TinyJudge: Unverifiable Constraint Alignment via Lightweight Specialist Ensembles

Anonymous ACL submission

Abstract

Instruction Following (IF) is a core capability of LLMs, requiring strict adherence to diverse constraints, ranging from verifiable ones (e.g., output length) to unverifiable ones (e.g., tone). Reinforcement learning with verifiable rewards has emerged as a paradigm for IF tasks, leveraging LLM-as-a-judge to assess unverifiable constraints. However, we empirically find that this approach remains a significant bottleneck, suffering from severe reward hacking and higher computational overhead. In this work, we first analyze the generalization capabilities of unverifiable constraints and discover that specific constraints exhibit distinct, high-generalization patterns. Motivated by this, we propose TinyJudge, a framework that employs an ensemble of specialized tiny language models ($\sim 0.6B$) to provide rewards for soft constraints. By distilling expertise from frontier models into these tiny models, it achieves high-precision, lightweight evaluation. Extensive evaluations across five benchmarks demonstrate that TinyJudge outperforms the baselines by $\sim 10\%$ in average performance and 12% in reward precision. Crucially, it also achieves a $3\times$ speedup in total training time. Our work provides a scalable and robust path for aligning LLMs with unverifiable human instructions.

1 Introduction

IF evaluates a model's ability to adhere to specific constraints, typically categorized into hard constraints (verifiable via rigid rules, e.g., length limits) and soft constraints (unverifiable by rules and requiring semantic judgment, e.g., tone or style) (Tam et al., 2024; Zhou et al., 2023). Current research is increasingly shifting focus from hard-only constraints toward scaling diverse constraints to enhance model generalization across real-world scenarios (Jiang et al., 2024; Zhang et al., 2025).

Recently, Reinforcement Learning from Verifiable Rewards (RLVR), driving the development of advanced models (Guo et al., 2025a; OpenAI, 2025), has emerged as a dominant paradigm for enhancing instruction-following capabilities. To further this progress, the research community is now scaling RLVR by integrating heterogeneous constraints to achieve broad generalization (Ren et al., 2025; Peng et al., 2025). This strategy combines rule-based rewards for verifiable hard constraints with LLM-as-a-judge for unverifiable soft constraints (Guo et al., 2025b; Pyatkin et al., 2025). However, this prevailing strategy relies heavily on the assumption that LLM-as-a-judge acts as a reliable evaluator. This raises a fundamental question: Is the current LLM-as-a-judge paradigm truly effective and robust enough to serve as a precise reward signal for RL training?

To address this question, we investigate the effectiveness of LLM-as-a-judge for unverifiable

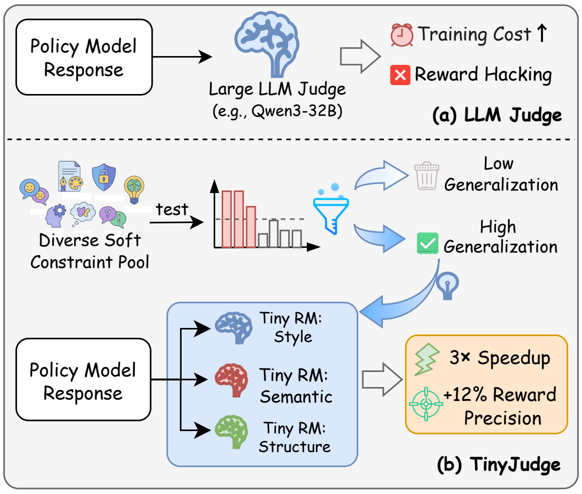


Figure 1: The challenges of current LLM-based reward in handling unverifiable soft constraints and our solution, TinyJudge. (a) Existing LLM-as-a-judge (**Top**): suffers from high computational overhead and reward hacking. (b) Motivated by insights from generalization analysis (**Middle**), we employ a limited number of tiny expert-level models as reward, achieving superior reward precision and speedup in judging (**Bottom**).

constraints in Section §3. Our empirical analysis reveals two critical issues: (1) Severe reward bias: When evaluating outputs against multiple constraints simultaneously, the LLM judge exhibits a significant tendency to overlook violations (i.e., failing to penalize errors), resulting in low reward precision. (2) Higher training overhead: Directly employing a frontier LLM as the reward model incurs a 3× increase in training time overhead. As illustrated in Figure 1, these issues severely restrict both the practicality and the performance ceiling of the method. Consequently, addressing these inefficiencies is a prerequisite for scaling RLVR to diverse, real-world IF tasks.

To this end, we first analyze the generalization capabilities of various unverifiable constraints. It reveals that some specific constraints exhibit significantly higher generalization than others. This suggests that decoupling evaluation, by assessing specific constraints individually rather than via a monolithic model for all constraints simultaneously, can mitigate reward bias. Motivated by this insight, we propose *TinyJudge*, a framework that employs tiny language models (e.g., 0.6B) to provide rewards exclusively for these high-generalization constraints. It distills expertise from frontier models into a specialized tiny model only for each high-generalization constraint. During RLVR training, reward signals are generated through a hybrid system: each unverifiable soft constraint is evaluated by the corresponding tiny expert model, while verifiable hard constraints are processed via code-based rules. This ensemble reward module provides high-precision, low-latency feedback to the policy model.

We evaluated our method across five IF benchmarks (e.g., IFEval and CFBench). Under identical experimental settings, TinyJudge achieves a ∼ 10% performance gain over the base model, while significantly outperforming other established baselines. The efficiency of our approach is equally notable. Compared to previous LLM-based reward systems, our method improves reward precision by 12%. Furthermore, TinyJudge achieves a 6× speedup in judging latency per response, resulting in a 3× reduction in total training time. Notably, its computational overhead is nearly identical to that of training with only hard constraints, demonstrating its extreme efficiency. In summary, this work paves the way for efficient and robust alignment with complex unverifiable human instructions.

2 Preliminaries

2.1 Task Definition: Instruction Following

Let $\mathcal{I}$ denote the space of natural language instructions and $\mathcal{Y}$ denote the space of model-generated responses. An instruction $I \in \mathcal{I}$ typically contains a core task accompanied by a set of n constraints $\mathcal{C} = \{c_1, c_2, \dots, c_n\}$. We categorize these constraints into two disjoint subsets:

- Hard Constraints: Objective requirements that can be verified via deterministic programs or rules (e.g., *"output in JSON format"*).
- Soft Constraints: Subjective qualities that require semantic understanding to evaluate (e.g., "*maintain a professional tone*").

