PAVIF: REINFORCEMENT LEARNING FOR INSTRUCTION FOLLOWING WITH THE POLICY AS ITS OWN VERIFIER

Anonymous authors
Paper under double-blind review

ABSTRACT

Instruction following is a core capability of large language models (LLMs), yet reinforcement learning for this task often relies on external verifiers to provide reward signals, introducing additional computational cost. This motivates us to investigate whether the policy can serve as its own verifier for instruction-following optimization. Our preliminary study reveals an intriguing phenomenon: verification training for the policy can also enhance its instruction-following potential. Building on this observation, we propose PAVIF, a two-stage reinforcement learning framework that internalizes verification within the policy. In the first stage, the policy is trained to verify its own responses using a self-verification dataset that we construct with responses spanning varying levels of instruction compliance. In the second stage, the policy uses its verification signals to reinforce instruction-compliant responses, turning the increased generation potential into more reliable instruction-following capability. Evaluations across six benchmarks show that PAVIF outperforms open-source baselines, with an average absolute improvement of 2.01% over the corresponding base models. Further analyses show that verification training improves the policy's verification accuracy, while PAVIF largely preserves general capabilities and reduces per-step training time by 23.2% compared with using an external verifier of the same scale. Our code is available at `https://anonymous.4open.science/r/PAVIF-310B`.

1 INTRODUCTION

Large language models (LLMs) are increasingly deployed as general-purpose assistants in real-world applications (Lou et al., 2024). In these scenarios, models are expected not only to fulfill the core task, but also to comply with user-specified requirements (Jiang et al., 2024; Qin et al., 2024; Zhang et al., 2025a). Failure to satisfy even a single constraint may render an otherwise correct response unusable. Therefore, improving instruction-following capability is crucial for LLMs to behave more controllably and reliably.

Instruction-following tasks often involve explicit, automatically verifiable constraints, providing a natural basis for reinforcement learning with verifiable rewards (RLVR) (Lambert et al., 2024; Qin et al., 2025; Pyatkin et al., 2025). These constraints are commonly categorized according to how they are verified (Peng et al., 2025). Hard constraints can often be verified programmatically. In contrast, soft constraints require semantic evaluation, for which existing methods typically rely on external verifiers, making reliable and efficient verification a key challenge. As illustrated in Figure 1, existing approaches employ either strong external LLMs at substantial computational cost (Peng et al., 2025) or smaller, separately trained verifiers with potentially limited generalization (Zeng et al., 2026). Even approaches that periodically refresh the verifier from the latest policy keep it separate and frozen during policy optimization, incurring additional model overhead (Ren et al., 2026b). Across these approaches, verification remains external to the policy rather than being acquired as an internal capability. To explore the benefits of internalizing verification, we conduct a preliminary experiment and find that verification training can enhance the policy's generation potential in instruction-following tasks. This raises our central question: *Can a policy learn to verify its own responses and use this feedback to improve instruction following?*

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.

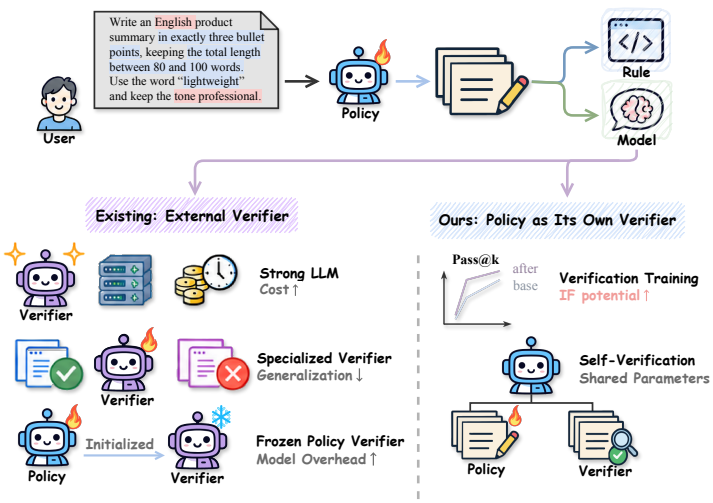


Figure 1: Overview of verification paradigms for complex instruction following.

In this paper, we propose **PAVIF** (**P**olicy **a**s a **V**erifier for **I**nstruction **F**ollowing), a reinforcement learning framework for instruction following in which the policy model serves as its own verifier. The framework comprises two stages. In the first stage, the policy learns to assess its own responses under the guidance of teacher scores. In the second stage, the policy performs both response generation and constraint verification, with its verification score serving as the main reward signal alongside rule-based and format rewards. By internalizing verification, PAVIF eliminates the need for external verifiers, reducing the associated computational and memory cost. To support verification training in the first stage, we introduce a simple, automated procedure to construct comprehensive self-verification data from existing instruction-following training data. Specifically, we selectively remove constraints to generate fully, partially, and non-compliant responses and obtain teacher scores against the original complete instructions.

We evaluate PAVIF across model scales on six representative instruction-following benchmarks covering multi-turn and multi-dimensional constraints. Across these settings, PAVIF achieves superior performance to existing open-source methods, revealing the potential of verification to enhance generation. Furthermore, our analysis reveals three key advantages of our approach. First, verification training increases the policy's average judgment accuracy on LLMBar, with absolute gains of 1.36% (Vanilla) and 1.22% (NoRules). Second, PAVIF delivers substantial gains in training efficiency, cutting per-step training time by 23.2%. Third, evaluations across four dimensions show that PAVIF largely preserves performance in knowledge, reasoning, and mathematics while achieving promising results in code generation.

Our main contributions are summarized as follows:

- We provide empirical evidence that verification training can benefit instruction-following generation, motivating further generation optimization through self-verification.
- We propose PAVIF, a two-stage reinforcement learning framework that equips the policy with self-verification capabilities and uses its own verification feedback to optimize instruction following.
- Extensive experiments demonstrate that PAVIF improves instruction-following performance and training efficiency while largely preserving general capabilities.

2 Related Work

2.1 Instruction Following in LLMs

Instruction following requires large language models to satisfy user-specified constraints when generating responses. Even correct responses to the core task may be deemed incorrect when they violate explicit requirements. To systematically assess this capability, several benchmarks have

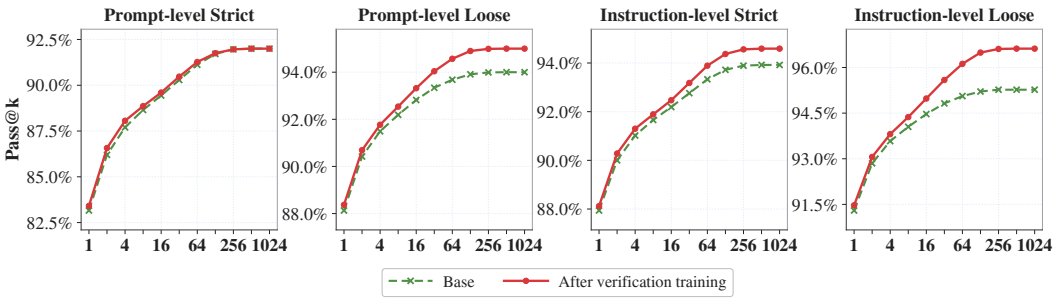


Figure 2: Pass@k on IFEval before and after verification training.

been developed to evaluate instruction following under diverse and complex constraints (Zhou et al., 2023; Jiang et al., 2024; Qin et al., 2024). Existing training-based approaches to improving instruction following primarily focus on supervised fine-tuning (SFT) (Sun et al., 2024; An et al., 2025; Hui et al., 2026), preference learning (Qi et al., 2025; Zhang et al., 2025b; Huang et al., 2025a;b; Yuan et al., 2025; Ren et al., 2025), and reinforcement learning with verifiable rewards (RLVR) (Qin et al., 2025; Ren et al., 2026a;b). Among these methods, SFT and preference learning are often limited by data quality and ambiguous feedback. By deriving rewards directly from constraint satisfaction, RLVR offers a promising alternative approach to improving instruction following, particularly for complex instructions.

2.2 Reinforcement Learning for Instruction Following

For instruction following, rewards are typically constructed by evaluating whether responses satisfy instruction-specific constraints (Peng et al., 2025; He et al., 2026; Xu et al., 2026). Hard constraints can be assessed by programmatic verifiers, whereas soft constraints generally require external language models or separately trained reward models, making verification accuracy and computational efficiency key challenges (Wen et al., 2026; Ren et al., 2026a). To address these challenges, TinyJudge (Zeng et al., 2026) employs lightweight specialists but retains separate verifier models. SEIF (Ren et al., 2026b) adopts multi-role alternating optimization, using a frozen verifier refreshed from the latest policy to provide adaptive rewards. Despite these advances, verification is still delegated to a separate component during policy optimization. In contrast, our PAVIF framework explicitly trains the policy to assess constraint satisfaction and then uses the same policy for both response generation and verification during Group Relative Policy Optimization (GRPO) (Shao et al., 2024).

