CONTEXTIF: ENHANCING INSTRUCTION-FOLLOWING THROUGH CONTEXT REWARD

Anonymous authors
Paper under double-blind review

ABSTRACT

While supervised fine-tuning (SFT) and preference learning (PL) are widely used to enhance the instruction-following ability of large language models (LLMs), they often struggle to generalize to novel or complex instructions and may compromise the models' general capabilities. In-context learning (ICL) emerges as a promising alternative due to its strong generalization without modifying the model's parameters, but its effectiveness is constrained by the reliance on high-quality, manually curated demonstration pools. To overcome this limitation, we propose ContextIF, a reinforcement learning (RL) framework for automatic context generation. Guided by comprehensive context reward, ContextIF is optimized by Group Relative Policy Optimization (GRPO). It aims to generate precise constraint summaries and optimal context demonstrations tailored to given instructions, thereby improving the instruction-following performance of target LLMs. We evaluate ContextIF on multiple representative instruction-following benchmarks using popular open-source LLMs. Experimental results demonstrate that ContextIF achieves substantial performance gains over existing SFT and ICL methods, while also generalizing effectively to unseen constraint conditions. Moreover, ContextIF preserves the parameters and general capabilities of the target models, offering strong adaptability and scalability. The code is provided in the Supplementary Materials.

1 INTRODUCTION

Large Language Models (LLMs) have achieved remarkable performance across diverse domains in natural language processing (NLP) (Touvron et al., 2023; OpenAI, 2023; GLM et al., 2024). As these models are increasingly deployed to develop agents across diverse domains, effective instruction following has become a critical factor for their practical application. Agents must comply with various constraints and instructions to ensure safe, trustworthy, and reliable interactions (Li et al., 2024; Tu et al., 2024). However, existing LLMs often struggle to adhere to the complex, multi-faceted constraints common in real-world instructions, limiting their effectiveness (Zhou et al., 2023; Sun et al., 2024; Qin et al., 2024b; Xia et al., 2024). Although numerous methods have been proposed to improve instruction-following capabilities, most studies focus on pre-training, Supervised Fine-Tuning (SFT), preference learning (PL) and reinforcement learning from human feedback (RLHF). These approaches require substantial high-quality data and computational resources and are susceptible to catastrophic forgetting, which leads to the loss of previously acquired knowledge and generalization ability (Lin et al., 2023). Moreover, they face considerable challenges in generalizing to unseen constrained tasks.

In-Context Learning (ICL) is a capability that enables LLMs to learn directly from demonstrations provided within the input prompt (Brown et al., 2020). By leveraging just a few task-specific demonstrations, LLMs can achieve strong performance across diverse tasks and rapidly adapt to new domains or problems without extensive fine-tuning. This capability is particularly crucial for enhancing instruction-following in scenarios involving numerous, complex, and multi-constraint requirements. Existing research has explored the importance of ICL in instruction-following tasks and investigated the impact of context (Zeng et al., 2025; Zhao et al., 2024; Li et al., 2025). Their findings indicate that ICL performance is highly contingent upon the quality of the provided examples, as different demonstrations can significantly affect the model's final adherence, underscoring the critical importance of context optimization for this task.

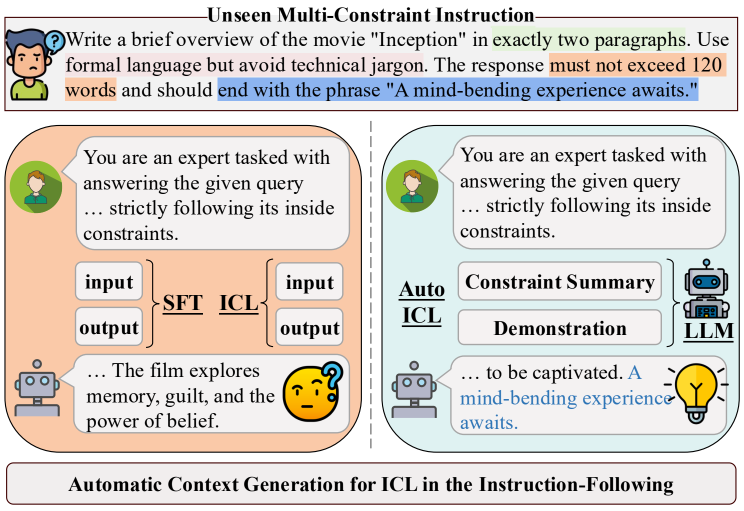
Unseen Multi-Constraint Instruction
Write a brief overview of the movie "Inception" in exactly two paragraphs. Use formal language but avoid technical jargon. The response must not exceed 120 words and should end with the phrase "A mind-bending experience awaits."
You are an expert tasked with answering the given query … strictly following its inside constraints.
input
output
SFT
ICL
input
output
… The film explores memory, guilt, and the power of belief.
You are an expert tasked with answering the given query … strictly following its inside constraints.
Auto ICL
Constraint Summary
Demonstration
LLM
… to be captivated. A mind-bending experience awaits.
Automatic Context Generation for ICL in the Instruction-Following

Figure 1: Comparison of conventional SFT/ICL methods (Left) with our proposed ContextIF (Right). Traditional SFT and ICL rely on extensive, high-quality human-annotated datasets, struggling to generalize to unseen constraints. In contrast, ContextIF enhances instruction-following performance by automatically generating high-quality constraint summaries and demonstrations.

While promising, the efficacy of ICL is fundamentally tethered to the quality and relevance of the provided demonstrations. Prior research has predominantly approached this challenge through two main paradigms: manual curation and retrieval-based selection (Wang et al., 2024). Manual curation, while capable of producing high-quality examples, is notoriously expensive, time-consuming, and fails to scale to the vast diversity of real-world instructions. Retrieval-based methods, which select examples from a pre-existing pool, offer better scalability but are fundamentally constrained by the coverage and quality of their source pool. This dependency becomes a critical bottleneck for complex instruction-following tasks, as illustrated in Figure 1. User instructions often contain nuanced, compositional, or entirely novel constraints for which suitable demonstrations simply do not exist in public datasets or general-purpose annotations. Consequently, retrieving context from static datasets is often insufficient for enhancing instruction-following capabilities across diverse domains. To address this limitation, existing studies have proposed that automatically generating specific context via LLMs is key to unlocking the full potential of ICL (Chen et al., 2023; Lee et al., 2025). However, self-generation without quality verification is often unreliable. The resulting demonstrations often lack the necessary structural rigor and semantic alignment to be effective, and can even degrade performance. Our work is motivated by the need for a principled framework to reliably generate optimal, task-specific context on the fly.

In this paper, we propose ContextIF, a novel framework that empowers LLMs to dynamically generate its own optimal context for instruction-following. Instead of retrieving from static pools, ContextIF employs a generator model to create high-quality, task-specific demonstrations on the fly for any given user query. The core of our framework is a reinforcement learning loop designed to train this generator model. The generation process involves two key steps: first, deconstructing the user query into a concise constraint summary, and second, constructing a parallel question-answer demonstration that exemplifies these constraints. To guide this complex generation task, we introduce a comprehensive context reward as the key innovation of our framework. This composite reward signal is engineered to assess both the structural correctness and the semantic quality of the generated context. The entire RL process is stabilized and optimized using Group Relative Policy Optimization (GRPO), enabling the generator model to learn how to produce context that maximally enhances instruction-following performance.

