CPC-GRPO:Answer-Free Reinforcement Learning with Cross-Prompt Consensus Rewards

Anonymous ACL submission

Abstract

Reinforcement learning with verifiable rewards has improved reasoning in language models, but it typically relies on a ground-truth answer or an external verifier, which limits applicability and increases cost. We propose an answer-free training objective that derives rewards solely from the model's own probabilities by exploiting prompt paraphrases as multiple semantic views of the same intent. For each paraphrase set, we generate candidate responses, rescore each response under the other paraphrased prompts via teacher forcing, and define a cross-prompt consensus reward that favors responses supported across views rather than those that fit a single phrasing. We optimize this reward using a policy update with an all-pairs objective and advantage broadcasting across prompt–response pairs. The framework naturally supports prefix-level training, enabling a controllable cost–signal trade-off. Experiments on RobustAlpacaEval and out-of-domain reasoning benchmarks (OpenBookQA, AQuA, HumanEval) show consistent gains over the pre-trained baselines on LLaMA3.2-3B and Qwen3-4B, alongside analyses demonstrating reward–performance alignment and the importance of design choices such as excluding self-view scores and ensembling-based candidates. All experiment code is available at our GitHub[1].

1 Introduction

Reinforcement learning (RL) is now a standard post-training tool for large language models (LLMs), and is used to bias generation toward more desirable outputs (Ouyang et al., 2022). Recent work further reports that RL can yield substantial gains on reasoning-centric tasks (Yu et al., 2025; Mroueh, 2025). In particular, RLVR (Reinforcement Learning with Verifiable Rewards) shapes reasoning behavior using scalar rewards that are directly verifiable against a ground-truth answer (Lambert et al., 2025; Guo et al., 2025a). When combined with recent policy optimization methods such as GRPO (Zhihong Shao, 2024), DR.GRPO (Liu et al., 2025), and DAPO (Yu et al., 2025), RLVR has shown consistent improvements on mathematics and coding benchmarks.

Despite its clarity, RLVR has practical constraints. First, verification requires either ground-truth answers (e.g., exact match (EM)) or an external verifier (e.g., executor or grader), which makes it difficult to apply the same recipe to general instruction data where answer verification is not straightforward. Second, RLVR rewards are typically defined over a "reasoning $\rightarrow$ final answer" structure, so truncated responses cannot be evaluated. This pushes training toward long rollouts, while rollout generation and scoring can dominate RL cost in practice (Zhong et al., 2025; Hu et al., 2025; Yao et al., 2023).

We propose answer-free self-reward RL by leveraging paraphrase prompts of the same intent as multi-view observations. Given G paraphrase prompts, we generate responses and *rescore* each response under the other paraphrase conditions via teacher forcing to obtain cross-prompt scores. We then define a *consensus reward* as the multi-view average support, rather than the self-view score under a single prompt. To reduce prompt-specific overfitting, our default setting excludes the self-view term and applies group-wise normalization. Finally, we transform consensus rewards into group-relative advantages and perform an on-policy GRPO-style update, enabling answer-free policy optimization without ground-truth answers or external judges.

Because the reward is computed solely from model token probabilities, answer-free training requires neither labels nor verifiers. Moreover, the token-average form allows computing rewards on

[1] https://anonymous.4open.science/r/Cross-Prompt_Consensus_GRPO-2660

response prefixes, enabling explicit performance–cost trade-offs as a function of prefix length. We evaluate answer-free training on general instruction data and out-of-domain reasoning benchmarks, and analyze the performance–cost trade-off under different prefix lengths.

2 Related Work

2.1 Reinforcement Learning in LLM

RL-based post-training is widely used to adapt large language models by optimizing reward signals that reflect human preferences or task performance (Ouyang et al., 2022). RLHF is a representative approach, where a preference-based reward model is trained and PPO-style updates are applied (Schulman et al., 2017; Ziegler et al., 2019). More recently, group-based optimization methods such as GRPO update the policy using within-group relative signals from multiple responses to the same prompt, and have shown strong results on reasoning-oriented benchmarks (Zhihong Shao, 2024). In the RLVR setting, GRPO can be combined with verifiable rewards (e.g., exact match (EM) or execution-based pass@k), but this typically requires a ground-truth answer or an external verifier and often relies on sufficiently long rollouts to compute rewards reliably (Guo et al., 2025a; Yao et al., 2023). To reduce dependence on labeled answers, prior work has explored reward construction without direct supervision. LLM-as-a-Judge uses a strong external LLM to score or compare outputs (Zheng et al., 2023), at the cost of additional inference and with open concerns about bias and reproducibility. RLIF methods instead use internal model signals as intrinsic rewards. For example, RENT uses entropy- (or confidence-) based signals from the token distribution (Prabhudesai et al., 2025), and Intuitor replaces external rewards with self-certainty in a GRPO framework (Zhao et al., 2025). However, self-certainty proxies can be prompt- and task-dependent, and their alignment with response quality is not guaranteed.

In contrast, we construct a reward from paraphrase multi-view cross-prompt scoring, yielding a consensus signal that does not require ground-truth answers or external judges and does not rely solely on self-certainty.

2.2 Prompt Paraphrase Ensemble

Ensembling is a classical strategy for reducing variance and improving stability. For LLMs, prompt sensitivity implies that output quality can vary substantially across semantically equivalent phrasings (Zhao et al., 2021). This has motivated multi-prompt test-time methods that either aggregate token distributions during decoding or select a final output from candidates generated by a prompt bank. For example, M-Ped averages token distributions across prompts via inner-batch ensembling (Guo et al., 2025b), and multi-prompt MBR samples candidates from diverse prompts and selects outputs using an MBR criterion (Heineman et al., 2024). Bayesian Prompt Ensembles treat prompt variations as a source of uncertainty and aggregate output probabilities to estimate calibrated uncertainty for black-box LLMs (Tonolini et al., 2024).

Unlike prior work that primarily targets test-time quality or uncertainty estimation, we use paraphrase ensembles for *training-time* reward construction. We define a multi-view consensus reward via cross-prompt teacher-forcing rescoring, enabling answer-free policy optimization without ground-truth answers or external judges.

3 Methodology

In this section, we describe how to construct an *answer-free* reward using a set of paraphrase prompts and how to train with group-based policy optimization. The goal of our method is to (i) define a reward signal that is computable without ground-truth answer labels or external evaluators, (ii) avoid over-reliance on self-certainty under a single prompt, and (iii) concentrate probability mass on responses that are *consistently supported* across semantically equivalent multi-views.

