Beyond Outcome: Trajectory-Driven Prompt Optimization via Multi-Dimensional Rewards

Zhixiang Liang[1,2,3*] **Yixiang Huang**[1*] **Chenjing Cai**[1†] **Hao Wu**[1]
Peihang Liu[1] **Qi Liu**[1,4] **Qiqi Zhong**[1,5] **Jiuchong Gao**[1†]
Jinghua Hao[1] **Renqing He**[1]

[1]Meituan [2]School of Artificial Intelligence, University of Chinese Academy of Sciences
[3]Beijing Key Lab of Molecular Imaging Technology Research and Translation,
CAS Key Lab of Molecular Imaging, CASIA [4]Peking University [5]UIBE
liangzhixiang2025@ia.ac.cn
{huangyixiang03, caichenjing, gaojiuchong}@meituan.com

Abstract

Automatic Prompt Optimization (APO) aims to improve the capabilities of Large Language Models (LLMs), yet existing methods remain limited by their result-oriented nature. Training-free approaches often converge to suboptimal prompts, while training-based methods suffer from reward sparsity and lack supervision of the optimization process. More importantly, both paradigms focus on obtaining the final prompt rather than learning the prompt optimization process itself. To address this limitation, we propose **ProPO**, a trajectory-driven **Pro**cess learning framework for **P**rompt **O**ptimization guided by multi-dimensional composite rewards. Specifically, ProPO introduces a historical trajectory reflection mechanism that leverages successful and failed optimization experiences for effective exploration. Concurrently, we adopt adaptive rubric rewards to provide fine-grained supervision and alleviate reward sparsity. More importantly, we design a novel Process Logic Reward that explicitly supervises the consistency of the optimization process. Extensive experiments demonstrate that ProPO, integrated with GRPO, establishes new state-of-the-art performance. Notably, ProPO achieves a remarkable **71.11%** accuracy on the highly complex AIME-2026 task and an absolute gain of **+25.70%** over the zero-shot baseline on Subjectivity Classification (Subj). Furthermore, empirical analyses reveal that our multi-dimensional reward mechanism provides finer-grained and denser supervision signals, enabling the model to learn the prompt optimization process itself and generate higher-quality prompts.

1 Introduction

LLMs have demonstrated unprecedented capabilities across diverse domains, including natural language processing, complex reasoning, and multi-agent systems [1, 2, 3, 4]. However, their performance on downstream tasks is notoriously sensitive to the input prompts. To alleviate the substantial labor costs of manual trial-and-error, Automatic Prompt Optimization (APO) has emerged to efficiently automate the prompt rewriting process.

Early APO explorations primarily conceptualized the LLM itself as an optimizer searching within a discrete text space [5, 6, 7].For instance, Automatic Prompt Engineer (APE) [8] utilizes LLMs as both proposers and scorers to filter candidate prompts. To navigate the vast discrete search space

*Equal contribution.
†Corresponding authors.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

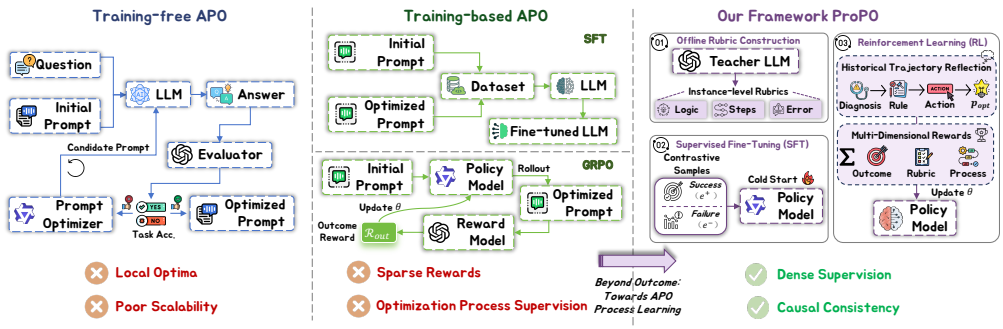


Figure 1: **Result-Oriented vs. Process-Oriented APO.** Existing APO paradigms face limitations: (Left) Training-free approaches can encounter local optima and scalability limitations during prompt search; (Middle) training-based RL methods can suffer from sparse rewards and insufficient supervision of the optimization process. (Right) ProPO aims to learn the prompt optimization process through three stages: (1) offline rubric construction establishes instance-level criteria for semantic supervision; (2) SFT provides a cold start from successful and failed execution trajectories; and (3) RL combines trajectory reflection with multi-dimensional rewards (Outcome + Rubric + Process), alleviating reward sparsity and providing explicit supervision over the prompt optimization process.

more efficiently, subsequent research introduced textual gradients [9, 10] or evolutionary algorithms [11, 12] to guide prompt updates through crossover and mutation mechanisms. Despite their success, training-free APO methods frequently become trapped in local optima as task complexity scales. To overcome this, recent efforts have introduced Reinforcement Learning (RL) to train policy models to automatically acquire optimization capabilities via environmental rewards [13, 14, 15] .However, these training-based methods are severely plagued by reward sparsity [16, 17]. More fundamentally, a critical blind spot persists across all existing APO paradigms: they are predominantly *result-oriented*. They focus primarily on acquiring the optimal prompt for a specific dataset, with limited attention to the intrinsic *process* of how such optimal prompts are derived. Figure 1 illustrates this fundamental distinction: existing training-free and training-based methods suffer from local optima and sparse rewards, whereas our proposed process-oriented framework learns the optimization process itself through trajectory-driven reflection and multi-dimensional rewards.

To bridge this fundamental gap, we propose **ProPO**, a trajectory-driven **Pro**cess learning framework for **P**rompt **O**ptimization guided by multi-dimensional composite rewards. This framework orchestrates a shift from merely searching for optimal prompts to explicitly learning the optimization process itself. During the RL exploration phase, we introduce a historical trajectory reflection mechanism. By adaptively sampling the complete reasoning chains of both positive and negative samples within the agent environment, we provide the policy model with high-quality contrastive feedback for self-reflection. To effectively mitigate reward sparsity, we devise an LLM-as-a-Judge Adaptive Rubric Reward. This mechanism imposes instance-level constraints, adaptively performing fine-grained semantic evaluations of candidate prompts. Crucially, to ensure logical consistency during the optimization process, we design a Process Logic Reward. This enforces the policy model to generate a structured Chain-of-Thought (CoT) that encompasses error diagnosis, rule summarization, and action execution, and jointly inspects its causal coherence. Process rewards are granted only when the reflective logic is flawlessly consistent, thereby genuinely internalizing the prompt optimization process within the model.

In summary, the main contributions of this paper are three-fold:

- **A Novel Process-Oriented Framework:** We propose a trajectory-driven process learning framework for prompt optimization guided by multi-dimensional composite rewards. This initiates a fundamental paradigm shift in APO, breaking away from traditional result-oriented searches toward explicitly learning the underlying prompt optimization process.
- **Contrastive Reflection & Adaptive Rubrics:** We introduce a reflection mechanism based on historical trajectories coupled with an Adaptive Rubric Reward. By utilizing the reasoning trajectories of positive and negative samples as contrastive feedback, we overcome the

stagnation caused by isolated reflection. Furthermore, the instance-level adaptive rubrics provide fine-grained evaluations, fundamentally alleviating reward sparsity in RL training.

- **Process Logic Reward:** We design a Process Logic Reward specifically tailored for prompt reflection trajectories. By conducting rigorous causal consistency checks on the structured CoT generated by the policy model, we enhance logical rigor and enable the model to better internalize the optimization methodology.

2 Related Work

2.1 Automatic Prompt Optimization

APO has demonstrated immense potential in exploring the capability boundaries of LLMs [18, 19, 20, 21]. However, efficiently discovering optimal prompts remains a critical challenge. In recent years, APO methodologies have generally diverged into two primary paradigms: training-free and training-based approaches. Training-free methods iteratively refine prompts through rich natural language feedback, primarily utilizing textual gradients [9, 10], evolutionary and genetic algorithms [11, 12], or historical reasoning trajectories [22, 23, 7]. Nevertheless, as downstream tasks scale in complexity, training-free methods inevitably struggle to yield optimal prompts relying solely on sparse textual feedback. Conversely, training-based methods aim to endow models with the intrinsic capability to generate or rewrite prompts. These approaches typically employ RL [14, 16, 17, 15, 24] to train a prompt optimizer based on online reward signals from downstream tasks. However, current training-based methods predominantly rely on outcome-based or format-based rewards, making them highly susceptible to reward hacking. More fundamentally, all existing APO paradigms focus predominantly on acquiring a high-quality prompt, with limited attention to the underlying *process* required to derive it. In contrast, our trajectory-driven framework explicitly learns the optimization methodology itself through historical trajectory reflection and multi-dimensional reward supervision.

2.2 Reinforcement Learning for LLMs

RL has been proven to significantly enhance LLMs, particularly in human alignment and complex reasoning capabilities [25, 26, 27, 28]. Early RL paradigms for LLMs were dominated by PPO [29], which suffers from exorbitant training costs and memory overhead. Consequently, DPO [30] was proposed, drastically reducing VRAM requirements via a cross-entropy loss on offline static data, albeit at the cost of sacrificing the model's online trial-and-error exploration capabilities in complex tasks. To address these limitations, GRPO [31] achieved a breakthrough by eliminating the need for a parameter-heavy value network while preserving robust online exploration capabilities. Building upon GRPO, researchers have continuously explored more stable and efficient RL variants [32, 33, 34, 35, 36, 37]. Our framework extends this by integrating multi-dimensional rewards that jointly supervise outcome, semantic alignment, and process logic.

3 Methodology

3.1 Problem Formulation

Given a frozen target LLM denoted as $\mathcal{M}_{\text{target}}$ and a task-specific dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ comprising N samples (where x_i represents the input query and y_i is the corresponding ground-truth label), we aim to learn a parameterized policy π_θ. This policy is trained to rewrite a human-crafted initial prompt p_{init} into an optimized prompt p_{opt}, thereby maximizing the overall expected task performance of $\mathcal{M}_{\text{target}}$ over the task distribution. The optimization objective is formalized as:

$$p_{\text{opt}} = \arg\max_{p \in \mathcal{P}} \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\mathbb{1}\big(\mathcal{M}_{\text{target}}([p; x]) = y\big) \right] \tag{1}$$

where $[\cdot;\cdot]$ denotes the sequence concatenation of the prompt and the input, and $\mathbb{1}(\cdot)$ is the indicator function. To efficiently achieve this objective and overcome the cold-start dilemma, we decouple the optimization paradigm into a progressive training pipeline comprising offline data preparation, supervised fine-tuning, and reinforcement learning with richer supervision signals.

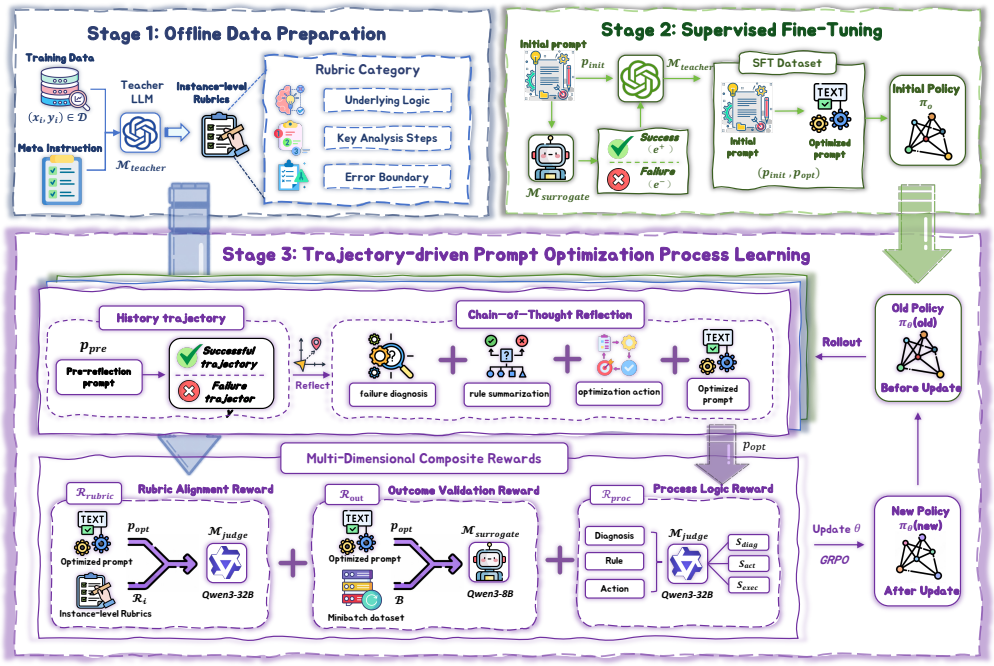


Figure 2: Overview of our trajectory-driven prompt optimization framework. The pipeline operates through three stages: (1) *Offline data preparation* utilizes a teacher model to generate SFT data and distill instance-level evaluation criteria; (2) *Supervised fine-tuning* endows the policy model with foundational rewriting capabilities;(3) *Reinforcement learning* optimizes the policy via GRPO with multi-dimensional rewards, providing richer supervision over the prompt optimization process.

3.2 Training Pipeline

Our framework organizes prompt optimization into a three-stage learning pipeline. First, offline data preparation constructs SFT data and instance-level rubrics from execution trajectories. Second, supervised fine-tuning equips the policy with foundational prompt rewriting capabilities. Finally, reinforcement learning further optimizes the policy through historical trajectory reflection and multi-dimensional reward supervision, enabling explicit learning of the prompt optimization process.

3.2.1 Phase 1: Offline Data Preparation

To construct training resources, we employ GPT-4.1 as a teacher model for offline data preparation. This phase encompasses two components: the generation of supervised fine-tuning (SFT) data and the construction of evaluation rubrics. To construct the SFT data, we first evaluate initial prompts on the target model to collect contrastive trajectories of execution successes (e^+) and failures (e^-). These contrastive trajectories are provided as feedback to GPT-4.1, guiding it to diagnose the root causes of failures and synthesize optimized prompts p_{opt}. This process yields the instruction-tuning dataset $\mathcal{D}_{\text{SFT}} = \{\langle p_{\text{init}}^{(j)}, p_{\text{opt}}^{(j)} \rangle\}_{j=1}^{M}$. Concurrently, to mitigate reward sparsity in the subsequent reinforcement learning phase, the teacher model analyzes the task characteristics of each sample to generate an instance-level evaluation rubric set $\mathcal{R}_i$. Categorized into underlying essential logic, key analytical steps, and strict error boundaries, these rubrics serve as dense supervision signals to ensure that optimized prompts strictly adhere to semantic constraints (detailed in Appendix G).