3 Preliminaries

3.1 Problem Formulation

A complex instruction-following task is defined by an instruction $q = (x, \mathcal{C})$, where x denotes the core task and $\mathcal{C} = \{c_1, \ldots, c_m\}$ denotes a set of explicit constraints. Following Peng et al. (2025), these constraints are divided into hard constraints, which can be verified programmatically, and soft constraints, which require semantic evaluation. Given an instruction q, the objective is to generate a response y that satisfies every constraint in $\mathcal{C}$.

3.2 Empirical Insights

We conduct a preliminary study to investigate whether verification training alone can benefit instruction-following generation. Concretely, we compare the model's generation performance before and after the verification training described in Section 4.3.

Experimental Setup. We randomly sample 100 prompts from each of IFEval and InFoBench to evaluate generation performance using pass@k, which measures the probability of obtaining at least one successful response in k samples. Detailed settings are provided in Appendix A.

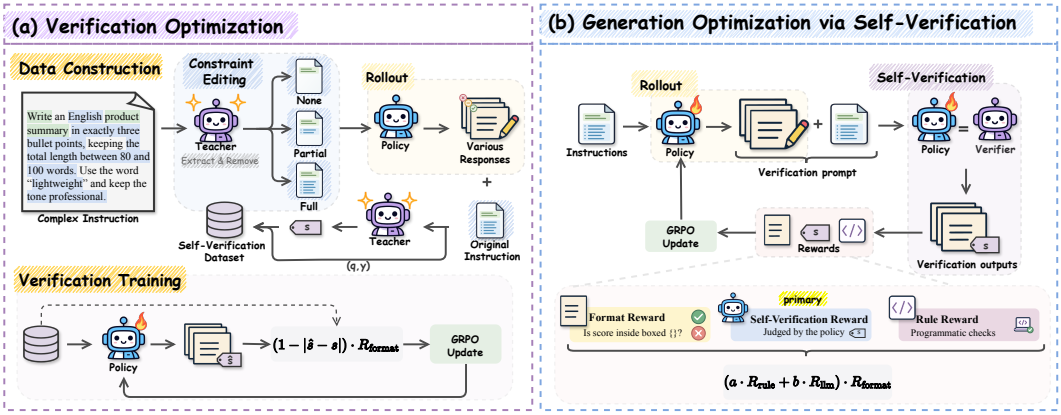


Figure 3: Overview of **PAVIF**. (a) Verification optimization trains the policy to assess constraint satisfaction. (b) Generation optimization uses the same policy to generate and verify responses, using self-verification scores as the primary reward for GRPO.

Generation Benefits from Verification Training. As shown in Figure 2, the model achieves higher pass@k on IFEval after verification training. Although the prompt-level strict curves converge at large k, these gains persist under the other evaluation settings. A similar trend is also observed on InFoBench (Appendix A.1), suggesting that acquiring verification capability can enhance the policy's potential to generate instruction-following responses. These observations motivate us to explore whether the policy's own verification feedback can further improve instruction following, as described in Section 4.4.

4 Method

4.1 Overview

We propose PAVIF, a reinforcement learning framework for instruction following that internalizes verification within the policy and uses its own verification feedback to optimize response generation. As illustrated in Figure 3, we first construct self-verification data (§ 4.2) and subsequently optimize the policy in two stages: (1) *Verification Optimization*, which equips the policy with an initial capability to assess constraint satisfaction (§ 4.3), followed by (2) *Generation Optimization via Self-Verification*, which builds on this capability by using the same policy for both response generation and verification (§ 4.4).

4.2 Data Construction

To equip the policy with self-verification capabilities (§ 4.3), we construct training data from a subset $\mathcal{Q}_{\text{ver}} \subseteq \mathcal{Q}$, where $\mathcal{Q}$ denotes the instruction set used for generation optimization (§ 4.4). Specifically, we selectively remove constraints to elicit candidate responses from the policy and employ a teacher model to score them against the original complete instructions.

Constraint Extraction and Removal. For each instruction $q = (x, \mathcal{C}) \in \mathcal{Q}_{\text{ver}}$, the teacher model extracts its constraints and selectively removes them while retaining the core task x. This produces three variants with increasing constraint coverage:

$$q^{(0)} = (x, \emptyset), \qquad q^{(p)} = (x, \mathcal{C}_p), \qquad q^{(f)} = (x, \mathcal{C}), \tag{1}$$

where $\mathcal{C}_p \subset \mathcal{C}$ denotes a subset of constraints, and $q^{(0)}$, $q^{(p)}$, and $q^{(f)}$ correspond to the core task with no, partial, and full constraint coverage, respectively. This variation in constraint coverage elicits policy responses with different levels of compliance with the original instruction, leading to a diverse and balanced distribution of verification scores.

Teacher Labeling. For each response y generated by the policy from $q^{(0)}$, $q^{(p)}$, or $q^{(f)}$, we construct a verification prompt q_{ver} by pairing y with the original complete instruction q. We then use the

teacher model's constraint-by-constraint judgments to obtain a verification score $s \in \{0, 0.5, 1\}$ for each example, yielding the self-verification dataset $\mathcal{D}_{\text{ver}} = \{(q_{\text{ver}}, s)\}$. Data construction details and prompts are provided in Appendix B.

4.3 Verification Optimization

In this stage, the policy learns to verify instruction compliance using $\mathcal{D}_{\text{ver}}$. As illustrated in Figure 3, given a verification prompt q_{ver}, the policy generates a verification response v containing a constraint-by-constraint analysis and a final score $\hat{s} \in \{0, 0.5, 1\}$. These scores correspond to non-compliance, partial compliance, and full compliance, respectively.

Using the teacher's verification score s as the reference, we define the training reward as

$$\mathcal{R} = (1 - |\hat{s} - s|) \cdot R_{\text{format}}, \tag{2}$$

where $R_{\text{format}} \in \{0, 1\}$ indicates whether the verification score can be correctly extracted from the output.

4.4 Generation Optimization via Self-Verification

In this stage, we optimize instruction following using the policy's own verification feedback. Given an instruction q, the policy first generates a group of candidate responses $\mathcal{Y}$. For each $y \in \mathcal{Y}$, we format (q, y) as a verification prompt q_{ver} and feed it back to the same policy, which generates a verification response v containing a self-verification score $\hat{s}$. This shared-parameter design allows the policy to leverage the verification capability acquired in the preceding stage.

The training reward integrates three components:

- **Format reward** $R_{\text{format}}(v)$: as defined in the verification optimization stage (§ 4.3).
- **Rule-based reward** $R_{\text{rule}}(y)$: applies programmatic checks to each hard constraint.
- **Self-verification reward** $R_{\text{llm}}(v)$: the score produced online by the policy itself, evaluating the overall satisfaction of all constraints.

The overall reward is:

$$\mathcal{R} = (a \cdot R_{\text{rule}}(y) + b \cdot R_{\text{llm}}(v)) \cdot R_{\text{format}}(v), \tag{3}$$

with weights dynamically determined by the constraint composition of each instruction:

$$a = \frac{n_{\text{rule}}}{n_{\text{all}} + n_{\text{rule}}}, \quad b = \frac{n_{\text{all}}}{n_{\text{all}} + n_{\text{rule}}}, \tag{4}$$

where n_{rule} denotes the number of hard constraints, n_{all} the total number of constraints.

Under this dynamic weighting scheme, the higher the proportion of hard constraints, the greater the weight assigned to R_{rule}. This allows it to serve as a reliable complement to R_{llm}, compensating for potential inaccuracies in the policy's assessment of hard constraints.

4.5 Optimization Objective

We adopt Group Relative Policy Optimization (GRPO) (Shao et al., 2024) for both optimization stages. As a variant of PPO, GRPO replaces the value network with group-level reward statistics. By eliminating the separate critic, it reduces memory overhead and enables efficient optimization. For each instruction q, the policy samples G responses $\{y_1, \ldots, y_G\}$ from the old policy and computes

$$A_i = \frac{r_i - \mu_G}{\sigma_G + \epsilon}, \qquad \mu_G = \frac{1}{G}\sum_{j=1}^{G} r_j, \qquad \sigma_G = \sqrt{\frac{1}{G}\sum_{j=1}^{G}(r_j - \mu_G)^2}. \tag{5}$$

The policy is then updated by maximizing

$$J_{\text{GRPO}}(\theta) = \mathbb{E}_{q\sim\mathcal{D},\,\{y_i\}\sim\pi_{\theta_{\text{old}}}}\left[\frac{1}{G}\sum_{i=1}^{G}\mathcal{L}_{\text{clip}}(\theta; q, y_i, A_i) - \beta D_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}})\right], \tag{6}$$

where $r_i = \mathcal{R}$ is the reward defined in Eq. 2 or Eq. 3 for the corresponding stage, $\mathcal{L}_{\text{clip}}$ denotes the token-level clipped surrogate objective, and β controls KL regularization.

Table 1: Performance comparison across six instruction-following benchmarks. Supplementary results are provided in Appendix E.