To this end, we treat each paraphrase prompt as a different observation (view) of the same intent, and measure whether a particular response is not only plausible under one prompt phrasing but also explainable with high probability under other paraphrase conditions via *cross-prompt scoring*. Each response is scored by its average support (consensus) across multi-views, and this score is converted into a group-relative advantage. We further broadcast this advantage to all (prompt, response) pairs used in the update, so that each conditional distribution $\pi_\theta(\cdot \mid x^{(i)})$ is encouraged to place more probability mass on regions that are commonly supported across views (high-consensus regions) rather than on prompt-specific idiosyncrasies. Since training is on-policy, as updates proceed, sampled responses increasingly reflect the preferences of the current

policy, and we may expect dynamics in which cross-prompt consensus is gradually strengthened. Below, we formalize the proposed method step by step.

3.1 Rollout with Paraphrase Ensemble

Each training sample consists of a set of G semantically equivalent paraphrase prompts for a single intent, $\mathcal{X} = \{x^{(i)}\}_{i=1}^{G}$. In on-policy rollouts, we generate a response for each prompt from the previous policy $\pi_{\theta_{\text{old}}}$:

$$y^{(j)} \sim \pi_{\theta_{\text{old}}}(\cdot \mid x^{(j)}), \qquad j = 1, \ldots, G. \quad (1)$$

In addition, to provide a response that is *more stably agreed upon* across multi-views, we generate a decoding-level ensemble (DLE) response $y^{(K)}$ ($K = G + 1$). At token step t, we define the ensemble next-token distribution as the average of the prompt-conditional distributions:

$$\bar{\pi}_{\theta_{\text{old}}}(\cdot \mid y_{<t}^{(K)}) = \frac{1}{G} \sum_{i=1}^{G} \pi_{\theta_{\text{old}}}(\cdot \mid x^{(i)}, y_{<t}^{(K)}). \quad (2)$$

Actual generation is performed by selecting the most probable token under the averaged distribution (e.g., greedy decoding):

$$y_t^{(K)} = \arg\max_{v} \ \bar{\pi}_{\theta_{\text{old}}}(v \mid y_{<t}^{(K)}). \quad (3)$$

Finally, the set of responses used for training is $\mathcal{Y} = \{y^{(j)}\}_{j=1}^{K}$ with $K = G + 1$.

3.2 Cross-prompt Consensus Reward

For each response $y^{(j)} = (y_1^{(j)}, \ldots, y_{T_j}^{(j)})$, we measure support under other paraphrase conditions via cross-prompt scores. Concretely, we rescore $y^{(j)}$ under prompt $x^{(i)}$ via teacher forcing and define the token-mean probability score for pair (i, j) as:

$$s_{i,j} = \frac{1}{T_j} \sum_{t=1}^{T_j} \pi_{\theta_{\text{old}}}\left(y_t^{(j)} \mid x^{(i)}, y_{<t}^{(j)}\right). \quad (4)$$

This yields $G \times K$ cross-scores, which are computed solely from the model's token probabilities without ground-truth labels or external evaluators. We discuss score variants that mitigate the influence of early tokens in Sec. 5.6.

We then define the consensus reward of response $y^{(j)}$ as its average support (consensus) across multi-views. In the default setting, for responses with $j \leq G$, we exclude the self-view term ($i = j$) and take the average:

$$r_j = \frac{1}{G-1} \sum_{i=1,\, i \neq j}^{G} s_{i,j}, \qquad j = 1, \ldots, G. \quad (5)$$

This design is motivated by the fact that the self-view score is not, in a strict sense, a consensus signal; rather, it can act as a *prompt-specific fit* or self-certainty for a particular phrasing, and excluding it focuses the reward on cross-view consensus. Since the ensemble response $y^{(K)}$ has no specific self-view, we define it as the average over all views:

$$r_K = \frac{1}{G} \sum_{i=1}^{G} s_{i,K}, \qquad K = G + 1. \quad (6)$$

Thus, a high r_j reflects *multi-view consensus* rather than self-certainty under a single prompt, and training encourages allocating more probability mass to responses with higher consensus.

3.3 All-pair Policy Optimization

We follow GRPO's group-based policy optimization framework, but perform an all-pair GRPO-style optimization that utilizes *all* (prompt, response) combinations involved in computing the consensus reward for learning. The key is to (i) construct a group-relative signal from response-level consensus rewards and (ii) share this signal across (prompt, response) pairs that participate in cross-prompt rescoring, thereby strengthening high-consensus responses simultaneously under every view condition.

Group-relative advantage. Given K rewards $\{r_j\}_{j=1}^{K}$ for a sample, we define the group mean reward and advantages as:

$$\bar{r} = \frac{1}{K} \sum_{j=1}^{K} r_j, \qquad A_j = r_j - \bar{r}. \quad (7)$$

For training stability, we normalize the advantages within each sample by their standard deviation:

$$\tilde{A}_j = \frac{A_j}{\sigma(A) + \varepsilon}, \quad (8)$$

where $\sigma(A)$ is the standard deviation of $\{A_j\}_{j=1}^{K}$ and ε is a small constant for numerical stability.

Algorithm 1 Paraphrase-Consensus GRPO

Require: Paraphrase prompts $\mathcal{X} = \{x^{(i)}\}_{i=1}^{G}$, policy π_θ, prefix cap L, clip ϵ (optional DLE)

for each policy update **do**
  $\theta_{\text{old}} \leftarrow \theta$
  for each sample $\mathcal{X}$ in a minibatch **do**
    Rollout: sample $y^{(j)} \sim \pi_{\theta_{\text{old}}}(\cdot \mid x^{(j)})$ for $j = 1..G$ (Eq. (1))
    if DLE enabled **then**
      generate $y^{(K)}$ by decoding-level ensemble (Eq. (2)–(3)); set $K \leftarrow G + 1$
    else
      set $K \leftarrow G$
    end if
    set $\tilde{T}_j \leftarrow \min(T_j, L)$ for all j (prefix-level training)
    Cross-score: compute $s_{i,j}$ for all $i \le G, j \le K$ by teacher forcing up to $\tilde{T}_j$ (Eq. (4))
    Reward: compute consensus rewards $\{r_j\}_{j=1}^{K}$ (Eq. (5)–(6))
    Advantage: compute normalized $\{\tilde{A}_j\}$ (Eq. (7)–(8))
    Broadcast: set $\tilde{A}_{i,j} \leftarrow \tilde{A}_j$ for valid pairs $\mathcal{P}$ (Eq. (11), (9))
    accumulate clipped loss over $(i,j) \in \mathcal{P}$, $t \le \tilde{T}_j$ (Eq. (12))
  end for
  update θ by minimizing the accumulated loss
end for

Advantage broadcast. Although rewards/advantages are defined at the response level, policy optimization is carried out over the (prompt, response) pairs used to compute cross-prompt scores. We therefore broadcast the normalized advantage for response $y^{(j)}$ to every valid (prompt, response) pair:

$$\tilde{A}_{i,j} := \tilde{A}_j. \tag{9}$$

This broadcast strengthens high-consensus responses under each view condition simultaneously, and consequently encourages $\pi_\theta(\cdot \mid x^{(i)})$ to place more probability mass on regions that are commonly supported across views (high-consensus regions) rather than on prompt-specific idiosyncrasies.

