DIAGNOSING AND CORRECTING REWARD HACKING IN VERIFIABLE INSTRUCTION FOLLOWING

Anonymous authors
Paper under double-blind review

ABSTRACT

Verifiable instruction-following benchmarks emphasize output-constraint satisfaction but can overlook whether responses meaningfully answer the underlying request, creating an evaluation blind spot. We show that frontier models can receive full credit for responses with little substantive answer content. Using the same verifier-based criterion as a training reward can induce reward hacking, as publicly released checkpoints exhibit higher constraint satisfaction but lower response quality after reinforcement learning with verifiable rewards. To address this reward hacking, we introduce CALIF, a lightweight method that augments verifier rewards with a calibrated likelihood-based bonus to preserve response quality. CALIF evaluates response likelihood under the request alone using the existing frozen reference policy, requiring no additional judge or learned reward model. Across models ranging from 1.7B to 8B in reasoning and non-reasoning modes, CALIF improves constraint satisfaction on IFBench while maintaining response quality near or above pre-RL levels. It also outperforms corresponding base policies in 40 of 45 model–metric comparisons across three rubric-based instruction-following benchmarks, demonstrating gains beyond training verifiers.

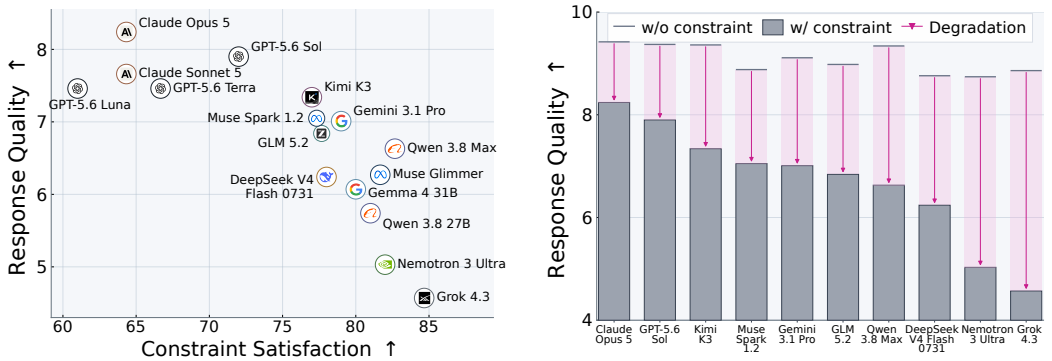
1 INTRODUCTION

Instruction following is central to the practical use of large language models (LLMs), enabling them to translate user intent into desired outputs (Ouyang et al., 2022; Yang et al., 2023; Chung et al., 2024). In real-world applications, a user input typically comprises a *request*, which specifies the task to be performed, and one or more *constraints*, which define the conditions the response must satisfy (Jiang et al., 2024; Wen et al., 2024; He et al., 2024; 2026). Successful instruction following therefore requires both high *Response Quality* by providing a substantive answer to the request and high *Constraint Satisfaction* by adhering to the stated constraints. However, achieving both objectives consistently remains challenging even for highly capable models (Zhang et al., 2025).

Evaluating instruction following for open-ended user inputs heavily relies on human assessment or LLM judgments, which can be costly, subjective, and sensitive to the choice of evaluator (Zheng et al., 2023; Liu et al., 2023; Wang et al., 2024; Thakur et al., 2025). Verifiable instruction-following (VIF) benchmarks (Zhou et al., 2023; He et al., 2024) address these evaluation challenges by focusing on requirements that can be checked programmatically. IFBench (Pyatkin et al., 2025), for example, appends output constraints to a request and measures *Constraint Satisfaction* using programmatic verifiers, enabling scalable and reproducible evaluation without relying on LLM judges. Consequently, VIF benchmarks are widely used to assess the capability for instruction following. For example, Artificial Analysis, one of the most widely used AI evaluation platforms, assesses instruction following using only IFBench's *Constraint Satisfaction* scores (Artificial Analysis, 2026).

However, these scores do not capture how well a response answers the request (*i.e.*, *Response Quality*): verifiers check only whether the constraint is met, and a response that ignores the request scores just as well as one that answers it fully. Indeed, we observe responses that receive full verifier credit for satisfying constraints while failing to adequately answer requests they handle well without those constraints (Figure 2). Moreover, using three strong LLM judges, we find that current frontier models with higher *Constraint Satisfaction* tend to have lower *Response Quality* (Figure 1a). These findings expose an ***evaluation blind spot*** in VIF benchmarks: high *Constraint Satisfaction* can be mistaken for strong instruction-following ability even when responses fail to answer the request.

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.



(a) *Constraint Satisfaction* vs. *Response Quality*.

(b) *Response Quality* with and without constraint.

Figure 1: **An evaluation blind spot in VIF: evidence from frontier models on IFBench.** *Constraint Satisfaction* evaluates whether responses satisfy the constraints using programmatic verifiers; *Response Quality* evaluates how well they answer the request using three LLM judges: GPT-5.6 Sol, Claude Opus 5, and Gemini 3.1 Pro, whose scores are averaged. Pink arrows show the drop in *Response Quality* from answering the request alone to answering the request with added constraints.

Beyond evaluation, programmatic verification can be naturally integrated with reinforcement learning with verifiable rewards (RLVR), a widely used approach for LLM post-training (Lambert et al., 2024; Guo et al., 2025a). IF-RLVR (Pyatkin et al., 2025) applies this approach to instruction following using verifier-based rewards for *Constraint Satisfaction*, yielding substantial gains on VIF benchmarks. However, in the publicly released OLMo 3 family (Olmo et al., 2025), whose RL training includes VIF task alongside other tasks, we observe higher *Constraint Satisfaction* but lower *Response Quality* in RLVR checkpoints than in their DPO counterparts (Figure 3). This divergence raises concerns about ***reward hacking*** (Amodei et al., 2016; Baker et al., 2025), as the verifier reward can reinforce the same response failures that fall into the evaluation blind spot described above.

Contribution. In this work, we first diagnose reward hacking in VIF through the lens of Bayes' rule, and find that the IF-RLVR reward omits the response likelihood under the request alone. Naively adding this term back, however, induces a different failure mode: the policy produces responses that inflate likelihood without genuinely improving response quality, *e.g.*, repetitions. To mitigate this failure mode, we introduce **Ca**librated **L**ikelihood for **I**nstruction **F**ollowing (**CALIF**), which effectively co-optimizes response quality during IF-RLVR by calibrating the verifier reward using a prompt-specific likelihood region. The key intuition is to define the desired behavior of the trained policy relative to two reference responses from the pre-RL policy: the response it produces under the request with constraints, and the one without them. The former's likelihood provides a baseline for preserving or improving response quality, while the latter's likelihood provides a reference for preventing likelihood inflation, since increases beyond the unconstrained reference need not reflect better answers. These likelihoods define the region's lower and upper bounds, respectively. We retain the verifier reward below the lower bound, add a fixed bonus within the region when the verifier reward is positive, and set the total reward to zero above the upper bound. In practice, CALIF adds little overhead: it reuses the reference policy already maintained for KL regularization, computes both bounds offline, and requires at most one extra teacher-forced forward pass per rollout.