We conduct extensive experiments on two leading open-source models: LLaMA3-8B-Instruct and Mistral-7B-Instruct. To evaluate instruction-following, we utilize a suite of four representative benchmarks: IFEval, Multi-IF, FollowBench, and LiveBench. Our findings demonstrate that backbone LLMs equipped with ContextIF consistently and significantly outperform strong baselines. These baselines include traditional SFT methods and various ICL strategies, even those that leverage demonstrations from manually curated pools or are generated by significantly larger models like GPT-4o. Furthermore, our results show two critical advantages of our RL-based approach. First, ContextIF excels at generalizing to unseen constraint types. Second, it preserves and even enhances the model's foundational capabilities, directly addressing the common challenge of catastrophic forgetting associated with fine-tuning methods.

Our main contributions are summarized as follows:

• We introduce ContextIF, a novel RL framework that empowers LLMs to dynamically generate its own optimal context for improving instruction-following, overcoming the static data limitations of traditional fine-tuning and ICL.

• We design a multi-faceted Context Reward that precisely quantifies the structural and semantic quality of generated demonstrations, providing the essential guidance signal for our RL framework.

• Extensive experiments show that ContextIF sets a new state-of-the-art for instruction-following among open-source models. Furthermore, our approach demonstrates superior generalization to unseen tasks and simultaneously enhances the model's general capabilities.

2 Related Work

We review two lines of related work: instruction following methods and In-Context augmentation.

2.1 Instruction following

Instruction following is a fundamental capability of LLMs, requiring them to understand and generate responses that satisfy complex human instructions (Li et al., 2023b; Dong et al., 2024). Recent research has primarily focused on constraints within instructions, such as keywords and length (Zhou et al., 2023). Numerous evaluation benchmarks have been developed to assess instruction-following ability under complex, multi-constraint contexts, including those based on synthetic instructions and rule-based evaluations (Zhou et al., 2023; Yao et al., 2023; Iso, 2022), as well as evaluations utilizing complex data and large language models (Jiang et al., 2024; Qin et al., 2024a; Wen et al., 2024). Based on these benchmarks, various methods have been proposed to enhance instruction adherence, mainly focusing on (1) collecting supervised fine-tuning data (Sun et al., 2024; Dong et al., 2024; Ren et al., 2025), including distillation from larger models, back-translation to generate new instruction-response pairs, and iterative response analysis for improvement (An et al., 2025); (2) gathering preference data, such as code verification (Dong et al., 2024) and model validation (An et al., 2025; Cheng et al., 2025); and (3) reinforcement learning with reward verification (Peng et al., 2025; Lambert et al., 2025). However, conventional approaches to improving instruction-following have centered on data collection and targeted training. This methodology is prone to several critical issues, including catastrophic forgetting of prior knowledge and a degradation of generalist abilities. Moreover, such models typically exhibit poor generalization when faced with novel constraints.

2.2 In-Context Augmentation

The ICL paradigm allows LLMs to perform new tasks without modifying their parameters by leveraging task-specific input-output pairs, known as demonstrations (He et al., 2025; Moeini et al., 2025). It is usually more effective than fine-tuning, allowing the model to adapt to new cases with fewer data requirements in the instruction-following (Zhao et al., 2024; Zhang et al., 2025). Previous approaches to improving ICL performance have primarily focused on optimizing the selection and ordering of a limited number of demonstrations (Wang et al., 2024; Dherin et al., 2025). Given the cost of manually crafting prompts, recent research has moved beyond selecting and ranking high-quality examples, and interest in LLM-generated prompt strategies has grown (Chen et al., 2023). However, previous ICL methods based on selection sorting heavily rely on large-scale demonstration pools for matching (Li et al., 2023a; 2025), and LLM-based methods generate contexts with low matching accuracy. Building on this, our work explores how to leverage reinforcement learning to provide rewards for context, thereby enhancing the ability of LLMs to generate contexts. Compared to previous methods, our approach offers a more precise example generation method and demonstrates strong generalization capabilities on unseen tasks under the guidance of reinforcement learning rewards.

3 METHODOLOGY

3.1 OVERVIEW OF CONTEXTIF

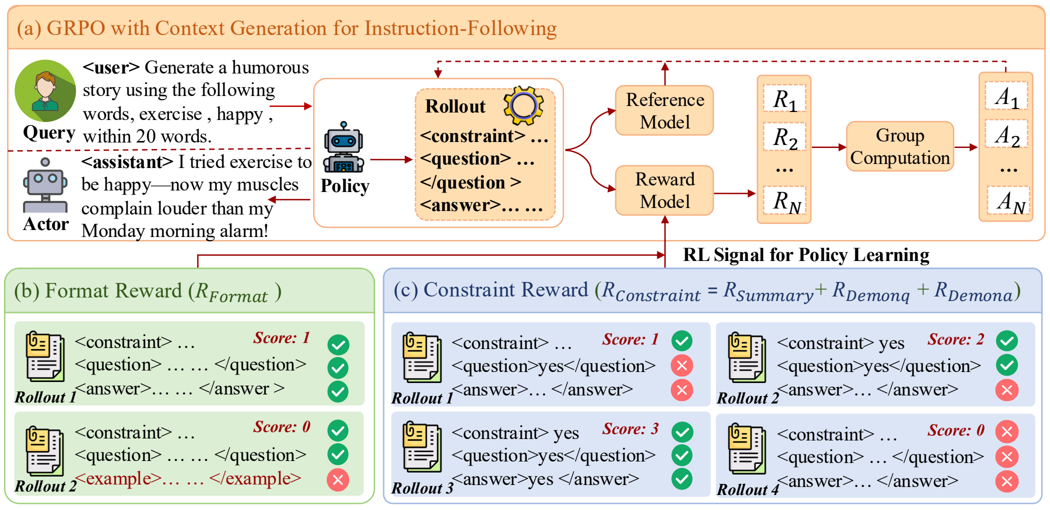
(a) GRPO with Context Generation for Instruction-Following
<user> Generate a humorous story using the following words, exercise , happy , within 20 words.
Query
<assistant> I tried exercise to be happy—now my muscles complain louder than my Monday morning alarm!
Actor
Policy
Rollout
<constraint> …
<question> …
</question >
<answer>… …
Reference Model
Reward Model
R_1 R_2 … R_N
Group Computation
A_1 A_2 … A_N
RL Signal for Policy Learning
(b) Format Reward (R_{Format})
Rollout 1 <constraint> … <question> … … </question> <answer>… … </answer > Score: 1
Rollout 2 <constraint> … <question> … … </question> <example>… … </example> Score: 0
(c) Constraint Reward ($R_{Constraint} = R_{Summary} + R_{Demonq} + R_{Demona}$)
Rollout 1 <constraint> … <question>yes</question> <answer>… </answer> Score: 1
Rollout 2 <constraint> yes <question>yes</question> <answer>… </answer> Score: 2
Rollout 3 <constraint> yes <question>yes</question> <answer>yes </answer> Score: 3
Rollout 4 <constraint> … <question> … </question> <answer>… </answer> Score: 0

Figure 2: An overview of the ContextIF framework. (a) The policy model, trained with GRPO, generates a constraint and demonstration context block based on a user query. This output is then evaluated by reward model to compute the final RL signal. (b) The Format Reward provides a binary signal for structural correctness. (c) The Constraint Reward provides a fine-grained score based on the semantic quality of the summary and the demonstration, guiding the policy toward generating task-optimal context for instruction-following.

