DO LLMS KNOW WHERE CONSTRAINTS BITE? TOWARDS FINE-GRAINED REWARD SIGNALS FOR INSTRUCTION FOLLOWING

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning with verifiable rewards has become the dominant post-training paradigm for instruction following. Current pipelines compress all constraint verdicts into a scalar, and broadcast the advantage uniformly across every token, while such coarse-grained credit assignment fails to localize responsible tokens for each constraint. In this paper, we investigate whether the policy's own generation distribution can be utilized for finer-grained credit. Through a pilot experiment, we find that for constraints within the model's capability, removing a constraint from the prompt causes the probabilities of exactly the tokens that implement it to drop sharply. However, for constraints beyond capability, no sampled response could satisfy them and the same contrast becomes unreliable. Based on this dichotomy, we propose LOCO, which performs constraint-level credit assignment: for constraints within the model's capability, we localize the responsible tokens by the leave-one-out contrast of each token; for constraints beyond it, we additionally provide the policy with the verbalized failure reason, so that the probability difference could localize the offending tokens. Moreover, when no sampled response satisfies all constraints, demonstration-conditioned self-distillation provides the policy with a golden response minimally edited from its best attempt to expand the model's capability boundary. Comprehensive experiments across eight instruction-following tasks confirm the effectiveness of our proposed LOCO.

1 INTRODUCTION

Instruction Following (IF) is one of the core capabilities for modern large language models (LLMs). Given an instruction that imposes explicit requirements on the format, length, content, language, or style of the response, the model is expected to comply with all of them (Zhou et al., 2023; Jiang et al., 2024). The capability underlies a wide range of applications, from producing machine-readable output for downstream tools, to controlling tone and structure in conversational agents. Therefore, instruction following has become a central objective of post-training (Ouyang et al., 2022; Rafailov et al., 2024) and among the most closely evaluated abilities of LLMs.

However, the task is far from solved. The difficulty for IF grows steeply with the number of constraints: once instructions carry more than ten simultaneous requirements, even strongest models miss a substantial fraction of them (Jiang et al., 2024). This difficulty is structural. First, a model's attention is finite and unevenly allocated, so longer and more fragmented instructions leave less capacity for each constraint, especially those buried in the middle of prompt. Second, complex instructions are rarely parallel either: they branch into conditional sub-requirements and chain tasks, forcing the model to track their composition. Moreover, constraints also interact implicitly, so joint satisfaction demands planning the response as a whole rather than meeting requirements one by one.

To boost LLM's instruction following capability, post-training methods have accordingly moved toward reinforcement learning with verifiable rewards (Dong et al., 2024; Peng et al., 2025; Qin et al., 2025). Specifically, each constraint is paired with a deterministic checker or an LLM judge, and the policy rollouts responses and get trained on the verdicts. Verdicts are often aggregated into a scalar reward: all-or-nothing (AON) rewarding a response only if every constraint passes, and constraint satisfaction rate (CSR) rewarding the fraction satisfied. However, both aggregations are coarse-

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.

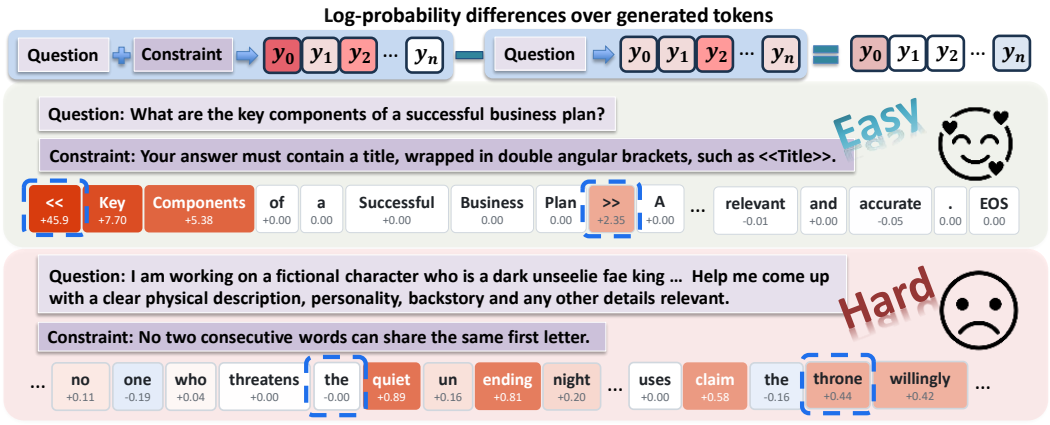


Figure 1: The constraint-ablation contrastive score could reliably attribute credit to constraints within LLM's capability, yet frequently fails to localize the tokens responsible for hard constraints.

grained: when all rollouts share the same scalar reward, the group-relative advantage vanishes and leaves no learning signal. Moreover, compliances or violations often trace back to a handful of pivotal tokens, yet the reward is broadcast uniformly over the response. A good instruction-following reward should meet two criteria: 1) it should *localize* the failed constraint and the responsible token; and 2) it should be *constructive*, indicating what a satisfying response looks like. In this way, training can not only learn from the failures but also expand the model's capability boundaries.

In this paper, we propose to perform constraint-level credit assignment via counterfactual reasoning. Unlike recent work (Xu et al., 2026) that applies an external token-level discriminator, we draw the signal from the policy's own probability shifts. Our premise is simple: a token generated *because* the presence of a constraint should lose probability if the constraint is removed. However, the signal is not always reliable. The reliability of generation probability shifts strongly depends on the model's own capability. As shown in Figure 1, for constraints within the model's capability boundary, the compliance-deciding tokens are produced because of the constraint, so removing it makes exactly their probabilities drop sharply. However, for constraints beyond LLM's capability, no tokens are reliably produced because of them, so the contrastive score becomes unreliable. The two regimes thus call for different treatments in credit assignment, and our method is designed based on it.

Specifically, we propose **LoCo** (**L**eave-**o**ne-out **Co**nstraint for Credit Assignment). For a constraint within the model's capability, LoCo removes it from the prompt and measures how each token's probability responds to this ablation. Tokens the constraint raises earn more reward, while those it lowers receive more penalty. For a constraint beyond the model's capability, we extract the failure reason from the verifier and recompute the probability shifts under an augmented prompt that shows the policy its own response together with this reason. We then combine this contrastive score with the constraint-level advantage and aggregate over constraints to assign credit at the token level. Moreover, a *demonstration-conditioned self-distillation* module uses an external strong model to make minimal edits to the best sampled response to make it satisfy every constraint, and distills the edited response back into the policy. For evaluation, we train Qwen3-4B-Instruct and Qwen3.5-9B on HIR-16K and evaluate on eight instruction following benchmarks. For in-domain benchmarks, LoCo improves the average constraint satisfaction by 2.6%–4.9% over GRPO trained with the same outcome reward. For out-of-domain benchmarks, it gains 3.2%–7.0%, with improvements holding at both model scales and architectures.

2 Related Work

2.1 Instruction Following

Instruction following requires models to satisfy the constraints specified in user instructions, such as formatting, length, and content requirements (Zhou et al., 2023; Jiang et al., 2024). Early approaches build instruction-response pairs and train with supervised fine-tuning or preference optimization (Wang et al., 2023; Xu et al., 2025; Ouyang et al., 2022; Rafailov et al., 2024), yet both generalize poorly to unseen or compositional constraints. More recently, reinforcement learning with verifiable

rewards constructs instruction data paired with executable checkers or LLM judges, and optimizes the policy against them (Dong et al., 2024; Peng et al., 2025; Qin et al., 2025; Zeng et al., 2026).

These RLVR pipelines, however, compress each response into one scalar reward, so the model cannot tell what failed or where. To address this, HiR (Zhang et al., 2025a) replays failed attempts as successes by rewriting the instruction to the constraints already satisfied, yet its sequence-level label still fail to tell where the failure lies; RTT (Xu et al., 2026) attributes each rubric to responsible tokens with a trained discriminator, yet it requires an auxiliary annotated model.

2.2 Credit Assignment

Post-training RL methods for LLMs differ mainly in the granularity estimation of advantage. PPO operates at the token level but must train a separate critic (Schulman et al., 2017), which makes the training instable; GRPO (Shao et al., 2024) and its extensions such as DAPO (Yu et al., 2025) and GSPO (Zheng et al., 2025a) discard the critic and instead broadcast one group-relative scalar across an entire response. Between these extremes lies a spectrum of intermediate supervision: process reward models trained on human step labels (Lightman et al., 2023), Monte Carlo estimation (Luo et al., 2024; Chen et al., 2024) or implicit rewards from outcome-trained models (Yuan et al., 2024).

A recent line of work assign token-level reward via policy-derived signals: e.g., entropy and surprisal concentrate (Simoni et al., 2025; Luo et al., 2026; Huang et al., 2025), attention entropy (Li et al., 2026), or gradient- and mechanism-based attribution (Zhang et al., 2026; Sun et al., 2026). These signals reflect the policy's uncertainty or a token's mechanical influence, which requires no auxiliary reward model or human step annotations.

3 Preliminary and Motivation

3.1 GRPO with Verifiable Constraints for Instruction Following

Let an instruction x decompose into K atomic, verifiable constraints, $x = \{c_1, \ldots, c_K\}$, for a rollout $y \sim \pi_{\theta_{\text{old}}}(\cdot \mid x)$, deterministic checkers (rule or LLM-based) return binary satisfaction labels $s_k(y) \in \{0, 1\}$, one per constraint. GRPO (Shao et al., 2024) samples a group of G rollouts $\{y_i\}_{i=1}^{G}$ for the same instruction and optimizes the clipped surrogate:

$$\mathcal{J}(\theta) = \mathbb{E}\Big[\sum_t \min\big(\rho_{i,t}\,\hat{A}_i,\ \text{clip}(\rho_{i,t}, 1-\epsilon, 1+\epsilon)\,\hat{A}_i\big)\Big], \qquad \rho_{i,t} = \frac{\pi_\theta(y_{i,t} \mid x, y_{i,<t})}{\pi_{\theta_{\text{old}}}(y_{i,t} \mid x, y_{i,<t})}, \quad (1)$$

with the group-normalized advantage $\hat{A}_i = (r_i - \mu_G)/\sigma_G$ computed from an outcome-level reward r_i. We use verifiable outcome rewards: either all-or-nothing satisfaction (AON), $r_i = \prod_k s_k(y_i)$, or the constraint satisfaction rate (CSR), $r_i = \frac{1}{K}\sum_k s_k(y_i)$.