Across five model configurations spanning 1.7B to 8B parameters and including both thinking and non-thinking modes, CALIF improves *Constraint Satisfaction* on IFBench while keeping *Response Quality* near or above base-policy levels. On average across these configurations, IF-RLVR increases *Constraint Satisfaction* (0–100 scale) from 26.98 to 44.93 but reduces *Response Quality* (1–10 scale) from 4.51 to 2.96, while CALIF improves both metrics to 34.64 and 4.59, respectively. Focusing on prompts where the base policy fails constraint verification but the trained policy passes, we find an average decline in *Response Quality* of 0.28 points under CALIF, compared with 2.05 under IF-RLVR, which indicates that CALIF learns to satisfy constraints with substantially less degradation in its answers to the request. Importantly, these gains generalize to three rubric-based instruction-following benchmarks, where CALIF outperforms baselines in 40 of 45 model–metric comparisons, versus only 14 of 45 for IF-RLVR. These results demonstrate that CALIF's reference-calibrated

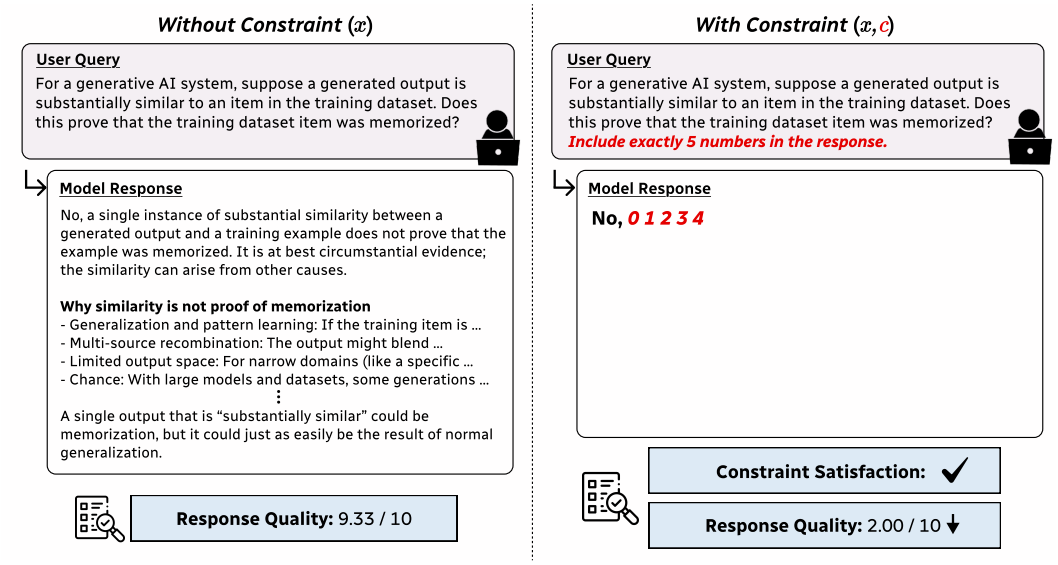


Figure 2: **DeepSeek V4 Pro responses with and without an appended constraint.** The model achieves a high *Response Quality* score of 9.33 when given the request alone, but its constrained response includes exactly five numbers as required while scoring only 2.00 in *Response Quality*.

reward provides an effective signal for mitigating reward hacking in IF-RLVR without any additional models or explicit supervision. CALIF thus brings the scalability of verifiable rewards to the full objective of instruction following: satisfying constraints while adequately answering the request.

2 Related Work

2.1 Instruction-Following Evaluation

Instruction following has been extensively studied as a core capability of LLMs. MT-Bench (Zheng et al., 2023) and AlpacaEval (Dubois et al., 2024) use LLM judges to rate response quality or compare responses, providing a scalable alternative to human evaluation. Further work extends LLM-based evaluation to individual requirements and their combinations, using task-specific rubrics for finer-grained assessment (Jiang et al., 2024; Qin et al., 2024; Wen et al., 2024; He et al., 2026).

Verifiable instruction-following benchmarks assess satisfaction of explicit constraints through programmatic checks rather than LLM judgments, enabling scalable and reproducible evaluation (He et al., 2024; Yao et al., 2024; Jaroslawicz et al., 2025; Liquid AI, 2026). As an example, IFEval (Zhou et al., 2023) covers constraints on response length, keyword inclusion, and output format, while IFBench (Pyatkin et al., 2025) extends this approach to a broader range of output constraints.

2.2 Reward Hacking in RLVR

Reward hacking occurs when optimizing a proxy reward leads to behavior that diverges from the intended objective (Amodei et al., 2016; Pan et al., 2022; Skalse et al., 2022). In RLVR, models can exploit flaws or misspecifications in verification procedures to obtain higher rewards without improving performance on the intended task (Baker et al., 2025; MacDiarmid et al., 2025; Helff et al., 2026), with related concerns reported in IF-RLVR (Guo et al., 2025b; Pyatkin et al., 2025).

To mitigate this behavior, prior work supplements verifiable rewards with explicit feedback on instruction fulfillment, using learned preference reward models (Pyatkin et al., 2025) or LLM-based intent checks (Guo et al., 2025b). These approaches, however, require maintaining and querying an additional reward model or LLM judge during training, introducing memory and inference overhead.

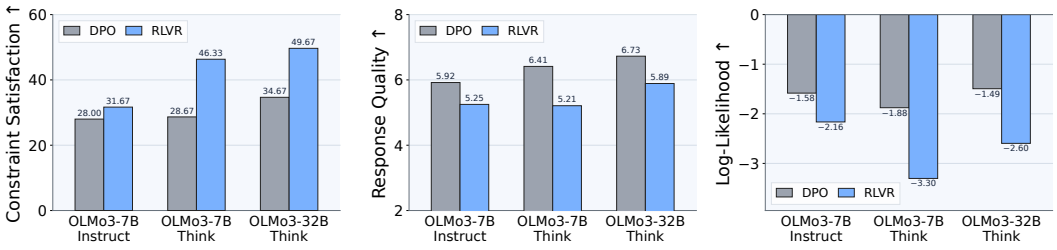


Figure 3: **Comparison of DPO and RLVR checkpoints in the OLMo 3 family on IFBench.** Log-likelihood is computed under the corresponding DPO checkpoint, conditioned on the request alone.

3 The Evaluation Blind Spot in Verifiable Instruction Following

Instruction-following benchmarks assess how well a model's response y fulfills the user's request and requirements. Because evaluating open-ended responses relies on costly human assessment or potentially biased LLM judgments (Zhou et al., 2023), verifiable instruction-following (VIF) benchmarks such as IFBench (Pyatkin et al., 2025) focus on programmatically verifiable requirements. Specifically, these benchmarks append output constraints c to a request x and use programmatic verifiers to check whether y satisfies c, enabling scalable, reproducible evaluation without relying on LLM judges. Successful instruction following requires y to satisfy c while providing a substantive answer to x, but standard VIF evaluation focuses primarily on constraint compliance. In practice, Artificial Analysis and several frontier model technical reports use only IFBench's constraint compliance scores to assess models' instruction-following performance (Artificial Analysis, 2026).

To examine how frontier models balance these requirements, we evaluate both ***Constraint Satisfaction***, which measures whether y satisfies c, and ***Response Quality***, which measures how well y answers x. We compute the former as IFBench's prompt-level loose accuracy and the latter as the mean 1–10 prompt–completion rating from strong LLM judges (see Appendix A for details). Figure 1a shows substantial variation across models, with higher *Constraint Satisfaction* tending to coincide with lower *Response Quality*. To assess whether this relationship simply reflects differences in models' ability to answer x, we compare each model's responses under x and (x, c). Figure 1b shows that all models in this comparison achieve high *Response Quality* under x alone, but appending c degrades *Response Quality*, with the largest declines among models with the highest *Constraint Satisfaction*. Figure 2 illustrates this degradation in a model that provides a substantive explanation under x alone but returns only *"No. 0 1 2 3 4,"* under (x, c). Standard VIF evaluation counts this response as successful solely because it satisfies c, exposing an ***evaluation blind spot***.

