy for RLVR training that retains all hard constraints while pruning soft constraints to at most one per instruction.

Motivation. Our diagnostic studies (§3, §4) reveal a consistent pattern: *LLM judges are most reliable under single-constraint evaluation but degrade sharply when a second constraint is introduced* (Figure 3). Specifically, adding a second soft constraint reduces reward precision by 2.5% and recall by 15.6%, with further degradation as constraint count increases. This unreliability enables policies to exploit judge biases, inflating training rewards without genuine capability gains (Figure 2).

Table 3: Evaluation results on the five IF benchmarks. Parentheses indicate scores on the `soft-only` subset for CFBench and FollowBench. *H1S* outperforms base models and ablation variants in 9 of the 10 benchmark–scale settings, the sole exception being FollowBench at 7B, where it trails *Mix* by 0.19%. ↑ denotes absolute improvement over the base instruction-tuned model. "–" indicates unavailable results from original papers;

Model	♣IFEval	♣Multi-IF	♣IFBench	♠CFBench	♠FollowBench	**Average**
Gemini-3.0-Pro	95.56	83.10	68.70	70.2	84.00	80.27
Claude-Sonnet-4.5	91.13	81.57	44.55	63.1	79.46	71.94
DeepSeek-V3.2	91.87	74.14	45.91	66.9	79.76	71.74
Qwen3-32B	87.06	70.70	25.17	64.2	69.61	63.31
IF-RLVR	87.80	–	53.70	–	–	–
RECAST	74.01	–	–	–	63.23	–
Qwen-IF	78.90	–	–	52	63.80	–
Qwen2.5-7B-inst	72.46	51.05	28.91	44.0 (47)	61.40 (65.46)	51.56
w/ *Hard-only*	80.78	58.89	31.63	49.3 (54)	68.96 (76.90)	57.85
w/ *Soft-only*	77.82	54.44	27.89	47.6 (52)	68.73 (76.12)	55.18
w/ *Mix*	78.37	58.25	29.59	51.2 (55)	**69.07** (**77.50**)	57.26
w/ *Hard*+1*Soft*	**85.25**$_{\uparrow 12.8}$	**66.70**$_{\uparrow 15.7}$	**38.13**$_{\uparrow 9.2}$	**55.1**$_{\uparrow 11.1}$(**56**)	68.88$_{\uparrow 7.5}$(76.21)	**62.81**$_{\uparrow 11.3}$
Qwen2.5-32B-inst	81.70	64.45	33.67	57.7(60)	73.06 (80.32)	61.98
w/ *Hard-only*	84.10	68.59	35.71	60.5(62)	75.12 (82.69)	64.70
w/ *Soft-only*	82.99	66.04	30.95	58.6(61)	74.19 (82.34)	62.43
w/ *Mix*	83.55	68.87	37.75	60.7(62)	74.99 (81.90)	65.03
w/ *Hard*+1*Soft*	**86.72**$_{\uparrow 5.0}$	**74.23**$_{\uparrow 9.8}$	**41.87**$_{\uparrow 8.2}$	**63.5**$_{\uparrow 5.8}$(**63**)	**76.95**$_{\uparrow 3.9}$(**83.0**)	**68.65**$_{\uparrow 6.7}$

Since reward precision, not constraint diversity, drives RLVR performance (§4.2), H1S confines LLM-based evaluation to this single-constraint regime, preserving soft guidance while avoiding multi-constraint reward hacking.

Method Overview. For each training instance we **retain all hard constraints and prune soft constraints to at most one**, yielding a hybrid reward structure in which the two constraint types play complementary roles: hard constraints supply high-precision, code-verifiable supervision, while the single retained soft constraint adds LLM-judged flexibility without exposing the reward signal to multi-constraint judge bias.

Formally, for an instruction x with hard-constraint set $C_{\text{hard}}(x)$ and soft-constraint set $C_{\text{soft}}(x)$, the pruned constraint set is:

$$C'(x) = C_{\text{hard}}(x) \cup \Pi\big(C_{\text{soft}}(x)\big), \qquad \big|\Pi\big(C_{\text{soft}}(x)\big)\big| \le 1, \tag{3}$$

where the reward is computed as strict binary over $C'(x)$: $R(x, y) = \mathbf{1}[\forall c \in C'(x), V_c(x, y) = 1]$.

Soft-Constraint Reduction. We realize this composition via a pruning operator Π that maps $C_{\text{soft}}(x)$ to a singleton subset whenever $|C_{\text{soft}}(x)| \ge 2$, i.e., retain a single constraint c^* and discard the rest:

$$\Pi\big(C_{\text{soft}}(x)\big) = \begin{cases} \{c^*\}, & c^* \sim \text{Uniform}\big(C_{\text{soft}}(x)\big), \quad |C_{\text{soft}}(x)| \ge 2, \\ C_{\text{soft}}(x), & |C_{\text{soft}}(x)| \le 1. \end{cases} \tag{4}$$

Hard constraints are left untouched, as code verifiers maintain near-perfect precision (Table 2). This confines every LLM-based reward assessment to the *single-constraint regime* where judge reliability is highest (Figure 3), suppressing the multi-constraint biases that enable reward hacking and yielding a reward signal that aligns more faithfully with true constraint satisfaction.

6 Experiments

6.1 Setup

Baselines. We compare against two groups of baselines. (1) *Leading LLMs*: Gemini-3.0-Pro, Claude-Sonnet-4.5, DeepSeek-V3.2, and Qwen3 (8B/32B), representing frontier and open-source

Table 4: Evaluation of general capabilities and cross-domain instruction following.

Model	Math	Knowledge	Writing	Cross-Domain IF	
	GSM8K	MMLU	WritingBench	MultiChallenge	AgentIF
Qwen2.5-7B	92.1	74.5	5.9	16.5	56.9
IF-RLVR	15.3$_{\downarrow 76.8}$	59.4$_{\downarrow 15.1}$	-	-	-
Qwen-IF	-	-	5.8$_{\downarrow 0.1}$	-	-
Hard+1*Soft*-7B	**92.3**$_{\uparrow 0.2}$	**73.1**$_{\downarrow 1.4}$	**6.0**$_{\uparrow 0.1}$	**19.1**$_{\uparrow 2.6}$	**58.6**$_{\uparrow 1.7}$
Qwen2.5-32B	95.4	78.0	6.1	21.6	60.6
Hard+1*Soft*-32B	**95.8**$_{\uparrow 0.4}$	**80.9**$_{\uparrow 2.9}$	**6.4**$_{\uparrow 0.3}$	**24.9**$_{\uparrow 3.3}$	**62.1**$_{\uparrow 1.5}$

instruction-following capabilities. (2) *RLVR-trained IF models* on the same base architecture: IF-RLVR (Pyatkin et al., 2025), RECAST (Guo et al., 2025), and Qwen-IF (Ren et al., 2025), which adopt diverse reward strategies (e.g., LLM judges, rule-based verifiers) and constraint compositions. All reported numbers follow the original papers or official checkpoints; details are provided in Appendix §F.2.