However, compliance is typically decided by a handful of pivotal tokens, yet both rewards broadcast the same scalar advantage uniformly across every token in response. A few constraint families such as keyword presence admit straightforward localization, but most of the constraints do not explicitly reveal which tokens decide their verdict. Delegating credit assignment to an external model introduces distribution shift and reward hacking as the policy updates (Zheng et al., 2025b), and value-network credit is systematically biased relative to policy-derived estimates (Cui et al., 2025; Yuan et al., 2025). Therefore, in this section, we ask whether the policy's own distribution can supply the missing token-level signal, and whether such self-supplied credit is reliable.

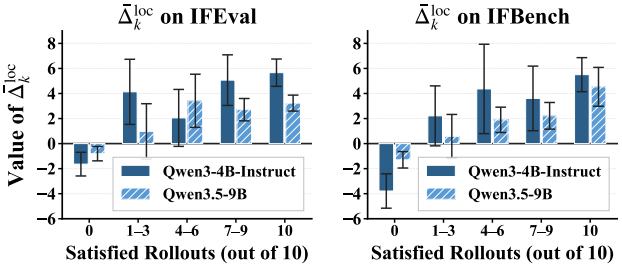
3.2 Do LLMs Know Where Constraints Bite?

Our hypothesis is simple: tokens that implement a constraint c_k derive their probability from its presence, so removing c_k from the prompt should lower exactly their probabilities. We quantify this with the *leave-one-out contrast* between the full prompt x and its ablated counterpart x^{-k}:

$$\Delta_k(y, t) = \log \pi_\theta(y_t \mid x, y_{<t}) - \log \pi_\theta(y_t \mid x^{-k}, y_{<t}). \quad (2)$$

A large positive $\Delta_k(y, t)$ thus marks token y_t as produced *because of* c_k.

The same expectation extends to violated constraints: if the policy knows where it fails, the constraint's presence should suppress the offending tokens rather than boost them. We test both sides on



	IFEval		IFBench	
SR(k/10)	Greedy	CAD	Greedy	CAD
0	0.0	0.0	0.6	0.0
1 - 3	20.9	33.3	9.1	18.2
4 - 6	55.0	70.0	75.0	81.2
7 - 9	78.2	92.7	81.8	95.5
10	99.7	100	98.3	100
Overall	82.8	85.2	31.0	33.0

Figure 2: **Left**: $\bar{\Delta}_k^{\text{loc}}$ for compliant and offending tokens. **Right**: Instruction Accuray on IFEval and IFBench under greedy decoding vs. context-aware decoding (CAD).

IFEval and IFBench, restricting the analysis to constraints whose responsible tokens admit precise localization, including compliant tokens $\mathcal{T}_k^+$ that carry out a satisfied constraint ($s_k(y) = 1$), and offending tokens $\mathcal{T}_k^-$ that break a violated one ($s_k(y) = 0$). For each constraint c_k in instruction, we sample ten rollouts and aggregate the contrast over the located tokens in both cases:

$$\bar{\Delta}_k^{\text{loc}} = \frac{\sum_{t\in\mathcal{T}_k^+} \Delta_k(y,t) - \sum_{t\in\mathcal{T}_k^-} \Delta_k(y,t)}{|\mathcal{T}_k^+| + |\mathcal{T}_k^-|}. \tag{3}$$

A positive $\bar{\Delta}_k^{\text{loc}}$ means that the constraint indeed boosts its compliant tokens or suppresses its offending ones. We utilize the number of satisfying rollouts among the ten samples to anchor the difficulty of c_k relative to the policy's capability. As shown in Figure 2 (left), $\bar{\Delta}_k^{\text{loc}}$ is consistently positive when most rollouts satisfy the constraint, and turns negative when almost all of them fail. Therefore, we conclude that the policy knows where to satisfy the constraint only for constraints within its capability. We leave further discussion in Appendix A.

Context-aware Decoding as Training-free Method. If the policy knows which direction to work in, we can pull it further along that direction at decode time. As long as the direction is correct, such amplification should improve instruction following. To this end, we learn from context-aware decoding (CAD) (Shi et al., 2023) and rescore each candidate token as:

$$s_\alpha(w) = \log p_{\text{x}^+}(w) + \alpha\big[\log p_{\text{x}^+}(w) - \log p_{\text{x}^-}(w)\big], \tag{4}$$

where p_{x^+} and p_{x^-} are the next-token distributions under the full prompt and its constraint-ablated counterpart. We similarly classify them with satisfied rollouts numbers. As shown in Figure 2 (right), a conservative α=0.2 consistently improves over greedy decoding on prompts within the model's capability (SR>0), lifting overall accuracy on both IFEval and IFBench, while for prompts beyond its capability (SR=0) the gain vanishes and even reverses on IFBench. This indicates that the signal provided by Eq. 2 localizes the responsible tokens reliably only when the constraint lies within the model's capability. We further discuss it in Appendix B.

4 Methodology

4.1 Constraint-wise Credit Assignment

Constraint-level Advantage. For each prompt x with constraint set $\mathcal{C}(x) = \{c_1, \ldots, c_K\}$, we sample a group of rollouts $\mathcal{G}(x) = \{y_1, \ldots, y_G\}$ from the current policy π_θ and score every response with a constraint verifier that returns per-constraint satisfaction labels $s_k(y) \in \{0, 1\}$. For each constraint, we define a *constraint-level* advantage within the group with:

$$A_k(y) = \frac{s_k(y) - \mu_k}{\sigma_k + \epsilon}, \qquad \mu_k = \frac{1}{G}\sum_{j=1}^{G} s_k(y_j), \quad \sigma_k^2 = \frac{1}{G}\sum_{j=1}^{G}\big(s_k(y_j) - \mu_k\big)^2. \tag{5}$$

However, the normalization degenerates whenever the within-group variation $\sigma_k = 0$, i.e., when a constraint is satisfied by every rollout or violated by all of them, in which case A_k vanishes and the constraint is dropped out of training even though it may still be valuable. To this end, we restore a minimal signal by injecting a small pseudo-advantage as follows:

$$\widetilde{A}_k = \begin{cases} +\gamma^+ & \text{if } s_k(y) = 1 \text{ for all } y \in \mathcal{G}(x), \\ -\gamma^- & \text{if } s_k(y) = 0 \text{ for all } y \in \mathcal{G}(x). \end{cases} \tag{6}$$

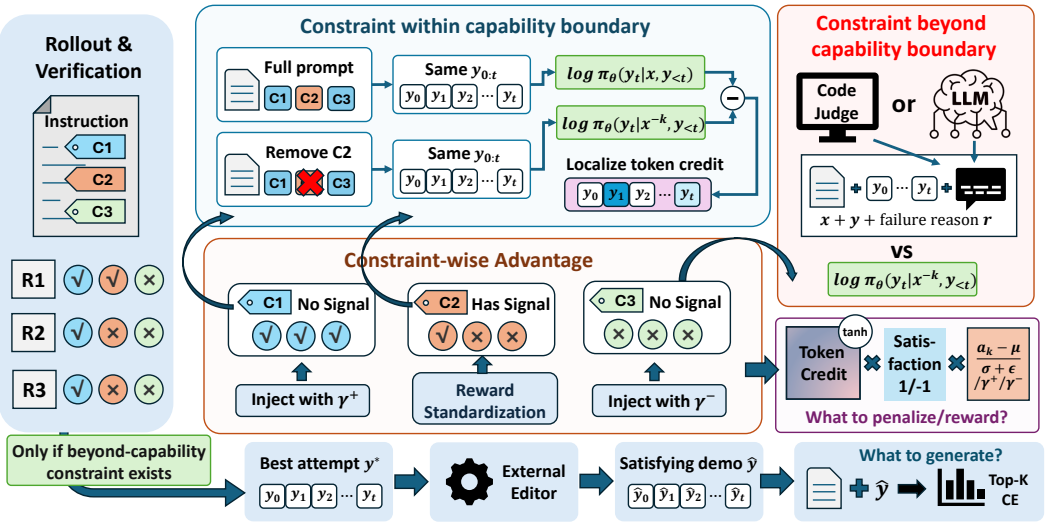


Figure 3: Framework of our proposed LoCo, with focuses on how reward credit is assigned differently to constraints falling inside or outside LLM's capability boundary.

The magnitudes are dynamic for each prompt: with N^+ all-satisfied and N^- all-failed constraints:

$$\gamma^+ \;=\; \gamma\, N^- / (N^+ + N^-), \qquad \gamma^- \;=\; \gamma\, N^+ / (N^+ + N^-), \tag{7}$$

The injected advantage keeps zero-variance constraints active in the credit computation. Moreover, the design ensures $N^+\gamma^+ = N^-\gamma^-$, so the injected advantages sum to zero within each prompt, preventing the overall advantage from drifting and thereby implicitly raising the reward baseline.