The goal of LLM instruction following is to learn a mapping function $f : \mathcal{I} \to \mathcal{Y}$, where $\mathcal{Y}$ represents the space of target responses that satisfy both the semantic intent and the explicit constraints defined in $\mathcal{I}$. Formally, given an instruction $I \in \mathcal{I}$, an LLM parameterized by θ generates a response $\hat{y}$ by modeling the conditional probability:

$$P(\hat{y}|I;\theta) = \prod_{t=1}^{T} P(y_t|I, y_{<t};\theta), \tag{1}$$

where T is the sequence length. The task objective is to minimize the discrepancy between the generated response $\hat{y}$ and the optimal response y^*.

2.2 GRPO as an RLVR algorithm

We employ Group Relative Policy Optimization (GRPO) (Shao et al., 2024) as our core RL algorithm. Unlike traditional PPO (Schulman et al., 2017; Engstrom et al., 2020) which relies on a value function critic, GRPO leverages group-based sampling to estimate baselines. This approach generates multiple candidate responses for the same instruction and computes advantage estimates through intra-group comparisons, effectively capturing the relative quality differences among responses.

Formally, for each input instruction x, the policy π_θ samples a group of G candidate responses $\{y_i\}_{i=1}^G$. The optimization objective is defined as:

$$\mathcal{J}(\theta) = \mathbb{E}_{\substack{x\sim P(X),\\ \{y_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}(Y|x)}} \Bigg[\frac{1}{G}\sum_{i=1}^{G} \frac{1}{|y_i|} \sum_{t=1}^{|y_i|} \Bigg\{ \min\left(\rho_{i,t}\hat{A}_{i,t}, \text{clip}\,(*)\,\hat{A}_{i,t}\right) - \beta \mathbb{D}_{\text{KL}}\left[\pi_\theta \| \pi_{\text{ref}}\right] \Bigg\} \Bigg], \tag{2}$$

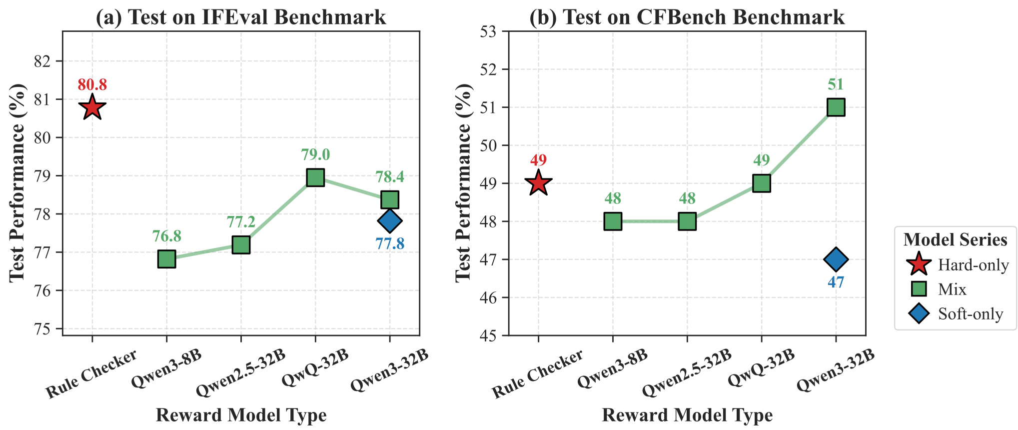


Figure 2: Test performance of the model under different reward models and training data configurations on IFEval (evaluating hard constraints) and CFBench (evaluating mixed constraints). Results show that: (1) the *hard-only* model outperforms the *soft-only* variant, and also performs comparably to the mixed-constraint model; and (2) a stronger reward model leads to better test performance.

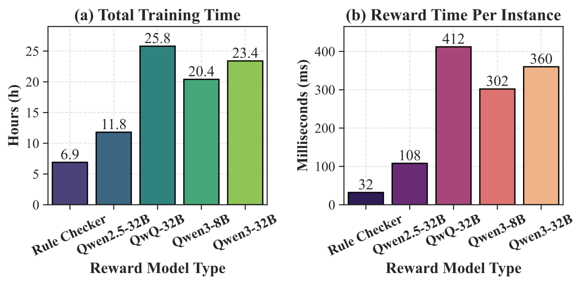


Figure 3: Comparison of training efficiency. (a) total training time (up to 100 iteration steps), (b) average reward latency per instance. The LLM-based approach significantly increases computational costs compared to rule-based rewards.

where the clip($*$) denotes clip $(\rho_{i,t}, 1-\varepsilon, 1+\varepsilon)$. Note that ε is a small constant for numerical stability. The advantage $\hat{A}_i$ is standardized within the group to reduce variance:

$$\begin{aligned} \mu &= \frac{1}{G}\sum_{i=1}^{G} r_i, \qquad \sigma = \sqrt{\frac{1}{G}\sum_{i=1}^{G}(r_i-\mu)^2+\epsilon} \\ \hat{A}_i &= \frac{r_i-\mu}{\sigma}, \qquad \rho_{i,t} = \frac{\pi_\theta(y_{i,t} \mid x, y_{i,<t})}{\pi_{\theta_{\text{old}}}(y_{i,t} \mid x, y_{i,<t})}, \end{aligned} \tag{3}$$

where r_i denotes the reward signal assigned by the reward model to the i-th response.

3 Pilot Experiments

In this section, we thoroughly evaluate the effectiveness of LLM-as-a-judge and present a key insight to improve performance.

3.1 Examining LLM-as-a-Judge

Experimental Setup. For model training, we utilize the *VerInstruct* dataset (Peng et al., 2025), comprising 22,000 instances. The dataset is partitioned into two constraint types: soft constraints (77.7%), which are evaluated via LLM-as-a-judge, and hard constraints (22.3%), which are validated through deterministic code-based rules. To investigate the influence of data distribution, we categorize the training data into three subsets: *hard-only*, *soft-only*, and the original *mix* set. We employ Qwen2.5-7B-Instruct (Qwen, 2025) 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, QwQ-32B (Team, 2025), Qwen3-8B (Yang et al., 2025), and Qwen3-32B (Yang et al., 2025) [1].