3.2.2 Phase 2: Supervised Fine-Tuning

Initializing reinforcement learning from scratch with sparse outcome-based rewards suffers from extremely low sample efficiency. Therefore, utilizing the curated dataset $\mathcal{D}_{\text{SFT}}$, we train the policy model π_θ to inject foundational rewriting prior knowledge. This is achieved by minimizing the

negative log-likelihood loss over the autoregressive generation of optimal prompts:

$$\mathcal{L}_{\text{SFT}}(\theta) = -\mathbb{E}_{(p_{\text{init}}, p_{\text{opt}})\sim\mathcal{D}_{\text{SFT}}}\left[\sum_{t=1}^{|p_{\text{opt}}|} \log \pi_\theta\big(w_t \mid w_{<t}, p_{\text{init}}\big)\right] \tag{2}$$

This initialization drastically reduces the difficulty of discovering high-quality prompts, establishing a solid foundation for the subsequent reinforcement learning phase.

3.2.3 Phase 3: Reinforcement Learning Exploration

Following the SFT phase, we introduce an interactive, feedback-driven stage based on the Group Relative Policy Optimization (GRPO) algorithm to achieve iterative policy evolution. During this phase, the model employs a historical trajectory-based reflection mechanism (detailed in Section 3.3) to adaptively sample and generate structured reasoning trajectories based on environmental interactions. These trajectories are then rigorously evaluated by a multi-dimensional composite reward function (detailed in Section 3.4), which integrates outcome validation, alignment with the offline-constructed rubrics, and process logic verification. The policy is updated by maximizing a clipped surrogate objective, regularized by the KL divergence from the SFT reference model π_{ref}:

$$\mathcal{L}_{\text{GRPO}}(\theta) = \mathbb{E}\left[\frac{1}{K}\sum_{k=1}^{K}\Big(\min\big(\rho_k A_k, \text{clip}(\rho_k, 1-\epsilon, 1+\epsilon)A_k\big) - \beta\mathbb{D}_{\text{KL}}(\pi_\theta\|\pi_{\text{ref}})\Big)\right] \tag{3}$$

where A_k denotes the intra-group relative advantage. Driven entirely by explicit process logic and dense multi-dimensional supervision, this mechanism ensures stable policy exploration.

3.3 Historical Trajectory-based Reflection Mechanism

To enable the model to generate better prompts, we design a reflection mechanism rooted in optimization trajectories. Relying on self-reflection without empirical grounding often leads to superficial modifications. Therefore, we incorporate trial-and-error feedback into the reasoning loop.

Given a task context c, the policy π_θ first outputs an initial version of the prompt. We then conduct exploratory trials within a Surrogate Execution Environment (e.g., Qwen3-8B [38]) to capture empirical execution successes (e^+) and failures (e^-). Conditioned on the historical prompt and this contrastive environmental feedback, the policy model π_θ sequentially deduces a structured CoT. This deduction process explicitly formulates a failure diagnosis (d), a rule summarization (r), and an optimization action (a), ultimately outputting the refined prompt p_{opt}.

We define this complete, structured reasoning trajectory as a sequence tuple τ:

$$\tau = \big(\underbrace{d, r, a}_{\text{CoT}},\ p_{\text{opt}}\big) \tag{4}$$

The meta-prompts utilized to guide this generation and subsequent reflection process are detailed in Appendix A.3. This trajectory exposes the model's reasoning steps for subsequent verification.

3.4 Multi-Dimensional Reward Formulation

Traditional outcome-based reward signals are often excessively sparse, struggling to guide the model in learning complex reflection logic and processes over long sequences. To address this, we propose a multi-dimensional composite reward mechanism $\mathcal{R}(\tau)$ that decouples downstream execution performance, semantic constraint adherence, and the logical rigor of the deduction process. This provides denser and more structured feedback signals for policy optimization.

Outcome Validation Reward ($\mathcal{R}_{\text{out}}$): This component measures the objective effectiveness of the optimized instruction. We deploy the prompt p_{opt} within the surrogate execution environment $\mathcal{M}_{\text{surrogate}}$ and compute its expected accuracy over a sampled mini-batch validation set $\mathcal{B}$:

$$\mathcal{R}_{\text{out}}(p_{\text{opt}}) = \frac{1}{|\mathcal{B}|}\sum_{(x_j, y_j)\in\mathcal{B}} \text{Eval}\big(\mathcal{M}_{\text{surrogate}}([p_{\text{opt}}; x_j]), y_j\big) \tag{5}$$

Adaptive Rubric Reward ($\mathcal{R}_{\text{rubric}}$): To evaluate prompt quality while ensuring that p_{opt} does not deviate from the original task intent at the semantic level, we employ a larger parameter model (e.g., Qwen3-32B [38]) as the evaluator $\mathcal{M}_{\text{judge}}$. This model scores the matching degree between p_{opt} and the offline-constructed instance-level tailored rubrics $\mathcal{R}_i$, normalizing the output:

$$\mathcal{R}_{\text{rubric}}(p_{\text{opt}}) = \mathcal{M}_{\text{judge}}(p_{\text{opt}}, \mathcal{R}_i) \in [0, 1] \tag{6}$$

Process Logic Reward ($\mathcal{R}_{\text{proc}}$): To enable the model to explicitly learn *how* to acquire high-quality prompts, we design a Process Logic Reward that supervises the prompt optimization process by evaluating the logical consistency of the optimization trajectory τ. Utilizing the larger parameter model, we discretize the inspection dimensions into three binary indicator variables: $s_{\text{diag}} \in \{0, 1\}$ evaluates whether the error diagnosis d accurately attributes the core essence of the negative sample failures; $s_{\text{act}} \in \{0, 1\}$ verifies whether the optimization action a effectively eliminates the diagnosed defects; and $s_{\text{exec}} \in \{0, 1\}$ determines whether p_{opt} practically executes the requirements of action a. The process reward is activated only when the aforementioned reasoning chain forms a closed logical loop:

$$\mathcal{R}_{\text{proc}}(\tau) = \mathbb{1}(s_{\text{diag}} = 1) \cdot \mathbb{1}(s_{\text{act}} = 1) \cdot \mathbb{1}(s_{\text{exec}} = 1) \tag{7}$$

Ultimately, the overall scalar reward for trajectory τ is composed of the superposition of signals from these three dimensions, seamlessly balancing outcome accuracy with process rationality:

$$\mathcal{R}_{\text{total}}(\tau) = \mathcal{R}_{\text{out}}(p_{\text{opt}}) + \mathcal{R}_{\text{rubric}}(p_{\text{opt}}) + \mathcal{R}_{\text{proc}}(\tau) \tag{8}$$

The complete algorithmic procedure, detailing the trajectory-driven prompt optimization process across its three stages, is provided in Appendix B.

4 Experiments

We evaluate ProPO on benchmarks covering classification, mathematical reasoning, and instruction following. We compare against representative methods and conduct ablation studies to analyze explicit supervision over the prompt optimization process and multi-dimensional rewards.

4.1 Experimental Setup

Datasets To thoroughly assess prompt quality and robustness, we employ six mainstream public benchmarks categorized into three core dimensions: (1) *Classification tasks* (SST-5 [39], Subj [40], TREC [41]) to examine strict output formatting and semantic boundary constraints; (2) *Mathematical reasoning tasks* (AIME-2026 [42], LiveBench-Math [43], abbreviated as LBench in the experimental tables) to evaluate the elicitation of deep CoT logic; and (3) *Instruction-following tasks* (IFBench [44]) to test precise control over multiple complex, long-context constraints.

Implementation Details We designate Qwen3-8B-Instruct as the core policy model (Actor) to execute prompt optimization, while the equivalently scaled Qwen3-8B serves as the surrogate execution model to provide empirical feedback. The training pipeline consists of a SFT initialization phase to inject rewriting priors, followed by RL exploration via the GRPO algorithm. Comprehensive implementation details, including Low-Rank Adaptation [45] (LoRA) configurations, RL hyperparameters, and hardware setups, are provided in Appendix A.

Baselines We horizontally compare our method against foundational baselines and state-of-the-art (SOTA) automatic prompt optimization approaches. The specific baselines include: (1) **Zero-shot Baseline**: The performance of original, unoptimized prompts; (2) **SFT**: Prompts generated by the policy model after only supervised fine-tuning; (3) **GRPO**: A baseline utilizing the standard GRPO algorithm but lacking multi-dimensional reward constraints; and (4) **Mainstream APO methods**: Including TextGrad, SPO, Trace, GEPA, StablePrompt,and PromptR1.

4.2 Main Results

The accuracy performance of all compared methods across the three major task categories, comprising six benchmark datasets, is presented in Table 1. Empirical results demonstrate that our proposed

Table 1: Performance comparison (%; mean ± std over seeds $\{1, 42, 2026\}$). LBench denotes LiveBench-Math. Best/second-best results are bold/underlined.

Method	Instruction	Math Reasoning		Classification			Avg.
	IFBench	AIME	LBench	SST5	Subj	TREC	
Training-free APO Methods							
TextGrad [10]	36.73±0.28	18.89±1.57	59.53±0.62	54.34±0.28	68.70±0.04	85.07±0.93	53.88
SPO [23]	25.06±3.54	55.55±3.14	57.63±0.05	54.74±0.48	72.93±0.20	72.60±0.82	56.42
Trace [22]	26.08±1.25	57.78±1.57	57.13±0.17	52.46±0.26	73.85±0.40	76.60±0.43	57.32
GEPA [12]	36.39±1.69	57.78±3.14	55.73±2.25	53.51±0.44	80.18±0.80	86.00±0.33	61.60
Training-based APO Methods							
Zero-shot	33.55±0.97	57.78±1.57	54.67±0.26	54.31±0.51	68.17±0.08	85.40±0.33	58.98
SFT	37.18±0.43	58.89±1.57	58.13±0.69	54.13±0.80	70.37±0.15	88.40±0.43	61.18
GRPO ($\mathcal{R}_{out}$)	38.08±0.50	57.78±5.67	58.70±1.72	55.93±0.32	82.42±0.54	89.73±0.09	63.77
StablePrompt [16]	27.66±0.58	58.89±1.57	62.10±1.56	54.88±0.72	77.63±0.41	87.87±0.25	61.51
PromptR1 [17]	38.66±0.32	63.30±4.77	58.27±0.61	55.76±0.07	79.90±0.11	77.40±0.85	62.22
ProPO	**40.59**±0.43	**71.11**±1.57	**63.93**±0.78	**58.71**±0.46	**93.87**±0.52	**93.20**±0.29	**70.24**

multi-dimensional reward-based optimization framework achieves significant and globally optimal performance across all evaluation dimensions. In classification tasks emphasizing format and boundary control (e.g., Subj and TREC), our method achieves absolute performance leaps of 25.70% and 7.8% over the unoptimized Zero-shot Baseline, respectively. Furthermore, it comfortably outperforms the SFT and GRPO baselines, thoroughly validating that the Process Logic Reward effectively guides the model to internalize task constraints. Notably, in the AIME-2026 task, which requires exceptionally deep logical chains, our method leads by a wide margin with an accuracy of 71.11%. This successfully overcomes the inherent bottleneck of mainstream APO methods, which frequently trap the optimization process in local optima within complex solution spaces.

Performance Across Policy Backbones. We further evaluate ProPO with four policy backbones from the Qwen and Llama families and different parameter scales. As shown in Figure 3, ProPO improves all evaluated backbones, with average gains ranging from 11.26 to 16.24 percentage points, indicating that the gains are not restricted to a single backbone configuration.

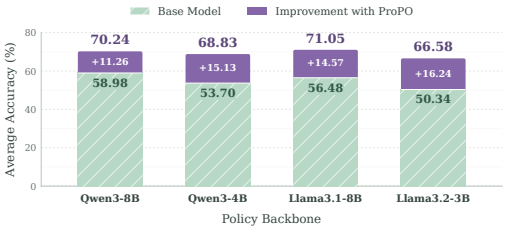


Figure 3: Average performance gains of ProPO across different policy backbones.

Specifically, Qwen3-8B, Qwen3-4B, Llama3.1-8B, and Llama3.2-3B improve by 11.26, 15.13, 14.57, and 16.24 percentage points, respectively. These results show that ProPO remains effective across different model families and scales, supporting its broader applicability across different policy architectures. Detailed benchmark-level results are provided in Appendix E.2.

4.3 Ablation Study

To investigate the independent contributions of each dimension within the composite reward signal, we ablate both the three reward dimensions and the individual consistency checks within the Process Logic Reward.

Table 2: Ablation of the multi-dimensional reward components. Results are mean ± standard deviation over seeds $\{1, 42, 2026\}$.

Method	IFBench	AIME	LBench	SST5	Subj	TREC	Avg.
GRPO ($\mathcal{R}_{out}$)	38.08±0.50	57.78±5.67	58.70±1.72	55.93±0.32	82.42±0.54	89.73±0.09	63.77
w/o $\mathcal{R}_{out}$	37.53±0.71	57.78±3.14	57.80±0.28	55.17±0.06	87.27±0.34	91.27±0.09	64.47
w/o $\mathcal{R}_{rubric}$	38.20±0.87	62.22±1.57	58.20±1.44	57.51±0.08	91.20±0.16	91.93±0.34	66.54
w/o $\mathcal{R}_{proc}$	39.68±0.58	62.22±4.16	61.13±0.38	57.70±0.51	91.30±0.46	92.07±0.09	67.35
ProPO (Full)	**40.59**±0.43	**71.11**±1.57	**63.93**±0.78	**58.71**±0.46	**93.87**±0.52	**93.20**±0.29	**70.24**

As shown in Table 2, the full multi-dimensional reward achieves the highest average accuracy of 70.24%, outperforming outcome-only GRPO by 6.47 percentage points. Removing $\mathcal{R}_{out}$ causes the largest drop to 64.47%, showing that execution feedback remains essential for assessing downstream prompt effectiveness. Removing $\mathcal{R}_{rubric}$ reduces the average accuracy to 66.54%, indicating that instance-level rubric supervision provides fine-grained optimization feedback. Removing $\mathcal{R}_{proc}$ yields 67.35%, showing that explicit supervision of the prompt optimization process provides additional benefits beyond outcome and rubric feedback.

Table 3: Ablation study of individual components in the Process Logic Reward. Results are mean $\pm$ standard deviation over seeds $\{1, 42, 2026\}$.