Capability-aware Contrast Calculation. As discussed, the constraint-level advantage fails to indicate *where* the constraint is decided. To localize the credit, we resort to the leave-one-out probability contrast, which measures how strongly each token's probability depends on the constraint's presence. As stated in Section 3.2, this contrast faithfully localizes the responsible tokens for constraints within the policy's capability, yet becomes unreliable for those beyond it. We therefore partition the constraint set of each prompt into *within-capability* and *beyond-capability* constraints, by whether the rollout group exhibits any viable strategy for the constraint:

$$\mathcal{C}^{\text{in}}(x) \;=\; \{\, c_k \in \mathcal{C}(x) \;\mid\; \max_{y \in \mathcal{G}(x)} s_k(y) = 1 \,\}, \qquad \mathcal{C}^{\text{out}}(x) \;=\; \mathcal{C}(x) \setminus \mathcal{C}^{\text{in}}(x), \tag{8}$$

and compute the contrast for the two regimes as:

$$\Delta_k(y,t) \;=\; \begin{cases} \log \pi_\theta\big(y_t \mid x,\, y_{<t}\big) \;-\; \log \pi_\theta\big(y_t \mid x^{-k},\, y_{<t}\big), & c_k \in \mathcal{C}^{\text{in}}(x), \\ \log \pi_\theta\big(y_t \mid x^{k}_{\text{cf}},\, y_{<t}\big) \;-\; \log \pi_\theta\big(y_t \mid x^{-k},\, y_{<t}\big), & c_k \in \mathcal{C}^{\text{out}}(x), \end{cases} \tag{9}$$

where the first branch restates the leave-one-out contrast of Eq. 2, under which the probability shift correctly localizes the tokens that implement c_k. For the second branch, the raw contrast becomes unreliable because the tokens to which the policy assigns higher probability are not necessarily those that would lead to satisfaction. We therefore repair it by conditioning the policy on evidence of its own failure, and construct a counterfactual context for each constraint $c_k \in \mathcal{C}^{\text{out}}(x)$ and rollout y:

$$x^k_{\text{cf}} \;=\; x \;\oplus\; y \;\oplus\; \phi(c_k, r_k), \tag{10}$$

here ϕ is a short feedback block stating that **why** c_k is violated and requesting a minimal rewrite, and r_k is a natural-language failure reason produced by a deterministic verbalizer that exposes the verifier's internal measurement (e.g., "the response contains 150 words; but at least 300 are required"); for semantic constraints (e.g., style, tone, or persona) without programmatic evidence, the judge model supplies r_k. Since the rewrite instruction inflates the probabilities globally, we further center the repaired contrast by its position-wise mean:

$$\widetilde{\Delta}_k(y,t) \;=\; \Delta_k(y,t) \;-\; \frac{1}{|y|} \sum\nolimits_{t'=1}^{|y|} \Delta_k(y,t'), \tag{11}$$

and use $\widetilde{\Delta}_k$ in place of Δ_k when forming the weight for $c_k \in \mathcal{C}^{\text{out}}(x)$. With concrete evidence of why it failed, the policy's probability shift concentrates on the tokens responsible for the violation.

Token-level advantage. After acquiring the contrast, we map it to a per-token weight as:

$$w_k(y,t) \;=\; \big(2\,s_k(y)-1\big)\,\tanh\big(\Delta_k(y,t)/\tau\big), \tag{12}$$

The bounded nonlinearity nature of $\tanh$ preserves the sign of the contrast while capping the influence of extreme tokens. Our credit assignment mechanism ensures that tokens the policy deems worthy of elevated probability receive greater reward when constraints are satisfied, while tokens the policy seeks to suppress but still persist in the output incur amplified penalty upon failure. The token-level advantage $A(y_i,t)$ deposits each constraint's advantage as follows:

$$A_{\text{tok}}(y,t)=\frac{1}{K}\sum_{k=1}^{K}A_k(y)\,w_k(y,t), \qquad A(y_i,t)=A_{\text{out}}(y_i)+\lambda\,A_{\text{tok}}(y_i,t), \tag{13}$$

where A_{out} is the outcome advantage computed from either the all-or-nothing satisfaction indicator or the constraint satisfaction rate, and λ controls the strength of the token-level correction. The design is deliberately conservative: the outcome advantage remains the leading term, so the token-level signal can redistribute credit within a response and will never override the outcome too much.

4.2 Expanding the Capability Boundary via Self-Distillation

For constraints beyond the capability boundary, the repaired contrast tells the policy *where* it went wrong, but LLM still fails to learn *what* a correct response should look like, as an all-failed group offers no positive example. We therefore distill a positive signal from a demonstration-conditioned teacher. Let $y^\star=\arg\max_{y\in\mathcal{G}(x)}\sum_k s_k(y)$ be the best rollout in the group; an external editor model minimally revises it into a demonstration $\widehat{y}$ that satisfies all failed constraints, which is appended to the prompt as an in-context demonstration, $\widetilde{x}=x\oplus\widehat{y}$. The teacher is the frozen reference policy conditioned on $\widetilde{x}$, which exploits the base model's in-context learning ability and remains fixed throughout training, while the policy is always scored under the original prompt x. At each position t, we retain the teacher's top-K tokens $\mathcal{V}_t=\mathrm{TopK}\big(\pi_{\text{ref}}(\cdot\mid\widetilde{x},\,y^\star_{<t}),\,K\big)$ with stop-gradient masses, and minimize the truncated cross-entropy as follows:

$$\mathcal{L}_{\text{dist}}(\theta)\;=\;\frac{1}{|y^\star|}\sum_{t=1}^{|y^\star|}\sum_{v\in\mathcal{V}_t}\bar{p}_t(v)\,\Big(\log\big[\,\mathrm{sg}\big(\pi_{\text{ref}}(v\mid\widetilde{x},\,y^\star_{<t})\big)\big]\;-\;\log\pi_\theta(v\mid x,\,y^\star_{<t})\Big). \tag{14}$$

The self-distillation loss is applied only when the instruction contains constraints beyond LLM's capability boundary, so that the teacher's demonstration-conditioned continuations are distilled into the policy exactly where it lacks a positive example, turning the best attempt $y^\star$ into a positive target that raises the capability ceiling.

4.3 Overall Objective

Different modules enter a single policy-gradient objective. For rollout y_i of prompt x, we optimize:

$$J(\theta)=\mathbb{E}_{x,\,\{y_i\}}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|y_i|}\sum_{t=1}^{|y_i|}\min\Big(\rho_{i,t}\,A(y_i,t),\;\mathrm{clip}(\rho_{i,t},1\pm\varepsilon)\,A(y_i,t)\Big)\right]-\eta\,\mathcal{L}_{\text{dist}}(\theta), \tag{15}$$

with $\rho_{i,t}=\pi_\theta(y_{i,t}\mid x,y_{i,<t})\,/\,\pi_{\theta_{\text{old}}}(y_{i,t}\mid x,y_{i,<t})$ the token-level importance ratio. The advantage $A(y_i,t)=A_{\text{out}}(y_i)+\lambda A_{\text{tok}}(y_i,t)$ deposits constraint-wise credit as in Eq. 13, where the contrast Δ_k that forms $w_k(y,t)$ is routed by the capability dichotomy of Section 4.1. The distillation term $\mathcal{L}_{\text{dist}}$ (Eq. 14) acts on exactly the region where the contrast cannot supply a positive target. Setting $\lambda=\eta=0$ recovers vanilla GRPO with an outcome reward exactly.

5 Experiments

5.1 Experimental Setup

Training Settings. We use HIR-16K (Zhang et al., 2025a) as the training dataset, which is a corpus of multi-constraint prompts whose constraints come from four families: the rule checkers of IFEval,

Table 1: Main results on eight instruction-following benchmarks (%). ID Average / OOD Average are macro-averages over the corresponding benchmarks. Subscripts denote *relative* improvement over the GRPO baseline $(\text{avg} - \text{avg}_{\text{GRPO}})/\text{avg}_{\text{GRPO}}$. **Best** in bold, second best underlined.

Method	In-Domain				Out-of-Domain					
	IFEval	IFB	MulDim	**Average**	CFB	InFo	FollowB	AdvIF	RoleM	**Average**
Qwen3-4B-Instruct										
Base	82.8	31.0	57.3	57.1	45.8	45.2	67.8	39.8	38.4	47.4
SFT	80.8	32.7	64.2	59.2	45.3	45.8	67.2	40.5	37.6	47.3
CSR as reward										
GRPO	84.5	35.3	80.8	66.9$_{-0.0}$	46.4	46.0	69.1	42.1	41.4	49.0$_{-0.0}$
Replay	86.3	35.7	82.0	68.0$_{\uparrow 1.7}$	46.0	48.4	70.3	43.2	43.0	50.2$_{\uparrow 2.4}$
R2T	86.7	39.7	81.2	69.2$_{\uparrow 3.5}$	47.0	48.8	69.9	42.6	42.3	50.1$_{\uparrow 2.3}$
LoCo (Ours)	**87.1**	**40.0**	**83.4**	**70.2**$_{\uparrow 4.9}$	**50.5**	**50.8**	**71.7**	**45.1**	**44.0**	**52.4**$_{\uparrow 7.0}$
AON as reward										
GRPO	84.8	36.3	80.6	67.2$_{-0.0}$	46.8	48.0	70.1	41.9	40.7	49.5$_{-0.0}$
Replay	85.8	35.7	79.7	67.1$_{\downarrow 0.2}$	47.3	**49.2**	70.7	43.5	41.4	50.4$_{\uparrow 1.9}$
R2T	84.7	37.7	**81.8**	68.1$_{\uparrow 1.2}$	46.2	49.0	70.4	42.5	42.0	50.0$_{\uparrow 1.1}$
LoCo (Ours)	**86.7**	**40.3**	81.2	**69.4**$_{\uparrow 3.2}$	**48.6**	49.0	**71.2**	**44.3**	**43.8**	**51.4**$_{\uparrow 3.8}$
Qwen3.5-9B										
Base	83.9	35.0	59.5	59.5	46.5	44.8	70.3	50.5	48.1	52.0
SFT	83.2	36.3	66.9	62.1	45.8	47.2	69.9	49.3	48.3	52.1
CSR as reward										
GRPO	86.7	38.3	83.5	69.5$_{-0.0}$	48.5	47.0	72.4	53.2	49.1	54.0$_{-0.0}$
Replay	87.4	39.7	85.5	70.9$_{\uparrow 2.0}$	47.8	**49.4**	73.6	53.7	49.4	54.8$_{\uparrow 1.4}$
R2T	87.1	40.0	84.0	70.4$_{\uparrow 1.2}$	49.5	47.6	74.5	51.8	48.6	54.4$_{\uparrow 0.7}$
LoCo (Ours)	**88.4**	**41.3**	**86.4**	**72.0**$_{\uparrow 3.6}$	**51.0**	49.2	**75.4**	**55.4**	**50.0**	**56.2**$_{\uparrow 4.0}$
AON as reward										
GRPO	86.9	38.7	84.1	69.9$_{-0.0}$	48.7	47.0	72.8	51.6	49.6	53.9$_{-0.0}$
Replay	86.0	39.3	85.7	70.3$_{\uparrow 0.6}$	**50.6**	48.0	72.3	52.4	49.9	54.6$_{\uparrow 1.3}$
R2T	86.7	**41.0**	85.0	70.9$_{\uparrow 1.4}$	49.0	47.4	73.1	51.2	**50.2**	54.2$_{\uparrow 0.4}$
LoCo (Ours)	**89.1**	40.0	**86.0**	**71.7**$_{\uparrow 2.6}$	**50.6**	**49.6**	**73.6**	**54.5**	50.1	**55.7**$_{\uparrow 3.2}$