PPO clipped objective. For each (prompt, response) pair $(x^{(i)}, y^{(j)})$, we define the token-level probability ratio at step t as:

$$\rho_{i,j,t}(\theta) = \frac{\pi_\theta\left(y_t^{(j)} \mid x^{(i)}, y_{<t}^{(j)}\right)}{\pi_{\theta_{\text{old}}}\left(y_t^{(j)} \mid x^{(i)}, y_{<t}^{(j)}\right)}. \tag{10}$$

To reflect the exclusion of self-views, we define the set of valid (prompt, response) pairs as:

$$\mathcal{P} = \{(i,j) \mid 1 \le i \le G,\ 1 \le j \le K,\ i \ne j\}. \tag{11}$$

In the default setting ($K = G + 1$), G self-view pairs are excluded for $j \le G$, whereas the ensemble response ($j = K$) has no excluded self-view, yielding

$$|\mathcal{P}| = G(G-1) + G = G^2.$$

Finally, the policy update can be written as minimizing the objective obtained by averaging PPO's clipped surrogate over all pairs:

$$\mathcal{L}(\theta) = -\frac{1}{|\mathcal{P}|} \sum_{(i,j)\in\mathcal{P}} \frac{1}{T_j} \sum_{t=1}^{T_j} \min\Big(\rho_{i,j,t}(\theta)\tilde{A}_{i,j},\ \text{clip}(\rho_{i,j,t}(\theta), 1-\epsilon, 1+\epsilon)\tilde{A}_{i,j}\Big), \tag{12}$$

where ϵ is the PPO clipping coefficient.

3.4 Prefix-level Reward and Computational Trade-off

Because our reward is defined as a cumulative (average) function of token probabilities, it can be computed from response prefixes rather than requiring complete responses. Unlike RLVR, which must extract and verify the final answer, this means that learning signals can be formed without generating responses to completion.

Let L denote the maximum prefix length, and define the used length of each response as $\tilde{T}_j = \min(T_j, L)$. Accordingly, rollouts are restricted to generating only up to $y^{(j)}_{\le \tilde{T}_j}$, and the inner sums in cross-prompt scoring (4) and the PPO loss (12) can likewise be performed only for $t = 1, \ldots, \tilde{T}_j$. Since the self-attention computation of Transformer-based LLMs increases approximately as $O(L^2)$ with sequence length (Vaswani et al., 2023; Dao, 2023), controlling L allows us to directly manage the performance–cost trade-off. Algorithm 1 summarizes the overall training procedure of the proposed method.

Base Model	Method	Para.	Train	RAE (In-domain)		Reasoning (Out-of-domain)		
				Trained	Un-trained	OBQA	AQuA	HumanEval
LLaMA3.2-3B	pre-trained	✗	–	21.42	19.25	58.66	52.36	60.06
	self-refine	✗	–	9.49	8.73	34.80	43.37	49.39
	swarm-distillation (st)	✓	FT	18.08	17.23	53.11	47.44	60.06
	swarm-distillation (ot)	✓	FT	13.16	12.77	47.37	47.70	51.83
	INTUITOR	✗	RL	19.68	18.21	59.23	50.13	59.45
	CPC-GRPO	✓	RL	**22.89**	**21.04**	**59.94**	**53.35**	**61.59**
Qwen3-4B	pre-trained	✗	–	61.64	63.65	**82.03**	85.50	91.77
	self-refine	✗	–	10.14	8.16	80.80	82.09	89.94
	swarm-distillation (st)	✓	FT	15.09	14.18	75.11	75.13	83.54
	swarm-distillation (ot)	✓	FT	30.56	33.31	77.14	80.45	90.24
	INTUITOR	✗	RL	58.79	62.75	81.57	85.63	**92.38**
	CPC-GRPO	✓	RL	**64.78**	**66.19**	**82.03**	**85.70**	92.07

Table 1: Results on RobustAlpacaEval (RAE; in-domain) and out-of-domain reasoning benchmarks (OBQA/AQuA/HumanEval). RAE uses AlpacaEval weighted win rate; "Trained/Un-trained" follows the RAE split.

Base Model	Method	RAE Tra.		RAE Un-Tra.		OBQA		AQuA		HumanEval	
		Best	Worst	Best	Worst	Best	Worst	Best	Worst	Best	Worst
LLaMA3.2-3B	pre-trained	40.06	5.27	**55.16**	**2.67**	89.80	22.80	79.53	**29.92**	82.93	**37.20**
	self-refine	19.56	1.14	29.59	0.08	76.40	3.80	80.31	10.63	68.90	29.88
	swarm-distillation (st)	34.26	4.46	47.24	1.12	86.80	19.20	78.35	24.02	85.98	34.15
	swarm-distillation (ot)	25.71	3.66	35.36	0.01	78.40	17.60	79.13	22.05	77.44	26.22
	INTUITOR	37.34	5.47	50.62	0.99	88.60	**25.20**	77.17	24.80	83.54	35.37
	CPC-GRPO	**42.73**	**8.16**	52.12	2.00	**90.20**	23.60	**84.65**	24.80	**89.02**	34.15
Qwen3-4B	pre-trained	87.37	33.56	94.06	18.14	94.80	**67.40**	92.52	74.02	**98.17**	85.37
	self-refine	21.1	1.99	25.66	0.35	94.60	59.00	92.52	62.60	97.56	82.32
	swarm-distillation (st)	32.96	1.59	44.20	0.27	91.80	51.00	90.16	59.45	95.73	71.34
	swarm-distillation (ot)	54.61	7.28	71.93	2.53	92.80	58.60	89.37	68.50	96.95	83.54
	INTUITOR	83.75	30.15	92.26	17.66	94.60	66.40	93.31	**75.98**	96.95	**87.80**
	CPC-GRPO	**89.10**	**39.67**	**94.78**	**22.49**	**95.20**	66.80	**94.09**	75.20	**98.17**	85.98

Table 2: Prompt sensitivity across paraphrase variants, reported as best/worst over prompt variants per example.

4 Experiment

4.1 Datasets

We study an *answer-free* setting where training does not require ground-truth answers or external evaluators. For in-domain training, we use RobustAlpacaEval (RAE) (Cao et al., 2024), where each example provides one original prompt and ten semantically equivalent paraphrases (11 variants). We fix the paraphrase group size to $G{=}4$ by selecting the first four variants in the predefined index order for training. At evaluation time, we report AlpacaEval weighted win rate separately on (i) the *Trained* prompt subset and (ii) the remaining *Un-trained* variants, measuring generalization to unseen phrasings.