Evaluation setup. For soft-constraint evaluation, we employ GPT-4o as the judge, strictly adhering to the benchmarks' official protocols (e.g., pointwise constraint decomposition and chain-of-thought reasoning). While LLM-based rewards can be noisy during training (§4), benchmark evaluation employs stronger models and more rigorous protocols to ensure reliability, as detailed in Appendix §D.

6.2 Main Results

Instruction-Following Performance. Table 3 reports the results. Our *H1S* models consistently outperform the base model, with average gains of 11.3% (7B) and 6.7% (32B). Under identical training settings, *H1S*-7B surpasses *Mix*-7B by 5.55% despite using fewer constraint types. This advantage extends to OOD soft-constraint subsets: *H1S* leads on the CFBench soft-only subset at both scales (56.0/63.0 vs. 55.0/62.0 for *Mix*) and on the FollowBench soft-only subset at 32B (83.0 vs. 81.9), while trailing *Mix* on FollowBench at 7B (76.21 vs. 77.50); precision-driven training therefore transfers to soft constraints, though the smaller scale does not fully close the gap. Notably, *H1S*-32B comes within $\sim 3.3\%$ of top-tier models such as DeepSeek-V3.2 and Claude-Sonnet-4.5, demonstrating that high-precision proxy training is a competitive alternative to LLM alignment.

General Capabilities Performance. Beyond IF, we assess general proficiency across three dimensions: mathematics (GSM8K (Cobbe et al., 2021)), general knowledge (MMLU (Hendrycks et al., 2021)), and writing (WritingBench (Wu et al., 2025)). We further evaluate robustness in practical multi-dimensional scenarios via Cross-domain IF benchmarks: multi-turn long-context dialogue (MultiChallenge (Deshpande et al., 2025)) and autonomous agent environments (AgentIF (Qi et al., 2025)). As shown in Table 4, published RLVR-trained IF models degrade sharply on general tasks (e.g., IF-RLVR drops 76.8% on math), whereas *H1S* largely preserves general abilities while remaining competitive on cross-domain IF; since these baselines are trained with different recipes and data, the comparison is indicative rather than controlled. This suggests that our method transfers beyond the trained constraint types rather than overfitting to narrow IF patterns (Appendix §E).

6.3 Training Efficiency Analysis.

Figure 6 shows the training curves and wall-clock costs of *H1S*-32B under identical settings. The *Hard-only* model is most efficient, relying solely on code-based verifiers; the *Soft-only* and *Mix* baselines incur high LLM-reward costs, averaging 16.92 and 19.63 minutes per step, whereas our method requires only 8.23 minutes. This cuts per-step latency by 51% relative to *Soft-only* and 58% relative to *Mix*, and H1S's reward distribution closely matches the *Hard-only* baseline, confirming a high-precision reward signal. Soft-constraint reduction thus improves reward precision while drastically reducing verification overhead.

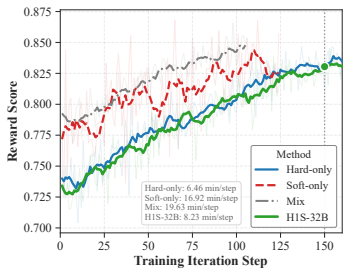


Figure 6: Training curves with wall-clock costs on Qwen2.5-32B-inst. *H1S*-32B cuts overhead and reward score aligns with *Hard-only* baseline.

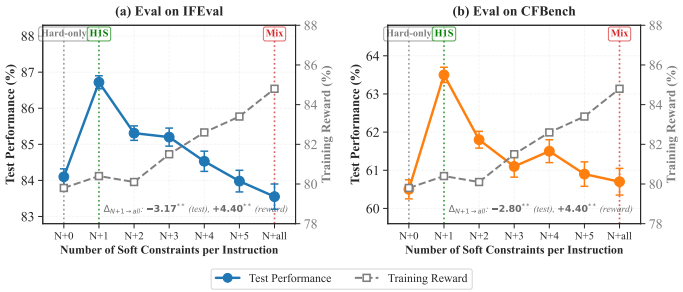


Figure 7: Ablation on soft-constraint budget k. Test performance follows an inverted-V pattern, peaking at *N+1* (our method *H1S*); training reward rises monotonically, confirming reward hacking beyond $k = 1$.

6.4 Ablation on Soft-Constraint Budget.

Figure 7 sweeps the number of soft constraints per instruction from 0 (i.e., *Hard-only*) to all (i.e., *Mix*), while keeping all hard constraints intact. All variants use the same number of training instances; only constraint composition differs. Each setting is trained with three random seeds and we report the mean over runs. Test performance follows an inverted-V pattern, peaking at *N+1*—our method *H1S*—with IFEval at 86.72% and CFBench at 63.5%. Both extremes are suboptimal: *N+0* under-covers soft constraint types, while *N+all* falls victim to reward hacking. Strikingly, the training reward rises monotonically from 79.80% at *N+0* to 84.83% at *N+all*, in the *opposite* direction to test performance. This divergence confirms that additional soft constraints merely give the policy more surface to exploit LLM-judge biases, inflating the reward without—and at the cost of—genuine capability. The optimal operating point is therefore $k = 1$, at the precision–diversity sweet spot that avoids both the under-coverage of *Hard-only* and the reward hacking of *Mix*, *confirming that high-precision rewards are the true enabler of effective RLVR training*. Since all settings share a single scale and three seeds, we interpret the inverted-V shape as ranking evidence rather than a precise optimum.

6.5 Discussion: Training vs. Evaluation Reliability

A critical question arises: *If LLM-based rewards are noisy during training, why are they considered reliable for benchmark evaluation?* We attribute this reliability gap to three differences. (1) **Judge Capability**: Training relies on cost-effective open-source judges (e.g., Qwen3-32B), whereas benchmarks use top-tier proprietary models (e.g., GPT-4o) with higher human alignment. (2) **Evaluation Protocol**: Benchmarks use rigorous protocols such as pointwise decomposition and CoT reasoning, which are designed to maximize LLM judge precision (as shown in §4), rather than the batch-scoring used in training. (3) **Error Propagation vs. One-Shot Assessment**: Training amplifies even minor systematic reward biases through thousands of gradient updates, leading to reward hacking, whereas evaluation is a one-shot measurement where errors do not compound. Empirical validation, including a multi-judge rank correlation study across top-tier LLMs, is provided in Appendix §D.1.

7 Conclusion

Current RLVR for instruction following suffers from reward hacking when scaling diverse soft constraints. We address this with H1S, which retains all code-verifiable hard constraints while limiting soft constraints to one per instruction. H1S outperforms mixed training on average, cuts training latency, and preserves general capabilities; we therefore advocate prioritizing reward engineering precision over naively scaling constraint diversity. We hope this motivates precision-aware reward design in other RLVR settings where verifiers remain noisy.