IFBench, and MulDimIF, with an extended set of open-ended constraints from Chatbot Arena. All methods are trained with verl (Sheng et al., 2025) on 8 A100 GPUs for 500 steps. We use Qwen3.7-max as the LLM for judge and minimal-edit repair.

Benchmarks. We evaluate on eight instruction-following benchmark, and split them by distance to the training distribution. The *in-domain* (ID) set, including IFEval (Zhou et al., 2023), IFBench (Pyatkin et al., 2025), and MulDimIF (Ye et al., 2026), which use the same constraint families and rule checkers from training data. The *out-of-distribution* (OOD) set contains: CFBench (Zhang et al., 2025b), InFoBench (Qin et al., 2024), FollowBench (Jiang et al., 2024), AdvancedIF (He et al., 2025) and RoleMRC (Lu et al., 2025), with details in Appendix C. We evaluate the performance for each method by reporting its instruction-level satisfaction rate, i.e., the percentage of prompts that satisfy **all** given constraints. We use Deepseek-v4-pro as the judge LLM for all evaluations.

Base models and baselines. We consider two base models of different scales and architectures, including Qwen3-4B-Instruct (Yang et al., 2025) and Qwen3.5-9B (Yang et al., 2025), with details in Appendix D. We compare LoCo with four training methods on each: (i) *SFT*, with golden responses generated by Deepseek-v4-pro; (ii) *GRPO* (Shao et al., 2024); (iii) *Replay* (Zhang et al., 2025a), which replays failed attempts as successes based on the constraints that have been satisfied in hindsight; and (iv) *R2T* (Xu et al., 2026), which trains an auxiliary model for credit assignment. All methods share the same training configurations in Appendix E.

5.2 Main Results

Our main results are reported in Table 1. To probe our method under outcome rewards of different granularity, we attach the token-level credit to both CSR and AON aggregations, each of which serves as A_{out} in Eq.13. Overall, LoCo attains the best ID and OOD averages on both model scales and under both aggregations. Relative to the GRPO backbone trained with the same verifiable outcome reward, LoCo lifts the in-domain average by up to 4.9% and the out-of-domain average by up to 7.0%. The gain is systematically larger under CSR than under AON. We attribute this to the information each baseline discards: CSR turns a partially-satisfied rollout into a non-zero yet position-blind scalar, so per-token localization is exactly what the reward lacks, whereas AON

Table 3: ISR Results (%) for LoCo with different RLVR methods. (Qwen3-4B-Instruct with CSR)

Method	In-Domain				Out-of-Domain					
	IFEval	IFB	MulDim	**Average**	CFB	InFo	FollowB	AdvIF	RoleM	**Average**
PPO	85.2	35.3	77.5	66.0$_{-0.0}$	**47.6**	46.6	67.5	41.3	39.5	48.5$_{-0.0}$
+LoCo	**87.1**	**40.3**	**81.9**	**69.8**$_{\uparrow 5.8}$	47.3	**49.4**	**70.3**	**44.0**	**43.4**	**50.9**$_{\uparrow 4.9}$
DAPO	84.7	35.0	81.3	67.0$_{-0.0}$	45.1	49.2	70.2	41.6	39.9	49.2$_{-0.0}$
+LoCo	**86.7**	**43.3**	**81.4**	**70.5**$_{\uparrow 5.2}$	**46.9**	**52.0**	**72.2**	**44.2**	**44.1**	**51.9**$_{\uparrow 5.5}$
GSPO	86.9	36.7	82.2	68.6$_{-0.0}$	46.7	46.8	67.2	43.3	41.2	49.0$_{-0.0}$
+LoCo	**87.1**	**40.7**	**83.2**	**70.3**$_{\uparrow 2.5}$	**48.2**	**50.0**	**69.1**	**44.8**	**41.6**	**50.7**$_{\uparrow 3.5}$

already concentrates its group-relative advantage on the few all-satisfied rollouts, leaving less for a token-level correction to add.

Preservation of general reasoning abilities. To check whether optimizing for instruction-following behavior erodes broader problem-solving competence, we evaluate the policy on three reasoning benchmarks that are orthogonal to instruction following, including MATH500 Lightman et al. (2023) for mathematics, HumanEval Chen et al. (2021) for coding, and MMLU-Pro Wang et al. (2024) for general tasks. As shown in Table 2, no consistent decline emerges. This echoes the conclusion of Zhang et al. (2025a), which likewise observe that instruction-following training leaves general capabilities essentially intact. This indicates that our fine-grained credit assignment introduces no additional cost to general ability.

Table 2: General-capability(%).

	Math500	HEval	MMLU-P
Qwen3-4B-Instruct			
Base	89.0	73.8	66.7
+LoCo	89.2$_{\uparrow 0.2}$	76.2$_{\uparrow 3.3}$	66.5$_{\downarrow 0.3}$
Qwen3.5-9B			
Base	87.4	85.4	76.4
+LoCo	87.6$_{\uparrow 0.2}$	84.2$_{\downarrow 1.4}$	76.8$_{\uparrow 0.5}$

Generality across RLVR methods. We further test whether LoCo can transfer to other RLVR methods. LoCo's modification sits mainly at the advantage level, leaving GRPO variants like DAPO and GSPO free to aggregate these advantages into its own objective. In contrast, PPO estimates its advantage from a learned critic via GAE rather than from a rollout group, so there is theoretically no group where per-constraint credit can be centered. We instead replace each constraint's advantage by its satisfaction sign and form the averaged constraint-level advantages onto the GAE advantage. As shown in Table 3, LoCo lifts both the in-domain and out-of-domain averages for all three methods.

5.3 Ablation Study

We ablate the mechanisms of LoCo, and retrain Qwen3-4B-Instruct under both CSR and AON, with ID/OOD averages reported. Specifically, **w/o C-level Advantage** variant drops the per-constraint decomposition of Eq. 5 and computes token-level advantages over all constraints as a union; **w/o evidence** removes the counterfactual repair of Eq. 10, so constraints beyond the policy's capability are scored by the raw leave-one-out contrast; and **w/o SDFT** deletes the self-distillation of Eq. 14, leaving all-failed groups only the negative repair signal. As shown in Table 4, every module contributes to the final performance, with the constraint-level advantage being the most critical and SDFT having a relatively smaller impact.

Table 4: Ablation study on Qwen3-4B-Instruct (%).

Variant	CSR		AON	
	ID Avg	OOD Avg	ID Avg	OOD Avg
LoCo (full)	**70.2**	**52.4**	**69.4**	**51.4**
w/o C-level Adv.	68.3$_{\downarrow 2.7}$	50.6$_{\downarrow 3.4}$	68.5$_{\downarrow 1.3}$	49.8$_{\downarrow 3.1}$
w/o evidence	69.2$_{\downarrow 1.4}$	51.0$_{\downarrow 2.7}$	69.1$_{\downarrow 0.4}$	51.0$_{\downarrow 0.8}$
w/o SDFT	69.8$_{\downarrow 0.6}$	52.1$_{\downarrow 0.6}$	68.9$_{\downarrow 0.7}$	51.2$_{\downarrow 0.4}$

5.4 Hyperparameter Analysis

We examine the sensitivity of LoCo to its four specific hyperparameters, including the token-credit weight λ (Eq. 13), the contrast temperature τ (Eq. 12), the injection magnitude γ (Eq. 7), and the self-distillation weight η (Eq. 15). We train Qwen3-4B-Instruct on 4,000 randomly sampled HIR-16K prompts for 3 epochs and report the in-domain ISR in Figure 4. We vary one hyperparameter while holding the others at their defaults, and report results under both CSR and AON outcome-reward so as to expose how our token-level credit interacts with the sparsity of the leading reward.