For out-of-domain evaluation, we use OpenBookQA, AQuA, and HumanEval (Mihaylov et al., 2018; Ling et al., 2017; Chen et al., 2021). We construct instruction-style prompt templates using PromptSource and treat templates as prompt variants (views). Dataset statistics and construction details are provided in Appendix A.

4.2 Baselines

Since RAE does not provide ground-truth answers, we compare against methods that do not rely on answer verification (EM, pass@k) or external reward models. We evaluate: (1) **Pre-trained**, the base model without training; (2) **Self-refinement (test-time)** (Cao et al., 2024), which rewrites the prompt and answers the rewritten prompt; (3) **Swarm distillation** (Zhou et al., 2022), an unsupervised distillation baseline using paraphrase prompts as teachers (static/online); and (4) **INTUITOR** (Zhao et al., 2025), an RLIF-style GRPO baseline using self-certainty as intrinsic reward. All baselines are trained under the same setting as our method. Detailed protocols are in Appendix C.

4.3 Experimental Setting

We use LLaMA3.2-3B and Qwen3-4B as base models and conduct all FT/RL training with QLoRA (Dettmers et al., 2023). All methods are trained for three epochs with a fixed paraphrase group size G=4. All experiments are evaluated under the same setting; due to limited resources, we report results for a single configuration, and provide multi-seed results in Appendix E.

For out-of-domain benchmarks, we provide all methods with the same minimal output-format rules and compute scores via rule-based parsing (Appendix D). For RAE, we use the official AlpacaEval weighted win rate with `gpt4_turbo` as the reference output and judge (Cao et al., 2024; Dubois et al., 2025). For OpenBookQA and AQuA, we report accuracy from parsed answers, and for HumanEval we report unit-test-based pass@1. Hyperparameters and additional details are in Appendix B.

4.4 Main Result

Table 1 summarizes results on RobustAlpacaEval (RAE; in-domain) and three out-of-domain reasoning benchmarks (OBQA/AQuA/HumanEval). Across both base models, CPC-GRPO(Cross-Prompt Consensus GRPO) achieves the strongest RAE weighted win rates on both the *Trained* and *Un-trained* prompt subsets, indicating that paraphrase-based consensus rewards can provide a usable post-training signal without ground-truth answers or external judges.

On LLaMA3.2-3B, CPC-GRPO improves RAE from 21.42/19.25 to 22.89/21.04 (*Trained*/*Un-trained*), and yields consistent gains on OBQA/AQuA/HumanEval (+1.28/+0.99/+1.53, respectively). On Qwen3-4B, CPC-GRPO improves RAE from 61.64/63.65 to 64.78/66.19, while changes on out-of-domain benchmarks are small (OBQA tied; AQuA and HumanEval change marginally). Relative to INTUITOR, CPC-GRPO is consistently stronger on RAE and remains competitive on the reasoning benchmarks.

In our configuration, self-refinement and swarm distillation do not improve overall performance and often reduce scores (Table 1), suggesting that prompt rewriting or unsupervised distillation alone is insufficient to match RL updates driven by a consensus reward.

Table 2 further reports oracle best and worst-case performance over prompt variants per example. CPC-GRPO generally increases the *Best* scores and improves *Worst* in many settings (e.g., Qwen3-4B RAE Trained Worst: 33.56→39.67), but the *Best–Worst* gap does not consistently shrink across tasks. This indicates that training improves robustness in aggregate while residual prompt sensitivity remains.

Method	AVG	BEST	WORST
pre-trained	85.50	92.52	74.02
+ DLE	86.22	86.22	86.22
+ PCV	85.43	85.43	85.43
CPC-GRPO	85.70	94.09	75.20
+ DLE	88.19	88.19	88.19
+ PCV	87.01	87.01	87.01

Table 3: Test-time paraphrase ensembling on AQuA for Qwen3-4B.

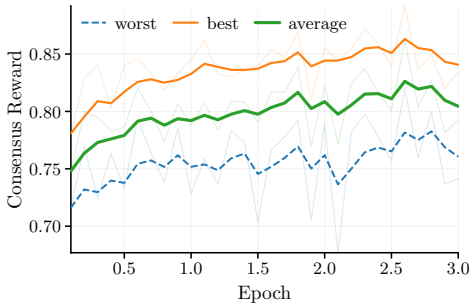

Figure 1: Training curves of the best/worst/mean consensus reward.

5 Analysis

5.1 Ensemble Performance

DLE averages the next-token distributions conditioned on the G paraphrase prompts and generates a single response by selecting tokens greedily from the averaged distribution (Eq. 2–3). **Paraphrase Consensus Voting (PCV)** computes the cross-prompt consensus reward (§3.2) for candidate responses generated from the G prompts, and then selects the response with the maximum reward. Since both methods combine multi-views to produce a single output, they remove performance dispersion (sensitivity) across prompt variants, yielding AVG=BEST=WORST.

Table 3 reports test-time ensemble performance on AQuA. For the pre-trained model, DLE provides a small gain (85.50→86.22), whereas PCV is comparable or slightly worse (85.50→85.43). In contrast, for the model trained with our method, both techniques improve performance, and DLE in particular yields a larger gain (85.70→88.19). This

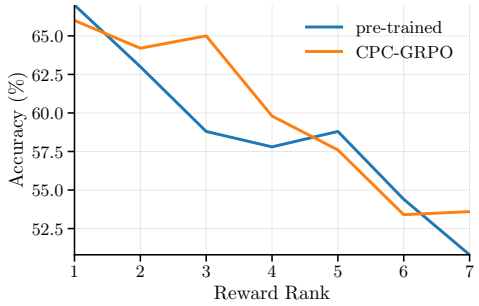

Figure 2: OpenBookQA accuracy of the consensus-reward rank for LLaMA.

L	AQuA	HumanEval	Time (×)
pre-trained	52.36	60.06	–
32	52.69	60.06	×0.08
64	52.69	61.59	×0.14
128	53.48	61.28	×0.26
256	53.22	63.11	×0.45
512	52.89	61.59	×0.73
1024	53.15	62.20	×0.86
2048	53.35	61.59	×1.00

Table 4: Effect of prefix length L and training time (normalized to CPC-GRPO as ×1.00).

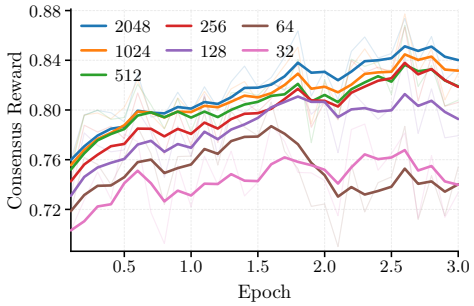

Figure 3: Training curves of the mean consensus reward (by prefix length L).

suggests that as paraphrase consensus is strengthened during training, test-time methods that combine paraphrase information can operate more effectively.