Model	IFEval		Multi-IF			IFBench		FollowBench		CFBench			InFoBench	Avg.
	Pr.S	Pr.L	Turn 1	Turn 2	Turn 3	Pr.S	Pr.L	HSR	SSR	CSR	ISR	PSR	Overall	
Open-source models														
Conifer-7B-DPO	30.31	35.67	20.61	14.08	11.56	14.00	16.00	36.58	60.40	25	7	9	78.84	34.07
TULU3-VerIF	81.70	84.47	80.02	65.75	54.51	27.00	28.33	56.17	76.40	42	14	19	83.21	58.67
UltraIF-8B-DPO	69.68	72.08	69.49	58.11	45.56	26.00	29.00	54.16	75.03	24	12	13	82.81	53.31
PARIF-Stage-II-8B	85.21	87.80	84.29	74.63	65.80	28.66	34.33	65.90	71.40	70	42	49	86.79	67.00
Qwen3-1.7B														
Base	68.57	72.27	67.44	54.85	43.95	19.00	21.66	59.92	77.67	55	26	31	81.59	55.65
+ SFT	70.61	73.94	**71.33**	53.27	42.04	**21.33**	**24.67**	56.47	74.65	53	24	30	80.32	55.39
+ VerIF	71.71	75.41	69.33	57.64	46.71	18.66	22.33	61.60	78.16	**60**	**29**	**35**	81.30	57.41
+ PAVIF (Ours)	**72.45**	**77.44**	70.60	**58.49**	**48.25**	19.66	24.66	**65.02**	**81.57**	53	25	30	**82.17**	**57.95**
Qwen3-4B-Instruct-2507														
Base	83.54	87.61	82.07	72.11	62.85	28.33	31.00	66.52	82.64	72	46	51	88.45	67.82
+ SFT	84.66	88.17	83.22	71.39	60.55	30.00	**35.33**	66.17	83.49	69	42	49	87.60	67.76
+ VerIF	86.87	89.09	83.90	73.67	64.44	30.66	32.66	69.55	79.22	72	46	53	88.00	68.84
+ PAVIF (Ours)	**88.35**	**91.68**	**84.31**	**74.65**	**65.27**	**31.66**	**35.33**	**71.02**	**85.78**	**73**	**49**	**55**	**88.63**	**70.71**
Qwen3-8B														
Base	83.54	85.76	83.19	73.61	64.52	26.33	30.33	75.66	88.25	68	38	46	**88.05**	67.90
+ SFT	82.62	85.40	83.71	70.56	61.05	**27.00**	30.67	75.11	85.62	**71**	**44**	**51**	86.99	67.88
+ VerIF	84.10	87.61	**84.50**	**74.69**	**66.27**	**27.00**	**31.33**	**76.17**	87.00	70	41	49	86.22	68.55
+ PAVIF (Ours)	**85.21**	**88.53**	83.80	74.53	65.91	**27.00**	**31.33**	75.74	**88.62**	69	40	47	87.43	**68.73**

5 EXPERIMENTS

5.1 BENCHMARKS AND EVALUATION METRICS

Instruction following. We evaluate our models on six instruction-following benchmarks covering both hard and soft constraints. IFEval (Zhou et al., 2023), Multi-IF (He et al., 2024), and IFBench (Pyatkin et al., 2025) focus on verifiable hard constraints, while FollowBench (Jiang et al., 2024), CFBench (Zhang et al., 2025a), and InFoBench (Qin et al., 2024) cover mixed and multidimensional constraints with soft requirements. For evaluation, we use gpt-oss-120b (Agarwal et al., 2025) and GPT-4o (Hurst et al., 2024) as the judge models. More details are provided in Appendix C.

Verification capability. We assess verification capability using LLMBar (Zeng et al., 2024), which tests whether a verifier can identify the instruction-following response from a pair of candidates.

General capabilities. We further examine general capability retention in knowledge, reasoning, mathematics, and code generation using MMLU (Hendrycks et al., 2021), ARC-Challenge (Clark et al., 2018), GSM8K (Cobbe et al., 2021), and HumanEval (Chen et al., 2021), respectively. We report accuracy for the first three and Pass@1 for HumanEval.

5.2 EXPERIMENTAL SETTINGS

Baselines. To contextualize the performance of our method, we compare it with the corresponding base models, SFT, and VerIF (Peng et al., 2025). We further evaluate Conifer-7B-DPO (Sun et al., 2024), TULU3-VerIF (Peng et al., 2025), UltraIF-8B-DPO (An et al., 2025), and PARIF-Stage-II-8B (Mu et al., 2026), covering both preference learning and reinforcement learning approaches.

Models. We conduct experiments on Qwen3 models, which achieve competitive performance across reasoning and general-purpose tasks (Yang et al., 2025). To test generalization across scales, we select three models ranging from 1.7B to 8B: Qwen3-1.7B, Qwen3-4B-Instruct-2507, Qwen3-8B. All experiments run in non-thinking mode. Training details are provided in Appendix D.

Table 2: Ablation results on the Qwen3-4B-Instruct-2507 model. Stage 1 and Stage 2 correspond to verification optimization (§ 4.3) and generation optimization via self-verification (§ 4.4), respectively. Detailed ablation results are provided in Appendix E.

Model	IFEval		Multi-IF	IFBench		FollowBench		CFBench	InFoBench	Avg.
	Pr.S	Pr.L	Avg.	Pr.S	Pr.L	HSR	SSR	ISR	Overall	
Ours	88.35	**91.68**	**74.74**	**31.66**	35.33	71.02	85.78	49	88.63	**69.05**
w/o Stage 1	85.58	90.01	73.74	31.33	**36.00**	72.01	84.70	50	89.10	68.78
w/o Stage 2	83.54	87.06	72.22	31.33	32.66	68.07	83.34	47	86.83	66.51
w/o Rule Reward	76.15	87.61	70.55	28.66	34.33	**72.33**	85.77	**52**	**89.55**	67.42
w/o LLM Reward	**88.72**	90.75	**74.74**	30.33	34.33	69.63	84.22	45	88.05	67.80
w/ Frozen Reward	87.43	90.20	74.55	31.33	33.66	70.06	84.84	46	88.26	67.93
Base	83.54	87.61	72.34	28.33	31.00	66.52	82.64	46	88.45	66.10
Joint Training	85.95	90.38	74.18	30.33	35.00	70.88	**85.84**	49	87.72	68.35

5.3 Main Results

As reported in Table 1, PAVIF improves instruction-following performance across the three Qwen3 scales, with the most consistent gains on Qwen3-4B-Instruct-2507. Our main observations are as follows:

Improves Instruction-Following Performance. A prominent finding stands out from our experiments: PAVIF consistently outperforms both the base models and SFT baselines across most benchmarks. Crucially, we observe clear improvements on hard and soft constraints, with gains of 3.83 points on IFBench and 2.67 points on CFBench over Qwen3-4B-Instruct-2507, suggesting that self-verification strengthens the model's instruction-following ability across diverse constraint types.

Outperforms Methods Relying on External Models. A notable result is that, without invoking any external model during generation optimization, our approach still outperforms VerIF on the majority of benchmarks. This advantage is most evident on Qwen3-4B-Instruct-2507, as evidenced by gains of 6.56 points on FollowBench (SSR) and 2.59 points on IFEval (Pr.L), with a similar trend observed at the 1.7B and 8B scales. Analysis of these results leads to the conclusion that the policy can provide effective feedback for instruction-following optimization, eliminating the need for a separate external LLM verifier during generation optimization.

Benefits Across Model Scales. PAVIF consistently improves over the corresponding base models across all three Qwen3 scales, with an average absolute improvement of 2.01 percentage points. These consistent improvements provide compelling evidence for the effectiveness and generality of PAVIF, underscoring its ability to enhance instruction following across models of varying capacities.

Effectively Unlocks the Potential of Strong Models. The benefit of PAVIF is most pronounced on Qwen3-4B-Instruct-2507, which exhibits strong baseline performance across both evaluation settings. Our method exhibits its most substantial performance gain on this model. This advantage may stem directly from the model's superior ability to harness self-verification signals to unlock latent instruction-following capabilities.

5.4 Ablation Studies

We conduct six ablation experiments on Qwen3-4B-Instruct-2507 to evaluate the contribution of each component. Results are reported in Table 2.

Effect of Each Training Stage. Both training stages contribute to instruction-following performance, with distinct effects across constraint types. Without verification training, performance deteriorates mainly on programmatically verifiable constraints and multi-turn instructions, suggesting that this stage provides a useful constraint-aware initialization for subsequent reinforcement learning. Furthermore, the absence of generation optimization affects nearly all constraint types, with the largest drops on IFEval and additional declines on fine-grained and multi-turn instruction following, showing that this stage is crucial for enhancing overall instruction-following performance.

Table 3: Accuracy (%) on LLMBar under the Vanilla (V) and NoRules (NR) settings. Avg. is the unweighted mean across the five subsets.