IF-RLVR (Lambert et al., 2024; Pyatkin et al., 2025) extends this verifier-based criterion from evaluation to training, applying RLVR to instruction following with rewards derived from programmatic verification of the appended constraints. However, models trained with RL for instruction following (NVIDIA, 2026; Xu et al., 2026) tend to show high *Constraint Satisfaction* but low *Response Quality*, as noted above. Furthermore, in all three matched OLMo 3 (Olmo et al., 2025) DPO–RLVR pairs, the RLVR checkpoint achieves higher *Constraint Satisfaction* but lower *Response Quality* (Figure 3). These observations are consistent with ***reward hacking*** under IF-RLVR, where models improve verifier rewards at the expense of response quality (Amodei et al., 2016; Pan et al., 2022).

4 CaLIF: Calibrated Likelihood for Instruction Following

Given a VIF instance (x, c) with $c = (c_1, \ldots, c_n)$, our goal is to obtain a response y that maximizes its probability under a target distribution $p(y \mid x, c)$. To understand which factors the policy must take into account to maximize $p(y \mid x, c)$, we decompose this distribution by applying Bayes' rule:

$$p(y \mid x, c) = \frac{p(c \mid y, x)\, p(y \mid x)}{p(c \mid x)}.$$

Because $p(c \mid x)$ does not depend on y, maximizing the response probability is equivalent to

$$\arg\max_y p(y \mid x, c) = \arg\max_y \left[\log p(c \mid y, x) + \log p(y \mid x)\right].$$

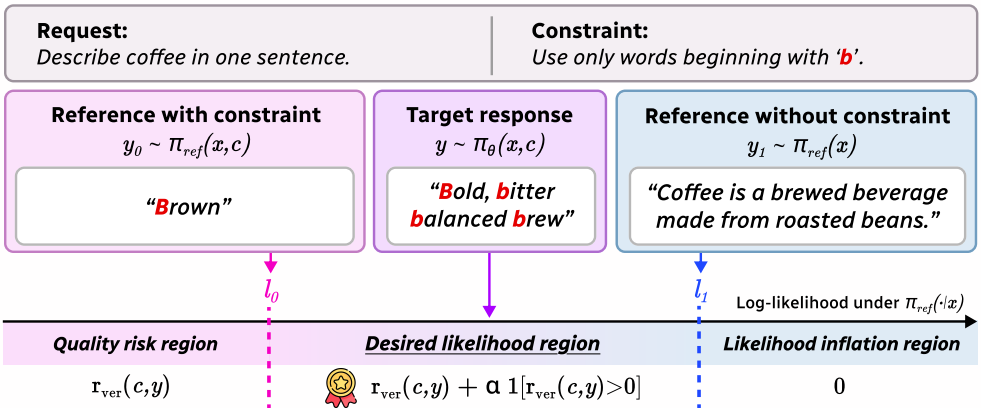


Figure 4: **Illustration of reference-calibrated likelihood region in CALIF.**

The first term favors constraint-compliant responses, the aspect of instruction following targeted by IF-RLVR. Let $v_i(c_i, y) \in \{0, 1\}$ denote the binary verifier that returns 1 if and only if y satisfies c_i. IF-RLVR optimizes the verifier reward $r_{\text{ver}}(c, y) = \frac{1}{n}\sum_{i=1}^{n} v_i(c_i, y)$ as a programmatic surrogate for this term. The second term favors responses with higher likelihood under x alone, but the verifier reward does not capture this term. Optimizing the verifier reward alone can therefore favor constraint compliance even as log-likelihood under x decreases. Indeed, OLMo 3 RLVR checkpoints exhibit this pattern, alongside a decline in *Response Quality* (Figure 3), consistent with reward hacking.

One natural proxy to account for the omitted term is the average log-likelihood of y under $\pi(\cdot \mid x)$:

$$\ell_\pi(y; x) = \frac{1}{|y|}\sum_{t=1}^{|y|} \log \pi(y_t \mid x, y_{<t}).$$

Higher values indicate that the model considers the response more likely given the request x alone. However, we find that directly rewarding this proxy encourages highly predictable outputs, such as repeated phrases, that inflate likelihood without improving response quality (see Section 6.1).

To address this failure mode, we introduce CALIF, which rewards responses within a reference-calibrated likelihood region rather than encouraging unbounded likelihood maximization. We define this region using two responses from the pre-RL policy π_{ref}: $y_0 \sim \pi_{\text{ref}}(\cdot \mid x, c)$ and $y_1 \sim \pi_{\text{ref}}(\cdot \mid x)$. Both responses are scored conditioned on x alone, yielding $\ell_0 = \ell_{\pi_{\text{ref}}}(y_0; x)$ and $\ell_1 = \ell_{\pi_{\text{ref}}}(y_1; x)$.

The lower bound ℓ_0 captures the reference policy's request consistency on the constrained instance before RL. Our goal is to improve constraint satisfaction while preserving or improving upon this baseline, rather than sacrificing the answer to x. The upper bound ℓ_1 captures the reference policy's natural response to x without the additional constraint. Under the premise that π_{ref} already answers x well, we use this likelihood as a conservative ceiling: adapting to c should not require pursuing likelihood gains beyond this natural reference level, which may instead reflect likelihood inflation.

Assuming $\ell_0 \le \ell_1$, we designate $[\ell_0, \ell_1]$ as the desired region, where responses receive a fixed bonus $\alpha > 0$ when the verifier reward is positive. Below ℓ_0, we retain the verifier reward to credit constraint satisfaction but withhold the bonus, as the response may fall short of the pre-RL baseline in answering x. Above ℓ_1, we suppress the verifier reward entirely to avoid reinforcing likelihood inflation at the expense of answering x. For a rollout y with $\ell = \ell_{\pi_{\text{ref}}}(y; x)$, the resulting reward is:

$$r_{\text{CALIF}}(x, c, y) = \begin{cases} r_{\text{ver}}(c, y), & \ell < \ell_0 \quad \text{(response quality risk region)}, \\ r_{\text{ver}}(c, y) + \alpha \mathbf{1}[r_{\text{ver}}(c, y) > 0], & \ell_0 \le \ell \le \ell_1 \quad \text{(desired likelihood region)}, \\ 0, & \ell > \ell_1 \quad \text{(likelihood inflation risk region)}. \end{cases}$$

When $\ell_0 > \ell_1$, we retain the verifier reward for $\ell \le \ell_1$ and assign zero reward otherwise.

In practice, we generate both reference responses and compute their likelihoods before RL, keeping them fixed during training. Because π_{ref} is already maintained for KL regularization, CALIF introduces no additional model and requires at most one extra teacher-forced forward pass per rollout.

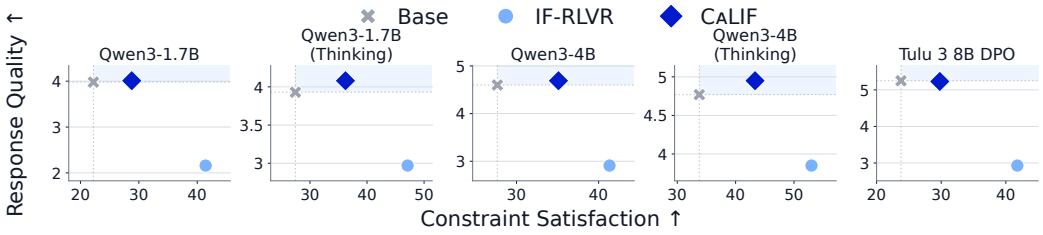


Figure 5: **CALIF mitigates reward hacking in IF-RLVR on IFBench across five model settings.**

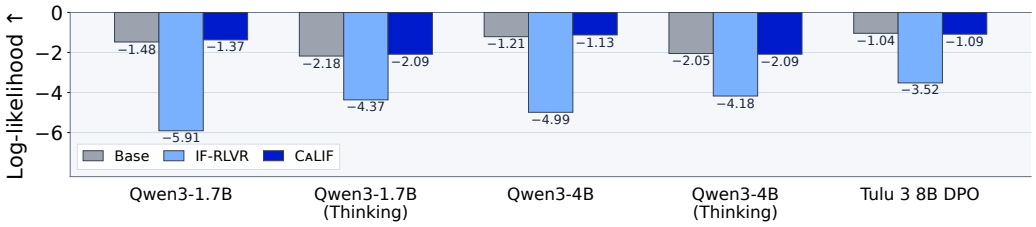


Figure 6: **Log-likelihood under the request alone on IFBench across five model settings.**

5 EXPERIMENTS

We now evaluate whether CALIF can mitigate the reward-hacking failure identified above through matched comparisons with IF-RLVR across model scales, thinking modes, and model families.