Test Performance. We train models separately on the *hard-only*, *soft-only*, and *mix* datasets and evaluate them on the hard constraint benchmark IFEval (Zhou et al., 2023) and mixed constraint benchmark CFBench (Zhang et al., 2025), details in Section §5.1. As illustrated in Figure 2, the model trained on *hard-only* data consistently outperforms the *soft-only* and *mix* variants (i.e., including both hard and soft constraint). In particular, the hard-only model outperforms the *soft-only* model (judged by Qwen3-32B) by 3.0% and the *mix* model by 2.4% on IFEval. This finding also holds on the out-of-distribution (OOD) benchmark CFBench. This observation suggests that using an

[1] The closed-source LLMs accessing via APIs are impractical as reward models in RLVR training due to API rate limits.

LLM as a judge for soft constraints does not confer broader OOC generalization to the model. This suggests that the LLM judge introduces undesirable biases, highlighting that naively employing an LLM as a reward for unverifiable soft constraints is imperfect and potentially harmful.

Time Efficiency. We evaluate the computational overhead in terms of total training time and per-reward latency under identical experimental settings. Specifically, we compare the *hard-only* model using rule-based rewards against the *mix* model that combines rule-based rewards (for hard constraints) and LLM-based rewards (Qwen3-32B for soft constraints). For reward latency of each instance, we report the average time required to judge a response across 1,000 randomly selected responses. As shown in Figure 3, employing an LLM-as-a-judge (e.g., Qwen3-32B) increases the total training time by $\sim 339\%$ and the reward latency of each instance by $11\times$ compared to the baseline. In contrast, rule-based rewards incur negligible overhead ($\sim$30 ms per instance). These results highlight the significant computational bottleneck introduced by LLM-based evaluation in the training loop.

Cross-Model Validation. To ensure the validity of the above observation, we scale our experiments across different model architectures (e.g., Qwen2.5 (Qwen, 2025), Qwen3 (Yang et al., 2025), LLama3.2 (Dubey et al., 2024)) and sizes (3B, 7B, 8B, 32B). We also evaluate the model on three additional benchmarks (see Section §5.1). The results are shown in Appendix §C Table 5. The soft-constraint models remain the worst across all settings, suggesting that this is a fundamental limitation of the LLM-as-a-judge paradigm rather than a model-specific artifact.

Visualization of Training. To further understand the above observation, we visualize the model training curves (Qwen3-32B as the reward for soft constraint) in Figure 4. We surprisingly find that soft-only models achieve significantly higher reward scores during training yet ultimately deliver lower test performance. This is mainly because soft-only models explore biases in LLM-as-a-judge to game the reward scores, rather than genuinely mastering constraint adherence. This finding indicates that LLM-based rewards lead to inflated reward scores, which do not translate into improved test performance. Therefore, we argue that directly using

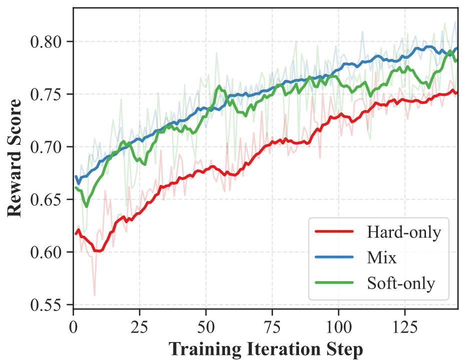


Figure 4: The training curves of models trained in hard-only, soft-only and mixed constraint data, respectively. The *Soft-only* model achieves higher reward scores than *Hard-only* model.

Reward Model	Hard Constraint Precision	Soft Constraint Precision
Rule Checker	**96.0**	–
Batch Judgment		
Gemini-2.5-pro	86.0	**86.3**
Qwen-3-32B	$76.5_{\downarrow 19.5}$	$74.5_{\downarrow 11.8}$
QwQ-32B	80.7	78.2
Qwen-2.5-32B	71.5	71.8
Point-wise Judgment		
Gemini-2.5-pro	$85.8_{\downarrow 0.2}$	$\mathbf{88.7}_{\uparrow 2.4}$
Qwen-3-32B	$82.6_{\uparrow 6.1}$	$83.5_{\uparrow 9.0}$
QwQ-32B	$83.8_{\uparrow 3.1}$	$83.9_{\uparrow 5.7}$
Qwen-2.5-32B	$74.5_{\uparrow 3.0}$	$75.8_{\uparrow 4.0}$

Table 1: Evaluation of reward reliability on hard and soft constraints of CFBench. Results show that: (1) LLM-based rewards achieve significantly lower reward precision than the rule checker; and (2) point-wise judgment consistently outperforms batch judgment across most LLM-as-a-judge models.

LLMs as rewards is currently unreliable.

3.2 Quantifying Reward Reliability

We evaluate the precision of reward signals from the reward model. Concretely, we take the responses generated by the base model on CFBench, and evaluate them as follows: (1) for responses involving hard constraints, we assess them using both a rule-based checker and an LLM-based reward model, respectively; (2) for responses involving soft constraints, we evaluate them solely with the LLM-based reward model. For comparison, we also test the frontier Gemini-2.5-Pro. We establish ground-truth labels by human experts, details in Appendix B. This allows us to quantify how well the reward models align with human evaluations.

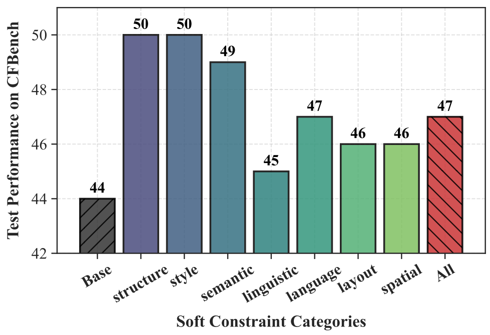


Figure 5: The generalization test performance of each soft constraint categories, on CFBench benchmarks. It shows that *style, structure*, and *semantic* exhibit higher performance compared to other categories.

Besides, we also compare point-wise judgment (assessing constraints individually) as an alternative to the conventional batch judgment (assessing all constraints simultaneously) for an instruction. Notably, pointwise judgment is more time-consuming than batch judgment.

For hard constraints, rule-based verifiers act as a near-perfect ground truth. In contrast, soft constraints rely on LLM-as-a-judge mechanisms, and we evaluate their agreement with the ground-truth labels. As shown in Table 1, we observe two key findings: (1) Although stronger LLMs achieve higher alignment with humans, they significantly underperform compared to the rule checker, e.g., the Qwen-3-32B lags behind by 19.5% compared to rule checker. This suggests that current LLM-based rewards suffer from a severe low reward-precision problem, which hampers the policy model's generalization in unverifiable constraint scenarios. (2) The *point-wise* judgment consistently outperforms batch judgment across open-sourced models, e.g., +6.1% for Qwen-3-32B on hard constraints. This suggests that when LLMs assess multiple constraints simultaneously, some form of bias interferes with reward precision.