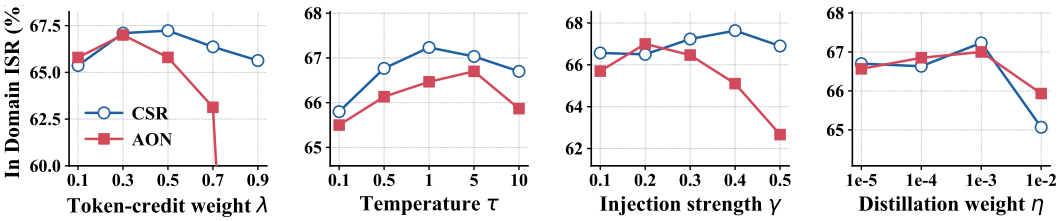
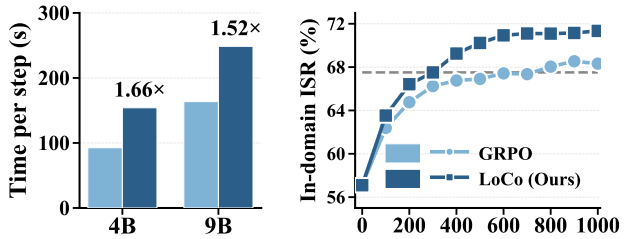


Figure 4: Hyperparameter analysis for LoCo. In-domain ISR (%) of LoCo on Qwen3-4B-Instruct as each of the four hyperparameters is swept under CSR and AON rewards.

Specifically, for token-credit weight λ, its effect is limited when CSR serves as the primary reward, while AON collapses from 67.0 at λ=0.3 to 21.2 at λ=0.9 once the auxiliary term overwhelms the outcome reward. The temperature τ attains a flat optimum at 1.0-5.0, with smaller values driving the tanh gate toward a near-binary sign that discards contrast magnitude, whereas larger values linearize it and attenuate the credit. The injection γ has little effect on CSR, but AON peaks early at γ=0.2 and declines sharply, since over-injection of pseudo-advantage without group-wise comparison introduces noise. The distillation weight η improves performance slowly below 10^{-3}, and decays once η either competes with the policy gradient. Overall, the CSR scheme is more robust to hyperparameter variations, whereas AON is comparatively more sensitive.

5.5 Further Discussion



		AdvIF	CFB
Qwen3.7-max	GRPO	26.6	37.6
	LoCo	28.4$_{\uparrow 6.8}$	40.6$_{\uparrow 8.0}$
Qwen3.8-max	GRPO	30.0	34.7
	LoCo	31.8$_{\uparrow 6.0}$	36.9$_{\uparrow 6.3}$
DeepSeek-V3.2	GRPO	44.3	54.2
	LoCo	46.4$_{\uparrow 4.7}$	55.4$_{\uparrow 2.2}$

Figure 5: **Left**: Training efficiency analysis. **Right**: ISR (%) under different LLM judges.

Training cost and efficiency. LoCo introduces additional modules, which require extra forward passes and make each optimization step slower than vanilla GRPO (as shown in Figure 5 left). This cost is quickly repaid: LoCo at 300 steps already surpasses GRPO at 600 steps, and GRPO never catches up to LoCo's 400-step result even after its 1000 steps training. LoCo thus reaches a higher ceiling while being more effective under a matched wall-clock budget.

Robustness to the judge model. Constraints such as stylistic or semantic requirements are supposed to be judged by externals LLM. To verify that LoCo remains strong across judges, we re-score the same policies under three different judges on two fully LLM-judged benchmarks (as shown in Figure 5 right). LoCo attains a higher instruction satisfaction rate than GRPO under every judge, showing that its gains are stable rather than an artifact of a particular verifier.

6 Conclusion

Reinforcement learning with verifiable rewards is now the standard recipe for instruction following, yet its pipelines compress constraint verdicts into a single scalar broadcast uniformly over the response, leaving the model unaware of which parts were right or wrong. We asked whether the policy's own generation distribution can supply this missing signal, and found a clean dichotomy: removing an in-capability constraint makes exactly its responsible tokens change their probability, whereas for out-of-capability constraints the same contrast turns unreliable. Based on this, we propose LoCo, which routes each in-capability constraint's advantage to its responsible tokens via the leave-one-out probability shift, and repairs the contrast for out-of-capability constraints by re-conditioning the policy on verbalized failure evidence. Across eight instruction-following benchmarks, LoCo consistently outperforms GRPO and strong RLVR baselines, with gains holding across model scales, RL algorithms, and LLM judges. We discuss limitations in Appendix G.

AI USE STATEMENT

In this work, LLM were used as assistive tools. Deepseek-v4-flash was employed for text polishing and writing refinement, and GLM-5.2 was used to assist with code generation. All LLM-assisted content was reviewed and verified by the authors.

ETHICS STATEMENT

This work studies post-training methods for instruction following and does not involve human subjects, crowdsourced annotation, or the collection of new personal data. All training data, evaluation benchmarks, and backbone models are publicly available and are used in accordance with their licenses, and the external LLM services used for judging and response editing are accessed under the providers' terms of use. Our pipeline relies on LLM judges only where deterministic checkers are unavailable, and failure evidence is verbalized by rule-based templates whenever possible, which limits the propagation of judge biases into training. We do not foresee uses of this work that pose risks beyond those inherent to general-purpose language models.

REPRODUCIBILITY STATEMENT

We have made every effort to ensure that our results are reproducible. The training corpus (HIR-16K), the eight evaluation benchmarks, and both backbone models are publicly available, and our pipeline is built upon the open-source verl framework. The experimental setup, including the judge models and evaluation protocol, is described in Section 5.1, with dataset details in Appendix C and backbone configurations in Appendix D. All training hyperparameters are listed in Table 8 and their sensitivity is analyzed in Section 5.4, while the prompts for failure-reason verbalization, counterfactual repair, and external minimal-edit repair are provided in Appendix H.

We provide a zip archive of the code in the **supplementary material**. The training environment consists of a single node with 8 GPUs (A100 80GB), Python 3.10+, PyTorch 2.10, CUDA 12.8, verl (0.8.0.dev0, installed from source), and vLLM 0.19. The exact versions of all required Python libraries are pinned in `requirements.txt`, and `README.md` provides instructions on how to run the training and evaluation pipeline.

REFERENCES

Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. Alphamath almost zero: Process supervision without process, 2024. URL https://arxiv.org/abs/2405.03553.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Jiacheng Chen, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, Jiarui Yuan, Huayu Chen, Kaiyan Zhang, Xingtai Lv, Shuo Wang, Yuan Yao, Xu Han, Hao Peng, Yu Cheng, Zhiyuan Liu, Maosong Sun, Bowen Zhou, and Ning Ding. Process reinforcement through implicit rewards, 2025. URL https://arxiv.org/abs/2502.01456.

Guanting Dong, Keming Lu, Chengpeng Li, Tingyu Xia, Bowen Yu, Chang Zhou, and Jingren Zhou. Self-play with execution feedback: Improving instruction-following capabilities of large language models, 2024. URL https://arxiv.org/abs/2406.13542.

Yun He, Wenzhe Li, Hejia Zhang, Songlin Li, Karishma Mandyam, Sopan Khosla, Yuanhao Xiong, Nanshu Wang, Xiaoliang Peng, Beibin Li, Shengjie Bi, Shishir G. Patil, Qi Qi, Shengyu Feng, Julian Katz-Samuels, Richard Yuanzhe Pang, Sujan Gonugondla, Hunter Lang, Yue Yu, Yundi Qian, Maryam Fazel-Zarandi, Licheng Yu, Amine Benhalloum, Hany Awadalla, and Manaal Faruqui. Advancedif: Rubric-based benchmarking and reinforcement learning for advancing llm instruction following, 2025. URL https://arxiv.org/abs/2511.10507.

Chengyu Huang, Zhengxin Zhang, and Claire Cardie. Hapo: Training language models to reason concisely via history-aware policy optimization, 2025. URL https://arxiv.org/abs/2505.11225.

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, 2024. URL https://arxiv.org/abs/2310.20410.

Gengyang Li, Zheng-Fan Wu, Siqi Bao, and Yunfang Wu. Not all tokens learn alike: Attention entropy reveals heterogeneous signals in rl reasoning, 2026. URL https://arxiv.org/abs/2605.07660.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023. URL https://arxiv.org/abs/2305.20050.

Junru Lu, Jiazheng Li, Guodong Shen, Lin Gui, Siyu An, Yulan He, Di Yin, and Xing Sun. Rolemrc: A fine-grained composite benchmark for role-playing and instruction-following, 2025. URL https://arxiv.org/abs/2502.11387.

Haipeng Luo, Qingfeng Sun, Songli Wu, Can Xu, Wenfeng Deng, Han Hu, and Yansong Tang. Stare: Surprisal-guided token-level advantage reweighting for policy entropy stability, 2026. URL https://arxiv.org/abs/2606.19236.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, and Abhinav Rastogi. Improve mathematical reasoning in language models by automated process supervision, 2024. URL https://arxiv.org/abs/2406.06592.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022. URL https://arxiv.org/abs/2203.02155.

Hao Peng, Yunjia Qi, Xiaozhi Wang, Bin Xu, Lei Hou, and Juanzi Li. Verif: Verification engineering for reinforcement learning in instruction following, 2025. URL https://arxiv.org/abs/2506.09942.

Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hannaneh Hajishirzi. Generalizing verifiable instruction following, 2025. URL https://arxiv.org/abs/2507.02833.

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, 2024. URL https://arxiv.org/abs/2401.03601.

Yulei Qin, Gang Li, Zongyi Li, Zihan Xu, Yuchen Shi, Zhekai Lin, Xiao Cui, Ke Li, and Xing Sun. Incentivizing reasoning for advanced instruction-following of large language models, 2025. URL https://arxiv.org/abs/2506.01413.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model, 2024. URL https://arxiv.org/abs/2305.18290.

MohammadHossein Rezaei, Anas Mahmoud, Zihao Wang, Utkarsh Tyagi, Advait Gosai, Razvan-Gabriel Dumitru, Aakash Sabharwal, Bing Liu, and Yunzhong He. Rubric-guided self-distillation: Post-training without rubric verifiers, 2026. URL https://arxiv.org/abs/2606.12507.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, EuroSys '25, pp. 1279–1297. ACM, March 2025. doi: 10.1145/3689031.3696075. URL http://dx.doi.org/10.1145/3689031.3696075.