We introduce ContextIF, a reinforcement learning framework designed to automate and optimize context generation for instruction-following tasks, thereby overcoming the limitations of both manually curated demonstration pools and context naively generated by LLMs themselves. As illustrated in in Figure 2, ContextIF includes a policy model, initialized from the same base LLM as the actor model, to act as a context generator. Given a user query, the model's task is to produce a self-contained ICL demonstration, which includes a precise constraint summary and a corresponding question-answer pair. This entire process is optimized using a context reward signal, which guides the model to generate context that is both structurally sound and semantically effective for the instruction-following task.

3.2 ICL ROLLOUT

The primary objective of the rollout phase in ContextIF is to autonomously generate a high-quality, task-specific context intended for ICL. This process begins with the policy model, as shown in Figure 2(a), which receives a user's query. Guided by a comprehensive system prompt, the policy's task is not to directly answer the query, but rather to perform a creative and analytical generation step.

Specifically, the policy executes a single, non-interactive rollout to produce a complete, self-contained XML block. This generated block is structured to serve as a perfect ICL context, containing three distinct and ordered tags: <constraint>, <question>, and <answer>. The <constraint>tag represents the model's analytical understanding of the initial query. The <question>and <answer>tags together form a new, parallel instruction-following pair that exemplifies the identified constraints.

This generated context block itself constitutes the entire trajectory for reinforcement learning process. Unlike traditional ICL where demonstrations are manually selected from a static pool, ContextIF dynamically crafts a new demonstration for each query. As depicted in Figure 2(a), this dynamically generated context is then evaluated by reward Model. The resulting reward signal, indicating the quality of the generated ICL demonstration, is then used to update the policy via group computation. This entire loop trains the policy to become an expert ICL demonstrator, capable of generating optimal context to enhance instruction-following. The specific prompts used for model inference and for the judge model are detailed in Appendix D.1.

3.3 Reward Design

Verifiable reward reinforcement learning has demonstrated strong empirical performance, becoming a key technique for enhancing LLMs and gaining widespread adoption. In our training of ContextIF, we similarly employ a composite reward that integrates a format-based verification reward and a constraint verification reward. Specifically, the format reward evaluates whether the model's output strictly conforms to the expected context XML structure, while the constraint reward assesses the quality of the constraint summary and the generated demonstration. Formally, the overall reward $\mathcal{R}_{\text{context}}$ is decomposed into two components: $\mathcal{R}_{\text{format}}$ and $\mathcal{R}_{\text{constraint}}$, which are detailed as follows.

Format Reward. Format reward $R_{\text{format}} \in \{0, 1\}$ Checks whether the model output contains the required constraint summary, demonstration question, and demonstration answer in the correct order specified by the ground truth:

$$\mathcal{R}_{\text{format}} = \begin{cases} 1, & \text{if all required fields appear and are in the correct order} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

Constraint Reward. $R_{\text{constraint}} \in [0, 3]$, evaluates the semantic quality of the generated content using a powerful judge model LLaMA3-70B-Instruct. This reward is an aggregate score composed of three distinct binary components. For each generated C-Q-A block, the judge model assesses three criteria: First, the *Constraint Summary* (r_{summary}) is awarded a score of 1 if the <constraint>tag accurately summarizes all constraints from the original query, and 0 otherwise. Second, the *Demonstration Question* (r_{demoq}) receives a score of 1 if the generated <question>contains a parallel constraint structure to the query, and 0 otherwise. Finally, the *Demonstration Answer* (r_{demoa}) is given a score of 1 if the generated <answer>faithfully follows all constraints within the generated question, and 0 otherwise. Total match score for each match is:

$$\mathcal{R}_{\text{constraint}} = r_{\text{summary}} + r_{\text{demoq}} + r_{\text{demoa}} \tag{2}$$

The final reward value R_{context} is finally derived as the sum of R_{format} and $R_{\text{constraint}}$:

$$R_{\text{context}} = R_{\text{format}} + R_{\text{constraint}} \tag{3}$$

In summary, our reward design explicitly distinguishes between structural adherence and semantic quality. By unifying the reward signal to encompass both strict XML format compliance and the fine-grained quality of the generated constraint summary and demonstration, our model is guided to produce outputs that are not only syntactically valid but also semantically optimal for the instruction-following task. This holistic evaluation is crucial for training an effective context generator and achieving success in downstream instruction-following scenarios.

3.4 RL Training with GRPO

To effectively train our context generation model using the composite context reward, we employ GRPO, a variant of Proximal Policy Optimization (PPO) particularly well-suited for LLM finetuning. GRPO stabilizes the training process by normalizing advantage values within groups of samples generated from the same initial query (Shao et al., 2024). Let $\pi\theta$ represent the current policy.

Normalized Advantage Across Query Groups. For each query Q, its responses derived from the rollout form a group G_Q consisting of multiple responses and their corresponding reward values:

$$G_Q = \{A, (s_1, r_1), (s_2, r_2), \ldots, (s_n, r_n)\} \tag{4}$$

where A denotes the verification for Q, and each reward r_i is computed as the sum of the format and constraint rewards associated with response s_i, i.e., $r_i = \mathcal{R}_{\text{format}}(s_i, A) + \mathcal{R}_{\text{constraint}}(s_i, A)$. For each group, we calculate the mean and standard deviation of the rewards:

$$\mu_Q = \frac{1}{n} \sum_{i=1}^{n} r_i \tag{5}$$

$$\sigma_Q = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(r_i - \mu_Q)^2} \tag{6}$$

Then, for each sample s_i in the group, we define the normalized advantage:

$$A_i(s_i|Q) = \frac{r_i - \mu_Q}{\sigma_Q + \eta} \tag{7}$$

where η is a constant to avoid division by zero.

4 Experiment

4.1 Baselines

In our experiments, we compare ContextIF with a series of strong baselines specifically optimized for instruction following, including Conifer (Sun et al., 2024), SPAR (Cheng et al., 2025), AutoIF (Dong et al., 2024) and UltraIF (An et al., 2025), which employs SFT and DPO. Additionally, we evaluated various industrial models, including GPT-4o (Hurst et al., 2024), QwQ-32B (Yang et al., 2024). A detailed description of each baseline is provided in Appendix B.

4.2 Backbone LLMs

LLaMA3 Series (Dubey et al., 2024) represent the state-of-the-art for open-source models of their size, a result of training on a massive 15T token dataset and an enhanced post-training alignment process, which together endow them with exceptional instruction-following capabilities.