Model	Natural		Adversarial								Avg.	
			Neighbor		GPTInst		GPTOut		Manual			
	V	NR	V	NR	V	NR	V	NR	V	NR	V	NR
IF-Verifier	70.50	71.50	53.73	**50.00**	54.35	50.00	57.45	61.70	53.26	51.09	57.86	56.86
Base	**88.00**	**87.00**	56.34	42.91	65.76	60.87	**65.96**	59.57	60.87	**58.70**	67.39	61.81
+ VerIF	87.00	85.00	53.36	40.30	65.22	58.70	62.77	59.57	**61.96**	56.52	66.06	60.02
+ PAVIF	87.00	84.50	56.72	43.28	68.48	61.96	64.89	58.51	60.87	**58.70**	67.59	61.39
+ Stage1	87.00	85.50	**60.82**	45.15	**71.20**	**63.04**	62.77	**62.77**	**61.96**	**58.70**	**68.75**	**63.03**

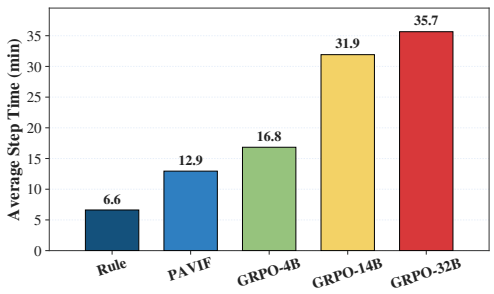


Figure 4: Average training-step time of different reward methods over one training epoch, using Qwen3-4B-Instruct-2507 as the policy model.

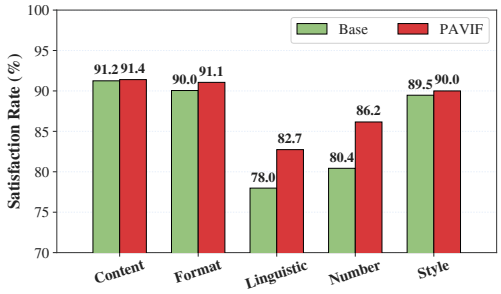


Figure 5: Constraint satisfaction rates across different constraint categories on InFoBench before and after training with PAVIF.

Contributions of Rule and LLM Rewards. Neither the rule reward nor the LLM reward fully compensates for the absence of the other, as removing either degrades overall performance. Specifically, performance on hard constraints deteriorates substantially without the rule reward, particularly on IFEval and Multi-IF, highlighting its role in precise constraint compliance. By providing feedback on constraints difficult to verify programmatically, the LLM reward improves multi-level and soft constraint following, although these gains do not extend to all constraint types.

Self-Verification vs. a Frozen Reward Model. Replacing the self-verification reward with a frozen reward model improves over the base model but falls short of ours on both hard and soft constraints, suggesting that the gains stem from the dynamic self-verification mechanism rather than a static reward model.

Two-Stage vs. Joint Training. Training on the combined data from both stages in a single GRPO run performs worse than our method, particularly on precise constraint following as measured by IFEval, demonstrating the importance of staged training.

5.5 Analysis of Verification Capability

To assess the effect of verification training, we compare the Stage 1 model against the base model and an open-source verifier on LLMBar. As shown in Table 3, the model trained for verification achieves the highest average accuracy among the evaluated models under both Vanilla and NoRules, indicating that **verification training improves instruction-following judgment**.

Given the observation from our preliminary experiments (§ 3.2) that verification training can benefit generation, we further examine whether generation optimization can similarly improve verification accuracy. Under Vanilla and NoRules, VerIF's average accuracy decreases by 1.33 and 1.79 percentage points relative to the base model, while PAVIF's decreases by 1.16 and 1.64 points relative to Stage 1. This contrast indicates that **generation optimization does not yield corresponding improvements in verification accuracy**, highlighting the importance of explicit verification training.

Table 4: Performance comparison of general capabilities between PAVIF and the baselines.

Model	Knowledge	Reasoning	Math	Coding	Avg. $\Delta\uparrow$
	MMLU ↑	ARC-C ↑	GSM8K ↑	HumanEval ↑	
Qwen3-1.7B	55.75	**43.26**	**68.08**	40.24	—
+ VerIF	**56.56** (+0.81)	41.81 (−1.45)	67.78 (−0.30)	40.85 (+0.61)	**−0.08**
+ PAVIF (Ours)	54.64 (−1.11)	40.78 (−2.48)	67.10 (−0.98)	**42.68** (+2.44)	−0.53
Qwen3-4B-Instruct-2507	70.62	**58.62**	85.52	**75.00**	—
+ VerIF	70.76 (+0.14)	57.85 (−0.77)	85.52 (+0.00)	74.39 (−0.61)	−0.31
+ PAVIF (Ours)	**70.78** (+0.16)	57.85 (−0.77)	**85.74** (+0.22)	**75.00** (+0.00)	**−0.10**
Qwen3-8B	**72.98**	56.66	87.57	63.41	—
+ VerIF	72.60 (−0.38)	56.57 (−0.09)	87.49 (−0.08)	64.02 (+0.61)	+0.02
+ PAVIF (Ours)	72.30 (−0.68)	**56.74** (+0.08)	**87.72** (+0.15)	**70.12** (+6.71)	**+1.57**

5.6 Analysis of Efficiency and Constraint Satisfaction

Computational Efficiency. As illustrated in Figure 4, under the same training configuration and reward-model scale, our PAVIF method requires 12.9 minutes per training step, compared with 16.8 minutes for GRPO-4B, which uses a frozen copy of the initial policy as its verifier. This represents a 23.2% reduction in training time, demonstrating that our method achieves higher training efficiency at the same model scale. Detailed experimental settings are provided in Appendix E.3.

Constraint-Following Improvements. To further investigate the performance gains, we analyze constraint satisfaction across categories on InFoBench and Multi-IF. As illustrated in Figure 5, the weaker numerical and linguistic categories achieve the largest gains of 5.8 and 4.7 percentage points, showing that PAVIF yields larger gains on constraint categories with lower initial satisfaction rates. The results on Multi-IF are reported in Appendix E.4.

5.7 Analysis of General Capability

Task-specific optimization for instruction following may come at the cost of general capabilities (Ouyang et al., 2022). To assess this effect, we compare models trained with different methods across general knowledge, reasoning, mathematical problem solving, and code generation.

As shown in Table 4, **PAVIF largely preserves general capabilities.** Its average score declines only slightly for the smaller models, while increasing by 1.57 points for Qwen3-8B. At this scale, PAVIF also achieves the highest average score among the compared training methods, exceeding the strongest baseline by 1.55 points.

Notably, PAVIF's performance on HumanEval reveals its potential for code generation. It matches the base model's performance for Qwen3-4B-Instruct-2507 and gains 6.71 points for Qwen3-8B, outperforming VerIF by 6.10 points. This result suggests that self-verification may provide an additional benefit for code generation, where explicit constraint checking is closely related to logical correctness and format compliance.

6 Conclusion

In this work, we introduce PAVIF, a reinforcement learning framework that improves instruction following through learned self-verification. Compared with approaches that rely on external verifiers or directly reuse existing judgment ability, PAVIF explicitly develops the policy's verification capability and turns it into feedback for its own generation optimization. Experiments show that PAVIF improves instruction-following performance while reducing per-step training time and largely preserving general capabilities. The observed gains in code generation further suggest that these benefits may extend beyond instruction following. Overall, these findings suggest that cultivating and using a model's own verification capability can support self-improvement, motivating further investigation into how this capability evolves during policy optimization and how its accuracy influences generation performance.

AI USE STATEMENT

In this work, we used generative AI tools to assist with data construction and annotation, as well as polish the manuscript. Specifically, we used Qwen3-235B-A22B-Instruct-2507 for constraint removal and verification scoring during verification dataset construction. The data produced with AI assistance were not used directly for training. Instead, we manually reviewed a subset to ensure the reliability of the resulting training data. In addition, we used GPT-6 to improve the language quality, readability, and grammatical correctness of the manuscript, rather than to directly generate substantive content. The AI-assisted manuscript text was manually reviewed and revised to ensure correctness and avoid hallucinations. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

Throughout this study, we adhere to the principles outlined in the ICLR Code of Ethics. Our work focuses primarily on improving the instruction-following capabilities of open-source LLMs through self-verification. The training and evaluation data are derived from publicly available datasets and benchmarks. Although a subset of the automatically constructed training data was manually reviewed, we acknowledge that the data may still reflect biases in the models used for generation and annotation.

REPRODUCIBILITY STATEMENT

To ensure the reproducibility of our research, we provide detailed descriptions of the methodology and experimental settings. The complete two-stage training procedure, reward functions, and automated construction of the self-verification dataset are described in Section 4. PAVIF is implemented using EasyR1 (Zheng et al., 2025), an open-source reinforcement learning framework built on veRL, and experiments are conducted using publicly available Qwen3 models and benchmarks. To further support reproducibility, our code is available at `https://anonymous.4open.science/r/PAVIF-310B`, and detailed training configurations and evaluation protocols are provided in the appendices.

REFERENCES

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K. Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025. doi: 10.48550/arXiv.2508.10925. URL https://arxiv.org/abs/2508.10925.