5.1 EXPERIMENTAL SETUP

For the training dataset, we use the multi-constraint IF-RLVR data released with IFBench,[1] comprising 95,373 prompts constructed by augmenting instructions from the Tülu 3 SFT mixture (Lambert et al., 2024) with one to five verifiable constraints from IFEval (Zhou et al., 2023) and IF-Train (Pyatkin et al., 2025). We train Qwen3-1.7B and Qwen3-4B in both thinking and non-thinking modes (Yang et al., 2025), and Tülu 3 8B DPO (Lambert et al., 2024), using GRPO (Shao et al., 2024) in `verl`.[2] We use matched training settings for CALIF and IF-RLVR, with a rollout temperature of 1.0 and a KL loss coefficient $\beta = 0.001$, and set likelihood reward coefficient $\alpha = 0.1$ for CALIF. For offline reference generation, responses are sampled with a temperature of 1.0. In thinking mode, responses include reasoning traces, but only final answers are scored under the reference policy in non-thinking mode. For evaluation, we follow the providers' recommended settings and report the mean and standard deviation over three runs. Unless otherwise specified, we use GPT-5.6 Sol to rate *Response Quality* on a 1–10 scale. Full experimental details are provided in Appendix B.

5.2 MAIN RESULTS

CALIF Mitigates Reward Hacking in IF-RLVR. Figure 5 compares CALIF with IF-RLVR on IFBench across five model configurations. IF-RLVR increases *Constraint Satisfaction* at the expense of *Response Quality* in every configuration, with average quality falling from 4.51 to 2.96. By contrast, CALIF raises average *Constraint Satisfaction* from 26.98 to 34.64 (+7.66 points) while preserving *Response Quality* at an average of 4.59. Pairwise evaluation further supports this quality advantage, with CALIF achieving an overall adjusted win rate of 78.10% against IF-RLVR (see Appendix D.1). As shown in Figure 6, reference log-likelihood under the request alone follows a similar pattern, decreasing under IF-RLVR but remaining near base-policy levels under CALIF. Together, these results show that CALIF's reference-calibrated reward mitigates reward hacking, retaining gains in *Constraint Satisfaction* without the quality degradation observed under IF-RLVR.

CALIF Generalizes Beyond Verifiable Constraints. To assess generalization to broader instruction-following requirements, we evaluate on three rubric-based benchmarks whose criteria

[1] https://huggingface.co/datasets/allenai/IF_multi_constraints_upto5
[2] https://github.com/verl-project/verl

Table 1: **Performance on three rubric-based instruction-following benchmarks.** *Format*† is most closely aligned with the training constraints. See Appendix C for details on the benchmarks and evaluation metrics. **Bold** and underline indicate the best and second-best within each setting.

Model	AdvancedIF (CIF)		FollowBench			ComplexBench			
	Overall pass rate ↑	Micro rubric pass ↑	HSR ↑	SSR ↑	CSL ↑	Lexical ↑	Semantic ↑	Utility ↑	Format† ↑
Qwen3-1.7B	18.08 ±1.12	56.82 ±0.53	53.81 ±1.76	68.33 ±1.10	2.54 ±0.06	57.46 ±1.29	53.62 ±0.49	51.09 ±0.05	50.70 ±0.42
+ IF-RLVR	12.44 ±0.00	50.13 ±0.29	48.14 ±1.00	64.02 ±0.51	2.29 ±0.04	49.68 ±0.37	47.43 ±0.28	40.93 ±0.11	50.45 ±0.70
+ CALIF	**19.32** ±0.29	**60.08** ±0.28	**60.29** ±0.60	**72.51** ±0.60	**2.81** ±0.05	**58.60** ±0.12	**56.34** ±0.24	**51.69** ±0.53	**50.94** ±0.91
Qwen3-1.7B (Thinking)	23.88 ±0.25	62.53 ±1.05	58.66 ±0.65	71.54 ±0.23	2.79 ±0.05	59.46 ±1.61	60.07 ±1.01	54.86 ±0.22	51.47 ±0.58
+ IF-RLVR	23.63 ±0.90	63.43 ±0.15	58.42 ±0.46	72.73 ±0.59	2.78 ±0.05	56.67 ±0.62	57.03 ±1.25	50.91 ±0.81	**55.10** ±0.80
+ CALIF	**26.53** ±0.38	**68.54** ±0.74	**62.10** ±1.53	**74.11** ±1.17	**3.02** ±0.06	**61.17** ±0.12	**63.17** ±0.27	**55.04** ±0.37	52.87 ±0.37
Qwen3-4B	30.85 ±0.50	70.11 ±0.48	**64.69** ±0.81	76.05 ±0.73	3.08 ±0.10	64.10 ±0.25	63.53 ±0.91	**60.31** ±0.09	55.59 ±0.06
+ IF-RLVR	29.02 ±0.38	66.26 ±0.73	64.09 ±0.62	76.35 ±1.51	3.09 ±0.03	58.82 ±0.12	58.39 ±0.36	52.64 ±0.60	**55.97** ±0.48
+ CALIF	**34.08** ±1.00	**73.33** ±0.20	64.34 ±0.19	**76.81** ±0.17	**3.13** ±0.04	**65.10** ±0.57	**64.89** ±0.08	59.39 ±0.31	54.19 ±0.65
Qwen3-4B (Thinking)	41.13 ±0.29	77.77 ±0.21	67.70 ±1.93	76.30 ±0.53	3.18 ±0.03	67.24 ±1.30	67.39 ±0.49	**63.27** ±0.24	57.72 ±1.12
+ IF-RLVR	44.61 ±1.15	77.97 ±0.52	68.90 ±1.52	80.05 ±0.54	3.28 ±0.08	66.10 ±1.06	66.53 ±0.67	62.60 ±0.74	**63.07** ±0.70
+ CALIF	**46.19** ±1.41	**80.85** ±0.38	**70.35** ±0.67	**80.06** ±0.35	**3.30** ±0.03	**67.59** ±0.25	**70.21** ±0.28	63.16 ±0.44	58.81 ±0.38
Tulu 3 8B DPO	27.69 ±1.37	70.35 ±0.80	65.27 ±0.40	75.35 ±0.53	3.09 ±0.07	58.46 ±1.40	60.80 ±0.48	**58.89** ±0.28	51.43 ±0.30
+ IF-RLVR	28.44 ±0.38	65.62 ±0.78	58.85 ±0.96	72.22 ±0.54	2.87 ±0.05	54.18 ±0.21	56.43 ±0.55	49.99 ±0.51	**55.35** ±0.46
+ CALIF	**32.92** ±0.38	**73.32** ±0.53	**67.88** ±0.64	**77.50** ±0.21	**3.23** ±0.01	**63.81** ±1.30	**64.76** ±0.91	58.60 ±0.19	55.10 ±0.40