3.3 Generalization Analysis of Constraints

To prevent bias of LLM-based rewards and gain insights for subsequent solutions, we analyze the generalization performance across all unverifiable soft constraint types. Specifically, to maximize coverage across all categories, we categorize soft constraints into seven categories: *style, structure, semantic, linguistic, language, layout, and spatial*. Detailed definitions and statistical distribution are provided in Appendix D.

We split the training data into seven subsets and perform GRPO training on each subset separately. We evaluate the test performance of models on CFBench benchmark (including all soft constraint categories and hard constraints), and show the results in Figure 5. We observe that constraints in the categories of *style, structure*, and *semantic* exhibit higher generalization performance compared to other categories. These results suggest that training on these three constraint categories yields greater generalization gains. We attribute this to the fact that these three constraints represent more foundational and generalized constraint patterns.

Overall, to improve reward precision, we can restrict LLMs to judge constraints one at a time, rather than assessing all constraints simultaneously. By training only on constraints that support strong generalization, we can mitigate LLM reward bias while also improving judgment efficiency.

4 Methods

This section introduces **TinyJudge**. Inspired by the above analysis, we incorporate only three high-generalization constraints alongside hard constraints. We fine-tune three tiny models to provide precise reward signals in the GRPO loop, as shown in the framework in Figure 6.

4.1 Specialist Reward Synthesis

In this offline phase, soft-constraint judgment knowledge is distilled from a powerful teacher model into a suite of lightweight specialist classifiers. This process ensures that the resulting reward signal is both highly precise and computationally efficient for subsequent GRPO training.

To fully cover the three soft constraint types, we first augment the raw query set $\mathcal{Q}$ (Section 3.1). For each query $q \in \mathcal{Q}$, we employ Gemini-3.0-Pro[2] to synthesize and integrate a latent soft constraint c_{soft} into the query to get the full instruction I. Subsequently, we generate a diverse response pool $\mathcal{Y}$ by sampling from a heterogeneous set of models, including Qwen2.5-7B/32B-Instruct (Qwen, 2025) and Llama3.2-3B (Dubey et al., 2024). This heterogeneity ensures that the specialists are exposed to varying levels of reasoning quality and common failure modes in instruction following.

Finally, for each triplet (q, c_{soft}, y), where $y \in \mathcal{Y}$, we utilize Gemini-3.0-Pro to provide a binary assessment r regarding the adherence to constraints.

[2]https://gemini.google.com/

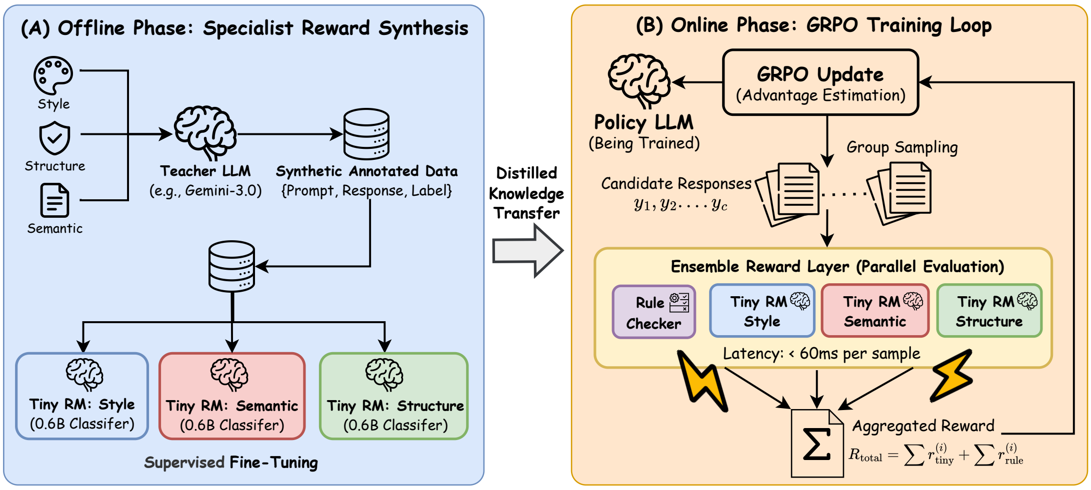


Figure 6: The constraint alignment framework (**TinyJudge**) via distilled specialist reward ensembles. It operates in two phases: (A) Specialist Distillation: we respectively distill three soft constraint types from a teacher LLM into a suite of lightweight, high-precision classifiers. (B) Accelerated GRPO Training: during RL training, these lightweight experts serve as a high-throughput ensemble reward model. They provide multi-constraint signals for a sample in milliseconds, enabling efficient alignment with unverifiable constraints.

we then fine-tune Qwen3-0.6B[3] as specialist judges. A suite of other lightweight models is also evaluated in Section 5. Let $\mathcal{M}_k$ denote the k-th specialist model; we optimize it using a supervised fine-tuning objective:

$$\mathcal{L}(\theta_k) = -\mathbb{E}_{(I,y,r)\sim\mathcal{D}_k} \sum_t \log P(r|I, y; \theta_k), \quad (4)$$

where $\mathcal{D}_k$ is the specialist-specific dataset tailored to a particular constraint type (e.g., style or tone).

4.2 Reward Engineering for GRPO Loop

In this online phase, TinyJudge computes a hybrid reward for each candidate response, enabling the model to align with both hard and soft constraints simultaneously.

Concretely, once the specialist judges $\mathcal{M}_k$ are distilled, they are integrated into the GRPO training loop. The total reward R_{total} is synthesized through an additive ensemble of rule-based logic and neural specialists:

$$R_{total}(q, r_i) = \frac{1}{N}\sum_{n=1}^{N} \mathcal{R}^n_{rule}(I, y_i) + \frac{1}{M}\sum_{k=1}^{M} \mathcal{M}_k(I, y_i), \quad (5)$$

where $\mathcal{R}_{rule}$ and $\mathcal{M}$ both are the binary evaluators for hard and soft constraints, respectively. The GRPO objective is optimized to achieve a perfect score in $\mathcal{R}_{total}$.

By utilizing tiny models as specialist judges and performing inference in parallel with the policy rollout, it can process a response in 10 milliseconds, effectively eliminating the latency bottleneck of LLM-based reward during the RL loop.