AI USE STATEMENT

In this work, we used generative AI tools (Gemini3.0-Pro) for editing and polishing manuscript text, including grammar correction, phrasing refinement, and structural feedback on drafts. We have not used generative AI tools for generating research ideas, designing experiments, writing code, producing figures or tables, or drafting mathematical proofs, and data collection, model training, and benchmark evaluation are not applicable to AI-assisted generation in this work.

We have reviewed all AI-assisted work. All edited text was manually verified by at least two authors against the original drafts to ensure factual accuracy, logical consistency, and alignment with our experimental evidence. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To ensure reproducibility, we have provided comprehensive documentation and resources throughout the paper and supplementary materials. Complete experimental configurations, including hyper-parameters, training schedules, and hardware specifications, are detailed in Appendix§F. Human annotation protocols for reward reliability evaluation (Table2) and inter-annotator agreement statistics are provided in Appendix§G. All benchmark evaluation scripts follow official protocols as cited in §6, with specific prompt templates and judge configurations documented in Appendix§C. Anonymous source code, including the GRPO training framework, H1S data composition module, and evaluation harness, has been submitted as supplementary material and will be released upon acceptance.

LIMITATIONS

Despite promising results, this work presents limitations that warrant further investigation.

- **Uniform Sampling Trade-off.** The pruning operator uses uniform sampling to avoid selection bias, but does not prioritize constraint difficulty or importance. In low-epoch regimes, critical soft constraints may be underrepresented. Adaptive sampling could improve efficiency but risks reintroducing bias.
- **Evaluation Proxy Gap.** Training uses open-source LLM judges while evaluation uses GPT-4o, creating a reward distribution asymmetry. Although multi-judge correlation supports reliability, absolute performance should be interpreted as relative rankings under a strong proxy rather than ground-truth measurements.

REFERENCES

Yapei Chang, Yekyung Kim, Michael Krumdick, Amir Zadeh, Chuan Li, Chris Tanner, and Mohit Iyyer. Bleuberi: Bleu is a surprisingly effective reward for instruction following. *arXiv preprint arXiv:2505.11080*, 2025.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024.

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

Muzhi Dai, Chenxu Yang, and Qingyi Si. S-grpo: Early exit via reinforcement learning in reasoning models. *arXiv preprint arXiv:2505.07686*, 2025.

Kaustubh Deshpande, Ved Sirdeshmukh, Johannes Baptist Mols, Lifeng Jin, Ed-Yeremai Hernandez-Cardona, Dean Lee, Jeremy Kritz, Willow E Primack, Summer Yue, and Chen Xing. Multichallenge: A realistic multi-turn conversation evaluation benchmark challenging to frontier llms. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 18632–18702, 2025.

Guanting Dong, Keming Lu, Chengpeng Li, Tingyu Xia, Bowen Yu, Chang Zhou, and Jingren Zhou. Self-play with execution feedback: Improving instruction-following capabilities of large language models. *arXiv preprint arXiv:2406.13542*, 2024.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv e-prints*, pp. arXiv–2407, 2024.

Mehdi Fatemi, Banafsheh Rafiee, Mingjie Tang, and Kartik Talamadupula. Concise reasoning via reinforcement learning. *arXiv preprint arXiv:2504.05185*, 2025.

Zhengkang Guo, Wenhao Liu, Mingchen Xie, Jingwen Xu, Zisu Huang, Muzhao Tian, Jianhan Xu, Yuanzhe Shen, Qi Qian, Muling Wu, Xiaohua Wang, Changze Lv, He-Da Wang, Hu Yao, Xiaoqing Zheng, and Xuanjing Huang. Recast: Expanding the boundaries of llms' complex instruction following with multi-constraint data, 2025. URL https://arxiv.org/abs/2505.19030.

Yun He, Di Jin, Chaoqi Wang, Chloe Bi, Karishma Mandyam, Hejia Zhang, Chen Zhu, Ning Li, Tengyu Xu, Hongjiang Lv, et al. Multi-if: Benchmarking llms on multi-turn and multilingual instructions following. *arXiv preprint arXiv:2410.15553*, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

Yuxin Jiang, Yufei Wang, Xingshan Zeng, Wanjun Zhong, Liangyou Li, Fei Mi, Lifeng Shang, Xin Jiang, Qun Liu, and Wei Wang. Followbench: A multi-level fine-grained constraints following benchmark for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4667–4688, 2024.

Joongwon Kim, Anirudh Goyal, Aston Zhang, Bo Xiong, Rui Hou, Melanie Kambadur, Dhruv Mahajan, Hannaneh Hajishirzi, and Liang Tan. A systematic examination of preference learning through the lens of instruction-following. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 11062–11082, 2025.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, et al. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*, 2025.

OpenAI. Introducing GPT-5, 2025. URL https://openai.com/index/introducing-gpt-5/.

Hao Peng, Yunjia Qi, Xiaozhi Wang, Bin Xu, Lei Hou, and Juanzi Li. Verif: Verification engineering for reinforcement learning in instruction following. *arXiv preprint arXiv:2506.09942*, 2025.

Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hannaneh Hajishirzi. Generalizing verifiable instruction following. *arXiv preprint arXiv:2507.02833*, 2025.

Yunjia Qi, Hao Peng, Xiaozhi Wang, Amy Xin, Youfeng Liu, Bin Xu, Lei Hou, and Juanzi Li. AGENTIF: Benchmarking large language models instruction following ability in agentic scenarios. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025. URL https://openreview.net/forum?id=FLiMxTkIeu.

Qwen. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Qingyu Ren, Qianyu He, Bowei Zhang, Jie Zeng, Jiaqing Liang, Yanghua Xiao, Weikang Zhou, Zeye Sun, and Fei Yu. Instructions are all you need: Self-supervised reinforcement learning for instruction following. *arXiv preprint arXiv:2510.14420*, 2025.

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

Jiao Sun, Yufei Tian, Wangchunshu Zhou, Nan Xu, Qian Hu, Rahul Gupta, John Wieting, Nanyun Peng, and Xuezhe Ma. Evaluating large language models on controlled generation tasks. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3155–3168, 2023.

Zhi Rui Tam, Cheng-Kuang Wu, Yi-Lin Tsai, Chieh-Yen Lin, Hung-yi Lee, and Yun-Nung Chen. Let me speak freely? a study on the impact of format restrictions on performance of large language models. *arXiv preprint arXiv:2408.02442*, 2024.

Kimi Team and Yifan Bai. Kimi k2: Open agentic intelligence, 2025. URL https://arxiv.org/abs/2507.20534.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, SH Cai, Yuan Cao, Y Charles, HS Che, Cheng Chen, Guanduo Chen, et al. Kimi k2. 5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*, 2026.