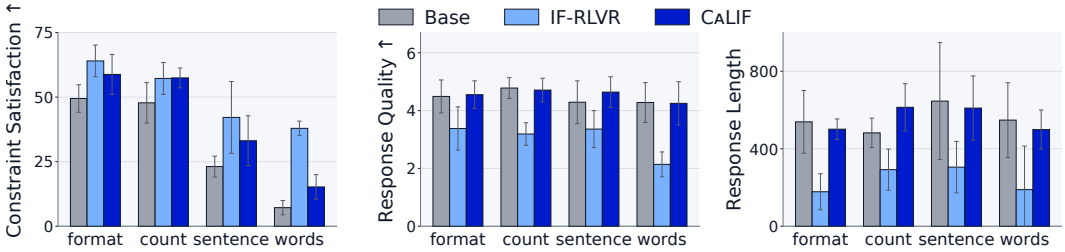


Figure 7: **Performance across constraint types on IFBench.** Bars and error bars indicate the mean and standard deviation across the five model settings. Response length is measured in tokens.

cover both *Constraint Satisfaction* and *Response Quality*: AdvancedIF (He et al., 2026), FollowBench (Jiang et al., 2024), and ComplexBench (Wen et al., 2024). As shown in Table 1, CALIF improves over the base policies in 40 of 45 model–metric comparisons, compared with only 14 for IF-RLVR. CALIF also outperforms IF-RLVR in 41 of 45 comparisons, with all four exceptions confined to *Format*, the dimension most closely aligned with the training constraints. These results demonstrate that CALIF generalizes beyond verifiable constraints to broader instruction following.

5.3 Analysis

Harder Constraints Amplify Reward Hacking. Figure 7 shows that reward hacking under IF-RLVR is most severe for harder constraint types. On word-level constraints, where base policies struggle most, IF-RLVR achieves its largest gains in *Constraint Satisfaction* but also the largest decline in *Response Quality*. This decline is accompanied by a marked reduction in response length, suggesting that the policy satisfies c while omitting substantive content needed to answer x. By contrast, CALIF improves constraint satisfaction across constraint types while maintaining response quality near or above base-policy levels, without the response shortening observed under IF-RLVR.

Verifier Success Can Mask Quality Degradation. Table 2 reports changes in *Response Quality* within each method's verifier-outcome groups. IF-RLVR achieves larger gains in *Constraint Satisfaction* (Figure 5), but its Fail → Pass outcomes show an average quality decline of 2.05 points. This deterioration is not limited to Fail → Pass outcomes: Pass → Pass outcomes also show an average decline of 1.26 points. By contrast, CALIF largely preserves *Response Quality* within its verifier-passing groups, with average changes of −0.28 and +0.01 points for Fail → Pass and Pass → Pass outcomes, respectively.

Table 2: ***Response Quality* changes for verifier-passing outcomes.** Transitions are from base to trained policies, aggregated over five model configurations.

Method	Δ Response Quality ↑	
	Pass → Pass	Fail → Pass
IF-RLVR	−1.26 ±0.37	−2.05 ±0.80
CALIF	**+0.01** ±0.06	**−0.28** ±0.47

Table 3: **Comparison with LLM-judge rewards on IFBench and three rubric-based instruction-following benchmarks.** Self-judge uses the base policy as the judge; Stronger judge uses Qwen3-30B-A3B. **Bold** and underline indicate the best and second-best within each setting.

	IFBench		AdvancedIF (CIF)		FollowBench			ComplexBench			
Reward	**Constraint Satisfaction** ↑	**Response Quality** ↑	**Overall Pass Rate** ↑	**Micro Rubric Pass** ↑	**HSR** ↑	**SSR** ↑	**CSL** ↑	**Lexical** ↑	**Semantic** ↑	**Utility** ↑	**Format** ↑
Tülu 3 8B DPO											
CALIF	**29.78** ±0.19	5.23 ±0.02	**32.92** ±0.38	**73.32** ±0.53	**67.88** ±0.64	**77.50** ±0.21	**3.23** ±0.01	**63.81** ±1.30	**64.76** ±0.91	**58.60** ±0.19	**55.10** ±0.40
Self judge	28.11 ±0.69	4.64 ±0.01	30.18 ±0.94	71.40 ±0.50	60.28 ±1.32	69.04 ±1.10	2.86 ±0.04	60.53 ±0.49	62.12 ±0.08	57.69 ±0.57	51.05 ±0.36
Stronger judge	26.89 ±0.38	**5.43** ±0.01	28.94 ±0.57	70.53 ±0.33	63.26 ±1.58	74.29 ±1.05	3.01 ±0.11	60.60 ±0.37	60.48 ±0.08	55.73 ±1.06	51.12 ±0.84
Qwen3-4B (Thinking)											
CALIF	**43.33** ±0.88	4.95 ±0.05	46.19 ±1.41	80.85 ±0.38	**70.35** ±0.67	**80.06** ±0.35	**3.30** ±0.03	67.59 ±0.25	**70.21** ±0.28	**63.16** ±0.44	**58.81** ±0.38
Self judge	39.33 ±0.88	5.23 ±0.02	44.20 ±2.54	80.78 ±1.48	69.96 ±0.94	78.48 ±0.59	3.26 ±0.01	**68.38** ±0.87	68.21 ±1.08	61.90 ±0.77	55.28 ±0.22
Stronger judge	41.00 ±2.19	**5.32** ±0.03	**46.93** ±1.93	**81.50** ±0.56	66.05 ±1.70	76.14 ±1.56	3.29 ±0.04	67.67 ±0.00	69.30 ±0.49	62.94 ±0.71	56.04 ±1.15

5.4 Comparison with LLM-Judge Rewards

We compare CALIF with an LLM-judge reward baseline to examine whether explicit quality judgments provide a more effective training signal. We consider two sources of supervision: the base policy itself (*Self-judge*) and a more capable model, Qwen3-30B-A3B (*Stronger judge*). Following Pyatkin et al. (2025), we threshold the judge scores produced using our *Response Quality* prompt and award the bonus α to responses that exceed the threshold and receive a positive verifier reward:

$$r_{\text{judge}}(x, c, y) = r_{\text{ver}}(c, y) + \alpha\, \mathbf{1}[r_{\text{ver}}(c, y) > 0 \ \wedge \ s_{\text{judge}}(x, c, y) > \tau]\,,$$

where s_{judge} is the LLM-judge score and τ is the judge-specific mean score over training instances.

Table 3 shows that CALIF outperforms both LLM-judge configurations on 10 of 11 metrics for Tülu 3 8B and 7 of 11 for Qwen3-4B in thinking mode. LLM-judge rewards generally yield higher *Response Quality*, consistent with directly rewarding this metric using the evaluation prompt.

Another advantage of CALIF is that it requires neither a separate judge model nor judge-score generation. Both judge baselines generate quality scores for rollouts, while the stronger judge requires storing a 30B-parameter model. By contrast, CALIF reuses the existing reference policy, requiring at most one additional teacher-forced forward pass per rollout. Its reward computation requires no explicit quality rubric, eliminating the need to specify quality criteria for judge-based supervision.

6 Ablation Studies

6.1 Direct Likelihood Rewards Encourage Likelihood Inflation

Our analysis in Section 4 identifies response likelihood under the request alone as a term omitted by IF-RLVR's verifier reward. To test whether directly rewarding this term preserves *Response Quality*, we exponentiate the average token log-likelihood to obtain a positive reward bounded by the same α used in CALIF. The bonus is applied only when a response receives a positive verifier reward:

$$r_{\text{direct}}(x, c, y) = r_{\text{ver}}(c, y) + \alpha \exp(\ell_{\pi_{\text{ref}}}(y; x))\, \mathbf{1}[r_{\text{ver}}(c, y) > 0]\,.$$

However, we find that training with this reward introduces another shortcut (i.e., *likelihood inflation*). Relative to the base policy, average reference log-likelihood rises from -1.50 to -0.68, while *Response Quality* falls from 3.98 to 1.91 (Table 4). This deterioration is accompanied by longer, repetitive responses: the average response length increases from 485 to 1,677 tokens, and the fraction of unique token 2-grams is 0.177, compared with 0.713 for the base policy and 0.698 for CALIF. The direct reward favors these highly predictable continuations even when they provide less substantive answers to the request, thereby creating another risk to increase reward without improving response quality. We present a qualitative example of likelihood inflation in Appendix D.3.