Method	IFBench	AIME	LBench	SST-5	Subj	TREC	Avg.
w/o S_{exec}	38.88 ± 0.97	$\mathbf{71.11 \pm 3.14}$	$\mathbf{64.20 \pm 0.43}$	58.26 ± 0.09	$\mathbf{94.08 \pm 0.17}$	93.00 ± 0.28	69.92
w/o S_{act}	38.88 ± 0.42	68.89 ± 1.57	62.07 ± 0.29	58.04 ± 0.53	93.08 ± 0.25	92.47 ± 0.41	68.91
w/o S_{diag}	37.99 ± 0.33	65.56 ± 1.57	61.80 ± 0.73	57.92 ± 0.24	91.88 ± 0.65	92.00 ± 0.16	67.86
w/o $\mathcal{R}_{\text{proc}}$	39.68 ± 0.58	62.22 ± 4.16	61.13 ± 0.38	57.70 ± 0.51	91.30 ± 0.46	92.07 ± 0.09	67.35
ProPO (Full)	$\mathbf{40.59 \pm 0.43}$	$\mathbf{71.11 \pm 1.57}$	63.93 ± 0.78	$\mathbf{58.71 \pm 0.46}$	93.87 ± 0.52	$\mathbf{93.20 \pm 0.29}$	**70.24**

We further examine the internal design of the Process Logic Reward by individually removing its diagnosis, action, and execution consistency checks while retaining the remaining components.

As shown in Table 3, removing any individual consistency check reduces the average accuracy. Among the three components, removing S_{diag} causes the largest decrease, lowering the average accuracy from 70.24% to 67.86%. Removing S_{act} and S_{exec} reduces the average accuracy to 68.91% and 69.92%, respectively. Removing the entire Process Logic Reward further reduces the average accuracy to 67.35%. These results indicate that diagnosis, action, and execution consistency jointly contribute to explicitly supervising the prompt optimization process.

Overall, the ablation results show that the three reward dimensions provide complementary supervision, while the three consistency checks within $\mathcal{R}_{proc}$ jointly support process-level supervision.Representative failure cases are presented in Section 4.4, and further analysis of Process Logic Reward integration strategies is provided in Appendix H.1.

4.4 Further Analysis

We further analyze ProPO from three complementary perspectives: trajectory-driven reflection, the granularity of rubric supervision, and computational efficiency. These analyses provide additional insights into how ProPO refines prompts, benefits from fine-grained evaluation criteria, and balances optimization effectiveness with computational cost.

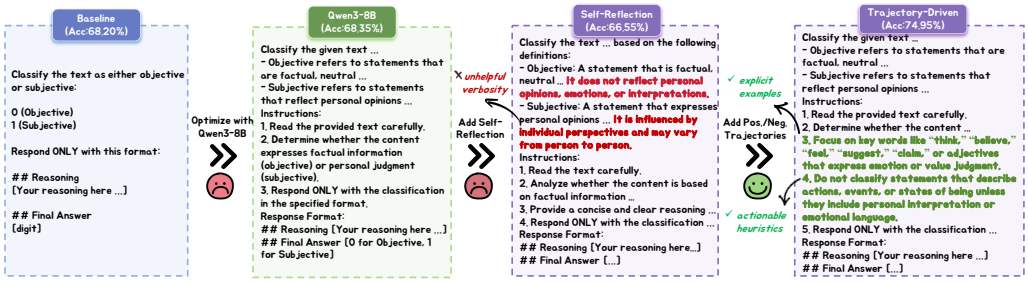


Figure 4: Qualitative comparison of prompt refinement strategies. Trajectory-driven reflection leverages successful and failed execution examples to produce more targeted prompt revisions.

Trajectory-Driven Reflection. To elucidate the necessity of empirical execution feedback, Figure 4 visually compares prompt evolution for the Subjectivity Classification task.

Empirical results indicate that a pure reflection mechanism is largely ineffective. When relying solely on historical prompts, the policy model struggles to determine a valid optimization direction, often modifying irrelevant details. As shown in Figure 4, this ungrounded reflection merely generates verbose, generic definitions, degrading accuracy from 68.35% to 66.55%.

In contrast, incorporating positive and negative reasoning trajectories provides critical empirical grounding. By contrasting successful and failed executions, the model can easily pinpoint the root

causes of errors. This trajectory-driven approach induces the generation of concrete exemplars and highly specific heuristics (e.g., targeted semantic indicators and explicit boundaries for action-based statements), which substantially boosts task accuracy to 74.95%. This visual evidence demonstrates that contrastive trajectory feedback is essential to guide the model away from spurious hallucinations and toward targeted, high-quality prompt optimization.

Table 4: Impact of rubric granularity on performance. Results are mean ± std over seeds $\{1, 42, 2026\}$. Δ denotes the change relative to w/o $\mathcal{R}_{rubric}$.

Rubric Strategy	IFBench	AIME	LBench	SST5	Subj	TREC	Avg.	Δ
Without Rubric	38.20±0.87	62.22±1.57	58.20±1.44	57.51±0.08	91.20±0.16	91.93±0.34	66.54	–
Manually Written	39.00±0.58	62.22±1.57	59.83±0.93	52.82±1.33	88.22±0.34	92.27±0.62	65.73	−0.81
Dataset-level	38.66±0.80	65.56±1.57	59.93±1.13	54.21±0.66	89.52±0.21	91.80±0.16	66.61	+0.07
Instance-level (Ours)	**40.59**±0.43	**71.11**±1.57	**63.93**±0.78	**58.71**±0.46	**93.87**±0.52	**93.20**±0.29	**70.24**	**+3.70**

Impact of Rubric Granularity. We further investigate how the granularity of rubric supervision affects prompt optimization. We compare manually written rubrics, dataset-level rubrics, and our instance-level rubrics against the setting without rubric supervision.

Table 4 reveals that rubric effectiveness critically depends on granularity. Instance-level rubrics achieve **+3.70%** average gain (with +8.89% on AIME), whereas manually written and dataset-level rubrics degrade performance by 0.81% and improve performance by 0.07%, respectively. This result stems from *semantic misalignment*: fixed prompts lack task adaptability, while dataset-level rubrics introduce noise when samples exhibit heterogeneous reasoning patterns. Conversely, our adative instance-level approach captures sample-specific constraints, enabling contextualized optimization. This validates a key insight: *fine-grained, adaptive evaluation is essential for process learning*. Generic rubrics fail to provide the precise supervision required for LLMs to internalize optimization mechanics. Further details on rubric granularity are provided in Appendix G.

Verification of the Step-wise Process Logic Effective prompt optimization demands strict logical consistency across the reflection trajectory. Figure 5 delineates how breakdowns in this cognitive chain degrade prompt quality across three canonical failure modes:

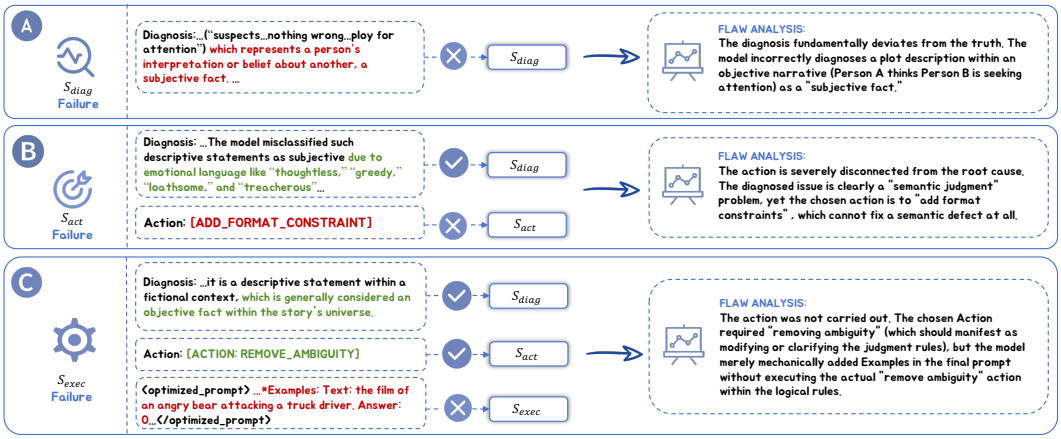


Figure 5: Visualization of process logic breakdowns. Maintaining an unbroken cognitive chain is imperative. (A) S_{diag} Failure introduces misguided rules via incorrect attribution; (B) S_{act} Failure prescribes misaligned actions, leaving core vulnerabilities unaddressed; (C) S_{exec} Failure executes superficial fixes. Our framework penalizes these disjointed trajectories to guarantee logical consistency.

(A) S_{diag} Failure (Diagnosis Error): Misattributing the root cause (e.g., mischaracterizing an objective narrative as a "subjective fact") prompts the formulation of unfounded rules. This introduces erroneous constraints that confound the downstream model and degrade performance on previously solved distributions.

(B) S_{act} Failure (Action Misalignment): Even with a correct diagnosis, prescribing a mismatched resolution (e.g., applying structural formatting to resolve a semantic ambiguity) leaves the core

vulnerability unaddressed. Injecting these extraneous instructions merely dilutes the model's attention without conferring performance gains.

(C) S_{exec} Failure (Execution Failure): Failing to implement a valid strategy (e.g., superficially inserting an example instead of revising the core judgment rules) misdirects the generation phase, rendering the refinement attempt entirely ineffective.

By rigorously verifying component-wise coherence, our framework penalizes these disjointed trajectories, ensuring that every prompt follows a cohesive and well-executed optimization rationale.

Computational Efficiency. We examine the computational cost and performance of ProPO. At the 20-step checkpoint, ProPO reaches 66.58% average accuracy in approximately 2.6 hours, already exceeding GEPA (61.60%), StablePrompt (61.51%), and PromptR1 (62.22%). Continuing training to 80 steps further improves the average accuracy to 70.24% within 12 hours. Under matched training settings on 8×A100 GPUs, ProPO outperforms StablePrompt and PromptR1 by 8.73 and 8.02 percentage points, respectively, while requiring less total runtime.

Table 5: Computational efficiency comparison.

Method	Time	/Step	GPU	Steps	Avg.
Training-free					
TextGrad	2.46h	8.20m	–	3	53.88
SPO	3.07h	7.68m	–	4	56.42
TRACE	7.94h	26.48m	–	3	57.32
GEPA	1.08h	0.67m	–	5	61.60
Training-based					
StablePrompt	14h	5.65m	8×A100	40	61.51
PromptR1	29h	39.90m	8×A100	30	62.22
ProPO@20	2.6h	8.05m	8×A100	20	66.58
ProPO	**12**h	8.05m	8×A100	80	**70.24**

ProPO has an average per-step latency of 8.05 minutes, compared with 5.65 minutes for StablePrompt and 39.90 minutes for PromptR1. Although training-free methods require less wall-clock time, they achieve lower average accuracy under our evaluation setting. Overall, ProPO achieves a favorable balance between computational cost and performance.A detailed latency breakdown is provided in Appendix D.1.

5 Limitations and Future Work

While our framework achieves strong empirical results, two primary challenges remain. First, the reliance on a teacher model for rubric generation imposes an upper bound on the policy's exploratory capabilities. Second, invoking large judge models during RL training incurs substantial computational overhead, particularly for long-trajectory sampling and scoring. Future work could explore self-evolving mechanisms where the policy model learns to autonomously refine its own optimization strategies through iterative self-assessment, progressively reducing dependency on external supervision. Additionally, extending our approach to open-ended creative tasks and multimodal prompt optimization represents another promising direction.

6 Conclusion

This paper proposes a trajectory-driven process learning framework for prompt optimization guided by multi-dimensional composite rewards, shifting the APO paradigm from searching for optimal prompts to explicitly learning *how* to derive them. We introduce a reflection mechanism that leverages historical reasoning trajectories of both positive and negative samples during RL exploration, enabling iterative deduction of superior prompts. We design a composite reward function combining outcome, adaptive rubric, and process logic rewards, providing fine-grained and denser signals. Extensive experiments across text classification, mathematical reasoning, and long-context instruction following show that by enabling the model to learn the prompt optimization process, our framework achieves better acquisition of high-quality prompts and outperforms existing state-of-the-art baselines, providing a new perspective for prompt optimization.

References

[1] Xinyuan Wang, Chenxi Li, Zhen Wang, Fan Bai, Haotian Luo, Jiayou Zhang, Nebojsa Jojic, Eric P Xing, and Zhiting Hu. Promptagent: Strategic planning with language models enables expert-level prompt optimization. *arXiv preprint arXiv:2310.16427*, 2023.

[2] Zhiheng Xi, Chenyang Liao, Guanyu Li, Zhihao Zhang, Wenxiang Chen, Binghai Wang, Senjie Jin, Yuhao Zhou, Jian Guan, Wei Wu, et al. Agentprm: Process reward models for llm agents via step-wise promise and progress. In *Proceedings of the ACM Web Conference 2026*, pages 4184–4195, 2026.

[3] Hui Huang, Yancheng He, Hongli Zhou, Rui Zhang, Wei Liu, Weixun Wang, Jiaheng Liu, and Wenbo Su. Think-j: Learning to think for generative llm-as-a-judge. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 31158–31166, 2026.

[4] Mert Cemri, Shubham Agrawal, Akshat Gupta, Shu Liu, Audrey Cheng, Qiuyang Mang, Ashwin Naren, Lutfi Eren Erdogan, Koushik Sen, Matei Zaharia, et al. Adaevolve: Adaptive llm driven zeroth-order optimization. *arXiv preprint arXiv:2602.20133*, 2026.

[5] Eshaan Agarwal, Joykirat Singh, Vivek Dani, Raghav Magazine, Tanuja Ganu, and Akshay Nambi. Promptwizard: Task-aware prompt optimization framework. *arXiv preprint arXiv:2405.18369*, 2024.

[6] Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution. *arXiv preprint arXiv:2309.16797*, 2023.

[7] Guanbao Liang, Yuanchen Bei, Sheng Zhou, Yuheng Qin, Huan Zhou, Bingxin Jia, Bin Li, and Jiajun Bu. Generalizable self-evolving memory for automatic prompt optimization. *arXiv preprint arXiv:2603.21520*, 2026.

[8] Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. In *The eleventh international conference on learning representations*, 2022.

[9] Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 7957–7968, 2023.

[10] Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. Textgrad: Automatic" differentiation" via text. *arXiv preprint arXiv:2406.07496*, 2024.