Qwen Team. Qwq-32b: Embracing the power of reinforcement learning, March 2025. URL https://qwenlm.github.io/blog/qwq-32b/.

Kuan Wang, Alexander Bukharin, Haoming Jiang, Qingyu Yin, Zhengyang Wang, Tuo Zhao, Jingbo Shang, Chao Zhang, Bing Yin, Xian Li, et al. Rnr: Teaching large language models to follow roles and rules. *arXiv preprint arXiv:2409.13733*, 2024.

Zhaoyang Wang, Jinqi Jiang, Huichi Zhou, Wenhao Zheng, Xuchao Zhang, Chetan Bansal, and Huaxiu Yao. Verifiable format control for large language model generations. *arXiv preprint arXiv:2502.04498*, 2025.

Yuning Wu, Jiahao Mei, Ming Yan, Chenliang Li, Shaopeng Lai, Yuran Ren, Zijia Wang, Ji Zhang, Mengyue Wu, Qin Jin, and Fei Huang. Writingbench: A comprehensive benchmark for generative writing, 2025. URL https://arxiv.org/abs/2503.05244.

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

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in LLMs beyond the base model? In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=4OsgYD7em5.

Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. *arXiv preprint arXiv:2503.18892*, 2025.

Yirong Zeng, Yufei Liu, Xiao Ding, Yutai Hou, Yuxian Wang, Wu Ning, Haonan Song, Dandan Tu, Qixun Zhang, Yuxiang He, et al. Tinyjudge: Unverifiable constraint alignment via lightweight specialist ensembles. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 26198–26212, 2026.

Tao Zhang, Chenglin Zhu, Yanjun Shen, Wenjing Luo, Yan Zhang, Hao Liang, Fan Yang, Mingan Lin, Yujing Qiao, Weipeng Chen, et al. Cfbench: A comprehensive constraints-following benchmark for llms. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 32926–32944, 2025.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *CoRR*, 2023.

Xinyu Zhu, Mengzhou Xia, Zhepei Wei, Wei-Lin Chen, Danqi Chen, and Yu Meng. The surprising effectiveness of negative reinforcement in llm reasoning. *arXiv preprint arXiv:2506.01347*, 2025.

A Related Works

Instruction Following. Instruction following requires models to generate responses that satisfy complex user instructions. Adherence to complex constraints, particularly those that involve a greater number and variety of conditions, remains a challenge for LLMs (Zhang et al., 2025; Tam et al., 2024; Sun et al., 2023). Earlier works favored sophisticated data synthesis, such as self-dialogue in AutoIF (Dong et al., 2024), rule extraction in RNR (Wang et al., 2024), or verifiable data generation in VFF (Wang et al., 2025), to scale supervised fine-tuning or DPO (Kim et al., 2025; Jiang et al., 2024; Chung et al., 2024; Pyatkin et al., 2025). The prevailing consensus, reflected in benchmarks like FollowBench (Jiang et al., 2024) and CFBench (Zhang et al., 2025), assumes that scaling constraint diversity (mixing hard and soft constraints) is essential for generalization. This view continues to shape the emerging RLVR training paradigm (Peng et al., 2025).

Reinforcement Learning with Verifiable Rewards. This approach has attracted growing attention for its effectiveness in incentivizing reasoning in LLMs with rule-based rewards (Yue et al., 2025; Zhu et al., 2025; Dai et al., 2025; Team & Bai, 2025). This paradigm has proven highly effective in reasoning domains, such as math and coding (Zeng et al., 2025; Fatemi et al., 2025; Liu et al., 2025). However, the application of RLVR has largely been confined to tasks with explicit ground truth. Driven by this diversity-centric view, researchers have turned to LLM-as-a-judge to provide reward signals for extending to unverifiable soft constraints (Pyatkin et al., 2025; Guo et al., 2025; Peng et al., 2025; Lambert et al., 2024). In contrast, our work challenges this trend by highlighting the primacy of reward precision: we show that strict alignment with high-precision constraints alone is sufficient to cultivate robust IF meta-skills, given reliable verifiers.

B Discussion and Future Works

Reward Precision vs. Diversity. We argue that expanding diversity is beneficial primarily when premised on high reward precision, a prerequisite that prior methods may have overlooked. Scaling constraint diversity pays off only when reward precision is sufficiently high. Low reward precision renders high constraint diversity counterproductive.

Therefore, a promising direction for future work is to explore frameworks that simultaneously achieve high precision and high diversity to enhance robustness. Potential pathways include: (i) **Mitigating Reward Hacking**: using stronger models or training dedicated reward models to maintain precision while scaling diversity. (ii) **Alternative Metrics**: Replace LLM-based soft constraints with deterministic strategies. For example, Chang et al. (2025) propose using string-matching metrics like BLEU as an effective substitute for LLM judges.

Table 5: Statistics and characteristics of the instruction following and cross-domain benchmarks.

Dataset	Samples	Constraint	Key Features	Eval. Method
IF Benchmarks				
IFEval (Zhou et al., 2023)	500	hard / 25	Verifiable, Objective	Code-based
IFBench (Pyatkin et al., 2025)	300	hard / 58	Unseen Constraint	Code-based
Multi-IF (He et al., 2024)	4,501	hard / 25	Multi-turn, Multilingual	Code-based
FollowBench (Jiang et al., 2024)	820	mix / 5	Multi-level Difficulty	Code + LLM
CFBench (Zhang et al., 2025)	1,000	mix / 25	Complex Scenarios	LLM
Cross-Domain Benchmarks				
AgentIF	707	-	Agentic Scenarios	Code + LLM
MultiChallenge	273	-	Multi-turn Context	LLM

C Benchmark and Evaluation Details

C.1 IF Benchmarks

To comprehensively assess the instruction-following capabilities of LLMs, we utilize five distinct benchmarks. These datasets cover hard and soft constraints, and also include generalization to unseen

constraints, multi-level difficulty, and multi-turn multilingual interactions. Table 5 summarizes the statistics of these datasets.

IFEval (Zhou et al., 2023) is a widely adopted benchmark designed to evaluate the ability of LLMs to follow objective and verifiable instructions. It consists of around 500 prompts containing 25 types of verifiable constraints (e.g., word count limits, formatting requirements).

Multi-IF (He et al., 2024) extends the scope of instruction following to multi-turn and multilingual settings. It contains 4,501 samples spanning 8 languages. The data set is constructed by expanding verifiable single-turn instructions into coherent three-turn dialogs. It serves as a stress test for maintaining instruction adherence over long contexts and across diverse linguistic distributions.

IFBench (Pyatkin et al., 2025) addresses the issue of model overfitting to common instruction datasets. It focuses on evaluating the generalization capabilities of models by introducing 58 novel, unseen, and challenging verifiable constraints. It employs strict code-based verification modules to measure performance on out-of-domain instructions.