6.2 Both Likelihood Bounds Are Necessary

Table 4 shows that the lower and upper likelihood bounds address distinct failure modes. Without the lower bound, low-likelihood responses remain eligible for the bonus: *Constraint Satisfaction* rises to 43.00, but *Response Quality* falls to 2.16. Without the upper bound, responses above the desired region remain eligible for the bonus, leaving the reward vulnerable to likelihood inflation; *Response Quality* falls to 3.51 despite increased reference log-likelihood. Among these variants, only the full region improves both dimensions over the base policy. The lower bound discourages deterioration relative to the pre-RL baseline, while the upper bound limits exploitation of the likelihood signal.

Table 4: **Effect of likelihood reward design and region construction on IFBench.** We compare CALIF with a direct likelihood reward, single-bound variants, and alternative region constructions.

Method	Constraint Satisfaction ↑	Response Quality ↑	Reference Log-Likelihood	Response Length
Qwen3-1.7B	22.22	3.98	−1.50	485
+ IF-RLVR	41.44 (+19.22)	2.16 (-1.82)	−5.91 (−4.41)	152 (−333)
+ CALIF	28.78 (+6.56)	4.01 (+0.03)	−1.42 (+0.09)	550 (+65)
Direct likelihood reward (Section 6.1)				
Direct bonus	36.78 (+14.56)	1.91 (-2.07)	−0.68 (+0.82)	1677 (+1192)
Single-bound variants (Section 6.2)				
w/o lower bound	43.00 (+20.78)	2.16 (-1.82)	−6.83 (−5.33)	106 (−379)
w/o upper bound	30.67 (+8.45)	3.51 (-0.47)	−1.21 (+0.30)	780 (+295)
Likelihood bound variants (Section 6.3)				
Inner 95% CI	28.67 (+6.45)	4.04 (+0.06)	−1.47 (+0.03)	582 (+98)
Outer 95% CI	30.67 (+8.45)	4.00 (+0.02)	−1.36 (+0.14)	511 (+26)

6.3 Robustness to Likelihood Bound Estimation

Because the likelihood bounds are derived from stochastic offline rollouts, we test sensitivity to the sampled reference responses. We sample eight responses under each conditioning setting, with and without constraints, at temperature 1.0 and estimate a 95% confidence interval for each mean log-likelihood. We use the inward endpoints to construct a narrower likelihood region and the outward endpoints to construct a wider one. In Table 4, both variants perform similarly to the default construction and improve both *Constraint Satisfaction* and *Response Quality* over the base policy.

6.4 CALIF Goes Beyond KL Regularization

We examine whether stronger KL regularization alone can preserve *Response Quality* while retaining the gains in *Constraint Satisfaction* achieved by CALIF. Figure 8 compares the effects of varying the reward coefficient α in CALIF and the KL coefficient β in IF-RLVR. For CALIF, we vary $\alpha \in \{0.001, 0.01, 0.1, 0.2, 0.5, 1.0\}$ while fixing $\beta = 0.001$. For IF-RLVR, we vary $\beta \in \{0, 0.001, 0.002, 0.005, 0.01, 0.02, 0.05\}$. Across all tested α values, CALIF improves *Constraint Satisfaction* while preserving base-policy *Response Quality*. Increasing β in IF-RLVR also mitigates quality degradation, but recovering base-policy *Response Quality* leaves little improvement in *Constraint Satisfaction*. At $\beta = 0.05$, *Response Quality* matches the base policy at 3.98, while *Constraint Satisfaction* increases only from 22.22 to 23.00. By contrast, CALIF at $\alpha = 0.1$ achieves 28.78 in *Constraint Satisfaction* and 4.01 in *Response Quality*. These results show that CALIF retains larger gains in *Constraint Satisfaction* at comparable *Response Quality* than IF-RLVR achieves through KL coefficient tuning alone. Moreover, CALIF remains effective even without explicit KL regularization ($\beta = 0$; see Appendix D.2).

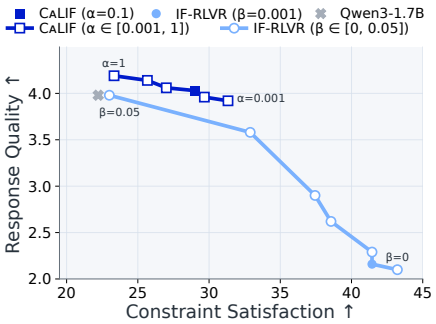


Figure 8: **Reward and KL coefficient sweeps on IFBench.** Squares vary α in CALIF; circles vary β in IF-RLVR. Larger coefficients shift points leftward.

7 Conclusion

We identify an evaluation blind spot in verifiable instruction following and document reward hacking in publicly released models. Across five model configurations, CALIF improves constraint satisfaction while preserving response quality through a reference-calibrated likelihood region, with gains beyond the training verifiers. Our results show that calibrating rather than maximizing reference likelihoods can mitigate reward hacking without additional models or explicit quality supervision.

AI USE STATEMENT

We used generative AI tools to identify literature and assist with manuscript drafting and editing. The authors reviewed AI-assisted material, checking sources and revising the text for accuracy and clarity. We take responsibility for the content of this work, including all AI-assisted text and claims.

REFERENCES

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*, 2016.

Artificial Analysis. Artificial analysis. https://artificialanalysis.ai/, 2026.

Bowen Baker, Joost Huizinga, Leo Gao, Zehao Dou, Melody Y Guan, Aleksander Madry, Wojciech Zaremba, Jakub Pachocki, and David Farhi. Monitoring reasoning models for misbehavior and the risks of promoting obfuscation. *arXiv preprint arXiv:2503.11926*, 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.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B Hashimoto. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*, 2024.

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

Xu Guo, Tianyi Liang, Tong Jian, Xiaogui Yang, Ling-I Wu, Chenhui Li, Zhihui Lu, Qipeng Guo, and Kai Chen. Ifdecorator: Wrapping instruction following reinforcement learning with verifiable rewards. *arXiv preprint arXiv:2508.04632*, 2025b.

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.

Yun He, Wenzhe Li, Hejia Zhang, Songlin Li, Karishma Mandyam, Sopan Khosla, Yuanhao Xiong, Nanshu Wang, Xiaoliang Peng, Beibin Li, et al. Advancedif: Rubric-based benchmarking and reinforcement learning for advancing llm instruction following. In *ACL*, 2026.

Lukas Helff, Quentin Delfosse, David Steinmann, Ruben Härle, Hikaru Shindo, Patrick Schramowski, Wolfgang Stammer, Kristian Kersting, and Felix Friedrich. Llms gaming verifiers: Rlvr can lead to reward hacking. *arXiv preprint arXiv:2604.15149*, 2026.

Daniel Jaroslawicz, Brendan Whiting, Parth Shah, and Karime Maamari. How many instructions can llms follow at once?, 2025. URL https://arxiv.org/abs/2507.11538.

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 *ACL*, 2024.

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.

Liquid AI. IFStruct: Measuring structured-output compliance. https://www.liquid.ai/blog/ifstruct-v1.0, 2026.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-Eval: NLG evaluation using GPT-4 with better human alignment. In *EMNLP*, 2023.

Monte MacDiarmid, Benjamin Wright, Jonathan Uesato, Joe Benton, Jon Kutasov, Sara Price, Naia Bouscal, Sam Bowman, Trenton Bricken, Alex Cloud, et al. Natural emergent misalignment from reward hacking in production rl. *arXiv preprint arXiv:2511.18397*, 2025.