Mistral-7B-Instruct (Jiang et al., 2023) is a highly influential model renowned for delivering remarkable performance and efficiency at the 7B parameter scale. Its effectiveness is attributed to architectural innovations like Grouped-Query Attention and Sliding Window Attention, making it a widely adopted and resource-efficient baseline for research.

4.3 Evaluation benchmarks

We use four representative instruction-following benchmarks to evaluate our method: IFEval (Zhou et al., 2023), the most commonly used dataset; Multi-IF (He et al., 2024), which includes multi-turn and multilingual instruction following; FollowBench (Jiang et al., 2024) and LiveBench (White et al., 2024) , which cover a comprehensive range of constraint types. For common capabilities, we employ a variety of specialized benchmarks: GSM8K (Cobbe et al., 2021a) and BBH (Suzgun et al., 2022) , MMLU (Hendrycks et al., 2020), and HumanEval (Chen et al., 2021b). A detailed description of each benchmark, along with specific evaluation protocols, is provided in Appendix C.

4.4 Main Results

All experimental results on four instruction-following benchmarks are presented in Table 1. Our analysis reveals that ContextIF not only significantly enhances the performance of its base models but also establishes a new state-of-the-art among open-source models of a similar scale. We draw the following key conclusions:

ContextIF Consistently and Substantially Improves Base Model Performance. The primary finding is the remarkable performance uplift provided by our framework. On the LLaMA3-8B-Instruct base, ContextIF-8B achieves significant improvements across all evaluated benchmarks. For instance, the average score across all four IFEval metrics increases from 77.11 to 83.35. This consistent and significant improvement validates the effectiveness and robustness of our reinforcement learning-based context generation approach.

ContextIF-8B Achieves State-of-the-Art Performance Among 8B Models. When compared with other specialized instruction-following methods fine-tuned from LLaMA3-8B-Instruct, ContextIF-8B demonstrates a clear advantage. It surpasses models such as Conifer-8B, SPAR-8B, and UltraIF-8B across all four metrics on IFEval. This superiority also extends to other benchmarks, including

Model	IFEval					MultiIF			FollowBench	LiveBench
	P(L)	I(L)	P(S)	I(S)	Avg.	Turn1	Turn2	Turn3	SSR	Score
GPT-4o	84.80	89.60	79.90	85.60	84.98	82.30	71.70	59.30	75.30	64.90
QwQ-32B	86.10	90.40	82.80	88.00	86.83	64.20	56.60	48.40	67.80	80.00
TULU 3	82.80	87.50	79.70	85.10	83.78	82.10	63.20	51.20	70.30	72.00
LLaMA3-70B-Instruct	84.04	89.21	77.76	84.53	83.89	63.83	52.24	43.92	62.90	67.50
LLaMA3-8B Models										
LLaMA3-8B-Instruct	77.02	84.05	69.44	77.94	77.11	63.83	52.24	43.92	62.90	46.70
Conifer-8B	79.50	85.50	75.60	82.70	80.83	66.00	53.80	41.90	64.76	46.90
UltraIF-8B	75.40	83.10	71.30	79.40	77.30	69.63	58.28	46.86	60.41	45.40
AutoIF-8B	76.93	82.02	68.13	77.55	76.16	62.63	51.53	42.53	59.50	45.70
SPAR-8B	81.15	87.05	79.11	85.13	83.11	72.55	60.46	51.32	68.80	49.80
ContextIF-8B	83.54	88.72	77.07	84.05	**83.35**	**74.32**	**62.58**	**53.51**	**69.37**	**59.90**

Table 1: Evaluation results of different models on IFEval, MultiIF, FollowBench(SSR), and LiveBench datasets. Pr. and Ins. stand for prompt and instruction levels, respectively. S and L represent strict and loose metrics for IFEval. For LiveBench, we only report the performance on the subset of instruction-following data.

the highly challenging multi-turn dialogue of MultiIF, where ContextIF-8B achieves the highest Turn3 score 53.51 among all LLaMA3-8B-Instruct variants. This suggests that our strategy of optimizing for task-optimal context is more effective than the SFT and DPO strategies employed by other methods.

ContextIF Competes with and Surpasses Larger and Proprietary Models. The most compelling result is how ContextIF closes the gap with much larger and more powerful models. On the IFEval metrics, the performance of ContextIF-8B is on par with significantly larger models like TULU 3 and LLaMA3-70B-Instruct, and remarkably close to the proprietary giant, GPT-4o. More strikingly, on the LiveBench benchmark, which specifically measures generalization to real-world instructions, ContextIF-8B achieves a score of 59.90. While this score does not match that of the larger-parameter QwQ-32B, it significantly surpasses its own base model 46.70 and outperforms other same-scale instruction-following models like SPAR-8B 49.80. This provides strong evidence that the tailored, high-quality contexts generated by our framework provide a superior signal for generalization, enabling smaller open-source models to match the performance of vastly larger counterparts on critical instruction-following tasks.

In summary, the results unequivocally validate our approach. By training a model to generate optimized context via our multi-faceted reward signal, ContextIF dramatically elevates the performance of its base models. It not only establishes leadership among models of a similar scale but also demonstrates a remarkable ability to compete with and, in the key aspect of real-world generalization, surpass proprietary and much larger models. This makes ContextIF a highly effective and parameter-efficient solution for advancing the state-of-the-art in instruction following. We demonstrate the robustness of our approach with consistent improvements on Mistral-7B-Instruct, with detailed results provided in Section E.2.

4.5 Comparison with ICL Strategies

To validate the effectiveness of our proposed ContextIF framework, we conducted a comprehensive comparison of various context generation strategies, with results presented in Table 2. We first observe that while a well-prompted zero-shot baseline can slightly improve the instruction-following capabilities of the LLaMA3-8B-Instruct model, employing randomly selected in-context demonstrations consistently degrades performance across all benchmarks. Our analysis suggests that randomly chosen contexts often introduce significant distracting information, negatively impacting the model's ability to follow instructions and underscoring that effective context is paramount. The viability of automatic high-quality context generation is established when using the LLM itself to generate context, an effect that is further amplified when employing a much larger model like GPT-4o. However, when we fine-tuned a dedicated LLaMA3-8B-Instruct model on 400 curated examples to specialize in context generation, its performance paradoxically fell below that of the base LLM context genera-

Model	IFEval					MultiIF			FollowBench	LiveBench
	P(L)	I(L)	P(S)	I(S)	Avg.	Turn1	Turn2	Turn3	SSR	Score
LLaMA3-8B-Instruct	77.02	84.05	69.44	77.94	77.11	63.83	52.24	43.92	62.90	46.70
+ zeroshot	75.42	82.73	72.58	80.14	77.72	68.37	55.82	46.25	63.12	48.40
+ select-context	74.31	82.37	71.35	79.86	76.97	61.23	51.14	42.21	60.43	43.70
+ LLM-context	78.87	85.17	76.52	83.33	80.97	70.82	57.51	49.63	65.25	50.50
+ tuneLLM-context	78.27	84.49	75.97	82.61	80.34	69.85	57.26	49.43	64.37	50.30
+ GPT4o-context	82.44	88.13	76.71	83.81	82.77	73.75	61.94	52.13	67.38	57.40
ContextIF-8B	**83.54**	**88.72**	**77.07**	**84.05**	**83.35**	**74.32**	**62.58**	**53.51**	**69.37**	**59.90**

Table 2: Performance comparison of ContextIF against various ICL strategies on the LLaMA3-8B-Instruct model.

tor. A closer analysis reveals overfitting on specific constraint types and a lower quality of generated demonstrations, highlighting the inherent limitations of a purely supervised approach for this task.