FollowBench (Jiang et al., 2024) evaluates the robustness of LLMs through a multi-level difficulty mechanism. It contains 820 instructions across five fine-grained categories (Content, Situation, Style, Format, Example). The benchmark is constructed by incrementally adding constraints to seed instructions, creating a difficulty gradient (Level 1 to Level N).

CFBench (Zhang et al., 2025) is a benchmark comprising 1,000 high-quality samples derived from the real world. It features a hierarchical taxonomy with 10 major constraint categories (e.g., style, logical rules, numerical) and over 25 subcategories. We report the average results across all samples. To ensure consistency with prior work, we use the official evaluation code, which retains only integers; in the main tables we report the mean over 5 runs, while the parenthesized `soft-only` subset scores follow the benchmark's integer output.

To rigorously test the generalization from hard to soft constraints without contamination from verifiable rules, we performed a strict filtering of the benchmark datasets (denoted as FollowBench-Soft and CFBench-Soft):

- **FollowBench-Soft**: We isolated the purely semantic categories: Content, Situation, and Style. We explicitly excluded the Format and Mixed categories, as these often contain implicit hard constraints (e.g., specific output structures) that could bias the evaluation.
- **CFBench-Soft**: We removed all instances involving numerical constraints (e.g., word counts, paragraph limits), ensuring that only unverifiable semantic requirements remained.

Cross-Domain Benchmarks Furthermore, to evaluate the model's robustness in practical, multi-dimensional scenarios, we utilize cross-domain benchmarks that examine instruction following across various domains.

AgentIF (Qi et al., 2025) addresses the challenge of following long and complex instructions in agentic scenarios. It consists of 707 manually annotated samples sourced from real-world agent applications, containing over 8,400 constraints.

MultiChallenge (Deshpande et al., 2025) focuses on evaluating frontier LLMs in realistic, complex multi-turn conversations. It features extended dialogues (up to 10 turns) designed to stress-test models across four critical dimensions: long-term instruction retention, inference and memory of scattered user information, reliable versioned editing, and contextual self-coherence.

D Reliability of LLM Judges in Benchmark Evaluation

A critical question arises: *If LLM-based rewards are noisy during training, why are they considered reliable for benchmark evaluation?* We argue that the reliability gap stems from three fundamental differences between the training and evaluation protocols: **Judge Capability**, **Evaluation Protocol**, and **Error Propagation**.

1. Judge Capability and Cost Constraints. During RLVR training, using state-of-the-art closed-source models (e.g., GPT-4o) as reward models is prohibitive due to high latency and API costs,

forcing us to rely on smaller open-source models (e.g., Qwen3-32B) which exhibit lower precision (Table 2). In contrast, standard benchmarks (e.g., FollowBench (Jiang et al., 2024), CFBench (Zhang et al., 2025)) employ top-tier proprietary models (e.g., GPT-4o, Claude-4.5-Sonnet) as judges. As shown in Table 2, these advanced models achieve significantly higher agreement with human annotations (e.g., Gemini-2.5-Pro achieves 86.3% precision on soft constraints), providing a much more trustworthy signal for evaluation.

2. Robust Evaluation Protocols. Training rewards are typically computed via simple batch prompts to maximize throughput, which exacerbates biases in multi-constraint scenarios (Figure 3). Benchmark evaluations, however, employ rigorous protocols to mitigate judge errors:

- **Pointwise Decomposition:** Complex instructions are broken down into atomic constraints, and each is evaluated independently. This aligns with our finding in §4 that LLMs perform best under single-constraint settings.
- **Chain-of-Thought (CoT) Reasoning:** Judges are prompted to generate step-by-step rationales before assigning scores, reducing impulsive biases.
- **Self-Consistency:** Some benchmarks use multiple sampling paths to aggregate judgments, further smoothing out random noise.

These strategies ensure that the evaluation signal is far more precise than the raw training rewards.

3. Error Accumulation vs. Independent Noise. The impact of judge errors differs fundamentally between the two phases. In **training**, systematic biases in reward signals can lead to *reward hacking*, where the policy model exploits judge weaknesses to inflate scores without improving actual capability, causing error accumulation and performance degradation (as seen in Figure 2). In **evaluation**, each judgment is an independent event. While individual judgments may contain noise, these errors are largely random and tend to cancel out over large test sets. Crucially, evaluation errors do not feed back into the model's parameters, preventing the catastrophic failure modes observed in training.

Why not apply benchmark protocols to training? While these robust evaluation strategies improve reliability, they are impractical for RLVR training due to: (1) **Latency:** Pointwise CoT evaluation is orders of magnitude slower than batch scoring, creating a bottleneck in the RL rollout phase. (2) **Cost:** Using GPT-4o for millions of training steps is financially unfeasible. (3) **Data Scale:** High-quality, human-verified test sets are small and static, whereas training requires diverse, large-scale, and dynamically generated data.

Therefore, while benchmark scores provide a high-confidence estimate of model capability, they reflect a different operational regime than training. Our method (H1S) specifically addresses the *training-phase* reliability issue by simplifying the reward structure, ensuring that the model learns from high-precision signals even when using cost-effective judges.

D.1 Multi-Judge Consistency Analysis

To address the potential concern that LLM-based evaluation may inherit the same reliability issues as LLM-based training rewards, we design a controlled experiment to quantify the consistency of benchmark evaluation across different SOTA judges.

Experimental Setup. We select five model variants derived from Qwen2.5-7B-Instruct: the base model, *Hard-only*, *Soft-only*, *Mix*, and our proposed *Hard*+1*Soft*. These models are evaluated on two soft-constraint benchmarks: FollowBench and CFBench. Three API-based advanced LLMs are employed as independent judges: (1) GPT-4o, (2) Claude-4.5-Sonnet, (3) Gemini-2.5-Pro. All judges follow the official evaluation protocols of each benchmark, including pointwise constraint decomposition and chain-of-thought reasoning, to ensure fair comparison.

Metrics. We adopt Spearman's rank correlation ρ: measures the monotonic relationship between two judges' score rankings. ρ ranges from -1 to 1, with values closer to 1 indicating stronger agreement. We do not report Kendall's τ: with only five model variants it is quantized to the coarse grid $\{0, 0.2, 0.4, 0.6, 0.8, 1.0\}$ and therefore cannot discriminate among the three judge pairs. We compute ρ over per-instance judge scores (N=820 for FollowBench, N=1,000 for CFBench) rather

than over the five model-level means; the latter are rank-identical across judges (Table 7) and would therefore give $\rho = 1.00$ by construction.

Table 6: Spearman's rank correlation ρ between three SOTA LLM judges on soft-constraint benchmarks. All pairwise correlations exceed 0.9, indicating that the relative performance ranking of models is highly consistent across different evaluators.