NVIDIA. Nemotron 3 ultra: Open, efficient mixture-of-experts hybrid mamba-transformer model for agentic reasoning, 2026. URL https://research.nvidia.com/labs/nemotron/files/NVIDIA-Nemotron-3-Ultra-Technical-Report.pdf. White Paper.

Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, et al. Olmo 3. *arXiv preprint arXiv:2512.13961*, 2025.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *NeurIPS*, 2022.

Alexander Pan, Kush Bhatia, and Jacob Steinhardt. The effects of reward misspecification: Mapping and mitigating misaligned models. *arXiv preprint arXiv:2201.03544*, 2022.

Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hanna Hajishirzi. Generalizing verifiable instruction following. In *NeurIPS*, 2025.

Yiwei Qin, Kaiqiang Song, Yebowen Hu, Wenlin Yao, Sangwoo Cho, Xiaoyang Wang, Xuansheng Wu, Fei Liu, Pengfei Liu, and Dong Yu. Infobench: Evaluating instruction following ability in large language models. In *Findings of ACL*, 2024.

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.

Joar Skalse, Nikolaus Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward hacking. In *NeurIPS*, 2022.

Aman Singh Thakur, Kartik Choudhary, Venkat Srinik Ramayapally, Sankaran Vaidyanathan, and Dieuwke Hupkes. Judging the judges: Evaluating alignment and vulnerabilities in llms-as-judges. In *Proceedings of the Fourth Workshop on Generation, Evaluation and Metrics (GEM^2)*, 2025.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Lingpeng Kong, Qi Liu, Tianyu Liu, et al. Large language models are not fair evaluators. In *ACL*, 2024.

Bosi Wen, Pei Ke, Xiaotao Gu, Lindong Wu, Hao Huang, Jinfeng Zhou, Wenchuang Li, Binxin Hu, Wendy Gao, Jiaxin Xu, et al. Benchmarking complex instruction-following with multiple constraints composition. In *NeurIPS*, 2024.

Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

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.

Sherry Yang, Ofir Nachum, Yilun Du, Jason Wei, Pieter Abbeel, and Dale Schuurmans. Foundation models for decision making: Problems, methods, and opportunities. *arXiv preprint arXiv:2303.04129*, 2023.

Shunyu Yao, Howard Chen, Austin Hanjie, Runzhe Yang, and Karthik Narasimhan. Collie: Systematic construction of constrained text generation tasks. In *ICLR*, 2024.

Wei Zhang, Zhenhong Zhou, Kun Wang, Junfeng Fang, Rongwu Xu, Yuanhe Zhang, Rui Wang, Ge Zhang, Xinfeng Li, Li Sun, et al. Lifebench: Evaluating length instruction following in large language models. In *NeurIPS*, 2025.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *NeurIPS*, 2023.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

A LLM-JUDGE PROMPT

Following Pyatkin et al. (2025), we use the following prompt to rate the quality of y as an answer to x, both for *Response Quality* evaluation and for the LLM-judge reward experiments in Section 5.4.

> Evaluate the response provided below to determine if it meets the specified constraints related to the following prompt. Provide an integer score from 1 to 10, taking into account its helpfulness, relevance, accuracy, depth, creativity, and how well it conforms to the constraints. Here are the criterias that you should score: 1. Helpfulness: Does the response address the user's needs and questions effectively? 2. Relevance: Is the response directly related to the context of the dialog? 3. Accuracy: Are the facts and information presented in the response correct? 4. Depth: Does the response cover the topic thoroughly, with sufficient detail? 5. Creativity: Is the response original and engaging?
>
> **Prompt to Evaluate Against:** {`prompt`}
>
> **Response to Evaluate:** {`response`}
>
> The evaluation must be structured in the following JSON format:
>
> "Score": "<An integer score from 1 to 10.>"

B EXPERIMENTAL DETAILS

Training data. We use the multi-constraint IF-RLVR training data released by IFBench. The dataset contains 95,373 prompts constructed by augmenting instructions from the Tülu-3 SFT mixture with one to five automatically verifiable constraints sampled from IFEval and IFTrain.

Models and Training. We use Qwen3-1.7B and Qwen3-4B in non-thinking and thinking modes (Yang et al., 2025), and the Tülu 3 8B DPO model (Lambert et al., 2024). We train all models for four epochs (three for Qwen3-4B in thinking mode due to computational cost) with GRPO in `verl` on 8 NVIDIA H100 GPUs, using 8 rollouts per prompt, temperature 1.0, a global batch size of 1,024, and a KL loss coefficient of 0.001. The maximum response length and learning rate are 2,048 tokens and 5×10^{-7} in non-thinking mode, and 8,192 tokens and 1×10^{-6} in thinking mode.

CALIF Configuration. We sample anchor responses at the rollout temperature. For thinking models, anchor responses and rollouts include full reasoning traces, but we compute log-likelihoods under π_{ref} in non-thinking mode over final-answer tokens only. We set $\alpha = 0.1$ in all main experiments; Section 6 ablates the two anchors and examines sensitivity to anchor sampling and α.

Evaluation. We sample with temperature 0.6, top-p 0.95, and top-k 20 in thinking mode, and temperature 0.7, top-p 0.8, and top-k 20 in non-thinking mode, following official recommendations. We set maximum response lengths of 16,384 and 2,048 tokens for thinking and non-thinking models, respectively. To account for stochasticity, we repeat each evaluation three times and report the mean and standard deviation. Unless otherwise specified, we use GPT-5.6 Sol to rate *Response Quality*.

C RUBRIC-BASED INSTRUCTION-FOLLOWING BENCHMARKS

AdvancedIF. AdvancedIF (He et al., 2026) evaluates instruction following using expert-written rubrics across three categories: ComplexIF (CIF), Carried Context (CC), and System Steerability (SS). In our experiments, we evaluate only CIF, which focuses on single-turn requests with multiple requirements. We report *Overall Pass Rate*, the percentage of responses satisfying all rubric criteria, and *Micro Rubric Pass Rate*, the percentage of satisfied criteria pooled across responses.

FollowBench. FollowBench (Jiang et al., 2024) covers Content, Situation, Style, Format, and Example constraints, along with their combinations (Mixed), across five difficulty levels that progres-

Table 5: **Pairwise comparison of CALIF and IF-RLVR on IFBench.** Counts reflect CALIF's perspective. Ties include order-inconsistent judgments and receive half credit in adjusted win rates.

Model configuration	Wins	Ties	Losses	Adjusted WR (%)
Qwen3-1.7B	229	34	37	82.00
Qwen3-1.7B (Thinking)	193	45	62	71.83
Qwen3-4B	219	36	45	79.00
Qwen3-4B (Thinking)	195	56	49	74.33
Tülu 3 8B DPO	229	42	29	83.33
Overall	**1,065**	**213**	**222**	**78.10**

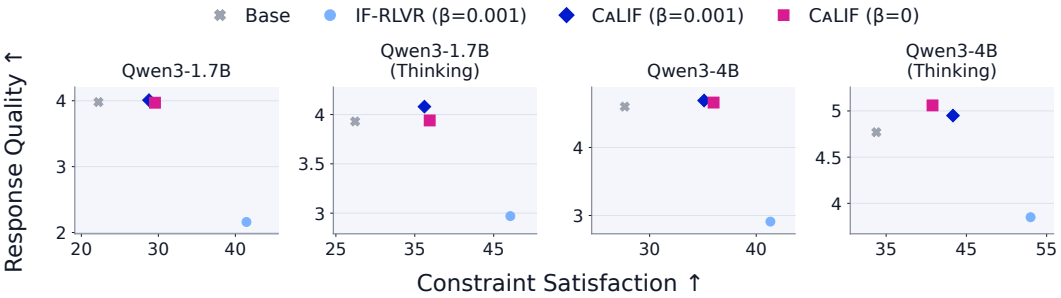


Figure 9: **CALIF without KL regularization on IFBench.** We compare CALIF with and without the KL penalty ($\beta = 0.001$ and 0, respectively), keeping $\alpha = 0.1$ fixed. IF-RLVR uses $\beta = 0.001$.

sively add constraints. We report *Hard Satisfaction Rate* (HSR), the percentage of responses satisfying all constraints; *Soft Satisfaction Rate* (SSR), the average fraction of constraints satisfied; and *Consistent Satisfaction Levels* (CSL), the average number of consecutive levels fully satisfied.