Kaikai An, Li Sheng, Ganqu Cui, Shuzheng Si, Ning Ding, Yu Cheng, and Baobao Chang. UltraIF: Advancing instruction following from the wild. In *EMNLP*, 2025. doi: 10.18653/v1/2025.emnlp-main.945. URL https://aclanthology.org/2025.emnlp-main.945/.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. doi: 10.48550/arXiv.2107.03374. URL https://arxiv.org/abs/2107.03374.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. *arXiv:1803.05457v1*, 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. doi: 10.48550/arXiv.2110.14168. URL https://arxiv.org/abs/2110.14168.

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. doi: 10.48550/arXiv.2410.15553. URL https://arxiv.org/abs/2410.15553.

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. doi: 10.18653/v1/2026.acl-long.820. URL https://aclanthology.org/2026.acl-long.820/.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *ICLR*, 2021. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

Hui Huang, Jiaheng Liu, Yancheng He, Shilong Li, Bing Xu, Conghui Zhu, Muyun Yang, and Tiejun Zhao. MuSC: Improving complex instruction following with multi-granularity self-contrastive training. In *ACL*, 2025a. doi: 10.18653/v1/2025.acl-long.523. URL https://aclanthology.org/2025.acl-long.523/.

Xiang Huang, Ting-En Lin, Feiteng Fang, Yuchuan Wu, Hangyu Li, Yuzhong Qu, Fei Huang, and Yongbin Li. Reverse preference optimization for complex instruction following. In *Findings of ACL*, 2025b. doi: 10.18653/v1/2025.findings-acl.1064. URL https://aclanthology.org/2025.findings-acl.1064/.

Tingfeng Hui, Pengyu Zhu, Bowen Ping, Ling Tang, Guanting Dong, Yaqi Zhang, and Sen Su. DecIF: Improving instruction-following through decomposition. In *ACL*, 2026. doi: 10.18653/v1/2026.acl-long.36. URL https://aclanthology.org/2026.acl-long.36/.

Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. GPT-4o system card. *arXiv preprint arXiv:2410.21276*, 2024. doi: 10.48550/arXiv.2410.21276. URL https://arxiv.org/abs/2410.21276.

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. doi: 10.18653/v1/2024.acl-long.257. URL https://aclanthology.org/2024.acl-long.257/.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Xinxi Lyu, et al. Tülu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024. doi: 10.48550/arXiv.2411.15124. URL https://arxiv.org/abs/2411.15124.

Renze Lou, Kai Zhang, and Wenpeng Yin. Large language model instruction following: A survey of progresses and challenges. *Computational Linguistics*, 2024. doi: 10.1162/coli_a_00523. URL https://aclanthology.org/2024.cl-3.7/.

Rongchuan Mu, Zexin Wang, Qianyu Wang, MingHua Ma, Zekun Wang, Ming Liu, and Bing Qin. PARIF: Pushing the Pareto frontier of instruction following and reasoning with curriculum reinforcement learning. In *ACL*, 2026. doi: 10.18653/v1/2026.acl-long.1136. URL https://aclanthology.org/2026.acl-long.1136/.

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. In *NeurIPS*, 2022. doi: 10.52202/068431-2011. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/b1efde53be364a73914f58805a001731-Paper-Conference.pdf.

Hao Peng, Yunjia Qi, Xiaozhi Wang, Bin Xu, Lei Hou, and Juanzi Li. VerIF: Verification engineering for reinforcement learning in instruction following. In *EMNLP*, 2025. doi: 10.18653/v1/2025.emnlp-main.1542. URL https://aclanthology.org/2025.emnlp-main.1542/.

Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hannaneh Hajishirzi. Generalizing verifiable instruction following. In *NeurIPS*, 2025. doi: 10.52202/085713-1645. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/46499a0622ecf568b72d17b61e45dbd5-Abstract-Datasets_and_Benchmarks_Track.html.

Yunjia Qi, Hao Peng, Xiaozhi Wang, Bin Xu, Lei Hou, and Juanzi Li. Constraint back-translation improves complex instruction following of large language models. In *CIKM*, 2025. doi: 10.1145/3746252.3761324. URL https://dl.acm.org/doi/10.1145/3746252.3761324.

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*, pp. 13025–13048, 2024. doi: 10.18653/v1/2024.findings-acl.772. URL https://aclanthology.org/2024.findings-acl.772/.

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. In *NeurIPS*, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/9baf31febefde7bd76023c2d2f13cbd7-Abstract-Conference.html.

Qingyu Ren, Jie Zeng, Qianyu He, Jiaqing Liang, Yanghua Xiao, Weikang Zhou, Zeye Sun, and Fei Yu. Step-by-step mastery: Enhancing soft constraint following ability of large language models. In *Findings of ACL*, 2025. doi: 10.18653/v1/2025.findings-acl.1004. URL https://aclanthology.org/2025.findings-acl.1004/.

Qingyu Ren, Qianyu He, Powei Chang, Jie Zeng, Zeye Sun, Fei Yu, Jiaqing Liang, and Yanghua Xiao. Instructions are all you need: Self-supervised reinforcement learning for instruction following. In *ACL*, 2026a. doi: 10.18653/v1/2026.acl-long.217. URL https://aclanthology.org/2026.acl-long.217/.

Qingyu Ren, Qianyu He, Jiajie Zhu, Xingzhou Chen, Jingwen Chang, Zeye Sun, Han Xia, Fei Yu, Jiaqing Liang, and Yanghua Xiao. SEIF: Self-evolving reinforcement learning for instruction following. *arXiv preprint arXiv:2605.07465*, 2026b. doi: 10.48550/arXiv.2605.07465. URL https://arxiv.org/abs/2605.07465.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. doi: 10.48550/arXiv.2402.03300. URL https://arxiv.org/abs/2402.03300.

Haoran Sun, Lixin Liu, Junjie Li, Fengyu Wang, Baohua Dong, Ran Lin, and Ruohui Huang. Conifer: Improving complex constrained instruction-following ability of large language models. *arXiv preprint arXiv:2404.02823*, 2024. doi: 10.48550/arXiv.2404.02823. URL https://arxiv.org/abs/2404.02823.

Bosi Wen, Yilin Niu, Cunxiang Wang, Pei Ke, Xiaoying Ling, Ying Zhang, Aohan Zeng, Hongning Wang, and Minlie Huang. IF-CRITIC: Towards a fine-grained LLM critic for instruction-following evaluation. In *ACL*, 2026. doi: 10.18653/v1/2026.acl-long.1147. URL https://aclanthology.org/2026.acl-long.1147/.

Tianze Xu, Yanzhao Zheng, Pengrui Lu, Lyumanshan Ye, Yong Wu, Zhentao Zhang, Yuanqiang Yu, Chao Ma, Jihuai Zhu, Pengfei Liu, et al. Rubrics to tokens: Bridging response-level rubrics and token-level rewards in instruction following tasks. In *COLM*, 2026. URL https://openreview.net/forum?id=QHqWI2cBDr.

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. doi: 10.48550/arXiv.2505.09388. URL https://arxiv.org/abs/2505.09388.

Weizhe Yuan, Ilia Kulikov, Ping Yu, Kyunghyun Cho, Sainbayar Sukhbaatar, Jason E Weston, and Jing Xu. Following length constraints in instructions. In *EMNLP*, 2025. doi: 10.18653/v1/2025.emnlp-main.1233. URL https://aclanthology.org/2025.emnlp-main.1233/.

Yirong Zeng, Yufei Liu, Xiao Ding, Yutai Hou, Yuxian Wang, Wu Ning, Haonan Song, Dandan Tu, Qixun Zhang, Yuxiang He, et al. TinyJudge: Unverifiable constraint alignment via lightweight specialist ensembles. In *ACL*, 2026. doi: 10.18653/v1/2026.acl-long.1204. URL https://aclanthology.org/2026.acl-long.1204/.

Zhiyuan Zeng, Jiatong Yu, Tianyu Gao, Yu Meng, Tanya Goyal, and Danqi Chen. Evaluating large language models at evaluating instruction following. In *ICLR*, 2024. URL https://arxiv.org/abs/2310.07641.

Tao Zhang, Chenglin Zhu, Yanjun Shen, Wenjing Luo, Yan Zhang, Hao Liang, Tao Zhang, Fan Yang, Mingan Lin, Yujing Qiao, et al. CFBench: A comprehensive constraints-following benchmark for LLMs. In *ACL*, 2025a. doi: 10.18653/v1/2025.acl-long.1581. URL https://aclanthology.org/2025.acl-long.1581/.

Xinghua Zhang, Haiyang Yu, Cheng Fu, Fei Huang, and Yongbin Li. IOPO: Empowering LLMs with complex instruction following via input-output preference optimization. In *ACL*, 2025b. doi: 10.18653/v1/2025.acl-long.1079. URL https://aclanthology.org/2025.acl-long.1079/.

Yaowei Zheng, Junting Lu, Shenzhi Wang, Zhangchi Feng, Dongdong Kuang, Yuwen Xiong, and Richong Zhang. EasyR1: An efficient, scalable, multi-modality RL training framework. https://github.com/hiyouga/EasyR1, 2025.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023. doi: 10.48550/arXiv.2311.07911. URL https://arxiv.org/abs/2311.07911.