In contrast, ContextIF-8B establishes a new state-of-the-art across all benchmarks. By leveraging a policy trained directly with our context reward signal, ContextIF-8B surpasses all other ICL strategies, including those powered by the significantly larger GPT-4o. For example, it achieves the highest scores across all four metrics on IFEval, outperforms all methods on the challenging multi-turn MultiIF benchmark with a score of 53.51 on Turn3, and sets the top score on both FollowBench at 69.37 and LiveBench at 59.90. These results unequivocally demonstrate that our reinforcement learning approach moves beyond mere context generation; it discovers a task-optimal context strategy. This learned strategy proves superior even to that of larger, general-purpose models, establishing a new and more effective paradigm for enhancing instruction-following.

4.6 Analysis on Unseen Constraints

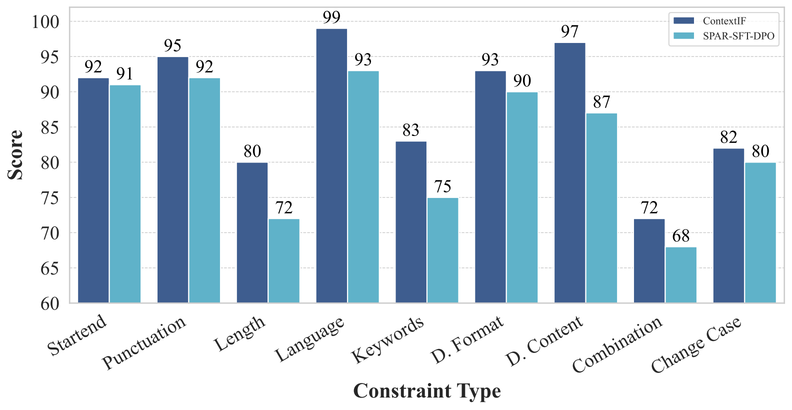

Figure 3: Prompt-level strict scores across different types of constraints on IFEval.

To analyze the generalization capability of ContextIF, we investigate its performance on constraint types that were absent or rare in our training data. Our ContextIF training set primarily consists of three common constraint types: content-based, style/tone, and format. To validate the model's generalization capabilities, we also excluded training data corresponding to the "Length" and "Keywords" constraint types for the SPAR-SFT-DPO baseline. We created specialized test subsets from the IFEval benchmark focusing on constraint types not covered in our training, such as length, Keywords, and language. The results, illustrated in Figure 3, demonstrate ContextIF's robust generalization capabilities. Across a wide range of these unseen constraint types, ContextIF consistently outperforms the baseline. For instance, on "Language", it achieves a near-perfect score of 99. On "Keywords" and "Length", it leads by 8 points respectively, even though neither model was explicitly trained on these specific constraints. This indicates that through its RL process of deconstructing and reconstructing instructions, ContextIF acquires a more fundamental and transferable understanding of instruction-following, rather than merely memorizing patterns. As expected, this strong performance

Method	Knowledge	Reasoning	Math	Coding	Average
	MMLU	**BBH**	**GSM8K**	**HumanEval**	**[All]**
LLaMA3-8B-Instruct	68.40	66.50	79.60	62.20	69.18
SPAR-SFT	67.80 (-0.6)	66.10 (-0.4)	78.60 (-1.0)	62.00 (-0.2)	68.63 (-0.55)
ContextIF-8B	70.10 (+1.7)	68.10 (+1.6)	80.40 (+0.8)	63.30 (+1.1)	70.48 (+1.3)

Table 3: Performance comparison on general capability benchmarks. We report 5-shot accuracy on MMLU, 3-shot accuracy on BBH, and Pass@1 on GSM8K and HumanEval. The numbers in parentheses indicate the performance change relative to the base model. Our ContextIF model not only preserves but enhances general capabilities, while the SFT baseline shows degradation.

is also evident on constraint types directly related to our RL training categories, such as "D. Content" and "D. Format". Notably, on the "D. Content" constraint, which aligns with our "content-based" training, ContextIF achieves a score of 97, surpassing the baseline by 10 points. This demonstrates the precision of the guidance provided by our Context Reward when the task is within the scope of its training. This dual finding—strong performance on both a wide array of unseen constraints and on seen constraint types—suggests that while incorporating richer constraint types into the training data could further boost performance, the ContextIF framework itself fosters a robust, generalizable instruction-following ability. We encourage the community to further explore how dynamically generated context can serve as a powerful tool for enhancing generalization in LLMs.

4.7 Analysis on General Capabilities

A crucial aspect of our investigation is to determine whether enhancing instruction-following capabilities via ContextIF compromises the model's foundational general abilities, a common drawback of SFT. To verify this, we evaluate our models across four diverse and challenging domains: general knowledge (MMLU), reasoning (BBH), mathematical problem solving (GSM8K), and coding (HumanEval). Table 3 presents the performance of our ContextIF-trained model in comparison to the original LLaMA3-8B-Instruct and the SFT baseline (SPAR). As hypothesized, the SPAR model exhibits a consistent performance degradation across all four general benchmarks, underscoring the risk of catastrophic forgetting associated with standard fine-tuning. The most significant drop is observed in GSM8K, with a decline of 1.0 point. In stark contrast, the ContextIF model not only preserves but demonstrates performance gains across every evaluated domain. Notably, it achieves its most significant improvements in knowledge and reasoning, with a 1.7-point increase on MMLU and a 1.6-point gain on BBH over the base model. These results strongly suggest that the underlying process of our RL framework—learning to deconstruct instructions, summarize constraints, and generate logically consistent examples—does not merely teach a narrow skill. Instead, it appears to refine the model's meta-learning and reasoning pathways. We believe that empowering a model to generate its own context for learning acts as a form of self-distillation, which reinforces its core capabilities, facilitating the development of a more general and versatile agent. To assess the necessity of each component in our reward design, we conduct a detailed ablation study in Section E.1.

5 Conclusion

In this study, we introduce ContextIF, a novel RL framework that enhances the instruction-following capabilities of LLMs by dynamically generating high-quality context via Context Reward. We reveal that, unlike traditional SFT and ICL methods that depend on complex and labor-intensive data curation, guiding an LLM to generate key constraints and high-quality parallel demonstrations as context through multi-faceted reward mechanism leads to significant improvements on instruction-following tasks. Critically, extensive experiments demonstrate that ContextIF not only excels in generalizing to unseen constraints but also bypasses the risk of catastrophic forgetting typically associated with SFT, preserving and even enhancing the model's general capabilities. This work showcases the immense potential of combining In-Context Learning with Reinforcement Learning for instruction-following, and we believe empowering models to craft their own context represents a scalable and effective path toward more generalizable and aligned language agents.