5 Experiments

5.1 Experimental Setup

Evaluation benchmarks. We evaluate the models on five representative IF benchmarks, including IFEval (Zhou et al., 2023), Multi-IF (He et al., 2024), and IFBench (Pyatkin et al., 2025), all of which focus exclusively on verifiable hard constraints. FollowBench (Jiang et al., 2024) and CF-Bench (Zhang et al., 2025) cover a comprehensive range of mixed constraint types, including soft constraints. The benchmark details are provided in Appendix §F. For evaluation, we use the Instruction Satisfaction Rate (ISR; Zhang et al., 2025) as the metric, which measures the model's ability to satisfy all constraints within a given instruction.

Baseline. For our baseline comparisons, we report the performance of leading LLMs. Additionally, we evaluate three recently RLVR-trained models derived from the same Base model: RECAST(Guo et al., 2025b) (verifiable data synthesis), IF-RLVR(Pyatkin et al., 2025) (multi-hard-constraint RL), and Qwen-IF(Ren et al., 2025) (label-free self-supervised reward from instructions). In the setup where the LLM is used directly as a reward model[‡], we train with a mix of three soft constraints alongside a hard constraint, aligning with our proposed method. More setup details

[3]To accelerate judgment, we disable the "thinking" mode.

Model	♣IFEval	♣Multi-IF	♣IFBench	♣CFBench	♠FollowBench	Average
Gemini-3.0-Pro	95.56	83.10	68.70	70.00	84.00	80.27
Claude-Sonnet-4.5	91.13	81.57	44.55	63.00	79.46	71.94
Deeepseek-V3.2	91.87	74.14	45.91	67.00	79.76	71.74
Qwen3-8B	85.77	70.36	24.48	55.00	67.28	60.58
Qwen3-32B	87.06	70.70	25.17	64.00	69.61	63.31
IF-RLVR†	87.80	—	53.70	—	—	—
RECAST†	74.01	—	—	—	63.23	—
Qwen-IF†	78.90	—	—	52.00	63.80	—
Qwen2.5-7B-inst	72.46	51.05	28.91	44.00	61.40	51.56
+ *Qwen3-8B*‡	77.82	57.15	29.59	48.00	68.64	56.24
+ *Qwen2.5-32B*‡	78.19	57.50	31.63	48.00	69.88	57.04
+ *Qwen3-32B*‡	79.48	57.08	30.95	49.00	69.74	57.25
+ TinyJudge-7B‡	$82.81_{\uparrow 10.35}$	$64.90_{\uparrow 13.85}$	$35.03_{\uparrow 6.12}$	$54.00_{\uparrow 10.00}$	$70.88_{\uparrow 9.48}$	$61.52_{\uparrow 9.96}$
Qwen2.5-32B-inst	81.70	64.45	33.67	57.00	73.06	61.98
+ *Qwen2.5-32B*‡	83.55	68.23	33.67	58.00	75.20	63.73
+ *Qwen3-32B*‡	84.47	68.29	35.71	60.00	74.35	64.56
+ TinyJudge-32B‡	$86.51_{\uparrow 4.81}$	$73.57_{\uparrow 9.12}$	$41.83_{\uparrow 8.16}$	$64.00_{\uparrow 7.00}$	$77.01_{\uparrow 3.95}$	$68.58_{\uparrow 6.60}$

Table 2: Evaluation results on the five IF benchmarks. ♣ and ♠ denote hard-only constraints and mixed constraints (including soft constraints) benchmark, respectively. For RLVR-trained baselines†, we report only previously published results because the model checkpoints are not publicly available. Models marked with the ‡ superscript are used as reward models.

shown in Appendix §C.

5.2 Main Results

Overall Performance. The results are shown in Table 2. We observe that TinyJudge consistently outperforms the baselines, e.g., achieving an average improvement of $\sim$ 10% for the 7B model and 6.6% for the 32B model. Furthermore, consistent performance across benchmarks shows that limiting training to three soft constraint types still yields robust behavior. Moreover, TinyJudge-32B (fast-think) outperforms the slow-thinking model Qwen3-32B by 5.47%. These results suggest that fine-tuning tiny models to judge constraints individually is an effective approach to enhancing IF capabilities. The model trained with these high precision rewards can serve as a competitive alternative to more advanced large-scale LLMs. To ensure the validity of the above findings, we scale our experiments across different model sizes (7B, 32B). As shown in Table 2, these findings are consistently observed, suggesting that TinyJudge is robust across different base models. Their training dynamics visualization is shown in Appendix C.3.

5.3 Analysis Experiments

Here, we present a detailed analysis, including ablation study on tiny models, training efficiency evaluation, and quantifying reward precision.

5.3.1 Ablation Study on Tiny Model

To investigate the impact of different tiny model configurations on reward modeling, we conduct an ablation study across three distinct settings. For all experiments, we employ Qwen2.5-7B-Instruct as the policy model in RLVR. We evaluate a range of tiny models, including the Qwen2.5 and Qwen3 series, with scales varying from 0.5B to 4B parameter size. The configurations are defined as follows:

- Three Tiny Models for Three (*3-for-3*, our): Following the *TinyJudge* approach, we employ three separate tiny models, each specialized for one of the three soft constraint types.
- One Tiny Model for Three (*1-for-3*): A single tiny model is trained on the combined data of the three specific soft constraint types.
- One Tiny Model for All (*1-for-all*): A single tiny model is trained on a mixture of all constraint data (i.e., the original *mix* set).

Results presented in Table 3 reveal several insights. Within the *3-for-3* setting, Qwen3 models exhibit consistent trends across different sizes (e.g., Qwen3-4B-Inst and Qwen3-0.6B differ by only 0.4 points). Notably, Qwen3 architectures exhibit better reward modeling performance than Qwen2.5 at the same model scale. Furthermore, we observe that the *3-for-3* configuration consistently yields

Tiny Model	♣IFEval	♣CFBench	Avg.
Base	72.46	44.00	58.23
Three Tiny Models for Three (Our)			
Qwen3-0.6B	<u>82.81</u>	<u>54.00</u>	<u>68.41</u>
Qwen3-1.7B	81.70	53.00	67.35
Qwen3-4B-inst	82.62	**55.00**	**68.81**
Qwen2.5-0.5B-inst	**83.36**	52.00	67.68
Qwen2.5-1.5B-inst	82.44	53.00	67.72
One Tiny Model for Three			
Qwen3-1.7B	80.07	51.00	65.54
Qwen3-4B-inst	80.62	53.00	66.81
One Tiny Model for All			
Qwen3-1.7B	78.74	46.00	62.37
Qwen3-4B-inst	81.70	46.00	63.85

Table 3: Ablation study on tiny models. The best result in each column is highlighted in bold, while the second-highest result is indicated with an underline formatting.