[11] Zeliang Tong, Zhuojun Ding, and Wei Wei. Evoprompt: Evolving prompts for enhanced zero-shot named entity recognition with large language models. In *Proceedings of the 31st international conference on computational linguistics*, pages 5136–5153, 2025.

[12] Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, et al. Gepa: Reflective prompt evolution can outperform reinforcement learning. *arXiv preprint arXiv:2507.19457*, 2025.

[13] Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric Xing, and Zhiting Hu. Rlprompt: Optimizing discrete text prompts with reinforcement learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3369–3391, 2022.

[14] Weize Kong, Spurthi Hombaiah, Mingyang Zhang, Qiaozhu Mei, and Michael Bendersky. Prewrite: Prompt rewriting with reinforcement learning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 594–601, 2024.

[15] PaweĹ Batorski, Adrian Kosmala, and Paul Swoboda. Prl: Prompts from reinforcement learning. *arXiv preprint arXiv:2505.14412*, 2025.

[16] Minchan Kwon, Gaeun Kim, Jongsuk Kim, Haeil Lee, and Junmo Kim. Stableprompt: Automatic prompt tuning using reinforcement learning for large language model. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 9868–9884, 2024.

[17] Wenjin Liu, Haoran Luo, Xueyuan Lin, Haoming Liu, Tiesunlong Shen, Jiapu Wang, Rui Mao, and Erik Cambria. Prompt-r1: Collaborative automatic prompting framework via end-to-end reinforcement learning. *arXiv preprint arXiv:2511.01016*, 2025.

[18] Kiran Ramnath, Kang Zhou, Sheng Guan, Soumya Smruti Mishra, Xuan Qi, Zhengyuan Shen, Shuai Wang, Sangmin Woo, Sullam Jeoung, Yawei Wang, et al. A systematic survey of automatic prompt optimization techniques. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 33066–33098, 2025.

[19] Lechen Zhang, Tolga Ergen, Lajanugen Logeswaran, Moontae Lee, and David Jurgens. Sprig: Improving large language model performance by system prompt optimization. *arXiv preprint arXiv:2410.14826*, 2024.

[20] Yaoning Yu, Ye Yu, Peiyan Zhang, Kai Wei, Haojing Luo, and Haohan Wang. Sipdo: Closed-loop prompt optimization via synthetic data feedback. *arXiv preprint arXiv:2505.19514*, 2025.

[21] Prashant Trivedi, Souradip Chakraborty, Avinash Reddy, Vaneet Aggarwal, Amrit Singh Bedi, and George K Atia. Align-pro: A principled approach to prompt optimization for llm alignment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 27653–27661, 2025.

[22] Ching-An Cheng, Allen Nie, and Adith Swaminathan. Trace is the next autodiff: Generative optimization with rich feedback, execution traces, and llms. *Advances in Neural Information Processing Systems*, 37:71596–71642, 2024.

[23] Jinyu Xiang, Jiayi Zhang, Zhaoyang Yu, Fengwei Teng, Jinhao Tu, Xinbing Liang, Sirui Hong, Chenglin Wu, and Yuyu Luo. Self-supervised prompt optimization. *arXiv preprint arXiv:2502.06855*, 2025.

[24] Hao Sun, Alihan Hüyük, and Mihaela van der Schaar. Query-dependent prompt evaluation and optimization with offline inverse rl. *arXiv preprint arXiv:2309.06553*, 2023.

[25] Kaiyan Zhang, Yuxin Zuo, Bingxiang He, Youbang Sun, Runze Liu, Che Jiang, Yuchen Fan, Kai Tian, Guoli Jia, Pengfei Li, et al. A survey of reinforcement learning for large reasoning models. *arXiv preprint arXiv:2509.08827*, 2025.

[26] Shuo Liu, Zeyu Liang, Xueguang Lyu, and Christopher Amato. Llm collaboration with multi-agent reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 32150–32158, 2026.

[27] Tian Xie, Zitian Gao, Qingnan Ren, Haoming Luo, Yuqian Hong, Bryan Dai, Joey Zhou, Kai Qiu, Zhirong Wu, and Chong Luo. Logic-rl: Unleashing llm reasoning with rule-based reinforcement learning. *arXiv preprint arXiv:2502.14768*, 2025.

[28] Yuxiang Wei, Olivier Duchenne, Jade Copet, Quentin Carbonneaux, Lingming Zhang, Daniel Fried, Gabriel Synnaeve, Rishabh Singh, and Sida I Wang. Swe-rl: Advancing llm reasoning via reinforcement learning on open software evolution. *arXiv preprint arXiv:2502.18449*, 2025.

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

[30] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

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

[32] Yuzhong Zhao, Yue Liu, Junpeng Liu, Jingye Chen, Xun Wu, Yaru Hao, Tengchao Lv, Shaohan Huang, Lei Cui, Qixiang Ye, et al. Geometric-mean policy optimization. *arXiv preprint arXiv:2507.20673*, 2025.

[33] Shihui Yang, Chengfeng Dou, Peidong Guo, Kai Lu, Qiang Ju, Fei Deng, and Rihui Xin. Dcpo: Dynamic clipping policy optimization. *arXiv preprint arXiv:2509.02333*, 2025.

[34] Shuaijie She, Yu Bao, Yu Lu, Lu Xu, Tao Li, Wenhao Zhu, Shujian Huang, Shanbo Cheng, Lu Lu, and Yuxuan Wang. Dupo: Enabling reliable llm self-verification via dual preference optimization. *arXiv preprint arXiv:2508.14460*, 2025.

[35] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale, 2025. *URL https://arxiv. org/abs/2503.14476*, 1:2, 2025.

[36] Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, et al. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.

[37] Shih-Yang Liu, Xin Dong, Ximing Lu, Shizhe Diao, Peter Belcak, Mingjie Liu, Min-Hung Chen, Hongxu Yin, Yu-Chiang Frank Wang, Kwang-Ting Cheng, et al. Gdpo: Group reward-decoupled normalization policy optimization for multi-reward rl optimization. *arXiv preprint arXiv:2601.05242*, 2026.

[38] 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.

[39] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

[40] Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting of the association for computational linguistics (ACL-04)*, pages 271–278, 2004.

[41] Ellen M Voorhees and Dawn M Tice. Building a question answering test collection. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 200–207, 2000.

[42] Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2026, 2026.

[43] Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Ben Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Sreemanti Dey, et al. Livebench: A challenging, contamination-limited llm benchmark. *arXiv preprint arXiv:2406.19314*, 2024.

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

[45] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

[46] Chen Wang, Zhaochun Li, Jionghao Bai, Yining Zhang, Hexuan Deng, Ge Lan, and Yue Wang. Distilled reinforcement learning for llm post-training. *arXiv preprint arXiv:2607.17247*, 2026.

A Implementation and Reproducibility

This appendix details the implementation of ProPO, including SFT and GRPO training configurations, data allocation, and prompt templates for prompt optimization, trajectory reflection, offline rubric construction, and reward evaluation. The main experiments use Qwen3-8B-Instruct as the prompt optimization policy, Qwen3-8B as the surrogate execution model, GPT-4.1 as the offline teacher model, and Qwen3-32B as the reward judge. All training experiments are conducted on 8 NVIDIA A100 GPUs.

A.1 Training Configuration

ProPO follows a progressive training pipeline consisting of supervised fine-tuning (SFT) for cold-start initialization, followed by online reinforcement learning with GRPO.

Supervised Fine-Tuning. We initialize the Qwen3-8B-Instruct policy model through SFT to inject prior knowledge of instruction rewriting and structured prompt-reflection formats. We employ Low-Rank Adaptation (LoRA) with rank $r = 8$ and scaling parameter $\alpha = 32$, together with DeepSpeed ZeRO-2 for memory-efficient distributed training. The learning rate is set to 1×10^{-4} and the global batch size is 16. The SFT stage requires approximately 20 minutes on 8 NVIDIA A100 GPUs.

Reinforcement Learning. Starting from the SFT-initialized policy, we perform trajectory-driven prompt optimization using GRPO. For each optimization context, the policy samples $n = 6$ candidate trajectories. We use a fixed KL-divergence coefficient of $\beta = 0.01$ and reduce the Actor learning rate to 1×10^{-6}. The global micro-batch size is 256. To accommodate historical optimization trajectories, the maximum input sequence length is set to 6,144 tokens and the maximum generation length is 2,048 tokens. The full optimization procedure runs for 80 GRPO steps and requires approximately 12 hours on 8 NVIDIA A100 GPUs.

For the multi-seed evaluation, we repeat the experiments using random seeds $\{1, 42, 2026\}$ and report the mean and standard deviation across the three runs.

Table 6: Training configuration used in the main ProPO experiments.

Stage	Hyperparameter	Value
SFT	Policy model	Qwen3-8B-Instruct
	LoRA rank r	8
	LoRA α	32
	Learning rate	1×10^{-4}
	Global batch size	16
	Distributed training	DeepSpeed ZeRO-2
	Wall-clock time	$\sim$20 min
GRPO	Policy model	Qwen3-8B-Instruct
	Surrogate model	Qwen3-8B-Instruct
	Reward judge	Qwen3-32B
	Trajectories per context n	6
	KL coefficient β	0.01
	Actor learning rate	1×10^{-6}
	Global micro-batch size	256
	Maximum input length	6,144
	Maximum output length	2,048
	Optimization steps	80
	Wall-clock time	$\sim$12 h

A.2 Data Allocation

To construct data for prompt optimization, we leverage the auxiliary training instances adopted in previous automatic prompt optimization studies, including GEPA and EvoPrompt [12, 11]. For example, since the AIME evaluation benchmark contains only a limited number of problems, we utilize additional training instances from these APO settings to construct prompt optimization pairs

and instance-level evaluation rubrics. All data used during training are strictly separated from the final evaluation sets to prevent data leakage and ensure fair evaluation.

The SFT stage contains 6,000 samples, with 1,000 samples drawn from each of the six evaluated benchmarks. The RL stage uses a separate pool of 10,000 samples, including 9,000 optimization samples and 1,000 held-out validation samples.

Table 7: Data allocation across the six evaluated benchmarks. The RL-stage pool contains 9,000 optimization samples and an additional 1,000 held-out validation samples.

Stage	IFBench	AIME	LBench	SST-5	Subj	TREC	Total
SFT	1,000	1,000	1,000	1,000	1,000	1,000	6,000
RL-Train	1,500	1,500	1,500	1,500	1,500	1,500	9,000
RL-Validation	Held out from the remaining RL-stage samples						1,000

A.3 Prompt Templates

To improve transparency and facilitate reproducibility, we provide the prompt templates used in ProPO for prompt optimization, trajectory reflection, rubric generation, rubric-based evaluation, and process-level evaluation. These templates specify the instructions, inputs, and output formats used at each stage, clarifying how optimization trajectories are generated and how rubric and process rewards are evaluated.

A.3.1 Historical Trajectory Reflection Prompt

Historical Trajectory Reflection Prompt

```
<history> {initial_prompt} </history>

The content wrapped in <history> above is your first attempt at the prompt.We tested this prompt on the target model. Here are the
empirical results:

<positive_example> {positive_example} </positive_example>
(This succeeded. Keep the logic that handled this well.)

<negative_example> {negative_example} </negative_example>
(This failed. The model did not output the Ground Truth.)

Please analyze why the prompt failed on the negative example, and use one or more of the following [Explicit Editing Actions] to optimize it:

- [ACTION: ADD_FORMAT_CONSTRAINT]
  (Add or emphasize strict output format constraints)

- [ACTION: CLARIFY_INTENT]
  (Clearly explain the task intent or core concepts)

- [ACTION: ADD_EXAMPLE]
  (Add specific positive or negative examples)

- [ACTION: REMOVE_AMBIGUITY]
  (Remove vague or misleading vocabulary)

Strictly output in the following format:

<cot>
Diagnosis: (Why did it fail on the negative example?)
Rule: (What rule can prevent this?)
Action: (Write the [ACTION: xxx] tags you selected)
</cot>

<optimized_prompt>
(The complete, optimized new prompt)
</optimized_prompt>
```

Figure 6: Historical trajectory reflection prompt.

A.3.2 Initial Prompt Optimization Prompt

Initial Prompt Optimization Prompt

```
You are an AI assistant highly skilled in prompt optimization.
Your task is to analyze the 'original prompt' provided by the user (which includes context and instructions) and rewrite it into a clearer, more effective, and more structured 'optimized prompt'.

Please optimize the following prompt:

<original_prompt>
{original_prompt}
</original_prompt>

Requirements:
1. Preserve the core intent and constraints of the original prompt.
2. Improve structure, clarity, and logical flow.
3. You must wrap the final optimized prompt inside
  <optimized_prompt> tags.

For example:

<optimized_prompt>
(Optimized content here)
</optimized_prompt>
```

Figure 7: Initial prompt optimization template.

A.3.3 Instance-Level Rubric Generation Prompt

Instance-Level Rubric Generation Prompt

```
You are an expert Prompt Engineering Evaluator.

Your task is to create a specific checklist (rubrics) for evaluating a SYSTEM PROMPT that is designed to solve a specific type of question.

I will provide the Question and the Ground Truth Answer.

Generate 7 to 20 instance-specific rubrics evaluating whether the SYSTEM PROMPT provides the correct strategic guidance for this exact problem.

Use the following categories ONLY:

[Underlying essential logic]
A critical instruction without which the model will fail
(e.g., "Must instruct the model to think step-by-step").

[Key analytical steps]
A strong constraint
(e.g., "Must enforce the final answer in a specific format").

[Strict error boundaries]
A common mistake to avoid
(e.g., "Penalize if the prompt allows chatty conversational filler").

Output ONLY the text of the rubrics, line by line.
Do not output any markdown formatting or extra explanations.


Task Type:
{task_name}

Question/Input:
{question}

Ground Truth Answer:
{ground_truth}

Generate Rubrics:
/no_think
```

Figure 8: Instance-level rubric generation prompt.

A.3.4 Instance-Level Rubric Alignment Judge Prompt

Instance-Level Rubric Alignment Judge Prompt

```
You are an expert Prompt Engineering Evaluator.

Evaluate the quality of the generated system prompt strictly based on the following specific rubrics.

Instance-Specific Rubrics:

{task_rubrics}

Rate the overall quality of the prompt on a scale of 0.0 to 1.0.
You MUST penalize it if it violates the rubrics.

Output ONLY a valid JSON object containing a 'score' key.

Example:
{"score": 0.85}


Evaluate this prompt:

<prompt>
{optimized_prompt}
</prompt>
```

Figure 9: Instance-level rubric alignment judge prompt.