Ethical Statement

The research conducted in this paper adheres to the ICLR Code of Ethics. Our work primarily focuses on enhancing the instruction-following capabilities of existing open-source Large Language Models. The datasets used for training and evaluation in our ContextIF framework were derived from publicly available instruction-following benchmarks, which consist of general-purpose, non-personal queries. We have made efforts to filter any potentially harmful or personally identifiable information during our data processing stage.

Reproducibility Statement

We are committed to ensuring the reproducibility of our research. All datasets used for training and evaluation in this work, including IFEval, MultiIF, FollowBench, and LiveBench, are publicly available benchmarks. The implementation details of our ContextIF framework, including the reward functions and the GRPO training algorithm, are described in Section Methodology. Our ContextIF framework is implemented on top of the publicly available veRL reinforcement learning library. The base models used in our experiments, including LLaMA3-8B-Instruct, are open-source and can be accessed via the Hugging Face Hub.

References

Kaikai An, Li Sheng, Ganqu Cui, Shuzheng Si, Ning Ding, Yu Cheng, and Baobao Chang. Ultraif: Advancing instruction following from the wild. *arXiv preprint arXiv:2502.04153*, 2025.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

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, 2021a. URL https://arxiv.org/abs/2107.03374.

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*, 2021b.

Wei-Lin Chen, Cheng-Kuang Wu, Yun-Nung Chen, and Hsin-Hsi Chen. Self-icl: Zero-shot in-context learning with self-generated demonstrations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 15651–15662, 2023.

Jiale Cheng, Xiao Liu, Cunxiang Wang, Xiaotao Gu, Yida Lu, Dan Zhang, Yuxiao Dong, Jie Tang, Hongning Wang, and Minlie Huang. Spar: Self-play with tree-search refinement to improve instruction-following in large language models, 2025. URL https://arxiv.org/abs/2412.11605.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021a. URL https://arxiv.org/abs/2110.14168.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021b. URL https://arxiv.org/abs/2110.14168.

Benoit Dherin, Michael Munn, Hanna Mazzawi, Michael Wunder, and Javier Gonzalvo. Learning without training: The implicit dynamics of in-context learning. *arXiv preprint arXiv:2507.16003*, 2025.

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

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

Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Dan Zhang, Diego Rojas, Guanyu Feng, Hanlin Zhao, et al. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *arXiv preprint arXiv:2406.12793*, 2024.

Pengfei He, Yingqian Cui, Han Xu, Hui Liu, Makoto Yamada, Jiliang Tang, and Yue Xing. Towards the effect of examples on in-context learning: A theoretical case study. *Stat*, 14(1):e70045, 2025.

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

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

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

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.

Hayate Iso. Autotemplate: A simple recipe for lexically constrained text generation. *arXiv preprint arXiv:2211.08387*, 2022.

Dongsheng Jiang, Yuchen Liu, Songlin Liu, Jin'e Zhao, Hao Zhang, Zhen Gao, Xiaopeng Zhang, Jin Li, and Hongkai Xiong. From clip to dino: Visual encoders shout in multi-modal large language models. *arXiv preprint arXiv:2310.08825*, 2023.

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.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training, 2025. URL https://arxiv.org/abs/2411.15124.

Dohyun Lee, Seungil Chad Lee, Chanwoo Yang, Yujin Baek, and Jaegul Choo. Exploring in-context example generation for machine translation. *arXiv preprint arXiv:2506.00507*, 2025.

Rui Li, Guoyin Wang, and Jiwei Li. Are human-generated demonstrations necessary for in-context learning? *arXiv preprint arXiv:2309.14681*, 2023a.

Xian Li, Ping Yu, Chunting Zhou, Timo Schick, Omer Levy, Luke Zettlemoyer, Jason Weston, and Mike Lewis. Self-alignment with instruction backtranslation. *arXiv preprint arXiv:2308.06259*, 2023b.

Xiaomin Li, Zhou Yu, Zhiwei Zhang, Xupeng Chen, Ziji Zhang, Yingying Zhuang, Narayanan Sadagopan, and Anurag Beniwal. When thinking fails: The pitfalls of reasoning for instruction-following in llms. *arXiv preprint arXiv:2505.11423*, 2025.

Yuanchun Li, Hao Wen, Weijun Wang, Xiangyu Li, Yizhen Yuan, Guohong Liu, Jiacheng Liu, Wenxing Xu, Xiang Wang, Yi Sun, et al. Personal llm agents: Insights and survey about the capability, efficiency and security. *arXiv preprint arXiv:2401.05459*, 2024.

Bill Yuchen Lin, Abhilasha Ravichander, Ximing Lu, Nouha Dziri, Melanie Sclar, Khyathi Chandu, Chandra Bhagavatula, and Yejin Choi. The unlocking spell on base llms: Rethinking alignment via in-context learning, 2023. URL https://arxiv.org/abs/2312.01552.

Amir Moeini, Jiuqi Wang, Jacob Beck, Ethan Blaser, Shimon Whiteson, Rohan Chandra, and Shangtong Zhang. A survey of in-context reinforcement learning. *arXiv preprint arXiv:2502.07978*, 2025.

OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

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.

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

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 Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 13025–13048, Bangkok, Thailand, August 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.772. URL https://aclanthology.org/2024.findings-acl.772/.

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. *arXiv preprint arXiv:2501.04945*, 2025.

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.

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.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. Challenging big-bench tasks and whether chain-of-thought can solve them, 2022. URL https://arxiv.org/abs/2210.09261.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Tao Tu, Anil Palepu, Mike Schaekermann, Khaled Saab, Jan Freyberg, Ryutaro Tanno, Amy Wang, Brenna Li, Mohamed Amin, Nenad Tomasev, et al. Towards conversational diagnostic ai. *arXiv preprint arXiv:2401.05654*, 2024.

Xubin Wang, Jianfei Wu, Yichen Yuan, Deyu Cai, Mingzhe Li, and Weijia Jia. Demonstration selection for in-context learning via reinforcement learning. *arXiv preprint arXiv:2412.03966*, 2024.

Bosi Wen, Pei Ke, Xiaotao Gu, Lindong Wu, Hao Huang, Jinfeng Zhou, Wenchuang Li, Binxin Hu, Wendy Gao, Jiaxing Xu, et al. Benchmarking complex instruction-following with multiple constraints composition. *Advances in Neural Information Processing Systems*, 37:137610–137645, 2024.

Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Ben Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Siddartha Naidu, et al. Livebench: A challenging, contamination-free llm benchmark. *arXiv preprint arXiv:2406.19314*, 4, 2024.

Congying Xia, Chen Xing, Jiangshu Du, Xinyi Yang, Yihao Feng, Ran Xu, Wenpeng Yin, and Caiming Xiong. Fofo: A benchmark to evaluate llms' format-following capability. *arXiv preprint arXiv:2402.18667*, 2024.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Shunyu Yao, Howard Chen, Austin W Hanjie, Runzhe Yang, and Karthik Narasimhan. Collie: Systematic construction of constrained text generation tasks. *arXiv preprint arXiv:2307.08689*, 2023.