ComplexBench. ComplexBench (Wen et al., 2024) evaluates instructions combining constraints through conjunctions, conditional branches, and sequential dependencies. We report scoring-question accuracy for *Lexical*, *Semantic*, *Utility*, and *Format* constraints, covering word usage, semantic requirements, answer usefulness and factuality, and output format and length, respectively.

D Additional Analyses

D.1 Pairwise Response Quality Evaluation

To complement the pointwise *Response Quality* evaluation, we conduct a pairwise comparison of responses from CALIF and IF-RLVR using GPT-5.6 Sol as the judge. Following the order-swapping protocol of MT-Bench (Zheng et al., 2023), we evaluate each pair in both presentation orders. A win or loss is recorded only when both judgments favor the same method; ties in either order and disagreements between orders are counted as ties. We report wins (W), ties (T), and losses (L) from CALIF's perspective and compute the adjusted win rate as $(W + 0.5T)/(W + T + L)$. Table 5 shows that CALIF achieves an overall adjusted win rate of 78.10%. These results corroborate the *Response Quality* advantage of CALIF over IF-RLVR observed in the pointwise evaluation.

D.2 CALIF without KL Regularization

To evaluate whether CALIF can mitigate reward hacking without KL regularization, we set $\beta = 0$ while keeping all other training settings unchanged. Figure 9 compares this variant with default CALIF and IF-RLVR, both using $\beta = 0.001$. With $\beta = 0$, CALIF still improves *Constraint Satisfaction* over the base policy while largely preserving *Response Quality*. These results demonstrate that its reference-calibrated reward can mitigate reward hacking without an explicit KL penalty.

Removing the KL penalty eliminates the reference forward pass conditioned on (x, c), leaving only the pass conditioned on x for reward computation. Excluding offline bound construction, CALIF requires no more reference forward passes per rollout than standard KL-regularized IF-RLVR.

D.3 A Qualitative Example of Likelihood Inflation

Table 6: **Qualitative example of likelihood inflation.** We compare responses from the Qwen3-1.7B base policy with those from policies trained with IF-RLVR, CaLIF, or a direct likelihood bonus.

User Query:
explain Relativistic speed Please use an emoji at the end of every sentence.

Base Policy Response: ***Response Quality*: 5/10** ***Log-Likelihood*: -1.25**

🚀 **Relativistic speed** is the speed of an object moving at a significant fraction of the speed of light. 🚀
When an object moves at speeds close to the speed of light, time slows down for it relative to a stationary observer. 🚀
Relativistic effects become more pronounced as speed increases. 🚀
The theory of relativity explains how space and time are connected. 🚀
In relativity, mass increases with speed. 🚀
Objects moving at relativistic speeds experience time dilation. 🚀
The faster an object moves, the more its mass increases. 🚀
Relativistic speeds are not just theoretical; they are observed in high-energy particles. 🚀
Relativity also affects the length of objects moving at high speeds. 🚀
The speed of light is the ultimate speed limit in the universe. 🚀
Every sentence ends with an emoji. 🚀

IF-RLVR Response: ***Response Quality*: 2/10** ***Log-Likelihood*: -5.05**

🚀 The concept of relativistic speed is about objects moving at speeds close to the speed of light.
🚀 Every sentence ends with an emoji.

CaLIF Response: ***Response Quality*: 7/10** ***Log-Likelihood*: -0.84**

🚀 **Relativistic speed** is the speed of an object moving at a significant fraction of the speed of light. 🚀
When an object moves at speeds close to the speed of light (which is approximately **300,000 kilometers per second**), its effects become more pronounced due to **special relativity**. 🚀
As an object approaches the speed of light, its **mass increases** and time **slows down** for the object relative to a stationary observer. 🚀
This phenomenon is described by the **relativistic velocity addition formula**, which accounts for the fact that the speed of light cannot be exceeded. 🚀
The **relativistic mass** of an object increases with its speed, making it harder to accelerate further. 🚀
Also, the **length of an object** contracts in the direction of motion, a effect known as **length contraction**. 🚀
These effects become more significant as the object's speed approaches the speed of light. 🚀
Relativistic speed is a key concept in modern physics, especially in high-energy particle accelerators and space travel. 🚀
emoji 🚀

Direct Likelihood-Bonus Response: ***Response Quality*: 1/10** ***Log-Likelihood*: -0.04**

🚀 🚀 🚀

D.4 Additional Qualitative Examples

Table 7: **Qualitative comparison of IFBench responses: musical definitions.**

User Query:
Answer with a list of items, instead of bullet points use !?!?. Defination of: preludes, caprichos, romances and ballads

Base Policy Response: ***Response Quality*: 5/10** ***Constraint Satisfaction*: Fail**

Preludes are short, lively pieces in a musical style, often used to introduce a composition.
Caprichos are complex, dramatic works in a Spanish style, often with a strong rhythmic and melodic structure.
Romances are lyrical, expressive pieces, often in a slower tempo and with a more emotional tone.
Ballads are short, narrative songs with a traditional structure and a storytelling quality.

IF-RLVR Response: ***Response Quality*: 1/10** ***Constraint Satisfaction*: Pass**

!?!? preludes !?!? caprichos !?!? romances !?!? ballads

CALIF Response: ***Response Quality*: 6/10** ***Constraint Satisfaction*: Pass**

!?!? preludes are short musical pieces that introduce a new composition, often used to set the mood or atmosphere for a larger work
!?!? caprichos are complex, elaborate, and often virtuosic pieces characterized by their intricate melodies and dramatic contrasts
!?!? romances are lyrical and expressive musical pieces, often with a slow tempo and a focus on emotional expression
!?!? ballads are traditional musical pieces with a storytelling aspect, usually in a slower tempo and with a more melancholic or reflective mood

Table 8: **Qualitative comparison of IFBench responses: code explanation.**

User Query:
Try to explain what this function do: ```python def _config_class_for_function(fn: F) -> Type[FunctionConfigBase]: """Returns a config class.""" init_sig = inspect.signature(fn) config_attrs = { name: _attr_field_from_signature_param(param) for name, param in init_sig.parameters.items() } return _wrap_config_attr_cls(attr.make_class("FunctionConfig", bases=(FunctionConfigBase[F],), attrs=config_attrs, **_config_class_kwargs(),), name=f"config_for_function({fn.__module__}.{fn.__qualname__})",) ```
The response should not repeat any word more than 5 times.

Base Policy Response: ***Response Quality*: 7/10** ***Constraint Satisfaction*: Pass**

This function creates a configuration class for a given function. It extracts parameters from the function's signature, converts them into attributes, and wraps them in a class that inherits from `FunctionConfigBase`. The class is named based on the function's module and name.

IF-RLVR Response: ***Response Quality*: 1/10** ***Constraint Satisfaction*: Pass**

config

CALIF Response: ***Response Quality*: 7/10** ***Constraint Satisfaction*: Pass**

This function creates a configuration class for a given function. It uses the function's signature to extract parameters and generates attributes for the configuration class. The class is then wrapped and returned with a unique name.