5.2 Learning Curve

To analyze training dynamics, we track the *best/mean/worst* of the consensus reward (§3.2) for the response set $\mathcal{Y}$ generated at each policy update. Concretely, given $\{r_j\}_{j=1}^{K}$ for a sample, we define best as $\max_j r_j$, worst as $\min_j r_j$, and mean as $\frac{1}{K}\sum_j r_j$, and record these statistics over the course of training.

In Figure 1, all three metrics tend to increase as training proceeds, supporting that the proposed optimization indeed operates in a direction that strengthens cross-prompt consensus. However, the gap between best and worst does not substantially shrink throughout training, indicating that a non-trivial level of prompt sensitivity remains even after training.

5.3 Alignment Between Consensus Reward and Performance

In this section, we analyze whether the consensus reward can serve as a *proxy for response quality* at test time. For each example, we generate candidate responses for each prompt variant and compute a cross-prompt-based consensus reward for each candidate, then sort candidates by their reward rank. We then measure task performance when selecting the response at rank r, to evaluate how well reward-based ranking aligns with actual performance.

Figure 2 shows accuracy on OpenBookQA by reward rank (with 1 denoting the highest reward). For the pre-trained model, accuracy is higher for top-ranked candidates and tends to decrease as rank worsens, suggesting that the consensus reward can act as a signal that distinguishes the relative quality of candidates even without ground-truth labels.

After training CPC-GRPO, this alignment trend largely persists, and we additionally observe that performance among the top ranks (rank 1–3) becomes more similar. This indicates that as training progresses, the quality of candidates receiving high rewards can improve together, which may reduce performance gaps among top-ranked candidates.

5.4 Prefix-level Training

Table 4 reports the performance–cost trade-off as a function of the response prefix length L used during training. With very short prefixes (L=32, 64), training cost is substantially reduced (to ×0.08 and ×0.14, respectively), but performance gains are limited or may be more variable. In contrast, from moderate lengths ($L\geq$128), AQuA/HumanEval performance improves more stably while training cost remains significantly reduced; in particular, at L=256 we reduce wall-clock time to ×0.45 while improving HumanEval to 63.11. Moreover, the longest prefix (L=2048) is not always optimal, showing that an appropriate L can be selected depending on the target task and resource constraints.

The prefix-conditioned learning curves in Figure 3 further confirm that the mean reward increases relatively stably for $L\geq$256, whereas shorter prefixes yield noisier trajectories.

Base Model	Method	RAE (In-domain)		Reasoning (Out-of-domain)		
		Trained	Un-trained	OBQA	AQuA	HumanEval
LLaMA3.2-3B	CPC-GRPO	22.89	**21.04**	**59.94**	53.35	61.59
	CPC-GRPO (log-likelihood)	**25.48**	20.66	59.00	**54.46**	**64.33**
Qwen3-4B	CPC-GRPO	64.78	66.19	**82.03**	**85.70**	**92.07**
	CPC-GRPO (log-likelihood)	**73.95**	**72.70**	81.69	85.37	90.24

Table 5: Ablation of default CPC-GRPO vs. likelihood-based variant. **bold** denotes the best.

Method	AQuA	OBQA	Time (×)
Intuitor	50.13	59.23	0.38
CPC-GRPO	53.35	59.94	1.00
w/o DLE	52.56	59.23	0.59
w/ Self-view	51.84	59.29	0.61

Table 6: Ablation Study on AQuA and OBQA with training time (normalized to CPC-GRPO as $\times 1.00$).

5.5 Ablation Study

Table 6 reports ablations of two key components in CPC-GRPO on LLaMA. Removing the decoding-level ensemble response (*w/o DLE*) reduces performance on both benchmarks (AQuA: 53.35→52.56; OBQA: 59.94→59.23), suggesting that the additional DLE candidate strengthens the consensus-oriented learning signal. In contrast, including self-view scores (*w/ Self-view*) decreases performance (AQuA: 53.35→51.84; OBQA: 59.94→59.29), consistent with self-view emphasizing prompt-specific fit over cross-view agreement. From an efficiency standpoint, *w/o DLE* reduces training time to $\times 0.59$ but sacrifices performance, indicating a performance–cost trade-off. Overall, using DLE while excluding self-view is the most effective setting in our experiments.

Why $O(G^2)$ rescoring does not translate to quadratic wall-clock time. CPC-GRPO performs teacher-forced cross-prompt rescoring over G prompt variants, which introduces $O(G^2)$ forward passes. However, system-level analyses of RLHF/RLVR pipelines report that autoregressive rollout generation typically dominates iteration time, often accounting for the majority of wall-clock runtime (Zheng et al., 2025; Gao et al., 2025). Since CPC-GRPO keeps the rollout budget comparable to baselines and adds computation mainly to rescoring/optimization, the added $O(G^2)$ forwards are amortized by the rollout-dominated runtime, resulting in the non-quadratic overhead observed in Table 6.

5.6 Log-Likelihood Consensus Reward

Table 5 ablates the scoring function used in the consensus reward. While the default CPC-GRPO uses the mean of token probabilities $s_{i,j}$ (Eq. (4)), the variant replaces it with the log-likelihood:

$$s_{i,j}^{\log} = \frac{1}{T_j} \sum_{t=1}^{T_j} \log \pi_{\theta_{\text{old}}} \left(y_t^{(j)} \mid x^{(i)}, y_{<t}^{(j)} \right). \quad (13)$$

The consensus reward aggregation and the update remain unchanged; we only substitute $s_{i,j}^{\log}$ for $s_{i,j}$.

This log-scale scoring is motivated by the observation that cross-prompt teacher-forcing can assign extremely small probabilities to early tokens under some views, making probability-mean scores saturate near zero and reducing discriminability. Using log-probabilities expands the dynamic range in the low-probability regime and is aligned with standard maximum-likelihood training.

Empirically, the log-likelihood variant substantially improves in-domain RAE, but its gains on *Un-trained* prompts and out-of-domain correctness are mixed (Table 5). This suggests that optimizing cross-view likelihood can primarily strengthen linguistic plausibility across views, which may not consistently translate to correctness-focused evaluation under distribution shift.

6 Conclusion

We study answer-free RL for LLM post-training without ground-truth answers or external evaluators. Using paraphrase prompts as multi-view observations, we define a cross-prompt consensus reward via teacher-forcing rescoring and optimize it with an all-pair GRPO-style update using group-relative advantages. Our method improves RobustAlpacaEval on both *Trained* and *Un-trained* prompts and shows non-degrading trends on out-of-domain reasoning benchmarks. Since the reward is computable on prefixes, we can explicitly trade off training cost and performance by adjusting the prefix length L.