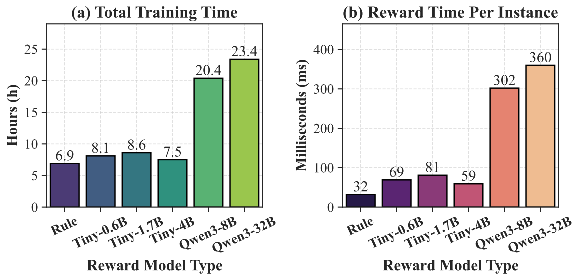


Figure 7: Comparison of training efficiency. Our tiny reward strategy significantly reduces computational cost compared to LLM-based rewards, achieving efficiency close to that of rule-based rewards.

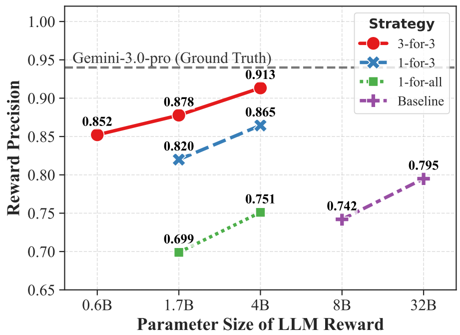


Figure 8: Quantifying reward precision in three soft constraint types across various reward models. Our proposed *3-for-3* reward strategy consistently achieves high reward precision.

superior results compared to the *1-for-all* and *1-for-3* settings using the same tiny model backbone. For instance, when using Qwen3-4B-Inst as the reward model, the *3-for-3* setting outperforms *1-for-all* and *1-for-3* by 2% and 5%, respectively. This evidence strongly supports the efficacy of our proposed architectural design for reward models.

5.3.2 Training Efficiency

To assess training efficiency, we measured the computational overhead under the same setup as Section 3.1. Figure 7 compares the Qwen3-Tiny reward model against traditional baselines in *3-for-3* setup. It reveals that our reward strategy reduces computational cost compared to direct LLM rewards by $\sim 3\times$. For example, Tiny-4B achieves a 312% speedup over Qwen3-32B in total training time, 610% for judging latency per response. Remarkably, the strategy achieves a processing speed comparable to that of rule-based rewards. These results demonstrate that our proposed strategy effectively mitigates the time overhead arising from LLM rewards, accelerating the training process.

5.3.3 Quantifying Reward Precision

We quantify reward precision to verify that our proposed TinyJudge effectively enhances reward reliability. We utilize a 5% held-out validation set and establish ground-truth labels by averaging five independent evaluations from Gemini-3.0-Pro. We then measure the alignment between various reward models (including Qwen3-Tiny configurations in *3-for-3*, *1-for-all*, and *1-for-3* setups) and the ground-truth. As illustrated in Figure 8, the tiny model within the *3-for-3* setup achieves the highest reward precision. For instance, the tiny 4B model outperforms Qwen3-32B by 12% in reward precision. This confirms that the TinyJudge strategy improves instruction-following generalization by directly improving the precision of reward signal.

6 Conclusion

In conclusion, this paper identifies and addresses the critical bottlenecks of reward bias and computational inefficiency in scaling soft constraints for RLVR. We propose a novel framework that replaces heavy LLM judges with an ensemble of specialized, tiny expert models tailored for high-generalization semantic constraints. It not only achieves a performance gain through higher reward precision but also delivers a $3\times$ reduction in total training time. These findings establish a highly efficient and robust paradigm for aligning models with complex, unverifiable human instructions without large-scale models.

Limitations

First, the selection and combination of constraints lack a systematic optimization framework. While we focus on high-generalization soft constraints, we have not yet established a theoretical or empirical boundary for the optimal number of expert models to be ensembled. Additionally, the reasons behind the strong generalization of these specific constraints warrant further investigation.

Second, the performance of tiny reward models is inherently capped by the teacher LLM. Since our training process relies on distilling expertise from frontier models, any systematic biases or reward inaccuracies present in the teacher model will inevitably be inherited by the specialized tiny judges.

Third, the *3-for-3* architecture faces scalability challenges at scale. Although the ensemble of tiny models offers high inference efficiency, a tiny model requires distillation and training whenever a new constraint type is introduced.

References

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

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

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

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

Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. 2020. Implementation matters in deep policy gradients: A case study on ppo and trpo. *arXiv preprint arXiv:2005.12729*.

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

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025a. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

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. 2025b. Recast: Expanding the boundaries of llms' complex instruction following with multi-constraint data. *Preprint*, arXiv:2505.19030.

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

Yuxin Jiang, Yufei Wang, Xingshan Zeng, Wanjun Zhong, Liangyou Li, Fei Mi, Lifeng Shang, Xin Jiang, Qun Liu, and Wei Wang. 2024. 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)*, pages 4667–4688.

Joongwon Kim, Anirudh Goyal, Aston Zhang, Bo Xiong, Rui Hou, Melanie Kambadur, Dhruv Mahajan, Hannaneh Hajishirzi, and Liang Tan. 2025. 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)*, pages 11062–11082.

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

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

OpenAI. 2025. Introducing GPT-5.

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

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

Qwen. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

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

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

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

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

Kimi Team and Yifan Bai. 2025. Kimi k2: Open agentic intelligence. *Preprint*, arXiv:2507.20534.

Qwen Team. 2025. Qwq-32b: Embracing the power of reinforcement learning.

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

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

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

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. 2025. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *arXiv preprint arXiv:2504.13837*.

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

Tao Zhang, Chenglin Zhu, Yanjun Shen, Wenjing Luo, Yan Zhang, Hao Liang, Fan Yang, Mingan Lin, Yujing Qiao, Weipeng Chen, et al. 2025. 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)*, pages 32926–32944.