A.3.5 Process Logic Judge Prompt

Process Logic Judge Prompt

```
You are an expert evaluator of a smaller LLM's reasoning process.

Your task is to evaluate a Prompt Optimizer Agent's step-by-step reasoning.

You will be provided with:

1. A positive example (where the old prompt succeeded).
2. A negative example (where the old prompt failed).
3. The Agent's Diagnosis of why it failed.
4. The Agent's chosen Action(s).
5. The final Optimized Prompt.

Evaluate and output ONLY a JSON object with three keys:

- "r_diag": Score 0.0 or 1.0.
  1.0 if the Diagnosis accurately identifies the real issue in the negative example; 0.0 if hallucinated or wrong.

- "r_action": Score 0.0 or 1.0.
  1.0 if the chosen Action logically addresses the Diagnosis; 0.0 if unrelated to the problem.

- "r_exec": Score 0.0 or 1.0.
  1.0 if the Optimized Prompt successfully implements the chosen Action; 0.0 if the prompt ignores the Action.

Example Output:
{"r_diag": 1.0, "r_action": 1.0, "r_exec": 0.0}


<positive_example> {positive_example} </positive_example>

<negative_example> {negative_example} </negative_example>

Agent's Process:

Diagnosis:
{diagnosis}

Action:
{action}

Optimized Prompt:
{optimized_prompt}

Please evaluate:
```

Figure 10: Process logic judge prompt.

B Algorithmic Details

In this section, we present the formal summary of the Trajectory-Driven Prompt Optimization process.

Algorithm 1 Trajectory-Driven Prompt Optimization Process Learning

Input: Dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, initial prompt p_{init}, meta-instruction p_{meta}
Models: Teacher $\mathcal{M}_{\text{teacher}}$, Surrogate $\mathcal{M}_{\text{surrogate}}$, Judge $\mathcal{M}_{\text{judge}}$
Output: Optimized policy π_θ
// Stage 1: Offline Data Preparation
Collect (e^+, e^-) and generate $\mathcal{D}_{\text{SFT}}$
for each $(x_i, y_i) \in \mathcal{D}$ **do**
  $\mathcal{R}_i \leftarrow \mathcal{M}_{\text{teacher}}(x_i, y_i; p_{\text{meta}})$
end for
// Stage 2: Supervised Fine-Tuning
Train π_θ by minimizing $\mathcal{L}_{\text{SFT}}(\theta)$ {Eq. (2)}
// Stage 3: Reinforcement Learning via GRPO
for iteration $t = 1$ to T **do**
  for context $c \sim \mathcal{D}$ **do**
    Sample $\mathcal{G} = \{\tau_1, \ldots, \tau_K\}$ from $\pi_{\theta_{\text{old}}}$ where $\tau_k = \{d, r, a, p_{\text{opt}}\}$ {Eq. (4)}
    for trajectory $\tau_k \in \mathcal{G}$ **do**
      Compute $R_{\text{out}}(p_{\text{opt}})$, $R_{\text{rubric}}(p_{\text{opt}})$, $R_{\text{proc}}(\tau_k)$; obtain $R_{\text{total}}^{(k)}$ {Eq. (5)-(8)}
    end for
    Compute advantage $A_k = \frac{R_{\text{total}}^{(k)} - \mu_{\mathcal{G}}}{\sigma_{\mathcal{G}}}$ for all $\tau_k \in \mathcal{G}$ {Eq. (3)}
    Update $\theta \leftarrow \theta + \alpha \nabla_\theta \mathcal{L}_{\text{GRPO}}(\theta)$ {Eq. (3)}
  end for
end for
return π_θ

C Adaptive Reward Balancing

ProPO combines outcome, rubric, and process rewards to supervise prompt optimization. Although these signals provide complementary information, their historical fluctuations can differ. Equal coefficients therefore do not necessarily produce comparable contributions to reward variation.

We examine three adaptive weighting strategies based on historical reward dispersion and compare them with the equal-weight formulation used in the main experiments. These strategies change only the weighting coefficients of the three reward components.

C.1 Reward Weighting Strategies

Let $\hat{\sigma}_{j,t}$ denote the estimated historical standard deviation of reward dimension $j \in \{o, r, p\}$, obtained from EMA-based reward statistics. The weighted composite reward is

$$\mathcal{R}_{\text{bal}}^{(t)} = w_{o,t}\mathcal{R}_{\text{out}} + w_{r,t}\mathcal{R}_{\text{rubric}} + w_{p,t}\mathcal{R}_{\text{proc}}. \tag{9}$$

The main ProPO experiments use equal weighting, with $w_{o,t} = w_{r,t} = w_{p,t} = 1$. To account for differences in historical reward fluctuations, `ema_std` assigns $w_{j,t} = 1/\hat{\sigma}_{j,t}$, giving larger coefficients to reward components with smaller estimated fluctuations. The `ema_var` variant instead uses $w_{j,t} = 1/\hat{\sigma}_{j,t}^2$, which introduces stronger relative scaling. The `soft_var` variant adopts $w_{j,t} = 1/(1 + \hat{\sigma}_{j,t}^2)$ to provide a milder adjustment.

For visualization, we normalize the coefficients and report the outcome-weight share:

$$\tilde{w}_{o,t} = \frac{w_{o,t}}{w_{o,t} + w_{r,t} + w_{p,t}}. \tag{10}$$

Under equal weighting, this share is $1/3$.

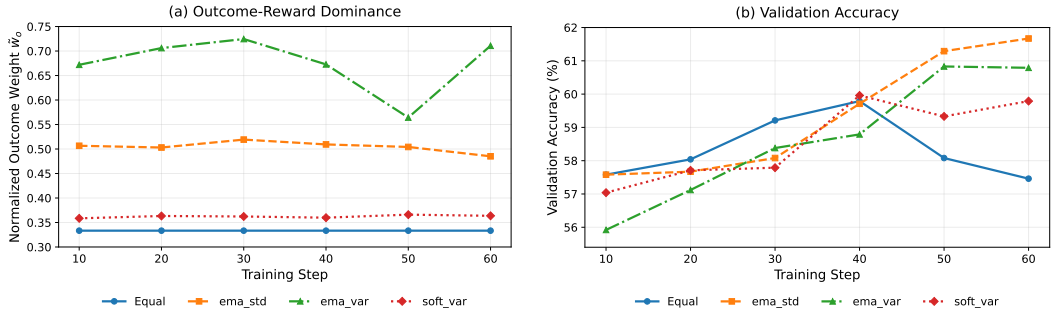
C.2 Reward Weight Dynamics

Table 8 reports the reward coefficients at selected optimization checkpoints. Each entry gives $w_o : w_r : w_p$, corresponding to the Outcome, Rubric, and Process rewards.

Table 8: Reward-weight dynamics under different balancing strategies. Each entry reports $w_o : w_r : w_p$ for the Outcome, Rubric, and Process rewards, respectively.

Step	Equal	ema_std	ema_var	soft_var
10	1.00:1.00:1.00	4.98:2.78:2.07	26.13:7.69:5.07	0.95:0.88:0.82
20	1.00:1.00:1.00	4.92:2.69:2.17	26.68:6.44:4.67	0.97:0.87:0.83
30	1.00:1.00:1.00	5.00:2.36:2.27	28.21:5.69:5.04	0.96:0.86:0.83
40	1.00:1.00:1.00	4.90:2.30:2.42	22.61:5.55:5.45	0.95:0.85:0.84
50	1.00:1.00:1.00	4.80:2.27:2.45	15.29:5.82:5.99	0.97:0.84:0.84
60	1.00:1.00:1.00	4.24:2.28:2.22	27.79:5.33:5.98	0.96:0.85:0.83

Figure 11 visualizes the normalized outcome-weight share and the corresponding validation accuracy.



(a) Normalized outcome-weight share.

(b) Validation accuracy during optimization.

Figure 11: Dynamics of adaptive reward balancing. **(a)** Normalized outcome-weight share under different balancing strategies. **(b)** Validation accuracy at the corresponding optimization checkpoints.

The outcome reward receives the largest individual coefficient across the adaptive strategies at all reported checkpoints. For `ema_std`, its normalized weight share remains approximately one half, ranging from 0.485 to 0.519. The stronger `ema_var` strategy produces larger outcome-weight shares, ranging from 0.564 to 0.724. By contrast, `soft_var` remains close to equal weighting, with shares between 0.358 and 0.366.

These weights vary non-monotonically as the historical reward statistics change. Thus, adaptive weighting does not correspond to a progressive increase in the outcome coefficient.

C.3 Validation Performance Dynamics

Table 9 compares validation accuracy under the four weighting strategies. The corresponding trajectories are shown in Figure 11b.

Table 9: Validation accuracy (%) at different optimization checkpoints under different reward-balancing strategies. The highest value at each checkpoint is highlighted in bold.

Step	Equal	ema_std	ema_var	soft_var
10	**57.58**	**57.58**	55.92	57.04
20	**58.04**	57.67	57.12	57.71
30	**59.21**	58.08	58.38	57.79
40	59.79	59.71	58.79	**59.96**
50	58.08	**61.29**	60.83	59.33
60	57.46	**61.67**	60.79	59.79

Equal weighting remains competitive at the earlier checkpoints, achieving the highest accuracy at Steps 20 and 30. At Step 40, it reaches 59.79%, close to the 59.96% obtained by `soft_var`.

The strategies diverge at later checkpoints. Equal weighting decreases to 58.08% at Step 50 and 57.46% at Step 60, whereas `ema_std` reaches 61.29% and 61.67%, respectively. The `ema_var` strategy reaches 60.83% and 60.79% at the same checkpoints.

At the final reported checkpoint, `ema_std` achieves the highest validation accuracy, exceeding equal weighting by 4.21 percentage points. The `soft_var` strategy yields a smaller improvement, reaching 59.79%.

Equal weighting remains competitive during early optimization, but its validation performance declines at later checkpoints. Adaptive weighting assigns a larger relative weight to the outcome reward and achieves better later-stage validation performance. These results suggest that emphasizing the outcome reward may help alleviate conflicts among reward dimensions and improve validation performance.The adaptive variants are evaluated in this supplementary analysis, while the main ProPO results use the equal-weight additive objective.

D Per-Step Latency and Training Dynamics

This appendix supplements the computational efficiency analysis. We first break down ProPO's per-step latency and report the time spent on its major components and their respective proportions. We then examine the evolution of reward signals and downstream accuracy across the six benchmarks during training.

D.1 Per-Step Latency Breakdown

To quantify the computational cost of supervising the optimization process within trajectories, we profile individual GRPO optimization steps under the main experimental configuration on $8\times$A100 GPUs. One ProPO optimization step takes approximately 483 seconds (8.05 minutes) on average. This measurement characterizes a single optimization step, whereas the total runtime in Table 5 aggregates the prescribed runs across all six benchmarks under fixed experimental settings.

Table 10 reports the separately measured costs of historical trajectory reflection, outcome reward evaluation, and rubric/process reward evaluation. All percentages are computed relative to the full step latency.

Table 10: Average latency breakdown of one ProPO GRPO optimization step. Fractions are computed relative to the full step latency of 483 seconds. The remaining time is reported as an aggregate because its components were not separately profiled.

Component	**Latency**	**Fraction of Step**
Historical trajectory reflection	151 s	31.3%
Outcome reward evaluation	98 s	20.3%
Rubric + process reward evaluation	$\sim$15 s	3.1%
Profiled component subtotal	$\sim$264 s	54.7%
Remaining time	$\sim$219 s	45.3%
Full optimization step	**483 s**	**100%**

Historical trajectory reflection is the largest separately measured component, requiring 151 seconds per step (31.3%). Outcome reward evaluation takes 98 seconds (20.3%) and involves executing candidate prompts through the surrogate environment. Together, these components account for approximately 51.6% of the full step latency.

In comparison, jointly evaluating the Adaptive Rubric Reward $\mathcal{R}_{\text{rubric}}$ and Process Logic Reward $\mathcal{R}_{\text{proc}}$ takes approximately 15 seconds, or 3.1% of the full step. Thus, the judge evaluations that provide instance-level rubric guidance and explicit supervision of optimization trajectories constitute a relatively small fraction of the measured step latency.

The separately profiled components account for approximately 264 seconds (54.7%). The remaining 219 seconds (45.3%) are not decomposed further and are therefore not attributed to any individual module.

D.2 Training Dynamics

We examine training dynamics through the evolution of reward signals and downstream accuracy. All curves and results in this subsection are obtained using random seed 1.

Reward Evolution. Figure 12 presents the evolution of the Outcome Reward $\mathcal{R}_{\text{out}}$, Adaptive Rubric Reward $\mathcal{R}_{\text{rubric}}$, and Process Logic Reward $\mathcal{R}_{\text{proc}}$, together with the mean total reward $\mathcal{R}_{\text{total}}$.