Judge Pair	**Spearman** ρ	
	FollowBench	CFBench
GPT-4o vs. Claude-4.5-Sonnet	0.90	0.91
GPT-4o vs. Gemini-2.5-Pro	0.93	0.95
Claude-4.5-Sonnet vs. Gemini-2.5-Pro	0.95	0.93
Average	**0.93**	**0.93**

Table 7: Soft-constraint subset scores (%) assigned by three independent LLM judges on CFBench. Numbers in parentheses indicate the rank under each judge. All five models maintain identical rankings across all three judges, confirming that the evaluation conclusion on CFBench is robust to judge selection.

Model	**GPT-4o**	**Claude-4.5-Sonnet**	**Gemini-2.5-Pro**
Qwen2.5-7B-inst	47.0 (5)	46 (5)	47 (5)
w/ *Soft-only*	52.0 (4)	51 (4)	51 (4)
w/ *Hard-only*	54.0 (3)	53 (3)	54 (3)
w/ *Mix*	55.0 (2)	54 (2)	55 (2)
w/ *Hard*+1*Soft*	**56.0** (1)	**55** (1)	**56** (1)

Results. As shown in Table 6, all pairwise correlations between the three judges are remarkably high: Spearman's ρ above 0.9. This indicates that the relative ordering of model performance is nearly invariant to the choice of judge.

Furthermore, Table 7 presents the absolute scores assigned by each judge. Despite minor numerical differences (within 1.0 percentage point across judges), all five models maintain **identical rankings** under all three judges:

$$\textit{Hard}{+}1\textit{Soft} > \textit{Mix} > \textit{Hard-only} > \textit{Soft-only} > \textit{Base}.$$

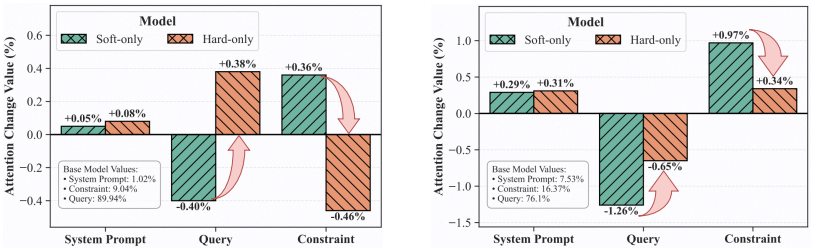
This consistency confirms that the superiority of our method is not an artifact of a specific judge's bias, but reflects genuine improvements in soft-constraint adherence.

Discussion. These results, combined with the structural differences between training-time rewards and benchmark evaluation (i.e., judge capability, evaluation protocol, and error propagation), provide strong evidence that LLM-based benchmark evaluation is substantially more reliable than LLM-based training rewards. The high inter-judge agreement further suggests that the evaluation signal is dominated by genuine model capability rather than judge-specific noise.

E Why Does High-precision Reward Training Generalize?

To provide mechanistic insights into why training on narrow hard constraints (with high-precision rewards) generalizes better than broad soft constraints, we investigate the shifts in average attention patterns across the final five layers of the Base, *soft-only*, and *hard-only* models.

In practice, we quantify the average attention weights of the model assigned to different prompt segments (i.e., system prompt, query, and constraint) and report the average weights in all instances (results illustrated in Figure 8). On IFEval, we observe that the Soft-only model shifts its focus, decreasing its attention to the raw query by 0.4% while increasing its attention to constraints by 0.36%. In contrast, the hard-only model increases the attention to the query by 0.38% while reducing the focus on constraints by 0.46%.



(a) Attention shift on IFEval. (b) Attention shift on CFBench.

Figure 8: The attention weight distribution shift relative to the base model. Each panel reports two groups of per-segment differences—*soft-only* − base and *hard-only* − base—on IFEval (a) and CFBench (b). On IFEval the two trained models shift in opposite directions, whereas on CFBench they move in the same direction, with smaller hard-only shifts.

On CFBench both models shift in the same direction, with smaller hard-only shifts (-0.65% vs. -1.26% on the query, $+0.34\%$ vs. $+0.97\%$ on the constraint). These shifts are averaged over the final five layers and all instances and are small in absolute terms (at most 1.3 percentage points on either benchmark), so we read them as descriptive trends rather than causal evidence. Given that the soft-only model underperforms the hard-only model, we hypothesize that: (1) The soft-only model improves IF performance by allocating more attention to the constraint text, but at the cost of reduced attention to the raw query, leading to degraded generalization. (2) In contrast, the hard-only model enhances IF without increasing attention to the constraint, suggesting that it has internalized IF capability; the freed-up attention capacity is redirected to the raw query segment, thereby improving generalization. This pattern is consistent with the view that, after training with high-precision rewards, the model has internalized the IF capability and acquired fundamental reasoning skills, thereby reducing the need to pay excessive attention to the constraint segment.

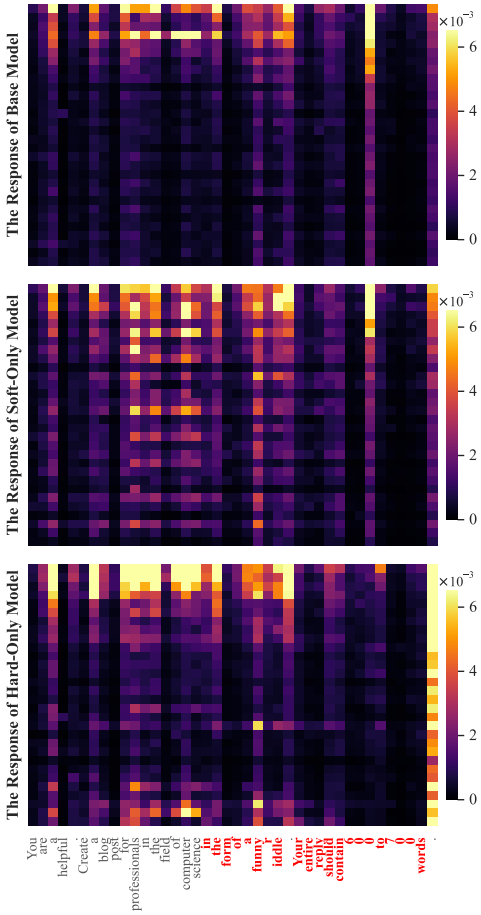


Figure 9: An attention heatmap analysis for a randomly selected sample from CFBench. The red text denotes constraint segment.

For the case study, Figure 9 provides a qualitative visualization of the attention maps for a randomly selected example. It shows that all models attend to the funny riddle (soft-constraint) region; the base and soft-only models show more attention on the numbers 600 to 700 (hard-constraint) than the hard-only model; but the hard-only model shows more attention on query segment. As a single illustrative case, this visualization is consistent with, but does not establish, the hypothesis that the hard-only model attends more sparsely to the trained constraint segments while maintaining attention on the original user query, and that the base and soft-only models distribute attention more diffusely.

Therefore, hard constraints serve as a high-precision proxy, internalizing the IF capacity. The model acquires a meta-skill that naturally transfers to unverifiable soft-constraint tasks, effectively learning how to follow rather than just what to follow.