A PRELIMINARY STUDY

Setup. We randomly sample 100 prompts from each of IFEval and InFoBench to evaluate the model before and after verification training. For both models, we use the same generation settings and sample $n = 1024$ responses per prompt on IFEval and $n = 64$ on InFoBench at temperature 0.7, using different random seeds across rollouts. No further selection or revision is applied to the generated responses. We report pass@k for $k \leq n$, with InFoBench responses evaluated by gpt-oss-120b.

Pass@k estimation. Following Chen et al. (2021), we estimate pass@k using the unbiased estimator

$$\widehat{\text{pass@}k} = \frac{1}{N}\sum_{i=1}^{N}\left[1 - \frac{\binom{n-c_i}{k}}{\binom{n}{k}}\right], \qquad 1 \leq k \leq n, \tag{7}$$

where $N = 100$ is the number of prompts, n is the number of responses sampled per prompt, and c_i is the number of successful responses for prompt i.

Interpretation. Pass@k measures the probability that at least one of k sampled responses satisfies all constraints, with pass@1 reflecting single-sample reliability. Its increase with k quantifies the benefit of additional sampling. Comparing pass@1 with pass@k at large k helps distinguish the ability to generate compliant responses from the consistency with which the model does so. In particular, low pass@1 but high pass@k indicates that compliant responses are attainable, although they are unlikely to be generated in a single attempt.

A.1 PRELIMINARY RESULTS ON INFOBENCH

While IFEval primarily evaluates hard constraints, InFoBench provides a more comprehensive evaluation of instruction following by including soft constraints. As in Section 3.2, we report pass@k on InFoBench for $k \in \{1, 2, 4, 8, 16, 32, 64\}$ using the same models and evaluation setup.

As shown in Figure 6, verification training improves overall performance on InFoBench. The model equipped with verification capability achieves higher pass@k at most evaluated sampling budgets, while the two curves nearly overlap at $k = 2$. Together with the IFEval results, these observations suggest that verification training can benefit generation for instructions involving both hard and soft constraints.

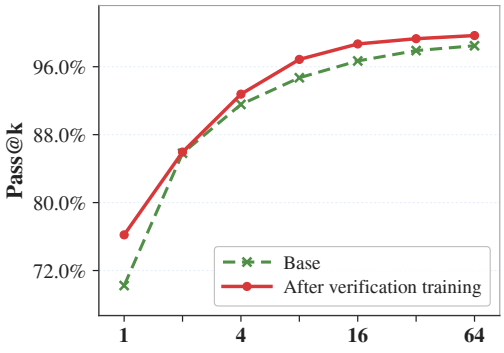


Figure 6: Pass@k on InFoBench before and after verification training.

B DATA CONSTRUCTION

Training data source. Our Stage 2 training data are drawn from the dataset released by Ren et al. (2026a)[1]. This work constructs multi-constraint instructions from 3,000 seed instructions using

[1] https://github.com/Rainier-rq/verl-if

Table 5: Composition of the training dataset.

Component	Examples
Instruction following	13,570
Mathematics	4,501
Science	1,929
Total	20,000

GPT-4o, covering 23 hard-constraint types and 25 soft-constraint types. Examples include JSON formatting and capitalization for hard constraints, and style and role requirements for soft constraints. Its instruction-following data contain one to five constraints per instruction. Table 5 summarizes the source paper's reported composition. These counts describe the full source dataset used for generation optimization, from which we randomly select 2,000 complex instructions for verification optimization.

Verification training data. We randomly select 2,000 instruction-following instructions from the Stage 2 training data to construct the Stage 1 verification dataset. For each selected instruction, we create three constraint-coverage settings: no constraints, partial constraints, and full constraints, as described in Section 4.2. For each setting, we select two responses from multiple policy-generated candidates, yielding six responses per original instruction and approximately 12,000 verification examples in total ($2{,}000 \times 3 \times 2 = 12{,}000$). Each candidate response is paired with the original complete instruction for teacher labeling and subsequent verification optimization with GRPO.

Verification labels. We use Qwen3-235B-A22B-Instruct-2507 to label these instruction–response pairs. The teacher first analyzes whether the response satisfies each constraint in the original instruction and we then obtain an overall verification score $s \in \{0, 0.5, 1\}$ from these scores, as described in Algorithm 1. During verification optimization, this score serves as a reference for computing rewards for the policy's verification outputs. This training develops the policy's self-verification capability by encouraging it to accurately assess how well the responses satisfy the given constraints.

Algorithm 1 Teacher Labeling

Require: Original instruction $q = (x, \mathcal{C})$ and candidate response y
Ensure: Verification score $s \in \{0, 0.5, 1\}$
1: Obtain the teacher's judgments for (q, y) using the prompt in Table 8
2: Extract scores $b_1, \ldots, b_K \in \{0, 1\}$, where $K = |\mathcal{C}|$
3: $\bar{b} \leftarrow \frac{1}{K} \sum_{i=1}^{K} b_i$
4: **if** $\bar{b} = 1$ **then**
5: $\quad s \leftarrow 1$
6: **else if** $\bar{b} \geq 0.5$ **then**
7: $\quad s \leftarrow 0.5$
8: **else**
9: $\quad s \leftarrow 0$
10: **end if**
11: **return** s

B.1 Prompt Templates for Data Construction

We first extract constraints using Table 6, then use Table 7 to remove all extracted constraints for the no-constraint variant or a subset for the partial-constraint variant. The full-constraint variant retains the original instruction. Table 8 provides the supplied teacher evaluation prompt, which requests a binary judgment for each constraint against the original complete instruction. The policy verification templates are provided in Appendix D.2.

Table 6: Constraint Extraction Prompt

Constraint Extraction

Role

You are a precise Instruction Analysis Expert, specialized in extracting core **Constraints** from complex natural language descriptions.

Task

Analyze the provided ‘[Instruction to Extract]‘. Identify all constraints, formatting requirements, style suggestions, and word counts. Note: The user’s core intent (e.g., writing, translation) is not considered a constraint.

Extraction Rules (extraction_logic)

1. **De-redundancy**: Remove polite phrases (e.g., ”Please help me,” ”Kindly”) and filler words; retain only core constraints.
2. **Atomization**: Break down complex, compound requirements into independent, simple conditions.
3. **Standardization**: Use ” %% ” as the logical connector in the output.
4. **Exclude Interference**: If the instruction includes start/end markers (e.g., ”start with XX”), these are constraints and must be extracted.
5. **Prohibited Terms**: Strictly forbid the output of tags like ‘[|Human|]‘, ‘[|AI|]‘, ‘<s>‘, ‘</s>‘, or the phrase ”Extracted Constraints.”

Output Format

Output the extracted constraint string directly. Do **not** include any introductory text (e.g., ”The result is:”), explanations, or repeated constraints.

Examples

[Instruction to Extract]: Write a story for me. Requirements: The word ”I” cannot appear, the ending must be a tragedy, total word count 500. The above is the instruction; it must start with ”[Answer Start]” and end with ”[Answer End]”.
[Extracted Constraints]: The word ”I” cannot appear %% The ending must be a tragedy %% Total word count 500 words %% Start with ”[Answer Start]” %% End with ”[Answer End]”

[Instruction to Extract]: Please translate this text into English. Requirements: Academic style, strictly no first-person pronouns, keep under 200 words, and output in table format.
[Extracted Constraints]: Academic style %% No first-person pronouns %% Fewer than 200 words%% Output in table format

Input

[Instruction to Extract]: instruction [Extracted Constraints]:

Table 7: Constraint Removal Prompt

Constraint Removal

Role

You are a senior text editing expert, skilled at streamlining instructions while maintaining natural and fluent language.

Task

Please remove all [Constraints to Delete] from the [Original Instruction], and reorganize the remaining core tasks into a logically coherent instruction.

Definitions

* **Core Task**: The user’s essential request (e.g., writing a story, summarizing, writing code).
* **Constraints to Delete**: Extracted restrictive conditions such as format, word count, punctuation, structure, etc.

Output Rules

1. **Return only the modified new instruction**. Strictly forbid any explanatory text (e.g., ”The modified instruction is as follows:”).
2. **Ensure Natural Fluency**: If removing the constraints makes the original sentence fragmented, please rewrite it based on the core task.

—

[Example]
[Original Instruction]: Please generate a report on quantum entanglement. The title must be marked with $()$. Chinese commas are not allowed. Please begin.
[Constraints to Delete]: The title must be marked with $()$
[Modified Instruction]: Please generate a report on quantum entanglement. Chinese commas are not allowed.

—

[Task Input]
[Original Instruction]: {instruction}
[Constraints to Delete]: {extracted_constraints}
[Modified Instruction]:

Table 8: Teacher Evaluation Prompt

Teacher Evaluation

I want you to act as a quality evaluator. You need to consider both the [User Instruction] and the [Model Answer], and evaluate the model answer by assigning a score of 0 or 1.

Specifically, the [Model Answer] is a response to the [User Instruction], and the [User Instruction] contains the requirements and evaluation criteria that the model answer should satisfy. You must strictly follow each evaluation point specified in the instruction and score the [Model Answer] accordingly. If the instruction requirement is fully satisfied, assign 1; if it is not satisfied, assign 0.