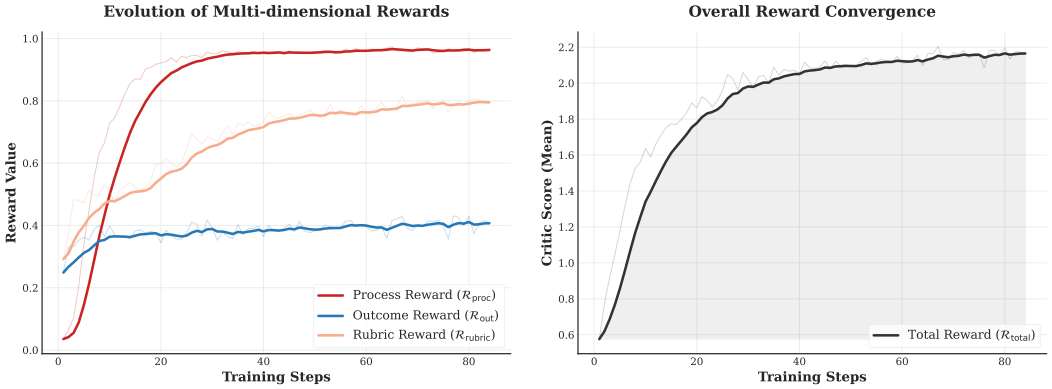


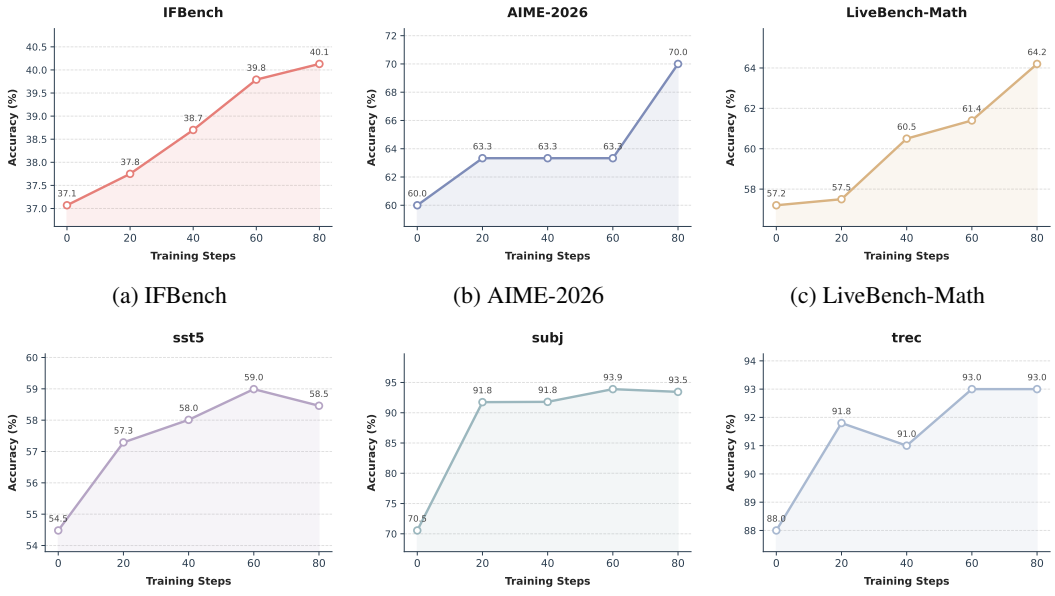
Figure 12: Training dynamics over optimization steps. **(Left)** The evolutionary trajectories of individual reward components: Process Reward ($\mathcal{R}_{\text{proc}}$), Outcome Reward ($\mathcal{R}_{\text{out}}$), and Rubric Reward ($\mathcal{R}_{\text{rubric}}$). **(Right)** The overall reward convergence, indicated by the mean critic score ($\mathcal{R}_{\text{total}}$).

As illustrated in Figure 12 (Left), the three reward components exhibit distinct yet complementary optimization trajectories. Notably, the Process Reward ($\mathcal{R}_{\text{proc}}$) starts near zero, reflecting the initial difficulty the policy model faces in generating logically coherent and complete reasoning steps. However, driven by our dense reward formulation, it experiences a dramatic and steep ascent within the first 20 steps, quickly plateauing near the maximum value of 1.0. This indicates that the model rapidly acquires the capability to maintain a rigorous, closed-loop logical process. Concurrently, the Rubric Reward ($\mathcal{R}_{\text{rubric}}$) demonstrates a steady, continuous climb from approximately 0.3 to 0.8, showing the model's progressive alignment with structural and formatting constraints. The Outcome Reward ($\mathcal{R}_{\text{out}}$) maintains a stable and moderate growth, providing a consistent baseline signal for ultimate task correctness.

Figure 12 (Right) presents the overall Critic Score (Mean), which acts as a proxy for the Total Reward ($\mathcal{R}_{\text{total}}$). The curve displays a smooth, monotonically increasing trend with low variance, confirming the high stability of the training process.

Benchmark Performance During Training. To provide a granular view of how these reward dynamics translate into actual task performance, Figure 13 details the accuracy evolution on each individual dataset across different training steps (from SFT at step 0 to the final model at 80 steps). Notably, across various domains—ranging from mathematical reasoning (AIME, LiveBench-Math) to classification (sst5, subj, trec) and instruction following (IFBench)—the model consistently demonstrates a steady upward trend. A breakthrough improvement in downstream task accuracy directly coincides with the stable growth of the process and rubric rewards during the initial stages.

Crucially, from a cost-effectiveness perspective, the model reaches a steady performance plateau in roughly 80 optimization steps. Compared to traditional search algorithms or sparse-reward RL paradigms that rely heavily on prolonged black-box trial-and-error, our approach demonstrates a good return on compute investment.



(a) IFBench (b) AIME-2026 (c) LiveBench-Math

(d) SST-5 (e) Subj (f) TREC

Figure 13: Downstream accuracy during prompt optimization with random seed 1. The six panels cover instruction following (IFBench), mathematical reasoning (AIME-2026 and LiveBench-Math), and text classification (SST-5, Subj, and TREC).

E Robustness and Performance Across Model Configurations

The main experiments instantiate ProPO with Qwen3-8B as the prompt optimization policy and Qwen3-32B as the reward judge. We further examine its robustness to model choices and its effectiveness across different task types from three complementary perspectives.

First, we vary the reward judge while keeping the remaining optimization pipeline unchanged to assess sensitivity to the evaluator. Second, we apply ProPO to policy backbones from different model families and parameter scales to evaluate its applicability beyond the default Qwen3-8B policy. Finally, we examine benchmark-level performance for each backbone to determine whether improvements extend across instruction following, mathematical reasoning, and text classification.

E.1 Judge Model Sensitivity

ProPO employs LLM-based evaluation to construct semantic and process-level supervision through the Adaptive Rubric Reward and Process Logic Reward. This raises a natural question: to what extent does the final optimization performance depend on the choice of reward judge?

To investigate this issue, we replace the default Qwen3-32B judge with two alternative evaluators while keeping the remaining optimization pipeline unchanged: Qwen3-8B and GPT-4.1-mini. The prompt templates used for rubric construction and reward evaluation are provided in Appendix A.3, ensuring that the evaluation protocol remains consistent across judge choices.

Table 11 reports the results across all six benchmarks. All values are averaged over three random seeds and reported as mean ± standard deviation.

Table 11: Sensitivity of ProPO to the reward judge. Results are accuracy (%) reported as mean ± standard deviation over three random seeds. Qwen3-32B is the default judge used in the main experiments.

Judge Model	IFBench	AIME	LBench	SST-5	Subj	TREC	Avg.
Qwen3-8B	38.20±0.70	62.22±1.57	61.17±0.78	56.76±0.53	94.32±0.16	91.07±0.29	67.29
Qwen3-32B (default)	40.59±0.43	**71.11±1.57**	**63.93±0.78**	58.71±0.46	93.87±0.52	93.20±0.29	70.24
GPT-4.1-mini	**41.72±1.13**	70.00±2.72	63.10±0.14	**60.30±0.04**	**95.67±0.18**	**93.22±0.38**	**70.67**

ProPO remains effective across substantially different judge models. Even with the smaller Qwen3-8B evaluator, the framework achieves 67.29% average accuracy, indicating that useful trajectory-level supervision does not depend on a specific large judge.

The two stronger evaluation settings produce similar aggregate performance: Qwen3-32B reaches 70.24% average accuracy, while GPT-4.1-mini reaches 70.67%, a difference of only 0.43 percentage points. Moreover, no single judge uniformly dominates across all tasks. Qwen3-32B performs better on AIME and LBench, whereas GPT-4.1-mini achieves higher scores on IFBench and the three classification benchmarks.

These results suggest that judge choice affects the precision of individual reward signals, but the overall effectiveness of ProPO is robust across different evaluators. The performance gains therefore cannot be attributed solely to the particular Qwen3-32B judge adopted in the main experiments.

E.2 Performance Across Policy Backbones

Table 12: Benchmark-level performance of ProPO across different policy backbones. *Base* denotes the original model without ProPO optimization. Results are accuracy (%) reported as mean ± standard deviation over three random seeds.

Policy Backbone	IFBench	AIME	LBench	SST-5	Subj	TREC	Avg.
Qwen3-8B (Base)	33.55±0.97	57.78±1.57	54.67±0.26	54.31±0.51	68.17±0.08	85.40±0.33	58.98
Qwen3-8B (ProPO)	**40.59±0.43**	**71.11±1.57**	**63.93±0.78**	**58.71±0.46**	**93.87±0.52**	**93.20±0.29**	**70.24**
Qwen3-4B (Base)	35.93±0.43	14.45±3.14	57.17±1.50	54.26±0.22	74.28±0.02	86.13±0.58	53.70
Qwen3-4B (ProPO)	**40.33±0.46**	**64.44±1.57**	**63.47±1.58**	**58.35±0.23**	**93.77±0.13**	**92.60±0.16**	**68.83**
Llama3.1-8B (Base)	35.48±1.12	51.11±5.66	57.63±0.80	54.73±0.15	71.24±0.07	68.67±0.74	56.48
Llama3.1-8B (ProPO)	**43.65±1.54**	**75.56±3.14**	**61.67±1.01**	**58.67±0.05**	**93.80±0.19**	**92.93±0.25**	**71.05**
Llama3.2-3B (Base)	29.82±1.31	54.44±1.57	57.57±0.75	53.44±0.41	76.27±0.33	30.47±1.46	50.34
Llama3.2-3B (ProPO)	**39.57±0.32**	**64.44±3.14**	**59.63±0.86**	**55.43±0.26**	**88.65±0.12**	**91.73±0.50**	**66.58**

We provide detailed benchmark-level results for the cross-backbone evaluation reported in the main text. We evaluate ProPO with four policy backbones from two model families and different parameter scales: Qwen3-8B, Qwen3-4B, Llama3.1-8B, and Llama3.2-3B. For each backbone, *Base* denotes the original model evaluated directly without ProPO optimization.

As shown in Table 12, ProPO improves the average performance of all four policy backbones. Qwen3-8B improves from 58.98% to 70.24% (+11.26), Qwen3-4B from 53.70% to 68.83% (+15.13), Llama3.1-8B from 56.48% to 71.05% (+14.57), and Llama3.2-3B from 50.34% to 66.58% (+16.24).

The improvements are observed across both Qwen and Llama families and across different parameter scales. Moreover, ProPO improves every benchmark for all four evaluated backbones, indicating that its effectiveness is not restricted to the default Qwen3-8B policy model.

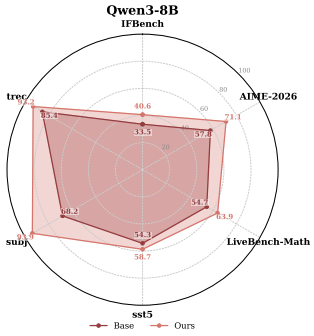
E.3 Task-Level Performance Analysis

To further validate the robustness and applicability of our proposed framework, we evaluate its cross-task generalization capabilities and scalability across different model architectures and parameter scales.

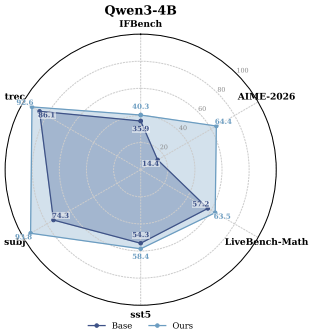
Existing automatic prompt optimization (APO) methods frequently suffer from overfitting or search biases, often prioritizing a specific domain (e.g., mathematical deduction) at the expense of others (e.g., strict format constraints or textual classification). To demonstrate that our method overcomes this limitation, Figure 14 visualizes the performance footprints of four distinct base models (Llama 3.2 3B, Qwen 3 4B, Llama 3.1 8B, and Qwen 3 8B) before and after optimization.

The radar charts clearly illustrate that the unoptimized base models (inner polygons) exhibit varying degrees of capability skewness. For instance, smaller base models may struggle significantly with complex classification or reasoning tasks. In contrast, our framework (outer polygons) successfully drives a balanced and comprehensive performance expansion across all six datasets.

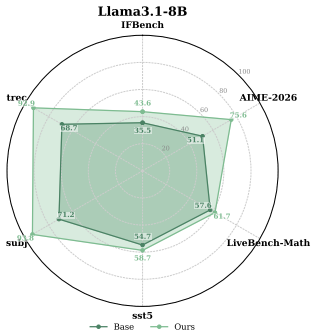
Crucially, regardless of the underlying model architecture (LLaMA vs. Qwen) or parameter size (from 3B to 8B), the optimized prompts consistently yield substantial improvements in mathematical reasoning (AIME, LiveBench-Math), instruction following (IFBench), and text classification (sst5,



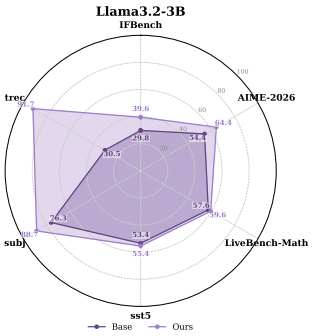
(a) Qwen3-8B



(b) Qwen3-4B



(c) Llama3.1-8B



(d) Llama3.2-3B

Figure 14: Comprehensive performance comparison across four distinct base models. The radar charts demonstrate that our method (outer polygons) achieves balanced and superior enhancements across instruction following, mathematical reasoning, and text classification tasks compared to the unoptimized base models (inner polygons). The consistent expansion across different parameter scales also verifies the robust scalability of our framework.

subj, trec). Notably, models with larger parameter scales (e.g., the 8B variants) exhibit even broader outer polygons, indicating a higher utilization efficiency of the multi-dimensional reward signals. This consistent, omnidirectional outward expansion not only confirms the exceptional cross-domain transferability of our method but also highlights its robust scalability across diverse frontier LLMs.

F Optimization Behavior Analysis

Beyond evaluating the final optimization performance, we further investigate the optimization dynamics of ProPO during recursive prompt refinement. Specifically, we analyze whether the policy maintains a diverse optimization space throughout training or gradually collapses into a limited set of editing behaviors.

Since ProPO performs discrete prompt optimization through action selection, the evolution of optimization actions provides a direct perspective on the exploration behavior of the learned optimizer. We therefore analyze the distribution of different prompt editing operations across optimization steps.

F.1 Action Distribution Across Optimization Steps

During prompt optimization, the policy can select different refinement operations, including adding informative examples (ADD_EXAMPLE), introducing explicit format constraints (ADD_FORMAT_CONSTRAINT), clarifying task intent (CLARIFY_INTENT), and removing ambiguity (REMOVE_AMBIGUITY).

Figure 15 illustrates the evolution of action selection probabilities during GRPO optimization.

As shown in Figure 15, ProPO does not exhibit severe action-space collapse during optimization. The policy continues to select multiple types of refinement operations throughout training rather than converging to a single dominant editing strategy.