F Detailed Experimental Results

Table 8: Evaluation results on five benchmarks. ♣ and ♠ denote hard-only constraints and mixed constraints (including soft constraints) benchmarks, respectively. Results show that on the **Pro* data hard-only training leads soft-only training on all five benchmarks (+3.6% on average) and matches the mixed variant on three of five (+0.9% on average); on the original data, hard-only training shows no clear advantage. We report the average accuracy over 5 runs and their standard deviation ($\mu \pm \sigma$). The *Soft-only* rows of *VerlIF* and *VerlIF-Pro* coincide because both models are trained on the same soft-only instances with the same seed.

Model	♣IFEval	♣Multi-IF	♣IFBench	♠CFBench	♠FollowBench	**Average**
*VerInstruct-Pro**						
Qwen2.5-7B	$72.46_{\pm 0.1}$	$51.05_{\pm 0.2}$	$28.91_{\pm 0.2}$	$44.0_{\pm 0.1}$	$61.40_{\pm 1.20}$	$51.56_{\pm 0.24}$
w/ *Hard-only*	$79.30_{\pm 0.2}$	$58.09_{\pm 0.08}$	$33.74_{\pm 0.4}$	$49.6_{\pm 0.1}$	$68.96_{\pm 0.56}$	$\mathbf{57.94}_{\pm 0.11}$
w/ *Soft-only*	$76.30_{\pm 0.09}$	$52.87_{\pm 0.06}$	$27.62_{\pm 0.3}$	$46.4_{\pm 0.1}$	$68.54_{\pm 1.35}$	$54.35_{\pm 0.27}$
w/ *Mix*	$76.63_{\pm 0.09}$	$57.27_{\pm 0.06}$	$30.61_{\pm 0.4}$	$51.4_{\pm 0.09}$	$69.24_{\pm 0.24}$	$57.03_{\pm 0.05}$
VerlIF						
Qwen2.5-7B	$72.46_{\pm 0.1}$	$51.05_{\pm 0.2}$	$28.91_{\pm 0.2}$	$44.0_{\pm 0.1}$	$61.40_{\pm 0.2}$	$51.56_{\pm 0.2}$
w/ *Hard-only*	79.67	56.62	31.29	46.3	62.85	55.34
w/ *Soft-only*	77.26	54.58	29.93	49.0	69.32	**56.02**
w/ *Mix*	77.08	54.84	30.27	47.2	67.28	55.33
VerlIF-Pro						
Qwen2.5-7B	$72.46_{\pm 0.1}$	$51.05_{\pm 0.2}$	$28.91_{\pm 0.2}$	$44.0_{\pm 0.1}$	$61.40_{\pm 0.2}$	$51.56_{\pm 0.2}$
w/ *Hard-only*	85.21	64.52	36.73	53.3	67.15	**64.98**
w/ *Soft-only*	77.26	54.58	29.93	49.0	69.32	56.02
w/ *Mix*	85.95	64.13	37.41	50.4	66.85	60.94

F.1 Robustness Analysis

In this section, we conduct supplementary analyses on additional training data and reward model to test the robustness of findings: (1) hard-only constraint training outperforms soft-only constraint trained models. (2) none of the OSS judges we evaluate is reliable enough to serve directly as an RLVR reward model.

F.1.1 Robustness Analysis of Training Data

To evaluate the sensitivity of our approach to training data composition and verifier quality, we conduct experiments across two datasets with varying hard/soft constraint ratios and verifier reliability levels.

Dataset Variants. The original *VerInstruct* (Peng et al., 2025) comprises 22k instances with a pronounced imbalance (77.7% soft-only vs. 22.3% hard-only) and relies on code verifiers that may contain implementation errors. To address these issues, we create *VerInstruct-Pro* by re-implementing all hard-constraint verifiers using Gemini-3.0-Pro to ensure correctness, and further downsample the soft-only subset to match the hard-only count, yielding the balanced variant *VerInstruct-Pro**. Similarly, we refine the hard-constraint verifiers in *VerlIF* (Ren et al., 2025) (20k samples, 60% soft) to produce *VerlIF-Pro*, while retaining the original version as a baseline.

Throughout, *Mix* refers to the original, unpruned pool of each dataset—i.e., its full mixture of hard- and soft-constraint instances—while *hard-only* and *soft-only* denote its hard and soft partitions. The two reward sources cover complementary constraint sets: code verifiers score every hard constraint, covering the *hard-only* subset as well as the hard-constraint portion of *Mix* instances, whereas the LLM judge scores every soft constraint, covering the *soft-only* subset as well as the soft-constraint portion of *Mix* instances. We refer the reader to the original papers for the distribution of constraint counts and types.

Results and Analysis. Table 8 reveals a critical interaction between verifier quality and training effectiveness. On the *original* data, hard-only training shows no clear advantage: it excels on hard benchmarks but underperforms on soft benchmarks, while soft-only training shows the opposite pattern. This contrasts sharply with results on the **Pro* data, where hard-only training leads soft-only training on all five benchmarks (+3.6% on average) and matches mixed training (+0.9% on average), the only exceptions being CFBench and FollowBench, where *Mix* leads by 1.8% and 0.3% respectively.

This discrepancy exposes a previously overlooked confounder: **both constraint types suffer from low reward precision in raw datasets, but for different reasons**. Hard constraints are plagued by *code-verifier noise* (false negatives due to buggy implementations), while soft constraints are vulnerable to *LLM reward hacking* (false positives due to judge biases). Under GRPO's optimization dynamics, false negatives are particularly detrimental as they suppress valid trajectories, whereas false positives merely introduce gradient noise that can be partially averaged out over groups. By leveraging top-tier LLMs to re-implement verifiers, our **Pro* refinement effectively eliminates false negatives in hard constraints, unleashing their true generalization potential. These results confirm that the efficacy of precision-first training is robust across dataset provenances and scales, provided that the underlying reward signal is sufficiently reliable.

This finding may also explain why prior work did not observe the superiority of hard-only training: buggy code verifiers in raw datasets introduced false negatives that masked the true generalization potential of hard constraints, making soft-constraint training appear competitively viable only because both reward signals were similarly compromised.

Statistical Test As reported in Table 8, our training pipeline exhibits high stability across multiple iterations. The standard deviations over five independent runs with different random seeds are consistently small, ranging from 0.05 to 1.35 percentage points across benchmarks (Table 8). This low variance demonstrates that the reported results for *VerInstruct-Pro** are robust to initialization noise and sampling stochasticity.