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

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

A Contribution

In summary, our contributions are as follows:

- Empirical Finding: We conduct a systematic analysis of LLM-as-a-judge in RLVR, revealing that existing paradigms suffer from severe leniency bias in multi-constraint scenarios and higher computational costs.
- Methodological Innovation: We propose **TinyJudge**, a scalable framework that distills reward signals into tiny, specialized expert models (e.g., 0.6B) based on constraint generalization analysis. This enables precise feedback for unverifiable soft constraints without relying on heavy online judges.
- Good Results: Extensive evaluations across five benchmarks show that TinyJudge achieves a $\sim$10% performance gain over baselines. Crucially, it improves reward precision by 12% and reduces training time by 3$\times$, matching the efficiency of rule-based methods.

B Details in Reward Reliability

To establish a rigorous ground-truth reference for validating reward signal reliability, we implemented a multi-stage human evaluation protocol.

Data Sampling. We conducted stratified random sampling from the CFBench validation set to ensure diverse coverage of difficulty levels and constraint types. Specifically, we sampled $N = 100$ instances from the Hard-Only subset (e.g., formatting, symbol restrictions) and $N = 100$ instances from the Soft-Only subset (e.g., stylistic tone, creative writing).

Expert Annotation Protocol. The annotation task was assigned to three independent human experts, all of whom are graduate-level researchers in NLP with high proficiency in English. To mitigate potential bias, we employed a **double-blind procedure**: annotators were blinded to the model sources and the automated reward scores. We developed a strict *Adherence Rubric* prior to annotation:

- **Hard Constraints:** Evaluated on a binary scale (Pass/Fail). A response is marked as specific 'Fail' if it violates any strictly verifiable rule (e.g., missing a JSON key).
- **Soft Constraints:** Evaluated on a 5-point Likert scale regarding the degree of instruction following (5=Perfect Alignment, 1=Complete Violation). For binary classification analysis, scores ≥ 4 are converted to 'Pass'.

Quality Assurance and Adjudication. To ensure label consistency, we calculated Fleiss' Kappa (κ) to measure Inter-Annotator Agreement (IAA). We observed a substantial agreement for hard constraints ($\kappa = 0.89$) and moderate-to-high agreement for soft constraints ($\kappa = 0.72$). For instances with significant disagreement (variance > 1.0 on the Likert scale or split votes on binary labels), a fourth senior expert acted as an adjudicator to determine the final ground-truth label. The final reference score is derived from the adjudicated consensus rather than a simple average.

C Additional Experiments

In this section, we report additional experimental results.

C.1 Implementation Details

For reinforcement learning, we implemented GRPO based on the MindSpeed-RL[4] training framework. Each RL training run completed on a cluster of 64 Ascend 910b NPUs (configured as 8 nodes × 8 NPUs). The hyperparameters used are detailed in Table 4.

C.2 Robustness Check

The result is shown in Table 5.

C.3 Training Dynamics Visualization.

We visualize the various model training curves in Figure 9. In this paper, we use the checkpoint at step 400 as the final model.

D Taxonomy of Soft Constraints

For model training, we utilize the *VerInstruct* dataset (Peng et al., 2025), comprising 22,000 instances. It includes two constraint types: soft constraints (77.7%), which are evaluated via LLM-as-a-judge, and hard constraints (22.3%). We analyze the generalization performance across all unverifiable soft constraint types in this dataset. Specifically, to maximize coverage across all categories, we categorize soft constraints into seven types: *style, structure, semantic, linguistic, language, layout, and spatial*. Their definition are shown in Table 6. The category distribution of soft constraints is summarized in Table 7.

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

Hyperparameter	**Value**
Data Configuration	
Global Batch Size	128
Max Prompt Length	2048
Max Response Length	6144
Micro Batch Size	4
Train Steps	400
Rollout Configuration	
Rollout Name	vllm
GPU Memory Utilization	0.6
Number of Rollouts	8
Temperature	1.0
Tensor Model Parallel Size	1
Top_P	1.0
RL Optimization	
Learning Rate	1e-6
LR Decay Style	constant
Mini Batch Size	128
KL Loss	0.001

Table 4: The configurations for RLVR training.

E Related Work

Instruction Following requires models to generate responses that satisfy user's complex instruction constraints. 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). Recently, the research community is actively advancing IF studies, evidenced by benchmarks such as FollowBench (Jiang et al., 2024) and CFBench (Zhang et al., 2025), shifting focus from hard-only constraints toward scaling diverse, structured constraints to improve generalization in complex real-world scenarios. To achieve this goal, the RLVR training paradigm (Peng et al., 2025) is a mainstream approach currently being actively explored.

Model	♣IFEval	♣Multi-IF	♣IFBench	♠CFBench	♠FollowBench	**Average**
Qwen2.5-7B-inst	72.46	51.05	28.91	44.00	61.40	51.56
w/ *hard-only*	80.78	58.89	31.63	49.00	68.96	**57.85**
w/ *soft-only*	77.82	54.44	27.89	47.00	68.73	<u>55.18</u>
w/ *mix*	78.37	58.25	29.59	51.00	69.07	57.26
Qwen2.5-32B-inst	81.70	64.45	33.67	57.00	73.06	61.98
w/ *hard-only*	84.10	68.59	35.71	60.00	75.12	64.70
w/ *soft-only*	82.99	66.04	30.95	58.00	74.19	<u>62.43</u>
w/ *mix*	83.55	68.87	37.75	60.00	74.99	**65.03**
Qwen3-8B	85.77	70.36	24.48	55.00	67.28	60.58
w/ *hard-only*	88.54	73.37	25.17	57.00	67.79	**62.37**
w/ *soft-only*	86.32	70.97	27.21	56.00	67.21	<u>61.54</u>
w/ *mix*	86.51	72.73	25.85	57.00	67.70	61.96
Llama3.2-3B-inst	74.12	40.99	20.74	17.00	50.40	40.65
w/ *hard-only*	78.00	53.00	25.85	22.00	53.65	46.50
w/ *soft-only*	74.86	44.75	24.82	22.00	54.11	<u>44.11</u>
w/ *mix*	79.30	51.31	24.48	25.00	53.61	**46.74**

Table 5: Evaluation results on five benchmarks for robustness checking. We use Qwen3-32B as the reward model for all *soft-only* and *mix* settings. ♣ and ♠ denote hard-only constraints and mixed constraints (including soft constraints) benchmarks, respectively. Bold and underlined values indicate the **best** and <u>worst</u> performances, respectively. The *hard-only* models consistently outperform *soft-only* models across various base model configurations, even on OOD mixed constraints benchmarks.