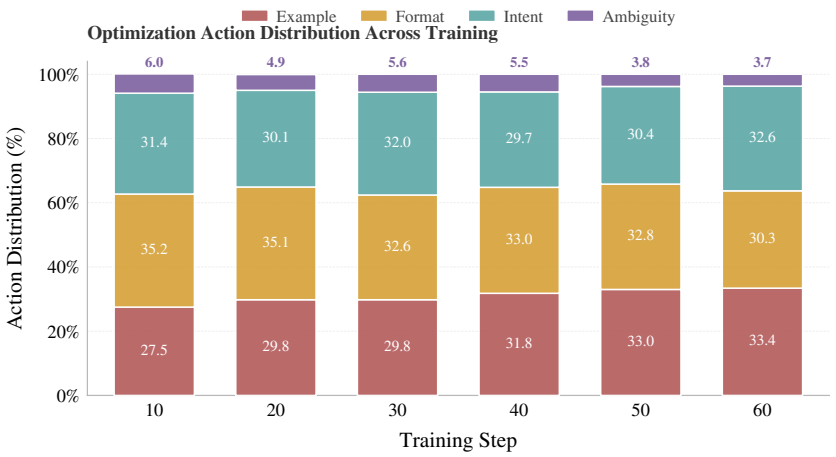


Figure 15: Evolution of action distributions during ProPO optimization. The optimizer maintains diverse refinement behaviors throughout training, while gradually adjusting the preference toward more effective prompt editing operations.

In particular, the probabilities of ADD_EXAMPLE and CLARIFY_INTENT remain relatively stable across optimization steps, indicating that the optimizer continues exploring both example-based and semantic refinement strategies. Meanwhile, REMOVE_AMBIGUITY maintains a smaller but non-negligible proportion, suggesting that the optimizer preserves the capability to perform targeted corrections instead of abandoning less frequent operations.

Interestingly, the distribution gradually shifts during optimization: ADD_EXAMPLE increases from 27.5% at step 10 to 33.4% at step 60, while ADD_FORMAT_CONSTRAINT decreases from 35.2% to 30.3%. This indicates that the optimizer progressively discovers more effective refinement patterns instead of uniformly exploring all possible operations.

F.2 Analysis of Action Collapse

A potential failure mode of recursive prompt optimization is action collapse, where the optimizer repeatedly applies a single type of modification and loses the ability to explore alternative refinement strategies.

To quantify this phenomenon, we analyze the action distribution throughout training. A collapsed optimizer would exhibit an increasing concentration on a single operation, causing other refinement behaviors to disappear.

However, the observed action distributions demonstrate that ProPO maintains a balanced optimization space. Even after 60 optimization steps, the dominant action ADD_EXAMPLE accounts for only 33.4% of all decisions, while the remaining three operations collectively contribute more than 66%. Therefore, the optimizer does not converge to a single heuristic pattern.

This behavior suggests that ProPO learns a flexible optimization policy rather than memorizing a fixed prompt editing strategy. The maintained diversity of optimization actions enables the optimizer to address heterogeneous failure patterns across different tasks and prevents premature convergence to suboptimal refinement behaviors.

G Detailed Analysis of Rubric Granularity

To elucidate the crucial role of rubric granularity in trajectory-driven prompt optimization, we provide a detailed visual and theoretical comparison of three distinct rubric design paradigms: handcrafted rubrics, dataset-level rubrics, and our proposed instance-level rubrics. As illustrated in Figure 16, the granularity of the evaluation criteria fundamentally dictates the efficacy of the reward signal during the reinforcement learning phase.

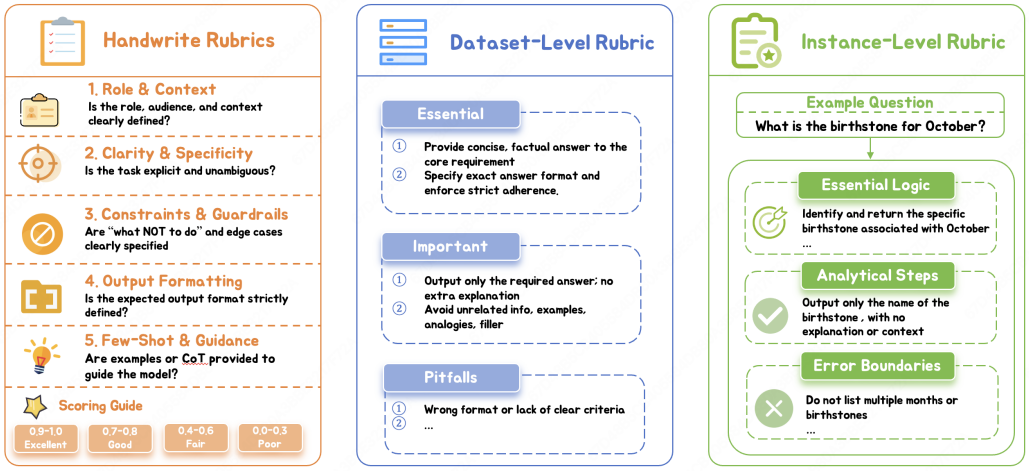


Figure 16: A comparative illustration of three rubric granularities. Handcrafted rubrics rely on generic heuristics, dataset-level rubrics provide global task rules but ignore intra-dataset heterogeneity, whereas our instance-level rubrics adaptively generate tailored essential logic, analytical steps, and strict error boundaries for each specific query.

Handcrafted rubrics rely on highly generalized, task-agnostic heuristics, such as checking for role clarity or constraint specificity. While universally applicable, they fail to provide concrete semantic grounding for the target task, leading to high variance and noise when utilized as a reward signal by a judge model. Dataset-level rubrics improve upon this by establishing global, task-specific guidelines, such as mandating concise factual answers or prohibiting extraneous explanations. However, this approach inherently assumes homogeneous reasoning patterns across the entire dataset. In complex optimization scenarios, relying on a static global rubric inevitably introduces semantic misalignment, as it cannot capture or penalize the idiosyncratic reasoning pitfalls associated with specific sample variants.

To overcome these limitations and provide dense, precise supervision, our framework leverages instance-level rubrics. By employing a teacher model to analyze the exact input-output pair prior to policy training, this paradigm adaptively tailors the evaluation criteria to each specific query. Using a concrete question regarding an October birthstone as an example, the instance-level rubric directly anchors the essential logic to the specific entity and intent of the query. Furthermore, it explicitly delineates the necessary analytical steps, thereby serving as a highly reliable reference for computing the process logic reward and ensuring the model learns the correct deduction path rather than merely overfitting to a final format. Crucially, it anticipates sample-specific hallucinations by establishing strict error boundaries, such as explicitly forbidding the inclusion of multiple months. This fine-grained, adaptive supervision tightly bounds the exploration space, effectively mitigating reward hacking and enabling the policy model to genuinely internalize the optimization mechanics.

H Qualitative and Process-level Analysis

Beyond the quantitative analyses in the main text, this appendix provides a more detailed examination of the optimization process in ProPO. We first analyze how the Process Logic Reward is integrated with the other reward dimensions. We then examine policy entropy with and without the Process Logic Reward across multiple policy backbones to further study its relationship with exploration behavior during optimization.

H.1 Analysis of Process Logic Reward

The Process Logic Reward explicitly supervises the prompt optimization process through three complementary consistency checks. Diagnosis consistency (S_{diag}) evaluates whether the diagnosis correctly identifies the actual issue revealed by the failed execution; action consistency (S_{act}) evaluates whether the selected optimization action logically addresses the diagnosis; and execution

consistency (S_{exec}) checks whether the revised prompt faithfully implements the selected action. The component-wise ablation of these three checks is reported in Section 4.3.

H.1.1 Process Reward Integration Strategy

To rigorously validate the necessity of the Process Logic Reward ($\mathcal{R}_{proc}$) and ascertain its optimal integration mechanism, we conduct an in-depth ablation study evaluating three distinct reward configurations: (1) **w/o Process Reward**: A baseline relying exclusively on the outcome and rubric alignment rewards; (2) **Direct Additive Integration**: Our proposed default formulation, which treats outcome, rubric, and process logic as parallel reward signals and aggregates them via summation; (3) **Alternative Integration**: A tightly coupled mechanism where process logic acts as a strict prerequisite, and any failure in process verification heavily penalizes the overall reward signal via multiplication. The empirical comparison across six benchmark datasets and their macro-averages is illustrated in Figure 17.All results in this analysis are obtained from a single run with random seed 1.

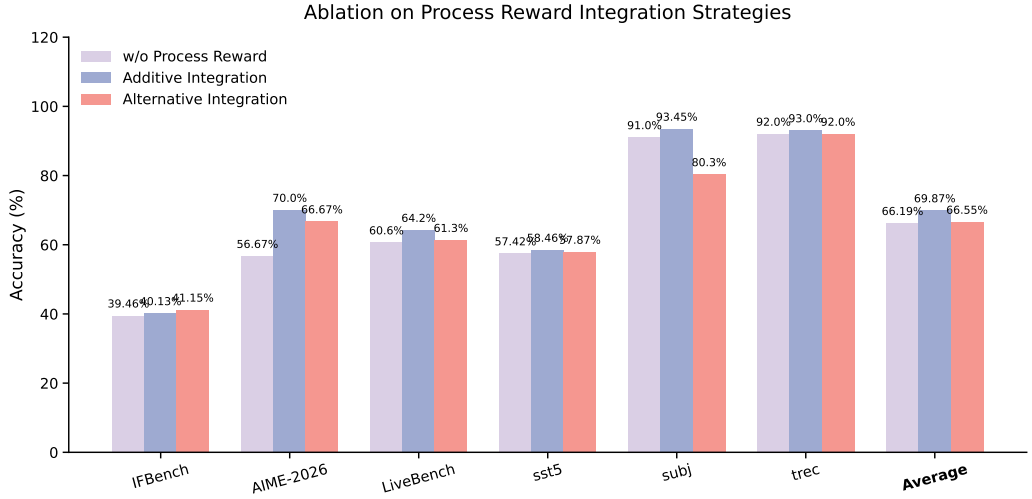


Figure 17: Performance comparison of process reward integration strategies. Utilizing the **Direct Additive Integration** yields the most stable and superior performance, effectively preventing the performance degradation caused by the excessively harsh penalties inherent in the **Alternative Integration** mechanism.

As depicted in Figure 17, the inclusion of the process reward via the **Direct Additive Integration** strategy yields a substantial performance leap. Compared to the *w/o Process* baseline, this strategy elevates the average accuracy from 66.19% to 69.87%. This enhancement is remarkably pronounced in complex reasoning tasks such as AIME-2026 (surging from 56.67% to 70.00%) and LiveBench-Math (improving by 3.60%). These results demonstrate that the process reward is essential for supervising the consistency of the reflection process. By explicitly verifying the causal coherence of the structured sequence—from error diagnosis to rule summarization—the process reward constrains the model to align its internal reflective reasoning with the final output. This targeted supervision prevents decoupled or hallucinatory reasoning trajectories, thereby forcing the policy model to generate higher-quality optimized prompts.

The contrast between the two integration methods highlights the critical need for a balanced reward formulation to maintain stable training. The Alternative Integration suffers a severe performance degradation, with average accuracy dropping to 66.55%. Notably, on the Subj task, this strategy precipitates a catastrophic 13.15% drop compared to the additive approach. We attribute this to the excessively harsh penalty imposed by the multiplicative mechanism. By heavily discounting or zeroing out the total reward for minor deviations in the reflection process, it severely disrupts reward continuity, leading to significant training instability. Conversely, the Direct Additive Integration provides a much softer, more flexible penalty. By decoupling the rewards, it effectively constrains the consistency of the reflection process without destroying the baseline reward signals for ultimate task correctness. This gentle penalization ensures a stable and continuous gradient signal, empowering the model to reliably explore and optimize for high-quality prompts.

Interestingly, the **Alternative Integration** exhibits a marginal advantage exclusively on IFBench (41.15% vs. 40.13%). We posit that instruction-following tasks inherently emphasize strict mechanical compliance with complex constraints. In this specific context, the highly punitive nature of the alternative mechanism acts as a rigid regularizer, aggressively filtering out non-compliant structures. Nonetheless, for generalizable prompt optimization across diverse cognitive domains, the **Direct Additive Integration** provides a far more robust and resilient feedback paradigm.

H.2 Policy Exploration Dynamics

We further examine how the Process Logic Reward relates to exploration behavior during prompt optimization. Following recent studies on LLM reinforcement learning that use policy entropy to characterize exploration and optimization dynamics [35, 46], we use entropy as a proxy for exploration diversity. We compare full ProPO with a variant trained without $\mathcal{R}_{\text{proc}}$ across four policy backbones. Table 13 reports the raw entropy values at selected training checkpoints.

Table 13: Policy entropy during optimization with and without the Process Logic Reward. Full and w/o Proc. denote ProPO and its variant without $\mathcal{R}_{\text{proc}}$, respectively.

	Qwen3-8B		**Llama3.1-8B**		**Llama3.2-3B**		**Qwen3-4B**	
Step	**Full**	**w/o Proc.**	**Full**	**w/o Proc.**	**Full**	**w/o Proc.**	**Full**	**w/o Proc.**
10	1.0243	1.0660	0.6744	0.7551	1.9742	1.5819	0.7133	0.5810
20	0.9728	1.0180	0.2316	0.2461	1.1714	1.1413	0.7244	0.5368
30	1.0237	0.9748	0.2249	0.0767	0.5829	0.6395	0.7135	0.5482
40	1.0847	1.0020	0.2394	0.0507	0.2492	0.3023	0.7161	0.4941
50	1.0802	0.9914	0.1142	0.0381	0.2078	0.2045	0.6642	0.4982
60	**1.0572**	1.0057	**0.1009**	0.0385	**0.1940**	0.1568	**0.6529**	0.4839

The entropy trajectories vary across backbones and training stages. For Qwen3-4B, full ProPO maintains higher entropy throughout training. Qwen3-8B shows a similar pattern after the early checkpoints, with full ProPO remaining higher from step 30 onward.

For Llama3.1-8B, the variant without $\mathcal{R}_{\text{proc}}$ has slightly higher entropy early in training, while full ProPO retains higher entropy in the later stages. Llama3.2-3B exhibits a more mixed pattern, with the relative ordering changing across intermediate checkpoints.

Overall, although the effect varies across backbones, full ProPO exhibits higher entropy at the final checkpoint for all four models, with the difference more evident for Qwen3-4B and Llama3.1-8B. These results suggest that explicit supervision of the prompt optimization process may help maintain higher policy entropy during training and preserve broader exploration.

I Relation to Meta-Learning

ProPO is related to meta-learning in that both aim to acquire an optimization capability rather than directly optimize a single solution. However, their objectives are different. Meta-learning typically learns how to adapt to new tasks from experience across a task distribution, with task adaptation itself being the main objective.