Weijia Shi, Xiaochuang Han, Mike Lewis, Yulia Tsvetkov, Luke Zettlemoyer, and Scott Wen tau Yih. Trusting your evidence: Hallucinate less with context-aware decoding, 2023. URL https://arxiv.org/abs/2305.14739.

Marco Simoni, Aleksandar Fontana, Giulio Rossolini, Andrea Saracino, and Paolo Mori. Gtpo: Stabilizing group relative policy optimization via gradient and entropy control, 2025. URL https://arxiv.org/abs/2508.03772.

Chenghao Sun, Chengsheng Zhang, Guanzheng Qin, Rui Dai, and Xinmei Tian. Weight patching: Toward source-level mechanistic localization in llms, 2026. URL https://arxiv.org/abs/2604.13694.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions, 2023. URL https://arxiv.org/abs/2212.10560.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark, 2024. URL https://arxiv.org/abs/2406.01574.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. Wizardlm: Empowering large pre-trained language models to follow complex instructions, 2025. URL https://arxiv.org/abs/2304.12244.

Tianze Xu, Yanzhao Zheng, Pengrui Lu, Lyumanshan Ye, Yong Wu, Zhentao Zhang, Yuanqiang Yu, Chao Ma, Jihuai Zhu, Pengfei Liu, Baohua Dong, Hangcheng Zhu, Ruohui Huang, and Gang Yu. Rubrics to tokens: Bridging response-level rubrics and token-level rewards in instruction following tasks, 2026. URL https://arxiv.org/abs/2604.02795.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Junjie Ye, Caishuang Huang, Zhuohan Chen, Wenjie Fu, Chenyuan Yang, Leyi Yang, Yilong Wu, Peng Wang, Meng Zhou, Xiaolong Yang, Tao Gui, Qi Zhang, Zhongchao Shi, Jianping Fan, and Xuanjing Huang. Muldimif: A multi-dimensional constraint framework for evaluating and improving instruction following in large language models, 2026. URL https://arxiv.org/abs/2505.07591.

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

Lifan Yuan, Wendi Li, Huayu Chen, Ganqu Cui, Ning Ding, Kaiyan Zhang, Bowen Zhou, Zhiyuan Liu, and Hao Peng. Free process rewards without process labels, 2024. URL https://arxiv.org/abs/2412.01981.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-rewarding language models, 2025. URL https://arxiv.org/abs/2401.10020.

Yirong Zeng, Yufei Liu, Xiao Ding, Yutai Hou, Yuxian Wang, Haonan Song, Wu Ning, Dandan Tu, Qixun Zhang, Bibo Cai, Yuxiang He, and Ting Liu. Precision over diversity: High-precision reward generalizes to robust instruction following, 2026. URL https://arxiv.org/abs/2601.04954.

Kongcheng Zhang, Qi Yao, Shunyu Liu, Wenjian Zhang, Min Cen, Yang Zhou, Wenkai Fang, Yiru Zhao, Baisheng Lai, and Mingli Song. Replay failures as successes: Sample-efficient reinforcement learning for instruction following, 2025a. URL https://arxiv.org/abs/2512.23457.

Tao Zhang, Chenglin Zhu, Yanjun Shen, Wenjing Luo, Yan Zhang, Hao Liang, Tao Zhang, Fan Yang, Mingan Lin, Yujing Qiao, Weipeng Chen, Bin Cui, Wentao Zhang, and Zenan Zhou. Cfbench: A comprehensive constraints-following benchmark for llms, 2025b. URL https://arxiv.org/abs/2408.01122.

Zheng Zhang, Ao Lu, Yuanhao Zeng, Ziwei Shan, Jinjin Guo, Lufei Li, Yexin Li, and Kan Ren. Grad2reward: From sparse judgment to dense rewards for improving open-ended llm reasoning, 2026. URL https://arxiv.org/abs/2602.01791.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization, 2025a. URL https://arxiv.org/abs/2507.18071.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Processbench: Identifying process errors in mathematical reasoning, 2025b. URL https://arxiv.org/abs/2412.06559.

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

A Additional Discussion for the Motivation

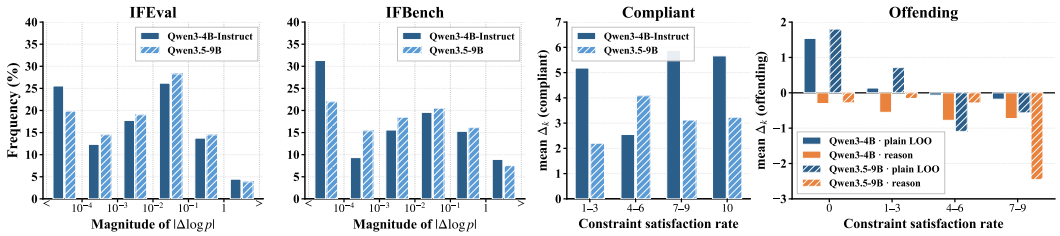


Figure 6: **Left two:** distribution of $|\Delta_k(y,t)|$ over all tokens on IFEval and IFBench. **Third:** mean contrast over compliant tokens $\mathcal{T}_k^+$. **Forth:** mean contrast over offending tokens $\mathcal{T}_k^-$ under the plain ablated prompt x^{-k} vs. a reason-augmented prompt, for both models.

Discussion 1: ablating c_k will not introduce a positional confound. In our design, deleting c_k from the prompt also shifts the position of every later token, which could in principle contaminate $\Delta_k(y,t)$. However, a positional effect is global: it would perturb the entire response roughly uniformly. The two left panels of Figure 6 show the opposite: over 80% of all tokens change by less than 0.1 in log-probability, so the constraint's influence is sparse and confined to a few tokens. We therefore attribute Δ_k to the constraint itself rather than to positional drift.

Discussion 2: the repair restores the sign beyond capability. The second branch of Eq. 9 claims that conditioning the policy on its own failure evidence $x_{\text{cf}}^k = x \oplus y \oplus \phi(c_k, r_k)$ (Eq. 10) makes the contrast point the right way for $c_k \in \mathcal{C}^{\text{out}}(x)$. We verify it in the right panel of Figure 6: under the plain ablated prompt, the offending tokens $\mathcal{T}_k^-$ carry a *positive* mean contrast at SR=0, indicating that LLM fails to recognize where it went wrong and instead treats its own output as correct. Meanwhile, the reason-conditioned contrast turns negative in every satisfaction bin for both models, indicating that the model can roughly localize the error from the failure reason. This supplies the missing evidence for the beyond-capability branch.

B Further Analysis for Context-Aware Decoding

Context-aware decoding (CAD) (Shi et al., 2023) appears in the main text as the training-free observation that motivates our signal (Section 3.2). Here, we view it as a stand-alone recipe and evaluate it at full scale. We first report the results on six benchmarks below:

Table 5: CAD results on six IF benchmarks (%), with CSR as the default reward.

Method	IFEval	IFB	MulDim	CFB	InFo	FollowB	**Average**
			Qwen3-4B-Instruct				
Base	82.8	31.0	57.3	45.8	45.2	67.8	55.7
CAD (α=0.2)	**85.2**	33.0	57.8	**47.7**	**47.4**	**70.1**	56.6$_{\uparrow 1.6}$
GRPO	84.5	**35.3**	**80.8**	46.4	46.0	69.1	**60.4**$_{\uparrow 8.4}$
LoCo (Ours)	87.1	40.0	83.4	50.5	50.8	71.7	63.9
			Qwen3.5-9B				
Base	83.9	35.0	59.5	46.5	44.8	70.3	56.7
CAD (α=0.2)	**87.3**	38.0	62.1	**49.6**	45.6	71.9	59.1$_{\uparrow 4.2}$
GRPO	86.7	**38.3**	**83.5**	48.5	**47.0**	**72.4**	**62.7**$_{\uparrow 10.6}$
LoCo (Ours)	88.4	41.3	86.4	51.0	49.2	75.4	65.3

CAD is a strong training-free baseline. As Table 5 shows, aside from the in-domain MulDimIF, CAD matches or surpasses the RL-trained GRPO on the remaining benchmarks, especially on the out-of-domain ones (CFBench, InFoBench, FollowBench). However, the method carries two practical problems. First, every decoding step must run two prefix prompts x^+ and x^- in parallel, which doubles both the per-step latency and the KV-cache footprint and makes it unattractive for practical deployment. Second, it is uncontrollable. As discussed in main text, once a constraint lies beyond the model's capability, the difference points the wrong way and CAD only amplifies noise. CAD has no mechanism to decide which constraints to trust or to give up the amplified.