Jie Zeng, Qianyu He, Qingyu Ren, Jiaqing Liang, Yanghua Xiao, Weikang Zhou, Zeye Sun, and Fei Yu. Order matters: Investigate the position bias in multi-constraint instruction following, 2025. URL https://arxiv.org/abs/2502.17204.

Yedi Zhang, Aaditya K Singh, Peter E Latham, and Andrew Saxe. Training dynamics of in-context learning in linear attention. *arXiv preprint arXiv:2501.16265*, 2025.

Hao Zhao, Maksym Andriushchenko, Francesco Croce, and Nicolas Flammarion. Is in-context learning sufficient for instruction following in llms? *arXiv preprint arXiv:2405.19874*, 2024.

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 USE OF LLMS

In the preparation of this manuscript, LLMs were utilized as a general-purpose writing assistance tool. Specifically, we employed LLM-based services to aid in proofreading, grammar correction, and rephrasing of sentences to improve clarity and style. The core scientific contributions, including the research ideation, experimental design, data analysis, and the formulation of conclusions, were conceived and executed entirely by the human authors. The LLMs' role was strictly limited to that of a language polishing tool. All content, including any text modified with the assistance of an LLM, was carefully reviewed, edited, and validated by the authors, who take full responsibility for the final version of this paper.

B DESCRIPTION OF BASELINES

Conifer (Sun et al., 2024) method generates a 13k instruction-following dataset through a three-stage synthesis pipeline. This process begins with seed instructions sourced from ShareGPT and applies query reframing, constraint generation, and recombination to create the final data. We directly utilize the full set as the baseline.

AutoIF (Dong et al., 2024) framework is its methodology for evaluating model responses using custom Python functions. These functions automatically verify adherence to a wide variety of manually designed constraints. For this baseline, we replicate their methodology on our base model to obtain the performance results.

UltraIF (An et al., 2025) is a framework for creating large-scale instruction-following data. It trains a specialized model, the UltraComposer, on a corpus of existing documents and datasets. This composer model is then used to synthesize the final instruction data. We reproduce their data generation and training pipeline to establish the baseline performance for this method.

SPAR (Cheng et al., 2025) is a self-play framework designed to generate high-quality preference data for instruction-following tasks. It addresses the issue of irrelevant content variations in standard preference pairs by using a tree-search self-refinement process. In this approach, an LLM iteratively refines its own responses to minimize distractions, creating cleaner and more comparable preference pairs for training. For our baseline, we fine-tune our base model using their officially released SFT and DPO datasets.

C EVALUATION BENCHMARKS

IFEval (Zhou et al., 2023) provides a benchmark for assessing the instruction-following capabilities of LLMs through verifiable prompts. It is composed of approximately 500 prompts that cover 25 distinct types of machine-verifiable instructions. In our evaluation, we report performance using both loose and strict accuracy metrics at both the prompt and instruction levels, adhering to the benchmark's standard protocol.

Multi-IF (He et al., 2024) extends the evaluation of instruction-following to more complex multi-turn and multilingual scenarios. Building upon IFEval, the benchmark contains 4,501 conversational dialogues across various languages, each structured into three turns. For our experiments, we report the average accuracy for each of the three conversational turns.

FollowBench (Jiang et al., 2024) is a benchmark designed to evaluate adherence to multi-level, fine-grained constraints. It is organized around five distinct constraint categories (Content, Situation, Style, Format, and Example) and emphasizes a compositional instruction design. We adopt the official evaluation protocol, which employs GPT-4-0125-preview as an automated judge to assess whether model outputs satisfy each individual constraint.

LiveBench (White et al., 2024) is a comprehensive benchmark featuring a diverse set of challenging tasks, including a dedicated subset for instruction-following. A key characteristic of the benchmark is its automated scoring system, which evaluates model outputs against objective ground-truth values. In our work, we utilize this specific instruction-following subset to assess the alignment capabilities of the models.

GSM8K (Cobbe et al., 2021b) is a benchmark composed of 8,500 high-quality grade school math word problems. It is specifically designed to challenge the multi-step mathematical reasoning capabilities of language models. For our evaluation, we report the model's overall accuracy on this dataset. the experiment.

HumanEval (Chen et al., 2021a) is a code generation benchmark consisting of 164 handcrafted programming problems. Each problem includes a function signature, a docstring, and a set of unit tests (averaging 7.7 per problem) for evaluation. The benchmark assesses a model's ability to synthesize functional code from natural language descriptions, thereby testing a combination of language comprehension, reasoning, and algorithmic skills. We report the Pass@1 metric in our experiments.

BBH (Suzgun et al., 2022) is a challenging subset of the Big-Bench benchmark, curated to include 23 tasks that current language models find difficult. The benchmark contains a total of 6,511 examples and is designed to rigorously evaluate a model's multi-step reasoning and problem-solving abilities. We report the average accuracy across all tasks in our experiments. In the experiment we report the accuracy metrics on BBH.

MMLU (Hendrycks et al., 2021) serves as a key benchmark for assessing the breadth of a language model's world knowledge and problem-solving abilities. The benchmark is composed of approximately 15,908 multiple-choice questions that span 57 distinct tasks. These tasks cover a wide spectrum of subjects, ranging from humanities like U.S. history to professional fields such as law and medicine. In the experiments, we report the average accuracy across all tasks.

D Experimental Details

D.1 Prompts of ConetxtIF

For the Actor Model, we use the following template during inference to incorporate either a zero-shot or a one-shot context:

Prompt Template for Response Generation Zero-shot

You are an expert tasked with answering the given query. Please provide a clear and concise response directly, without introductory phrases such as "What a great question", "Here is the answer", or similar expressions.
Focus solely on addressing the query. Now please answer the given query while strictly following its inside constraints.
Query: {query}

Prompt Template for Response Generation One-shot

You are an expert tasked with answering the given query. Please provide a clear and concise response directly, without introductory phrases such as "What a great question", "Here is the answer", or similar expressions.
Focus solely on addressing the query. Now please answer the given query while strictly following its inside constraints.

Here is an example of the constraints contained therein and one that contains constraints of the same type.

Constraints: {constraint}
Example: {Example}

Here is the Query:
Query: {query}

For the Policy Model, we use the following prompt template to generate constraint summaries and corresponding demonstrations for given queries:

Prompt Template for Context Generation

You are an expert in identifying constraints in instructions and constructing high-quality, parallel examples. Your objective is to analyze a given user query, identify its constraints, and then generate a new, similar example. Your final output MUST be a single, valid XML block and nothing else.