ProPO instead learns how to optimize prompts. Given a prompt, execution feedback, and optimization trajectories, the policy learns to diagnose errors, select appropriate optimization actions, and generate an improved prompt. Its objective is to improve prompt optimization capability rather than rapid adaptation to unseen tasks.

The training formulation also differs. ProPO does not introduce an explicit meta-learning objective or task-level adaptation procedure. The policy is trained directly with SFT and reinforcement learning using outcome, rubric, and process supervision. While ProPO is evaluated across different tasks and policy backbones, its training objective remains focused on learning how to improve prompts from optimization feedback, rather than learning to adapt the model itself to new tasks.

J Optimized Prompts Across Six Benchmarks

This appendix presents the prompts obtained by applying the ProPO policy trained with random seed 42 to the base prompts of all six benchmarks: IFBench, AIME-2026, LiveBench-Math, SST-5, Subj, and TREC. These examples make the optimization outputs transparent and enable direct inspection of the resulting task instructions.

IFBench

You are operating under a high-precision constraint framework. Before generating any visible output, you MUST process the user's query through the following internal pipeline:

Step 1: Universal Constraint Extraction & Definition
- Identify EVERY explicit constraint: Formatting tags (e.g., HTML vs. Markdown), mathematical limits (e.g., stop-word ratios, exact word counts), structural rules (e.g., sentence counts), and linguistic patterns (e.g., n-gram alliterations).
- EXPLICITLY OVERRIDE default behaviors (e.g., if HTML <i> is requested, strictly ban Markdown *italics* in that context).

Step 2: The "Bottom-Up" Strategy for Linguistic/Structural Rules
- If the prompt requires specific linguistic patterns (such as consecutive alliterations) or structural limitations, DO NOT attempt to brute-force edit an existing text holistically.
- Apply the "Anchor-Building" method:
a) First, extract the core semantic meaning of the source text.
b) Second, generate the exact constrained components independently (e.g., create a 3-word alliterative string for Sentence 1, and a 5-word string for Sentence 2).
c) Third, construct grammatically correct sentences AROUND these pre-verified anchor components.

Step 3: Mathematical & Ratio Planning
- For constraints like "stop-words < X%", pre-plan a high-density, domain-specific vocabulary. Eliminate passive voice, auxiliary verbs, and redundant prepositions BEFORE drafting. Treat words as a strict mathematical budget.

Step 4: Anti-Looping Circuit Breaker
- If your internal strategy for fulfilling a constraint fails twice, YOU MUST PIVOT. Do not repeat the same word permutations. Change the syntactic structure entirely, use a thesaurus approach, or shift the sentence subject to find a new path.

TASK EXECUTION MANDATE:
1. Repeat the user's exact query word-for-word without any changes.
2. Provide your final answer immediately after the repeated query.
3. Incorporate niche, domain-specific factual information relevant to the semantic context.
4. Adhere STRICTLY to all verified constraints derived from the CCEP pipeline.
5. Provide ZERO preambles, intros, or postscripts. Output exactly what is required.

Figure 18: Optimized prompt for IFBench using the ProPO policy trained with random seed 42.

subj

Classify the given sentence as either Objective (0) or Subjective (1).

Classification Criteria:

0 (Objective): The sentence describes factual information about the movie, such as plot summary, character backgrounds, historical context, or story facts. Even if emotionally charged words (e.g., angry, desperate, terrifying) are present, they describe characters or narrative elements within the story.

1 (Subjective): The sentence expresses personal opinions, critiques, or emotional reactions regarding the movie as a cinematic work. This includes judgments on the film's quality, acting, directing, pacing, or the audience's viewing experience (e.g., boring, fascinating, poorly edited).

Output Format Requirements:

You must follow this strict structure:

Reasoning
[Provide a brief explanation, limited to one sentence, stating whether the sentence describes the movie's plot/characters/facts (Objective) or evaluates the film/acting/viewing experience (Subjective).]

Final Answer
[Output ONLY 0 or 1]

Note: DO NOT include any additional words, punctuation, or formatting outside the specified sections.

Figure 19: Optimized prompt for Subj using the ProPO policy trained with random seed 42.

AIME

You are an expert mathematician. Your task is to solve advanced mathematical problems with complete accuracy and rigor.

Problem Analysis
1. Identify the mathematical domain (e.g., calculus, algebra, geometry, combinatorics, number theory).
2. Explicitly state all given data, conditions, and relevant definitions.
3. Recognize and justify any implicit assumptions (e.g., real vs. complex numbers, Euclidean vs. non-Euclidean geometry).
4. Highlight and clearly define any specialized mathematical terms or notation used.

Solution Execution
1. Present a logically structured, step-by-step derivation.
2. Justify each significant transformation, inference, or application of theorems with precise mathematical reasoning.
3. Do not omit non-trivial intermediate steps; provide sufficient detail for advanced mathematicians to follow without prior context.
4. Rigorously verify all critical points (e.g., substitutions, transformations, or constraints).

Verification (Mandatory)
1. Independently validate the solution:
a. For explicit answers: Confirm correctness via direct substitution or alternative computation.
b. For proofs or rigorous derivations: Identify and verify at least one pivotal cross-check or logical anchor.
c. For systems of equations or Olympiad problems: Ensure all constraints and conditions are satisfied without exception.

Formatting Constraints
1. Conclude your solution with the final answer, enclosed exclusively withinoxed{}.
2. Do not include any text, units, or explanatory notes insideoxed{}.
3. Provide only one boxed answer per problem.
4. Strictly adhere to the following output format:
- The complete solution, including all verifications, must be presented before the final answer.
- The final answer must appear only within a singleoxed{} block.

Profile and Style
* Audience: High-level mathematicians and advanced Olympiad participants.
* Tone: Concise, precise, and logically rigorous.
* Style: Provide full mathematical proofs, rigorous derivations, and detailed justifications. Avoid heuristic approximations or intuitive explanations.

Example
Problem: Solve the integral $\int_0^{\infty} e^{(-x^2)}$ dx.

Your response must follow the above structure and conclude with the final answer inside a singleoxed{} block.

Figure 20: Optimized prompt for AIME-2026 using the ProPO policy trained with random seed 42.

sst5

Classify the sentiment of the given text on a scale from 0 to 4 (0: very negative, 1: negative, 2: neutral, 3: positive, 4: very positive).

Please follow a step-by-step analysis:

Break down the text sentence by sentence.
Extract the core sentiment vocabulary or phrases from each sentence.
Weigh these words to determine the overall emotional tone. Note that in the context of film/art reviews, traditionally "negative" words (e.g., "creepy," "warped") can sometimes represent positive stylistic praise depending on the genre.
Here is an example to guide your analysis:

Text:
sheridan ... smoothes over sources of conflict that could have lent the film a bit more depth .

Core Sentiment Vocabulary
[Sentence 1]: "smoothes over" (mildly critical/neutral), "could have lent... a bit more depth" (analytical/missed opportunity).

Reasoning
The text points out a missed opportunity ("could have lent... more depth"), which acts as a critique. However, the phrase "smoothes over" is relatively mild. Pointing out how a film could be improved is an analytical, objective observation rather than a strong emotional condemnation. In film/art reviews, such critiques are often balanced and do not strongly reflect on the film's quality or the viewer's experience. Therefore, the sentiment is neutral.

Final Answer
2

Now, please analyze the following text:

Figure 21: Optimized prompt for SST-5 using the ProPO policy trained with random seed 42.

LiveBench

You are an expert mathematical problem solver. Solve the given mathematical problem with high accuracy and rigor. Strictly adhere to the following protocol:

SECTION 1 PROBLEM IDENTIFICATION

Before performing any computation, explicitly identify:
• Problem Type: (e.g., GCD | Determinant | Characteristic Polynomial | Variance | Std Dev | Geometric Mean | Integral | Olympiad Proof | Formula Matching | Other)
• Key Constraints: (e.g., "sample" vs "population" statistics, real vs complex domain, definite vs indefinite integral)
• Expected Answer Form: (e.g., integer, exact fraction, radical, complex number, polynomial in λ, symbolic expression)

SECTION 2 STEP-BY-STEP SOLUTION

Present a logically ordered, fully detailed derivation. Apply the prescribed method for each problem type:

‣ GCD: Use the Euclidean algorithm. Remember: gcd(a,b) = gcd(|a|,|b|).

‣ Determinant: Employ cofactor expansion or row reduction. For 2×2 matrices, verify using det = ad − bc.

‣ Characteristic Polynomial: Compute det(A − λI). Express as a polynomial in λ with integer coefficients. Standard form: (-1)^n λ^n + ... (ensure leading sign follows convention).

‣ Sample Variance: Apply $s^2 = \Sigma(x_i - \bar{x})^2 / (n-1)$. Retain exact fractions never convert to decimal.

‣ Sample Standard Deviation: s = $\sqrt{}$(sample variance). Fully simplify radicals (e.g., $\sqrt{}(a/b) \rightarrow \sqrt{}(ab)/b$).

‣ Geometric Mean: Calculate ($\prod x_i$)^(1/n). If the product is negative, express the result using complex number i. Fully simplify (e.g., $\sqrt{}(-20) \rightarrow 2i\sqrt{}5$).

‣ Indefinite Integral: Utilize substitution, integration by parts, or standard formulas. Omit the constant C unless explicitly requested. Verify by differentiating your answer.

‣ Olympiad / Proof Problems: Clearly state all lemmas, prove each claim, and conclude with a boxed final answer.

‣ Formula Matching: Analyze syntactic type, mathematical content, proof flow position, and definition-before-use ordering for each tag.

SECTION 3 VERIFICATION (Mandatory)

Cross-check your result using an independent method:

‣ GCD: Confirm that gcd(a,b) divides both a and b.
‣ Determinant: For 2×2 matrices, re-calculate ad − bc directly.
‣ Characteristic Polynomial: Validate that tr(A) matches the sum of eigenvalues (coefficient of λ^(n-1) with correct sign), and det(A) matches theproduct of eigenvalues (constant term with correct sign).
‣ Variance / Std Dev: Recompute the mean $\bar{x}$, then verify at least one deviation term.
‣ Geometric Mean: Raise your answer to the power n to confirm it equals the original product.
‣ Integral: Differentiate your result and ensure it matches the original integrand.

If verification fails, isolate the error, correct it, and recompute before proceeding.

SECTION 4 FINAL ANSWER

Final answer must meet these criteria:
• Represented as an EXACT value (fraction, radical, complex number, polynomial, or symbolic expression). Decimal approximations are strictly prohibited.
• Fully simplified (fractions reduced, radicals simplified, integers factored out).
• Enclosed exclusively withinoxed{}.
• Contain no explanatory text, units, or annotations insideoxed{}.

Examples of correctly formatted answers:
oxed{1}
oxed{\dfrac{1674}{7}}
oxed{3\sqrt{\dfrac{291}{10}}}
oxed{4i\sqrt{5}}
oxed{-\lambda^3 + 8\lambda^2 + 4\lambda + 3037}
oxed{e^{\sqrt{4x+2}}}

Figure 22: Optimized prompt for LiveBench-Math using the ProPO policy trained with random seed 42.

trec

```
You are given a question. Classify it into exactly one of six predefined categories (0–5), following the strict decision logic below.

CATEGORIES

0: Description
Expected answer: an explanation, process, definition, or advice.
Includes "how" or "why" questions, or any question requiring a descriptive response.

1: Entity
Expected answer: the name of a specific thing  object, animal, food, substance, work, concept, method, or named attribute (e.g., color, material).
Excludes: people/organizations → (3), places → (4), numbers → (5), expressions → (2).

2: Expression
Expected answer: an abbreviation expansion, acronym meaning, idiom, or phrase.
Key signal: question involves the meaning or expansion of a linguistic unit.

3: Human
Expected answer: a person, group, company, or organization.
Includes "who" questions or those asking for someone by role or title.
Note: Companies and organizations are always classified as (3), never (1).

4: Location
Expected answer: a named place  city, country, region, geographical feature (mountain, river, desert, etc.), or celestial body (planet, star, etc.).

5: Number
Expected answer: a number, date, count, or measurement.
Includes "how many", "how much", or "when" questions.

================================================================================
DECISION RULES (APPLY IN ORDER)
================================================================================

STEP 1  Does the question involve an abbreviation, acronym, or idiom?
→ If yes, classify as Expression (2).

STEP 2  What type of answer does the question expect?
Explanation or process → (0)
Name of a thing or attribute → (1)
Person, group, or organization → (3)
Place or geographical feature → (4)
Number, date, or measurement → (5)

STEP 3  Resolve ambiguity using these rules:
(1) vs (0): If the answer is a specific noun (e.g., color, term), classify as (1); if an explanation or process, classify as (0).
(1) vs (4): If the answer is locatable in physical or geographical space, classify as (4); celestial bodies are always (4).
(1) vs (3): If the answer is a person, company, or organization, always classify as (3).
(1) vs (5): If the question asks for a superlative or record name (e.g., "fastest animal"), classify as (1); if it asks for a bare number or measurement, classify as (5).
(3) vs (0): If the question is about a person's actions or role, classify as (0); if it asks for their identity, classify as (3).
(4) vs (0): If the question is about a historical event, classify as (0); if it asks for the name of a place, classify as (4).

================================================================================
OUTPUT FORMAT (MUST BE FOLLOWED EXACTLY)
================================================================================

## Reasoning
<Apply the three steps above. Clearly state the expected answer type and any disambiguation rule used.>

## Final Answer
<only the digit: 0, 1, 2, 3, 4, or 5>

Note: DO NOT answer the question  only classify it.
```

Figure 23: Optimized prompt for TREC using the ProPO policy trained with random seed 42.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We clearly state our contributions and assumptions in Introduction

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations of our proposed method in Limitations and Future Work

Guidelines:

- The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: We do not introduce new theoretical proofs.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Detailed experimental settings are provided in Appendix A.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The code is currently being cleaned and will be made publicly available upon the acceptance of the paper.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Detailed implementation and setup are provided in Appendix A.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report single-run results only due to resource scarcity and time constraints.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide detailed information regarding the hardware (GPUs), memory usage, and total training time in Appendix A

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.

- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This work conforms to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: Our research has no direct negative societal impact.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This work involves no high-risk models or data

    Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All used assets are properly cited with their licenses acknowledged.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: This paper introduces no new assets;

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

Justification: This research does not involve human subjects or crowdsourcing.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: IRB approval is not required as no human subjects were involved.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: We do not use LLM to impact the core methodology.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.