Comparison between CAD and LoCo. We now compare LoCo with context-aware decoding to show that how our token-level credit assignment implement the contrast that CAD adds at decode time. For clarity we assume a single constraint, so the full and ablated prompts reduce to x^+ and x^- and the leave-one-out contrast of Eq. 2 becomes:

$$d(w) = \log \pi_\theta\big(w \mid x^+\big) - \log \pi_\theta\big(w \mid x^-\big)\,, \tag{16}$$

$d(w)$ is now precisely the term CAD adds to the log-score. For our LoCo, the token-level advantage adds a token-wise reward built entirely from $d(w)$ to the GRPO advantage. To see how A_{tok} implements $d(w)$, we drop the group normalizer for constraint-level advantage computation and the $\tanh(\cdot/\tau)$ gate, so that the token-wise reward reduces to λd itself. Since d is a detached reward, the loss term for one sampled token w_t is:

$$\ell_\lambda(w_t) \;=\; -\,\lambda\, d(w_t)\, \log \pi_\theta(w_t \mid x^+), \qquad w_t \sim \pi_\theta(\cdot \mid x^+). \tag{17}$$

We then differentiate $\log \pi_\theta(w_t \mid x^+)$ with respect to the logit $z(w)$ of a candidate w. Due to the softmax identity $\partial \log \pi_\theta(w_t)/\partial z(w) = \delta_{w,w_t} - \pi_\theta(w)$, we take the on-policy expectation:

$$\begin{aligned}
\mathbb{E}_{w_t\sim\pi_\theta}\left[\frac{\partial\, \ell_\lambda(w_t)}{\partial z(w)}\right] &= -\lambda \sum_v \pi_\theta(v \mid x^+)\, d(v)\big(\delta_{w,v} - \pi_\theta(w \mid x^+)\big) \\
&= -\lambda\, \pi_\theta(w \mid x^+)(d(w) - \sum_v \pi_\theta(v \mid x^+)\, d(v)) \\
&= -\lambda\, \frac{\partial}{\partial z(w)}\, \mathbb{E}_{w\sim\pi_\theta}[d(w)]\,,
\end{aligned} \tag{18}$$

This shows that the mean gradient of the correction is precisely the gradient of the expected contrast, so minimizing ℓ_λ takes a single step up $\mathbb{E}_{w\sim\pi_\theta}[d(w)]$, which is the same d that CAD adds to the log-score. Folded into post-training, this tilt is incorporated in the parameter weights and introduces no extra inference cost for deployment. It is also controllable: we amplify only when the direction is correct, and when the raw contrast points the wrong way we fall back on the reason-conditioned repair to steer the model back on course.

Comparison between RGSD and LoCo. In open-ended domains, Rubric-Guided Self-Distillation Rezaei et al. (2026) trains a student that is blind to the rubric toward a teacher that sees it, so the teacher's edge is also the rubric-conditioned shift. However, in instruction following domains, the student already observes the constraint (rubric), so our teacher is the same policy pushed one step further along d. This yields an *implicit* teacher: the student's own distribution nudged in the right direction and distilled back into itself, rather than imported from an external oracle.

C Evaluation Datasets

We evaluate LoCo on eight established instruction-following benchmarks.

IFEval (Zhou et al., 2023) contains 541 prompts annotated with 25 types of verifiable instructions on keywords, length, format, and language, each checked by a deterministic program under the official strict and loose protocols.

IFBench (Pyatkin et al., 2025) applies the same protocol to 300 test prompts whose 58 newly designed constraint types are deliberately out of distribution for IFEval-style checkers.

MulDimIF (Ye et al., 2026) draws 1,200 test prompts from a framework crossing three constraint patterns (listing, incorporation, and example), four categories (content, format, language, and length), and four difficulty levels, with every constraint verified by code.

CFBench (Zhang et al., 2025b) collects 1,000 manually curated prompts over 200+ real-life scenarios and 50+ NLP tasks, with constraints organized into 10 primary categories and annotated as primary or secondary; an LLM judge returns a 0/1 verdict per constraint.

FollowBench (Jiang et al., 2024) builds on 820 initial instructions spanning six constraint categories (content, situation, style, format, example, and mixed) and adds one constraint per level up to five difficulty levels, combining rule-based checks with LLM judging.

InFoBench (Qin et al., 2024) decomposes 500 instructions into 2,250 yes/no questions that an LLM judge answers individually, yielding the decomposed-requirement following ratio.

AdvancedIF (He et al., 2025) contains 1,645 dialogs whose prompts and rubrics were written and cross-reviewed by human experts, covering complex single-turn instruction following, multi-turn carried-context following , and system-prompt steerability. Each dialog carries a rubric of up to 20 yes/no criteria about the final turn.

RoleMRC (Lu et al., 2025) contains 1,400 samples covering role-related and role-unrelated prompts, including answer, refusal, and nested variants, each annotated with knowledge-base and stylistic constraints; an LLM judge returns a 0/1 verdict for each constraint.

Table 6: Evaluation datasets. "Rule" denotes deterministic programmatic checkers and "Judge" denotes per-constraint LLM judging with the official prompts.

Benchmark	#Items	Constraints	Verification
IFEval (Zhou et al., 2023)	541	25 verifiable types	Rule
IFBench (Pyatkin et al., 2025)	300	58 new verifiable types	Rule
MulDimIF (Ye et al., 2026)	1,200	3 patterns, 4 categories, 4 levels	Rule
CFBench (Zhang et al., 2025b)	1,000	10 primary categories	Judge
FollowBench (Jiang et al., 2024)	820×5 levels	6 constraint categories	Rule + Judge
InFoBench (Qin et al., 2024)	500	2,250 decomposed questions	Judge
AdvancedIF (He et al., 2025)	1,645	up to 20 expert rubrics	Judge
RoleMRC (Lu et al., 2025)	1,400	KB + style constraints	Judge

D BACKBONE MODELS

We build our policies on two open-weight dense decoder-only LLMs, Qwen3-4B-Instruct and Qwen3.5-9B, which we introduce below; Table 7 details their configurations, and the main architectural difference between the two are formalized afterwards.

Table 7: Configuration of the two backbones. n_q/n_{kv}: numbers of query and KV heads; d_h: head dimension; n_k/n_v and d_k/d_v: numbers and dimensions of the linear-attention key/value heads.

	Qwen3-4B-Instruct	**Qwen3.5-9B**
Parameters	∼4B	∼9B
Layers	36 (full attention)	32 (24 linear, 8 full)
Hidden / FFN size	2560 / 9728	4096 / 12288
Full attention (n_q/n_{kv}, d_h)	32 / 8, 128	16 / 4, 256
Linear attention (n_k/n_v, d_k/d_v)	–	16 / 32, 128 / 128
Attention type	softmax GQA	Gated DeltaNet (3:1) + gated GQA
RoPE (base, rotated dims)	10^6, 100%	10^7, 25%
Max context	256K	256K
Vocabulary	151,936	248,320

Qwen3-4B-Instruct. An instruction-tuned dense transformer from the Qwen3 family with approximately 4 billion parameters, post-trained with SFT and GRPO-based reinforcement learning to support hybrid thinking and non-thinking modes; we use the 2507 non-thinking checkpoint, which natively supports 256K context with a 151,936-token vocabulary, and delivers strong instruction following and reasoning for its scale while remaining cheap enough for large-scale RL training.

Qwen3.5-9B. A dense hybrid-attention model from the Qwen3.5 family with approximately 9 billion parameters, instruction-tuned with a multi-token prediction objective and natively multimodal (text, image, and video via a 27-layer vision tower); it stacks 24 Gated DeltaNet linear-attention layers with 8 full-attention layers for efficient long-context inference, supports 256K context with a 248,320-token vocabulary, and excels at long-context reasoning and multimodal understanding.

Full attention. Both models use softmax attention with grouped-query attention (GQA) in their full-attention layers, where the output for a query $\mathbf{q}_t \in \mathbb{R}^{d_h}$ attends over the cached prefix:

$$\mathbf{o}_t = \mathrm{softmax}\left(\frac{\mathbf{q}_t \mathbf{K}_{\le t}^\top}{\sqrt{d_h}}\right)\mathbf{V}_{\le t}, \qquad \mathbf{K}_{\le t} = [\mathbf{k}_1, \ldots, \mathbf{k}_t], \tag{19}$$

and the KV cache grows linearly with the sequence length.

Linear attention. In Qwen3.5-9B, 24 of the 32 layers follow the Gated DeltaNet pattern. Each Gated DeltaNet layer replaces the prefix KV cache with a fixed-size recurrent state $\mathbf{S}_t \in \mathbb{R}^{d_k \times d_v}$ per value head, updated by the gated delta rule as follows:

$$\mathbf{S}_t = \alpha_t \mathbf{S}_{t-1} - \beta_t \mathbf{k}_t \left(\mathbf{k}_t^\top \mathbf{S}_{t-1} - \mathbf{v}_t^\top\right), \qquad \mathbf{o}_t = \mathbf{S}_t^\top \mathbf{q}_t, \tag{20}$$

where $\alpha_t, \beta_t \in (0, 1)$ are input-dependent forget and input gates, $\mathbf{q}_t, \mathbf{k}_t$ are produced after a short causal convolution (kernel size 4) with SiLU; both attention types of Qwen3.5-9B are additionally modulated by an output gate. Unlike full attention which costs $O(T)$ memory and computation per token, the recurrence here costs $O(1)$ memory and $O(d_k d_v)$ computation per token.

E Training Configuration

The training configuration of LoCo is shown below.

Table 8: Training configuration of LoCo for the two backbones.

	Qwen3-4B-Instruct	**Qwen3.5-9B**
Rollout & Sampling		
Temperature	1.0	1.0
top-p / top-k	1.0 / -1	1.0 / -1
Group size G	6	6
Max prompt length	2,048	2,048
Max response length	1,600	1,600
Optimization		
Learning rate	1×10^{-6}	5×10^{-7}
Train batch size	64	64
PPO mini-batch	64	64
Micro-batch / GPU	2	1
Log-prob micro-batch (actor/ref)	8 / 16	1 / 1
Dynamic batch size	enabled	disabled
KL-loss coefficient	1×10^{-4}	1×10^{-2}
Entropy coefficient	0	0
Gradient clip	1.0	1.0

F Training with different judge models

In the main text, we use Qwen3.7-max for constraint satisfaction judge, and also for the LLM that supplies additional failure reasons and edits the best attempt into a demonstration. This is because Qwen3.7-max supports higher concurrency in our serving setup. Here we instead switch to DeepSeek-V4-Pro, which keeps the training-time judge consistent with the one used for evaluation. The results are reported below, where LoCo still delivers a substantial overall improvement.

Table 9: Main results for Qwen3-4B-Instruct, with the LLM judge replaced by **DeepSeek-v4-pro**.