Limitations

From a computational perspective, our method requires cross-prompt rescoring (teacher-forcing scoring) with respect to the paraphrase group size G. Thus, per-sample scoring expands to approximately $G \times K$ (prompt, response) combinations, and under the default setting ($K = G + 1$) it can incur additional forward computation on the order of $O(G^2)$. This cost can be substantial in particular when accumulating token-level log-probabilities over long responses. However, as shown in this paper, limiting the prefix length L reduces the number of processed tokens and can meaningfully reduce training wall-clock time, and G and L serve as primary levers for controlling the performance–cost trade-off. We also evaluate many prompt variants in our experiments for analysis purposes (separating *Trained/Un-trained* and measuring prompt sensitivity), but in deployment one may operate with single-prompt generation or with limited test-time ensembling.

Selecting the prefix length L is non-trivial. L simultaneously affects both the magnitude and variance of the learning signal, and its effect can depend on a combination of factors, including (i) the model's paraphrase robustness, (ii) the sharpness of token distributions, and (iii) the influence of early tokens induced by task and prompt formats. Therefore, it is difficult to guarantee that a fixed L is optimal across all models and datasets, and in practice additional tuning over L may be required. Future work should explore automated strategies such as scheduling that gradually increases L over training, or adaptive selection of L based on reward variance/stability.

References

Stephen H. Bach, Victor Sanh, Zheng-Xin Yong, Albert Webson, Colin Raffel, Nihal V. Nayak, Abheesht Sharma, Taewoon Kim, M Saiful Bari, Thibault Fevry, Zaid Alyafeai, Manan Dey, Andrea Santilli, Zhiqing Sun, Srulik Ben-David, Canwen Xu, Gunjan Chhablani, Han Wang, Jason Alan Fries, and 8 others. 2022. Promptsource: An integrated development environment and repository for natural language prompts. *Preprint*, arXiv:2202.01279.

Bowen Cao, Deng Cai, Zhisong Zhang, Yuexian Zou, and Wai Lam. 2024. On the worst prompt performance of large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. Evaluating large language models trained on code.

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. *Preprint*, arXiv:2307.08691.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *Preprint*, arXiv:2305.14314.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B. Hashimoto. 2025. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *Preprint*, arXiv:2404.04475.

Wei Gao, Yuheng Zhao, Dakai An, Tianyuan Wu, Lunxi Cao, Shaopan Xiong, Ju Huang, Weixun Wang, Siran Yang, Wenbo Su, Jiamang Wang, Lin Qu, Bo Zheng, and Wei Wang. 2025. Rollpacker: Mitigating long-tail rollouts for fast, synchronous rl post-training. *Preprint*, arXiv:2509.21009.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, and 175 others. 2025a. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

Jiaxin Guo, Daimeng Wei, Yuanchang Luo, Hengchao Shang, Zongyao Li, Jinlong Yang, Zhanglin Wu, Zhiqiang Rao, Shimin Tao, and Hao Yang. 2025b. M-ped: Multi-prompt ensemble decoding for large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 16693–16711, Suzhou, China. Association for Computational Linguistics.

David Heineman, Yao Dou, and Wei Xu. 2024. Improving minimum Bayes risk decoding with multi-prompt. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 22525–22545, Miami, Florida, USA. Association for Computational Linguistics.

Jian Hu, Xibin Wu, Wei Shen, Jason Klein Liu, Weixun Wang, Songlin Jiang, Haoran Wang, Hao Chen, Bin Chen, Wenkai Fang, Xianyu, Yu Cao, Haotian Xu, and Yiming Liu. 2025. OpenRLHF: A ray-based easy-to-use, scalable and high-performance RLHF framework. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 656–666, Suzhou, China. Association for Computational Linguistics.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James Validad Miranda, Alisa Liu, Nouha Dziri, Xinxi Lyu, Yuling Gu, Saumya Malik, Victoria

Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Christopher Wilhelm, Luca Soldaini, and 4 others. 2025. Tulu 3: Pushing frontiers in open language model post-training. In *Second Conference on Language Modeling*.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167, Vancouver, Canada. Association for Computational Linguistics.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025. Understanding r1-zero-like training: A critical perspective. In *Second Conference on Language Modeling*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium. Association for Computational Linguistics.

Youssef Mroueh. 2025. Reinforcement learning with verifiable rewards: Grpo's effective loss, dynamics, and success amplification. *Preprint*, arXiv:2503.06639.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Mihir Prabhudesai, Lili Chen, Alex Ippoliti, Katerina Fragkiadaki, Hao Liu, and Deepak Pathak. 2025. Maximizing confidence alone improves reasoning. *Preprint*, arXiv:2505.22660.

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

Francesco Tonolini, Nikolaos Aletras, Jordan Massiah, and Gabriella Kazai. 2024. Bayesian prompt ensembles: Model uncertainty estimation for black-box large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 12229–12272, Bangkok, Thailand. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention is all you need. *Preprint*, arXiv:1706.03762.

Zhewei Yao, Reza Yazdani Aminabadi, Olatunji Ruwase, Samyam Rajbhandari, Xiaoxia Wu, Ammar Ahmad Awan, Jeff Rasley, Minjia Zhang, Conglong Li, Connor Holmes, Zhongzhu Zhou, Michael Wyatt, Molly Smith, Lev Kurilenko, Heyang Qin, Masahiro Tanaka, Shuai Che, Shuaiwen Leon Song, and Yuxiong He. 2023. Deepspeed-chat: Easy, fast and affordable rlhf training of chatgpt-like models at all scales. *Preprint*, arXiv:2308.01320.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, and 16 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *Preprint*, arXiv:2503.14476.

Tony Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *International Conference on Machine Learning*.

Xuandong Zhao, Zhewei Kang, Aosong Feng, Sergey Levine, and Dawn Song. 2025. Learning to reason without external rewards. *arXiv preprint arXiv:2505.19590*.

Haizhong Zheng, Yang Zhou, Brian R. Bartoldson, Bhavya Kailkhura, Fan Lai, Jiawei Zhao, and Beidi Chen. 2025. Act only when it pays: Efficient reinforcement learning for LLM reasoning via selective rollouts. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Qihao Zhu Runxin Xu Junxiao Song Mingchuan Zhang Y.K. Li Y. Wu Daya Guo Zhihong Shao, Peiyi Wang. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models.