To further validate the superiority of hard-only strategy, we conducted independent t-tests to compare the *Hard-only* strategy against the *Soft-only* and *Mix* baselines. The results in Table 9 demonstrate that *Hard-only* constraints yield statistically significant improvements on all three hard-constraint benchmarks and on CFBench ($p < 0.05$); on FollowBench the gap to *Soft-only* is not statistically significant ($p = 0.22$), consistent with the comparable scores in Table 8. Specifically, on the ♣Multi-IF benchmark, the *Hard-only* strategy outperforms the *Soft-only* baseline with extreme significance ($p = 1.75 \times 10^{-7}$). Even when compared to the *Mix* strategy, which includes partial soft constraints, the *Hard-only* approach maintains a significant lead ($p < 0.05$ on every benchmark where both variants were evaluated, and $p < 0.01$ on IFEval), confirming that strict adherence to hard constraints is pivotal for enhancing model instruction-following capabilities.

Table 9: Statistical Test Evaluation results on five benchmarks. ♣ and ♠ denote hard-only constraints and mixed constraints benchmarks, respectively. P-values are calculated relative to the Hard-only strategy.

Model	♣IFEval	♣Multi-IF	♣IFBench	♠CFBench	♠FollowBench
VerInstruct-Pro*					
Hard vs Soft	0.003**	1.75e-7***	0.001**	0.002**	0.22
Hard vs Mix	0.008**	0.031*	0.041*	0.013*	0.003**
VerlIF-Pro					
Hard vs Soft	0.01**	1e-4***	0.001**	0.002**	-
Hard vs Mix	0.01**	1e-4***	0.001**	0.002**	-

Note: $^{*}P < 0.05$, $^{**}P < 0.01$, $^{***}P < 0.001$.

F.1.2 Robustness Analysis of Reward Model

We employ Qwen2.5-7B-Instruct as the base model. To analyze the scaling and reasoning capabilities of reward models, we select a diverse suite of LLMs to serve as judges, specifically: Qwen2.5-

32B(Qwen, 2025), QwQ-32B (Team, 2025), and Qwen3 series (Yang et al., 2025) (i.e., Qwen3-8B, Qwen3-32B, and Qwen3-235B-inst). During training, hard constraints are strictly enforced via a rule-based checker, while soft constraints are evaluated using an LLM-based reward judge.

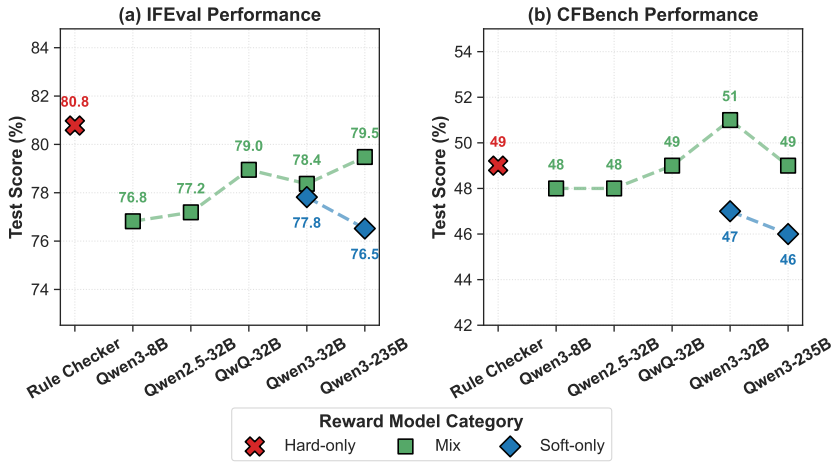


Figure 10: Test performance of the model under different reward models on IFEval (evaluating hard constraints) and CFBench (evaluating mixed constraints). Results show that: (1) the *hard-only* model outperforms the *soft-only* variant, and also performs comparably to the mixed-constraint model; and (2) a stronger reward model still fails to outperform the rule-based verifier.

As illustrated in Figure 10, the reward model trained exclusively on hard-only data outperforms its soft-only counterpart and remains on par with the mix variants (which incorporate both hard and soft constraints). For example, on the IFEval benchmark, the hard-only model (evaluated via Qwen3-32B) surpasses the soft-only and mix models by 3.0% and 2.4%, respectively. This finding also holds on the OOD benchmark CFBench. Furthermore, even scaling to more powerful open-source (OSS) reward models fails to bridge the gap with the rule-based checker (hard-only). These results highlight that naively employing OSS LLMs to reward unverifiable soft constraints is suboptimal and potentially harmful.

F.2 Setup Details

Implementation Details For reinforcement learning, we implemented GRPO based on the MindSpeed-RL[1] training framework. Each RL training run for the 7B model completed within 24 hours on a cluster of 64 Ascend 910b NPUs (configured as 8 nodes × 8 NPUs). For optimization stability, we incorporated KL divergence regularization with a coefficient of 0.001 using the low-variance KL implementation, while enabling gradient checkpointing for memory efficiency. The hyperparameters used are detailed in Table 10.

Baseline RLVR-trained Models We conduct comparisons against three representative recent RLVR-trained models to improve the instruction-following capabilities of language models, namely:

- RECAST (Guo et al., 2025), a framework that empowers models to handle complex, multi-constraint instructions by utilizing a verifiable data synthesis pipeline and RL with verifiable constraints.
- IF-RLVR (Pyatkin et al., 2025), training RLVR with multi-constraints per instance to enhance a model's ability to follow diverse and complex hard constraint instructions.
- Qwen-IF (Ren et al., 2025), a label-free self-supervised RL method that enhances IF by deriving reward signals directly from input instructions and utilizing constraint decomposition to address sparse rewards.

[1] https://gitcode.com/Ascend/MindSpeed-RL

Table 10: The configurations for RL training with GRPO.

Hyperparameter	Value
Data Configuration	
Global Batch Size	128
Max Prompt Length	2048
Max Response Length	6144
Micro Batch Size	4
Train Steps	600
Rollout Configuration	
Rollout Name	vllm
GPU Memory Utilization	0.6
Number of Rollouts	8
Temperature	1.0
Tensor Model Parallel Size	4
Top_P	1.0
RL Optimization	
Learning Rate	1e-6
LR Decay Style	constant
Mini Batch Size	128
KL Loss	0.001

G Details in Reward Reliability

Ground-Truth Label Curation. In this part, we establish ground-truth labels by human experts. To construct a gold-standard labels for validating automated reward signals, we curated a representative subset of 200 instances, equally split between verifiable hard-constraint tasks (sampled from IFEval) and semantic soft-constraint tasks (sampled from CFBench). We recruited three domain experts for blind evaluation. To quantify evaluation reliability, we calculated Fleiss' Kappa (κ) to measure Inter-Annotator Agreement (IAA). We observed near-perfect consistency for hard constraints ($\kappa = 0.91$) and substantial agreement for soft constraints ($\kappa = 0.78$). For instances exhibiting significant divergence (defined as conflicting binary verdicts), a designated senior lead researcher served as an adjudicator to determine the final ground truth. Thus, the final ground truth reflects this adjudicated consensus rather than a simple averaging of individual ratings, ensuring robust and reliable benchmark scores.