Method	In-Domain				Out-of-Domain					
	IFEval	IFB	MulDim	**Average**	CFB	InFo	FollowB	AdvIF	RoleM	**Average**
GRPO	85.7	34.7	80.0	66.8$_{-0.0}$	48.5	48.6	70.7	42.7	41.0	50.3$_{-0.0}$
LoCo (Ours)	**86.7**	**40.7**	**83.6**	**70.3**$_{\uparrow 5.3}$	**50.9**	**51.8**	**72.6**	**44.2**	**44.4**	**52.8**$_{\uparrow 4.9}$

G Limitations and Future Work

Our proposed LoCo inherits several limitations from its design. First, the token-level signal is self-supplied but not free. It require extra forward passes, so a single optimization step is measurably more expensive than vanilla GRPO. Although we showed this overhead is repaid within a matched wall-clock budget, it still raises the peak memory and engineering complexity of training. Second, our credit assignment supposes that an instruction can be decomposed into atomic, independently verifiable constraints; while this holds for our training dataset HIR-16K, it is less clean for constraints that are compositional or mutually dependent. Third, our training dataset consists of

single-turn instructions with relatively short responses, so whether the with/without-constraint probability contrast that LoCo relies on remains valid for multi-turn dialogues or much longer responses remains to be verified, since accumulated context may dilute each token's credit.

Our future work mainly revolves around two aspects. First, we would like to extend LoCo to long-context and multi-turn instruction following, where constraints stated early may only be decided many turns later. In this case, a token's credit can be diluted as responses grow, calling for a capability partition estimated over the dialogue history rather than a single rollout group. Second, we will move beyond our assumption that constraints are atomic and independent toward modeling relations among them, so that credit assignment reflects how each constraint interacts with each other.

H Prompts

H.1 Failure-Reason Verbalization

The verbalized failure evidence is produced locally whenever possible. Every family of executable checkers is mapped to a verbalizer template that embeds the values measured from the response, so the evidence is always concrete and checkable; our rule-based checkers (12,814 constraint instances, 4,159 unique checker codes) are covered by 24 semantic groups. Table 10 lists representative templates, and the box below shows actual rendered outputs.

Table 10: Representative verbalizer templates for executable checkers. Curly-brace placeholders are filled with values measured from the response.

Checker semantics	Template
All required keywords must appear	`{m} of {K} required keyword(s) are missing: {missing}; the checker requires all of {all} to appear (case-insensitive).`
Total length $\geq N$ words	`The response contains {wc} words; the checker requires at least {N} words.`
Total length in $[\ell, h]$ words	`The response contains {wc} words; the checker requires between {lo} and {hi} words.`
Each sentence $\leq N$ words	`{n_over} of {n_sent} sentences exceed {N} words (longest: {max} words); each sentence must contain at most {N} words.`
Question marks forbidden	`The response contains {n} question mark(s); question marks are not allowed.`
Each paragraph $\leq N$ sentences	`The longest paragraph contains {max_s} sentences; each paragraph must contain at most {N} sentences.`
Each keyword exactly once	`"{k}" appears {n} times; the checker requires exactly one occurrence of each keyword: {all}.`
Every word $\geq N$ characters	`The response contains {n_short} of {n_words} words shorter than {N} characters (shortest: "{w}", {len} chars); every word must have at least {N} characters.`

When a checker yields no measured evidence, the pair escalates to a single API call, shown below; if that also fails, the evidence falls back to the generic sentence *"An automated check of the response determined this constraint is NOT satisfied."*

H.2 Counterfactual Repair Credit

For constraints that no rollout in the group satisfies, the policy is re-conditioned on its own response together with the verbalized failure evidence; the token-level probability shifts between this augmented prompt and the leave-one-out prompt provide the credit signal.

Counterfactual repair credit: re-conditioning prompt

```
{instruction}

Your previous response:
{response}

Feedback on your previous response:
- Constraint: "{constraint}"
- Status: NOT satisfied
- Evidence: {reason}

The response above violates the quoted constraint. Rewrite your
    complete response. Keep the original wording, structure and
    content wherever they do not conflict with the constraint, and
    change only what is necessary to satisfy it.
```

H.3 External Minimal-Edit Repair

External repair: when all-failed constraint exists

```
You are a helpful assistant. Your task is to revise a response so it
    satisfies ALL of the failed constraints listed below.

Question:
{question}

All constraints:
{criteria_list}

Failed constraints (ALL of these must be satisfied):
{failed_constraints}

Verification details for each failed constraint:
{rule_info}

Current response (best attempt -- highest overall satisfaction rate
    in the group):
{response}

Instructions:
- Output the COMPLETE revised response that satisfies ALL failed
    constraints simultaneously.
- Keep as much original content intact as possible. Only change what
    is necessary.
- Do NOT output just the changed parts. Output the entire response
    from start to finish.
- Wrap the entire revised response in <response></response> tags.
- Do NOT add any explanation or commentary outside the tags.

Example output format:
<response>The complete revised response text here, satisfying all
    failed constraints.</response>
```

H.4 Self-Distillation Prompt

During self-distillation, the frozen reference policy scores the student's own rollout tokens under a demonstration-augmented prompt, built by appending the externally repaired response to the instruction as an in-context example:

Self-distillation: demonstration-conditioned teacher prompt

```
{instruction}

To satisfy all constraints, here is a good example:
{repaired_response}

After understanding the reference response, please try to answer the
    question and follow all constraints using your own approach below.
```

I CASE STUDY

In this section, we qualitatively examine how LoCo's fine-grained reward signal is allocated across the tokens of a response, and present two examples from HIR-16K.

Case 1: All constraints are verified by deterministic programs.

Instruction: Can you help me find a gift for my friend who loves books? They have everything and are hard to shop for. I need your advice on something unique and thoughtful.

Constraints:

- Your ENTIRE response should be in English. ✓
- Your ENTIRE response should be in all capital letters. ×
- Your response should contain AT LEAST 4 SENTENCES. ✓
- Your response must contain AT LEAST 2 placeholders represented by square brackets, such as [gift idea 1]. ✓
- At the end of your response, please explicitly add a postscript starting with P.S. ✓
- Finish your response with this exact phrase ANY OTHER QUESTIONS? ✓

Response: HOW ABOUT GIVING THEM A PERSONALIZED BOOK CLUB SIGNSHEET THAT INCLUDES THEIR FAVORITE AUTHORS AND TOP TITLES, WITH A PLACEHOLDER FOR THEM TO ADD NEW BOOKS [GIFT IDEA 1]? YOU COULD ALSO CREATE A HANDWRITTEN JOURNAL WITH PROMPT QUESTIONS FOR BOOK DISCUSSIONS, INCLUDING [GIFT IDEA 2] TO ENCOURAGE DEEPER REFLECTION. THIS WAY, THE GIFT IS UNIQUE, PERSONAL, AND MATCHES THEIR LOVE FOR LITERATURE WITHOUT REPETITION. IT GOES BEYOND STANDARD BOOKS BY FOCUSING ON INTERACTION AND MEMORY. P.S. MAKE SURE TO INCLUDE A SMALL NOTE SAYING "EVERY BOOK YOU READ IS A NEW PART OF YOUR STORY" TO make it truly meaningful. ANY OTHER QUESTIONS?<end>

What to penalize? The only failed check is the all-capitals rule. Specifically, "make" is the first lowercase token in the whole response, so it receives the greatest penalty of the entire response.

What to reward? The capitalized tokens satisfy the rule and earn more reward: "HOW", "P.S.", "ANY" carry the strongest blue. Since the majority of constraints are satisfied, most tokens receive rewards overall.

Case 2: Some of the constraints are verified by external LLM judge.

Instruction: Provide a comprehensive plan to design a survey form that accommodates a text input field for users to offer additional remarks. The field must have a maximum character limit of 100. Also provide a detailed explanation of how you would modify the HTML implementation of the form to meet this new requirement.

Constraints:

- Allow no more than 35 words in each sentence. ✓
- Your response should not contain question marks. ✓
- Use transitional phrases and sentences to connect the different steps in the plan and to guide the reader through the process. ×
- Avoid using overly technical jargon or complex terminology that may confuse non-experts, and refrain from using bullet points or numbered lists with less than three items. ✓
- The main theme of the response is to provide a comprehensive plan for designing a survey form with a text input field for additional remarks, and the focus is on the steps to achieve this goal, including modifying the HTML implementation. ✓

- The response should follow a step-by-step approach, with each step building upon the previous one to provide a comprehensive plan for designing a survey form. ✓

- The response should begin with a sentence acknowledging the request, such as 'Certainly! Here is a comprehensive plan to design a survey form that accommodates a text input field for users to offer additional remarks:' ✓

- The response should use numbered lists (1., 2., 3., etc.) to outline the steps involved in designing the survey form and modifying the HTML implementation, with each step on a new line and a period at the end of each step. ✓

Response: Certainly! Here is a comprehensive plan to design a survey form that accommodates a text input field for users to offer additional remarks:

1. Begin by opening the HTML file to view the current form structure.
2. Insert a new input field after existing questions in the form layout.
3. Set the input type to text and add a label for clarity.
4. Apply a maxlength attribute with value 100 to control input length.
5. Confirm that each user enters no more than 35 words per sentence.
6. Test the form with sample entries to ensure limits are applied correctly.
7. Adjust spacing and style so users can see the field clearly.
8. Review the form for flow and adaptation to mobile devices.
9. Finalize the HTML file with all changes saved and validated.<end>

What to penalize? The failed check demands transitional phrases between the steps, and penalty focuses exactly on the bare imperative verbs that open each list item, e.g., "2.Insert", "3.Set", "5.Confirm", "6.Test" and "7.Review", where a connective phrase should have appeared.

What to reward? The checks the plan does satisfy, including the numbered list and the step-by-step approach, stay bright. For example, the verbs that execute the plan ("Certainly", "HTML", "add") are strongly rewarded. The model separates the missing transitions from the correct plan content.