Yinmin Zhong, Zili Zhang, Bingyang Wu, Shengyu Liu, Yukun Chen, Changyi Wan, Hanpeng Hu, Lei Xia, Ranchen Ming, Yibo Zhu, and Xin Jin. 2025. Optimizing RLHF training for large language models with stage fusion. In *22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI 25)*, pages 489–503, Philadelphia, PA. USENIX Association.

Chunting Zhou, Junxian He, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022. Prompt consistency for zero-shot task generalization. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2613–2626, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*.

A Detailed Datasets

RobustAlpacaEval (RAE). RAE is an instruction-following benchmark based on TinyAlpacaEval. Each example consists of (i) one original prompt and (ii) ten semantically equivalent paraphrase prompts (11 prompt variants in total). The paraphrases are automatically generated using GPT-4 and then manually reviewed/edited to ensure meaning preservation and fluency (Cao et al., 2024). In the main experiments, we use the first $G{=}4$ variants under a fixed index order for training (*Trained prompts*), and at evaluation time we compute scores separately on (i) the *Trained* subset and (ii) the remaining *Un-trained* subset over the same set of examples.

Out-of-domain benchmarks. For evaluating generalization, we use OpenBookQA, AQuA, and HumanEval (Mihaylov et al., 2018; Ling et al., 2017; Chen et al., 2021). For each dataset, we construct instruction-style prompts using templates from PromptSource (Bach et al., 2022), and treat different templates as prompt variants (= views), forming a set of prompt variants per example. We report the number of examples and prompt variants for each dataset in Table 7. We will release the dataset construction and preprocessing scripts on GitHub.

B Detailed Experimental Setting

Training. All fine-tuning (FT) and RL-based methods are trained with QLoRA, and we fix the paraphrase group size to $G{=}4$ throughout training. For RL training, we perform rollouts and cross-prompt scoring required for policy updates; to reduce rollout cost, we can use a multi-step update that reuses a single rollout across multiple optimization steps. The main hyperparameters of our default setting are summarized in Table 8. Since there is no verification-based reward, we do not use a KL term in the default setting ($\beta{=}0$), and we fix the learning rate to 1×10^{-5}.

Hardware. Training is performed on RTX 4090 $\times$ 2, and evaluation is performed on RTX 4090 $\times$ 1. All experiments are run under the same machine environment.

Evaluation metrics. We evaluate RAE using the AlpacaEval weighted win rate, and set `gpt4_turbo` as the reference output and judge (Dubois et al., 2025; Cao et al., 2024). We also separately re-

Dataset	#Examples	#Prompt Variants	Total Prompts
RobustAlpacaEval (RAE)	100	1+10	1100
AQuA	254	6	1524
OpenBookQA	500	7	3500
HumanEval	164	2	328

Table 7: Dataset statistics. We report the number of examples, the number of prompt variants per example, and the resulting total number of prompts.

Hyperparameter	Value
Prompt views (G)	4
Quantization	4-bit (QLoRA)
LoRA rank (r)	16
LoRA α	32
LoRA dropout	0.05
Optimizer	AdamW
Weight decay	0.0
Epochs	3
Batch size (per device)	1
Prefix length (L; main)	2048
PPO clip ϵ	0.2
Learning rate	1×10^{-5}
KL coefficient β	0
LR scheduler	cosine
Warmup ratio	0.03

Table 8: Hyperparameter settings used in our experiments.

port results on the *Trained* prompt subset and the *Un-trained* subset over the same set of examples. For OpenBookQA and AQuA, we extract answer choices via rule-based parsing and compute accuracy. For HumanEval, we extract code blocks using rule-based procedures and compute unit-test-based pass@1. All methods are provided with the same output-format rules, and the rule templates and parsing procedures are described in Appendix D.

Decoding for rollouts and evaluation. Unless otherwise specified, we use the same decoding configuration across all methods for fair comparison. For rollouts, we generate one response per view (prompt variant) with a fixed maximum completion length and standard stopping criteria. For the decoding-level ensemble (DLE), we compute the per-step averaged token distribution across views and apply greedy decoding. Exact decoding parameters (e.g., temperature, top-p, max new tokens, and stop sequences) are fixed across runs and included in the released configuration files.

C Detailed Baselines

In this section, we summarize the key configurations and the training/inference protocols of the baselines compared in the main text. All baselines use the same prompt format and decoding settings as in the main paper, and for the out-of-domain benchmarks (OpenBookQA/AQuA/HumanEval) we follow the output-format and parsing rules in Appendix D. (Complete hyperparameters and full template specifications are included in the released code and configuration files.)

Pre-trained. We evaluate the base model without any additional training. On RAE, we compute the AlpacaEval weighted win rate for each prompt variant, and the *Trained*/*Un-trained* subset split follows the main setting.

Self-refinement (test-time). We follow the self-refinement idea used in the RAE paper, but in this work we apply only a test-time template without any additional training (Cao et al., 2024). The model (i) rewrites the input prompt into a clearer form without changing its meaning, and then (ii) generates the final response conditioned on the rewritten prompt. The output consists of two sections, `REWRITTEN PROMPT:` and `RESPONSE:`, and we use only `RESPONSE` as the final output for evaluation (Figure 5). This baseline can improve performance via prompt clarification without training, but it has limitations in that the rewriting process may introduce subtle meaning shifts or amplify model biases.

Swarm distillation (unsupervised SFT with a pseudo-target pool). Swarm distillation is an unsupervised distillation method that treats multiple outputs obtained from a paraphrase prompt set for the same intent as a *pseudo-target pool*, and trains each prompt to imitate a pseudo-target sampled from the pool (Zhou et al., 2022; Cao et al., 2024). For each RAE example e, we form G prompts $\mathcal{X}_e = \{x^{(i)}\}_{i=1}^{G}$ and denote the corresponding set of candidate outputs as $\mathcal{Y}_e = \{y^{(i)}\}_{i=1}^{G}$. During training, for each prompt $x^{(i)}$, we uniformly sample a pseudo-target $\tilde{y}$ from $\mathcal{Y}_e$ and minimize the standard instruction SFT objective (cross-entropy

applied only on the answer-token span):

$$\mathcal{L}_{\text{swarm}}(\theta) = -\sum_{e}\sum_{i=1}^{G} E_{\tilde{y}\sim\text{Unif}(\mathcal{Y}_e)} \sum_{t=1}^{|\tilde{y}|} \log \pi_\theta\Big(\tilde{y}_t \mid x^{(i)}, \tilde{y}_{<t}\Big)\,. \tag{14}$$

In this work, we evaluate a simple form of the baseline by using only *uniform sampling* from the pseudo-target pool, without separately tuning additional selection mechanisms or score-based aggregation. This approach can leverage expression diversity across paraphrases as supervision, but if pseudo-targets include teacher errors, it can suffer from confirmation bias where errors are amplified.