Output format:

1. Extract each constraint from the [User Instruction] in order, separating them with "&&".
2. For each line, first output the extracted constraint, followed by the corresponding score (0 or 1) in \box{}.
3. Directly output your evaluation. Do not output any other content.

You may refer to the example below.

[Example 1]:
[User Instruction]:Please perform the following edit on the given sentence "I love eating apples."\n\n1. Replace "I" with "Xiaoming."\n\nOnly output the edited sentence and present the answer without any commas.
[Model Answer]: "Okay, the modified sentence is: Xiaoming loves eating apples."
[Your Evaluation]:
Consistent with the reference answer; replacement is accurate\box{1} && Contains extra redundant information\box{0} && present the answer without any commas\box{0}

Below is the task input:

[User Instruction]: {question}
[Model Answer]: {answer}
[Your Evaluation]:

C Benchmarks and Evaluation Details

This section describes the benchmarks and evaluation settings used in our experiments. The contents are organized as follows:

- Appendix C.1: instruction-following benchmarks for assessing constraint satisfaction.
- Appendix C.2: general capability benchmarks covering knowledge, scientific reasoning, mathematical reasoning, and code generation.
- Appendix C.3: the LLMBar benchmark and evaluation setup for assessing verification capability.
- Appendix C.4: judge configurations and evaluation metrics.

C.1 Instruction-Following Benchmarks

IFEval. IFEval (Zhou et al., 2023) evaluates compliance with objectively verifiable instructions, such as word-count limits, keyword requirements, and output-format constraints. Each prompt contains one or more requirements that can be checked programmatically, enabling reproducible evaluation of precise instruction following.

Multi-IF. Multi-IF (He et al., 2024) extends IFEval to multi-turn and multilingual interactions. It contains three-turn conversations across eight languages and assesses whether models can continue to follow instructions as the conversation progresses. We report results separately for the three turns.

IFBench. IFBench (Pyatkin et al., 2025) evaluates generalization in precise instruction following through 58 diverse, verifiable constraint types. It introduces challenging output requirements beyond those covered by earlier benchmarks, testing whether instruction-following improvements extend to a broader range of constraints.

FollowBench. FollowBench (Jiang et al., 2024) evaluates fine-grained constraints in five categories: content, situation, style, format, and example. Its multi-level design progressively adds constraints to an initial instruction, allowing evaluation of how models handle increasing instruction complexity.

CFBench. CFBench (Zhang et al., 2025a) is a Chinese benchmark for following complex constraints in realistic user instructions. It covers diverse application scenarios and constraint categories,

and evaluates compliance from multiple perspectives, including individual constraints, complete instructions, and requirement priorities.

InFoBench. InFoBench (Qin et al., 2024) evaluates instruction following by decomposing complex instructions into simpler, individually assessable requirements. Its Decomposed Requirements Following Ratio measures compliance with these requirements, providing a fine-grained assessment across diverse constraint categories.

C.2 General Capability Benchmarks

MMLU. MMLU (Hendrycks et al., 2021) measures knowledge and problem-solving ability through multiple-choice questions spanning 57 subjects, including the humanities, social sciences, and STEM. We use it to assess general knowledge retention and report accuracy.

ARC-Challenge. ARC-Challenge (Clark et al., 2018) contains challenging grade-school science questions selected because they are difficult for retrieval and word co-occurrence baselines. It tests the ability to combine scientific knowledge with reasoning. We report accuracy.

GSM8K. GSM8K (Cobbe et al., 2021) consists of grade-school mathematics word problems that require multiple reasoning and arithmetic steps. We use it to assess mathematical reasoning and report final-answer accuracy.

HumanEval. HumanEval (Chen et al., 2021) evaluates code generation through Python programming problems specified by function signatures and docstrings. Generated programs are assessed for functional correctness using unit tests. We report Pass@1.

C.3 Verification Capability Benchmark

LLMBar. LLMBar (Zeng et al., 2024) is a meta-evaluation benchmark for assessing an evaluator's ability to judge instruction following. It contains 419 instances, each consisting of an instruction, two candidate responses, and a gold label identifying the response that correctly follows the instruction. The benchmark comprises a Natural set and an Adversarial set containing four subsets (Neighbor, GPTInst, GPTOut, and Manual). The adversarial examples test whether evaluators can distinguish instruction-compliant responses from superficially appealing but noncompliant ones. We compare the Stage 1 checkpoint with the base model and VerIF under the Vanilla (with evaluation rules) and NoRules (without evaluation rules) prompting settings, reporting accuracy for each subset and the unweighted average across all five subsets (Table 3).

C.4 Evaluation Details

Evaluation setup. For model-based evaluation, we adopt gpt-oss-120b as the judge for the main results in Table 1 and the ablation results in Table 2. Supplementary results using GPT-4o are provided in Appendix E.1 and Appendix E.2. Each table uses a single judge for its model-based evaluations. The general capability evaluations report accuracy for MMLU, ARC-Challenge, and GSM8K, and Pass@1 for HumanEval.

Score aggregation. For Tables 1, 2, 11, and 12, Avg. is computed by first averaging the reported metrics within each benchmark and then taking the unweighted mean across the six benchmarks. In Table 2, CFBench is represented by ISR only.

D Training and Implementation Details

D.1 Baseline Settings

Base. For each model setting, Base denotes the corresponding checkpoint before our task-specific training. It serves as the reference for measuring changes in instruction following and general capabilities.

Table 9: Training hyperparameters for the two stages.

Hyperparameter	Stage I	Stage II
Max response length	8192	8192
Responses per prompt	5	5
Sampling temperature	1.0	1.0
Global batch size	256	256
Micro batch size for update	1	1
Micro batch size for experience	4	4
Learning rate	1×10^{-6}	1×10^{-6}
Clip ratio	0.2	0.2
KL loss coefficient	0.08	0.01

Conifer-7B-DPO. Conifer (Sun et al., 2024) constructs complex instruction-following data using GPT-4 and adopts a progressive learning scheme with easy-to-hard instructions and process feedback. The model undergoes supervised fine-tuning followed by DPO.

TULU3-VerIF. VerIF (Peng et al., 2025) combines rule-based code verification with LLM-based verification to provide rewards for reinforcement learning on the VerInstruct dataset. TULU3-VerIF is the TULU3-based model trained with this approach.

UltraIF-8B-DPO. UltraIF (An et al., 2025) decomposes real-world user prompts into simpler queries, constraints, and evaluation questions, and trains an UltraComposer to synthesize complex instructions. The evaluation questions guide response filtering for subsequent model alignment; we include its 8B DPO variant.

SFT. The SFT baseline is trained on instruction–response pairs constructed using Qwen3-235B-A22B-Instruct-2507. It uses supervised response prediction conditioned on the input instruction, in contrast to the verification objective used in PAVIF's first-stage GRPO training.

VerIF. VerIF (Peng et al., 2025) combines programmatic checks with external LLM judgments to provide reinforcement learning rewards. In our implementation, we use IF-Verifier-7B, the trained verifier released by VerIF, and the same training data as PAVIF.

D.2 Training Configuration

Implementation and hyperparameters. We implement GRPO using the EasyR1 framework and train on 4 nodes, each equipped with 4 NVIDIA L40S GPUs (16 GPUs in total). We use Fully Sharded Data Parallel (FSDP) with full sharding, gradient checkpointing, and actor parameter and optimizer offloading. Rollout tensor parallelism is set to 2.

Table 9 summarizes the key settings for both stages. Each update uses 256 prompt groups, with five responses sampled per prompt at temperature 1.0 and a maximum response length of 8192 tokens. The per-GPU micro-batch sizes are 1 for policy updates and 4 for log-probability computation. Both stages use AdamW with a constant learning rate of 1×10^{-6} and a clipping parameter of $\epsilon = 0.2$. KL regularization is applied through a separate actor loss using the low_var_kl estimator, with coefficients of 0.08 in Stage 1 and 0.01 in Stage 2; the reward-level KL coefficient is zero. Training data are shuffled with seed 1, and overlong prompts are filtered.

E Full Experimental Results

This section collects the complete comparison and ablation results, together with a supplementary analysis of multi-turn constraint following. The results are organized as follows:

- Appendix E.1: supplementary Qwen3 comparisons evaluated with GPT-4o (Table 11).
- Appendix E.2: detailed ablation results evaluated with GPT-4o (Table 12).
- Appendix E.3: settings for evaluating computational efficiency.
- Appendix E.4: multi-turn constraint-following analysis on Multi-IF (Figure 7).

Table 10: Self-Verification Prompt

Self-Verification

You are a rigorous Instruction-Following Evaluation Expert. Your mission is to extract each explicit constraint from the [User Instruction] and verify whether the [Model Answer] satisfies it.

Tasks

1. **Extract Constraints**: Identify and extract all functional, formatting, word count, or content-based constraints from the [User Instruction].
2. **Point-by-Point Verification**: Compare the [Model Response] against each extracted constraint.
3. **Detailed Assessment**: Provide specific evidence or reasons for success/failure for every single constraint.
4. **Final Scoring**: Assign a final score of 0, 0.5, or 1 based on overall compliance.