The required XML format is: <constraint>Summary of constraints</constraint>
<question>Newly generated question with similar constraints</question>
<answer>Answer that follows the new question's constraints</answer>

Here are the examples of correct execution you must follow:

Example 1:
Input: Generate a five-sentence story using the following words, need, money, computer, bought, happy.
Output:
<constraint>Generate a five-sentence story using the specified words</constraint>
<question>Generate a five-sentence story using the following words, cat, garden, sunshine, play, joy</question>
<answer>The cat wandered into the garden on a sunny day. It began to play among the flowers with great joy. The warm sunshine made the garden feel magical. Birds chirped happily as the cat chased butterflies. At the end of the day, the cat rested peacefully, content and joyful.</answer>

Example 2:
Input: Write an extravagant session plan to learn about java. Make sure to include a postscript starting with P.P.S.
Output:
<constraint>Include a postscript starting with P.P.S in the session plan</constraint>
<question>Write a detailed study plan to learn Python. Make sure to include a postscript starting with P.P.S.</question>
<answer>Study Plan to Learn Python:\n\n1. Introduction to Python: Understand the basics of Python syntax and data types.\n2. Control Structures: Learn about loops, conditionals, and functions.\n3. Data Structures: Explore lists, dictionaries, sets, and tuples.\n4. Object-Oriented Programming: Understand classes, objects, and inheritance.\n5. Modules and Packages: Learn how to organize code and use libraries.\n6. Practical Projects: Build small projects like a calculator or a to-do app to apply knowledge.\n\nP.P.S. Remember to practice coding daily and explore online resources to deepen your understanding.</answer>

Example 3:
Input: List the pros and cons of using two different names for the same thing. Make sure the word synonyms appears at least 3 time.
Output:
<constraint>The word "synonyms" must appear at least 3 times in the response</constraint>
<question>List the advantages and disadvantages of using multiple terms for the same concept. Make sure the word synonyms appears at least 3 times.</question>
<answer>Using multiple terms as synonyms for the same concept can have both pros and cons. On the positive side, synonyms can enrich communication by providing variety and preventing repetition, making the text more engaging. Additionally, synonyms can help clarify meaning by offering alternative expressions that might resonate better with different audiences. However, the use of synonyms can also lead to confusion if the terms are not perfectly interchangeable or if the audience is unfamiliar with some of the synonyms used. Moreover, excessive reliance on synonyms might dilute the precision of the message. Overall, while synonyms enhance language flexibility, careful consideration is needed to maintain clarity.</answer>

Prompt Template for Context Generation

Example 4:
Input: Answer must be in lowercase letters only throughout the entire response.
Output:
<constraint>Answer must be in lowercase letters only throughout the entire response</constraint>
<question>What is another word for paintings? Answer in lowercase letters only, throughout your entire answer.</question>
<answer>artworks</answer>

You will now be given a user query. Your task is to process this query according to the rules and examples above.
CRITICAL INSTRUCTION: Do NOT use generic placeholder text like "Constraint description goes here", "Question description goes here", or any similar variations in your output. You must generate specific, new content derived from the user query.
Query: {query}
Now, based on the user query provided above, generate your response in the specified XML format.

D.2 Implementation Details

For our primary experiments, both the policy model and the actor model were initialized from the same base model. The policy model was then fine-tuned using the GRPO algorithm as described in Section 3.4. The actor model's weights were kept frozen during all stages of training and evaluation to serve as the target LLM for instruction-following. This decoupled setup ensures that our method enhances instruction-following capabilities without degrading the model's pre-existing general knowledge.

E Experiment Results

E.1 Ablation Studies

We conduct ablation studies to investigate how each component of our context reward contributes to the model's instruction-following capabilities and to assess their necessity. The results are presented in Table 4. In the "w/o Format" setting, the format reward is removed. In the "w/o Summary", "w/o Demoq", and "w/o Demoa" settings, the rewards for the constraint summary, demonstration question, and demonstration answer are respectively excluded. As shown in Table 4, the demonstration answer reward (w/o Demoa) and the constraint summary reward (w/o Summary) prove to be the most critical components for enhancing complex instruction-following. Removing the answer faithfulness reward results in the most severe performance drop across all benchmarks, with the IFEval average score plummeting from 83.75 to 78.50. The removal of the summary reward similarly leads to significant performance degradation.

Perhaps the most insightful finding is the relative impact of the Format Reward. While its removal (w/o Format) degrades performance, the drop is less severe than when the summary or faithfulness signals are omitted. This highlights that while a consistent XML structure is beneficial, the core challenge in instruction-following lies in correctly interpreting and executing the instruction's semantic intent. Our base model, already adept at structured outputs, gains the most from the rewards that specifically target this deeper understanding. The Constraint Summary Reward teaches the model to first accurately deconstruct an instruction into its core constraints. Subsequently, the Answer Faithfulness and Question Relevance rewards guide the model to faithfully reconstruct these constraints into a new, logically coherent demonstration. It is this learned process of deconstruction and reconstruction, rather than mere structural mimicry enforced by the Format Reward alone, that proves essential for true instruction-following. The synergy of all these reward components is therefore critical, validating our multi-faceted approach to enhancing this core capability.

Model	IFEval	MultiIF	FollowBench	LiveBench
	Avg.	Turn3	SSR	Score
ContextIF-8B	83.75	53.51	69.37	59.90
w/o Format	81.13	51.21	67.48	56.10
w/o Summary	79.22	50.88	65.05	52.10
w/o Demoq	81.15	50.93	66.11	55.80
w/o Demoa	78.50	49.15	64.20	51.70

Table 4: Ablation results for the different components of our context reward.

E.2 Instruction-following Evaluation Results.

To further validate the model-agnostic nature and generalizability of our ContextIF framework, we replicated our experiments on a different base model, Mistral-7B-Instruct. The results, presented in Table 5, demonstrate that ContextIF's superiority is not limited to the LLaMA3 architecture. Our ContextIF-7B consistently and significantly outperforms all other 7B models across every evaluated benchmark. This is particularly evident on fine-grained and multi-turn tasks; our model sets a new state-of-the-art on both the strict instruction-level accuracy of IFEval 70.18 and the challenging Turn3 accuracy of MultiIF 44.51. Furthermore, it also achieves the top score on the comprehensive LiveBench benchmark at 54.70. These findings strongly indicate that the core principle of ContextIF—training a model to generate its own optimal context via a nuanced reward signal—is a fundamental and transferable technique. It effectively enhances instruction-following capabilities irrespective of the underlying model architecture, confirming the robustness and widespread applicability of our proposed framework.

Model	IFEval				MultiIF			FollowBench	LiveBench
	P(L)	I(L)	P(S)	I(S)	Turn1	Turn2	Turn3	SSR	Score
					Mistral-7B-Instruct Models				
Mistral-7B-Instruct	53.30	64.19	48.49	59.31	46.16	34.53	28.62	60.87	50.20
Conifer-7B	54.89	64.98	50.46	60.91	46.20	46.27	34.74	61.57	51.80
UltraIF-7B	54.93	65.22	52.86	62.23	48.46	44.25	42.58	61.41	50.50
AutoIF-7B	54.90	65.23	52.87	62.23	47.63	37.41	32.58	59.50	51.60
SPAR-7B	58.25	68.11	56.56	66.19	51.42	48.46	38.13	67.13	53.70
ContextIF-7B	**64.51**	**73.26**	**61.30**	**70.18**	**69.92**	**59.08**	**44.51**	**68.87**	**54.70**

Table 5: Evaluation results of different models on IFEval, MultiIF, FollowBench(SSR), and LiveBench datasets. Pr. and Ins. stand for prompt and instruction levels, respectively. S and L represent strict and loose metrics for IFEval. For LiveBench, we only report the performance on the subset of instruction-following data.