RLVR has garnered attention for its ability to incentivize reasoning in LLMs through rule-based rewards (Yue et al., 2025; Zhu et al., 2025; Dai et al., 2025; Team and Bai, 2025). This paradigm has demonstrated remarkable efficacy in reasoning domains, e.g., math and coding (Zeng et al., 2025; Fatemi et al., 2025; Liu et al., 2025). However, the application of RLVR remains largely restricted to tasks with explicit, verifiable ground truths. To extend this success to soft constraints in IF, researchers have relied on LLM-as-a-judge to provide reward signals (Pyatkin et al., 2025; Guo et al., 2025b; Peng et al., 2025; Lambert et al., 2024). In contrast, our work identifies the LLM reward hacking bottlenecks. To address it, we introduce expert-level tiny models designed to minimize training overhead and mitigate reward hacking.

F Benchmark Details

To comprehensively assess the instruction-following capabilities of LLMs, we utilize five distinct benchmarks. These datasets cover hard and soft constraints, and also assess generalization to unseen constraints, multi-level difficulty, and multi-turn multilingual interactions. Table 8 summarizes the statistics of these datasets.

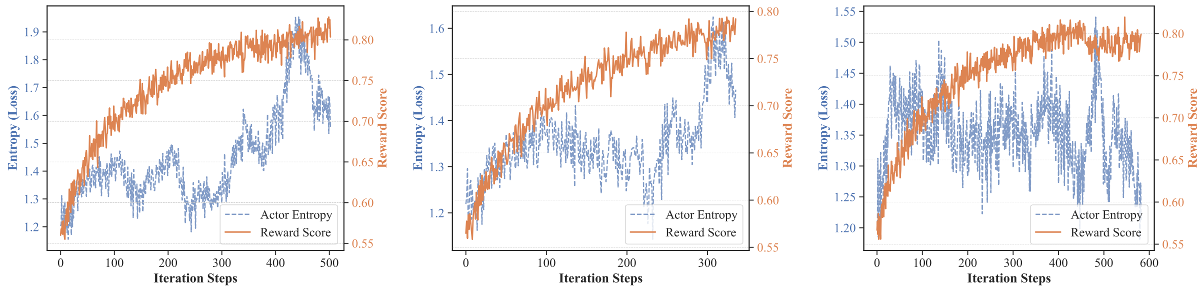
IFEval (Zhou et al., 2023) is a widely adopted benchmark designed to evaluate the ability of LLMs to follow objective and verifiable instructions. It consists of around 500 prompts containing 25 types of verifiable constraints (e.g., word count limits, formatting requirements).

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 dataset is constructed by expanding single-turn verifiable instructions into coherent three-turn dialogues. It serves as a stress test for maintaining instruction adherence over long contexts and across diverse linguistic distributions.

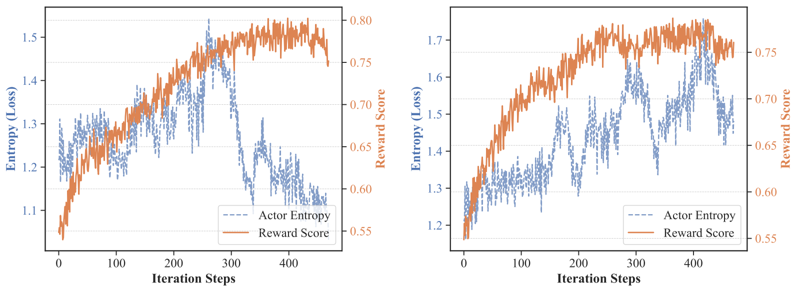
IFBench (Pyatkin et al., 2025) addresses the issue of model overfitting to common instruction datasets. It focuses on evaluating the generalization capabilities of models by introducing 58 novel, unseen, and challenging verifiable constraints. It employs strict code-based verification modules to measure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



(a) Reward with Qwen3-0.6B in *3-for-3*. (b) Reward with Qwen3-1.7B in *3-for-3*. (c) Reward with Qwen3-4B in *3-for-3*.



(d) Reward with Qwen2.5-0.5B in *3-for-3*. (e) Reward with Qwen2.5-1.5B in *3-for-3*.

Figure 9: Visualization of training dynamics for various tiny models.

style, logical rules, numerical) and over 25 subcategories. CFBench is designed to simulate complex tasks, including both an *Easy* and *Hard* subset to test models on varying degrees of constraint complexity.

Category	Definition	Example
Style	Specifies the overall atmosphere (vibe), narrative tone, or the persona (role) of the writer.	Answer in the persona of a weary film noir detective; the tone should be cold, cynical, and filled with a sense of fatalism.
Structure	Focuses on the abstract logical layout or imitation of specific non-fixed formats, emphasizing logical flow.	Adopt an 'echoing' structure: ensure the final paragraph provides a specific response to the philosophical hypothesis in the first paragraph.
Semantic	Regulates the introduction or exclusion of specific semantic content, or the use of particular rhetorical devices.	Explain code refactoring using a 'pruning a bonsai' metaphor, and strictly avoid mentioning any specific programming language names.
Linguistic	Concerns atomic-level grammatical constructions, parts-of-speech restrictions, or morphological rules.	Do not use any passive voice constructions, and ensure that no single sentence exceeds a length of 15 words.
Language	Governs rules for switching between languages and language alignment within specific cultural contexts.	Respond in Chinese, but naturally embed authentic English technical terms when discussing system architecture.
Layout	Focuses on the visual distribution of text, paragraph density, and the specific formatting of lists.	The response must be divided into three paragraphs of equal length, and all list items must start with a custom symbol (e.g., $\diamond$).
Spatial	Mandates the absolute spatial position or visual weight of specific information within the text stream.	At the very end of the response, naturally embed a specific closing statement and bold every technical term mentioned.

Table 6: Definitions and Examples of Constraint Categories

Constraint Type	Original Data Keys	Count
Style	Desired_Writing_Style	15,343
Structure	Hierarchical_Instructions	16,456
Semantic	Semantic_elements, Specific_Literary_Devices	34,782
Linguistic	Specific_Grammatical, Morphological	36,162
Language	Multi-lingual_Constraints	17,185
Layout	Paragraphs_Constraints, Item_Listing_Details	24,868
Spatial	Specific_Sentence, Special_Format, Key_Formatting	17,340

Table 7: Summary of soft constraint categories distribution

Dataset	Size (Samples)	Constraint Types	Key Features	Eval. Method
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	mixed / 5	Multi-level Difficulty	Code + LLM
CFBench (Zhang et al., 2025)	1,000	mixed / 25	Complex Scenarios	LLM

Table 8: Statistics and characteristics of the instruction following benchmarks.