Scoring Criteria

\boxed{1}: Perfect compliance. Followed all extracted constraints with zero omissions.
\boxed{0.5}: Partial compliance. Met at least half, but not all, of the extracted constraints.
\boxed{0}: Failed. Violated more than half of the extracted constraints.

Output Format (Strictly Enforced)
Below is my evaluation of the response:

1. **Constraint Decomposition and Verification**:
[Constraint 1] && [Score 0/1]: [Briefly explain if the constraint was met and why]
[Constraint 2] && [Score 0/1]: [Briefly explain if the constraint was met and why]
(Continue for all identified constraints)

2. **Detailed Assessment Summary**:
//Provide your holistic evaluation here.
//You must explicitly state where the model excelled and where it deviated, hallucinated, or failed in formatting.
//If the model performed perfectly, explain how it precisely aligned with the complex instructions.

Based on my assessment, the final score is:
Score=\boxed{[0, 0.5, or 1]}

Here is your task input:

[User Instruction]: {question}
[Model Answer]: {answer}
[Your Evaluation]:

Table 11: Performance comparison of different methods across Qwen3 model scales on six instruction-following benchmarks. VerIF and PAVIF use the same training data. Model-based judgments use GPT-4o. Avg. is the unweighted mean across the six benchmarks, with metrics averaged within each benchmark.

Method	IFEval		Multi-IF			IFBench		FollowBench		CFBench			InFoBench	Avg.
	Pr.S	Pr.L	Turn 1	Turn 2	Turn 3	Pr.S	Pr.L	HSR	SSR	CSR	ISR	PSR	Overall	
Qwen3-1.7B														
Base	68.57	72.27	67.44	54.85	43.95	19.00	21.66	49.26	59.08	76	42	53	79.64	56.16
+ VerIF	71.71	75.41	69.33	57.64	46.71	18.66	22.33	51.44	60.34	**80**	**48**	**60**	**81.07**	58.60
+ PAVIF (Ours)	**72.45**	**77.44**	**70.60**	**58.49**	**48.25**	**19.66**	**24.66**	**53.21**	**62.42**	76	43	54	80.53	**58.71**
Qwen3-4B-Instruct-2507														
Base	83.54	87.61	82.07	72.11	62.85	28.33	31.00	66.19	77.26	87	65	74	88.00	70.44
+ VerIF	86.87	89.09	83.90	73.67	64.44	30.66	32.66	68.66	79.16	**88**	65	**75**	87.56	71.85
+ PAVIF (Ours)	**88.35**	**91.68**	**84.31**	**74.65**	**65.27**	**31.66**	**35.33**	**76.45**	**80.63**	**88**	**66**	**75**	**89.60**	**73.79**
Qwen3-8B														
Base	83.54	85.76	83.19	73.61	64.52	26.33	30.33	62.77	68.78	86	59	70	**87.64**	68.64
+ VerIF	84.10	87.61	**84.50**	**74.69**	**66.27**	**27.00**	**31.33**	64.31	69.67	**87**	**62**	**72**	87.16	69.67
+ PAVIF (Ours)	**85.21**	**88.53**	83.80	74.53	65.91	**27.00**	**31.33**	**64.54**	**70.38**	**87**	61	**72**	87.16	**69.79**

E.1 Supplementary Comparisons with GPT-4o

Table 11 compares Base, VerIF, and PAVIF across the three Qwen3 model scales, using GPT-4o for model-based evaluation. These results supplement the main comparison with an alternative judge. Method differences should be assessed within the same evaluation setting.

E.2 Detailed Ablation Results with GPT-4o

Table 12: Ablation results on the Qwen3-4B-Instruct-2507 model. Model-based judgments use GPT-4o.

Method	IFEval		Multi-IF			IFBench		FollowBench		CFBench			InFoBench	Avg.
	Pr.S	Pr.L	Turn 1	Turn 2	Turn 3	Pr.S	Pr.L	HSR	SSR	CSR	ISR	PSR	Overall	
PAVIF	88.35	**91.68**	84.31	**74.65**	65.27	**31.66**	35.33	76.45	80.63	**88.00**	66.00	75.00	89.60	**73.79**
w/o stage 1	85.58	90.01	84.16	73.47	63.60	31.33	**36.00**	71.53	77.57	**88.00**	67.00	74.00	**90.36**	72.74
w/o stage 2	83.54	87.06	82.16	71.99	62.52	31.33	32.66	63.65	71.70	86.00	65.00	74.00	88.13	70.05
w/o Rule Reward	76.15	87.61	81.31	70.30	60.04	28.66	34.33	71.10	77.91	**88.00**	**69.00**	**77.00**	90.00	71.07
w/o LLM Reward	**88.72**	90.75	**85.07**	74.45	64.69	30.33	34.33	72.31	80.96	86.00	66.00	73.00	87.78	72.70
w/ Frozen Reward	87.43	90.20	83.95	74.14	**65.55**	31.33	33.66	**77.28**	**82.87**	**88.00**	65.00	75.00	87.96	73.32
Base	83.54	87.61	82.07	72.11	62.85	28.33	31.00	66.19	77.26	87.00	65.00	74.00	88.00	70.44
Joint Training	85.95	90.38	84.33	73.98	64.24	30.33	35.00	68.84	76.60	87.00	66.00	76.00	88.62	72.11

Table 12 reports the Qwen3-4B-Instruct-2507 ablations using GPT-4o for model-based evaluation. The table includes all three Multi-IF turns and the three CFBench metrics.

E.3 Efficiency Evaluation Settings

Experimental setup. We compare the training efficiency of five configurations using Qwen3-4B-Instruct-2507 (Figure 4). All methods follow the generation optimization settings in Appendix D.2 and use identical input prompts. PAVIF and the GRPO baselines use the verification prompt in Table 10. Rewards are computed as follows:

- **Rule:** Only rule-based rewards are used.
- **PAVIF:** In addition to rule-based rewards, the policy provides rewards by evaluating all constraints in its own responses, with shared parameters for generation and verification.
- **GRPO-4B:** In addition to rule-based rewards, a separate, frozen Qwen3-4B-Instruct-2507 model serves as the reward model and scores the policy's responses.
- **GRPO-14B:** In addition to rule-based rewards, a separate, frozen Qwen3-14B serves as the external reward model.
- **GRPO-32B:** In addition to rule-based rewards, a separate, frozen Qwen3-32B serves as the external reward model.

The 4B, 14B, and 32B suffixes denote the external reward-model scale; the policy base remains Qwen3-4B-Instruct-2507 in all cases.

Timing metric. We report the average wall-clock time per training step over one training epoch during generation optimization. This comparison measures per-step training efficiency and excludes verification-data construction and Stage 1 verification training. PAVIF requires 12.9 minutes per step, compared with 16.8 minutes for GRPO-4B, corresponding to a reduction of $(16.8 - 12.9)/16.8 \times 100\% = 23.2\%$ after rounding. This comparison holds the policy and verifier scale at 4B while contrasting parameter-sharing self-verification with a separate frozen verifier.

E.4 Multi-Turn Constraint-Following Improvements on Multi-IF

While InFoBench evaluates single-turn responses against both hard and soft constraints, Multi-IF assesses compliance with hard constraints across multiple turns. We therefore examine constraint satisfaction across its three turns to complement the analysis in the main text. As shown in Figure 7,

PAVIF improves constraint satisfaction by 2.2, 2.5, and 2.4 percentage points in the first, second, and third rounds, respectively, with improvements in 26 of the 27 evaluated settings. These results indicate that the gains extend across all three rounds and are observed in most evaluated settings, supporting PAVIF's effectiveness in multi-turn constraint following.

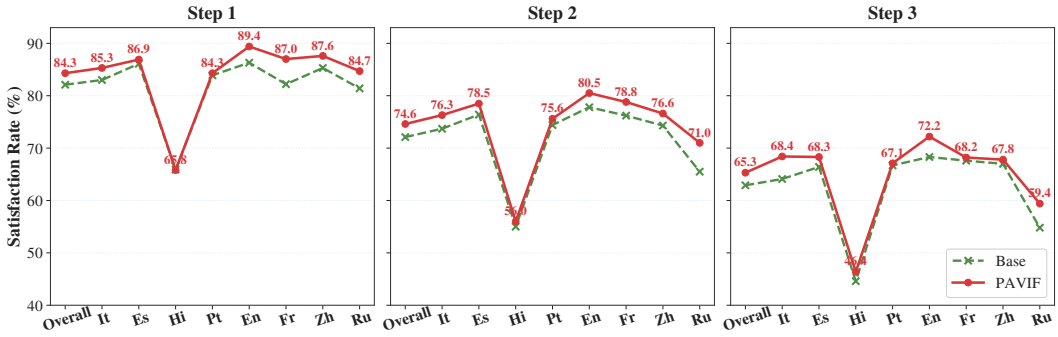


Figure 7: Constraint satisfaction rates across three rounds on Multi-IF before and after PAVIF training.