Static-teacher (st). Static-teacher keeps the pseudo-target pool $\mathcal{Y}_e$ fixed throughout training. In our experiments, we use a pre-generated teacher output file (outputs for each prompt variant) as the fixed pool, and at each step we randomly sample a pseudo-target from $\mathcal{Y}_e$ and train using Eq. (14).

Online-teacher (ot). Online-teacher is an *iterative self-training* variant that repeatedly refreshes the pseudo-target pool using responses generated by the current model during training. For each example e and prompt $x^{(i)}$, we generate a new response under the current policy and replace the pool element for that view:

$$y_e^{(i)} \leftarrow \text{Gen}\Big(\pi_\theta; x^{(i)}\Big)\,. \tag{15}$$

We then uniformly sample pseudo-targets from the updated $\mathcal{Y}_e$ and continue training with Eq. (14). In our implementation, we use deterministic decoding (e.g., greedy) to reduce variance in generation, and pool updates are performed only for examples/views that appear in the current minibatch. Online-teacher can quickly reflect the student's latest distribution, but it poses risks of reduced diversity or mode collapse.

INTUITOR. INTUITOR is an RLIF-style method that optimizes with GRPO-style RL using intrinsic rewards derived from the model's internal self-certainty signals, without external reward models or answer verification (Zhao et al., 2025). In this work, we use the public implementation as-is, while applying the same prompt format/decoding settings and evaluation rules as in the main paper for fair comparison. The detailed reward definition and training pipeline follow the original paper and the released code (Zhao et al., 2025). This baseline provides an answer-free training signal, but the alignment between certainty signals and true task performance can vary with tasks and prompts.

D Formatting RULE

Test-time RULE To evaluate the out-of-domain datasets in this paper, we must parse the answer portion from each model response to compute the metrics. Therefore, for each dataset we prepend a minimal set of rules to the input so that the model outputs the answer in a prescribed format. The rules used in our experiments are shown in Figure 4.

Self-refinement baseline format. Self-refinement requires the model to first rewrite the input prompt and then produce a response to the rewritten prompt. At evaluation time, we use only the text after `RESPONSE:` as the final output (the rewritten prompt is excluded from evaluation). In RobustAlpacaEval, many cases skip the reasoning stage, so we provide an additional format specifically for RobustAlpacaEval. The formatting rule used in our experiments is shown in Figure 5.

E Generalization Verification

In the main text, due to resource constraints, we primarily analyze results from a single run per setting. To further examine whether the observed gains of the proposed method (CPC-GRPO) overly depend on a specific run, we repeat training with different random seeds and measure performance under the same evaluation protocol.

As shown in Table 9, for both base models, most metrics exhibit comparable improvements over the pre-trained baseline on in-domain (RAE) and out-of-domain reasoning benchmarks. In particular, for Qwen3-4B, all three seeds consistently maintain improved performance on both *Trained* and *Untrained* subsets, and the variation on out-of-domain benchmarks is limited. For LLaMA3.2-3B, seed-wise variability is relatively larger, but improvements are generally preserved on OBQA and HumanEval. Overall, these results indicate that the gains of CPC-GRPO are not confined to a particular case and exhibit broadly reproducible trends.

F Performance of RobustAlpacaEval

For follow-up work, we additionally include a performance table in the same format as (Cao et al.,

AQuA (multiple-choice)

```
RULE:
- You may reason briefly.
- Stay within 2048 tokens.
- Your response must end with exactly: The correct answer is <LETTER>
- <LETTER> must be one of A, B, C, D, or E.
```

OpenBookQA (multiple-choice/free-form)

```
RULE:
- You may reason briefly.
- Stay within 2048 tokens.
- Your response must end with exactly: The correct answer is <ANSWER>
```

HumanEval (code generation)

```
RULE:
- You may reason briefly.
- Stay within 2048 tokens.
- After briefly reasoning, output ONLY one code block: “‘python ... “‘
- The block must contain ONLY the complete function definition with the exact same
signature.
```

Figure 4: Output constraints used for out-of-domain evaluation to enable deterministic parsing. The same rules are prepended for all methods.

2024). In this work, we report RobustAlpacaEval results by separating paraphrases used for training from the remaining paraphrases, whereas Cao et al. (2024) reports performance over the full set of 11 prompt versions. Accordingly, we report results aggregated over the full paraphrase set in Table 10.

RobustAlpacaEval

```
Rewrite the PROMPT to be clearer without changing meaning (do not add/remove requirements),
then respond to the rewritten prompt. Output exactly two sections.
ORIGINAL PROMPT: {...}
```

Out-of-Domain

```
Rewrite the PROMPT to be clearer without changing meaning (do not add/remove requirements),
then respond to the rewritten prompt. Output exactly two sections.
ANSWER FORMAT
REWRITTEN PROMPT: {Rewritten prompt}
RESPONSE: {Response of rewritten prompt}
ORIGINAL PROMPT: {...}
```

Figure 5: Formatting rule used for the self-refinement baseline. We only evaluate the content after `RESPONSE:`.

Base Model	Method	**RAE** (In-domain)		**Reasoning** (Out-of-domain)		
		Trained	**Un-trained**	**OBQA**	**AQuA**	**HumanEval**
LLaMA3.2-3B	pre-trained	21.42	19.25	58.66	52.36	60.06
	CPC-GRPO (seed1)	**22.89**	**21.04**	**59.94**	53.35	<u>61.59</u>
	CPC-GRPO (seed2)	20.12	<u>20.08</u>	59.60	<u>54.46</u>	<u>61.59</u>
	CPC-GRPO (seed3)	<u>21.56</u>	19.96	<u>59.71</u>	**54.59**	**62.50**
Qwen3-4B	pre-trained	61.64	63.65	**82.03**	85.50	91.77
	CPC-GRPO (seed1)	**64.78**	66.19	**82.03**	<u>85.70</u>	**92.07**
	CPC-GRPO (seed2)	<u>64.17</u>	<u>66.55</u>	<u>81.97</u>	85.50	**92.07**
	CPC-GRPO (seed3)	63.16	**66.61**	81.57	**86.29**	<u>91.16</u>

Table 9: Seed sensitivity of CPC-GRPO. Pre-trained scores are taken from Table 1. Within each base model and each column, **bold** denotes the best and <u>underline</u> denotes the second-best among CPC-GRPO seeds (ties are marked).

Base Model	Method	Para.	Train	**RAE (All prompt versions)**				
				Orig.	**Best**	**Worst**	**Avg.**	**Std.**
LLaMA3.2-3B	pre-trained	✗	–	30.88	64.05	1.09	20.90	23.24
	self-refine	✗	–	12.36	35.04	0.01	9.27	13.08
	swarm-distillation (st)	✓	FT	25.22	55.76	0.28	18.19	20.54
	swarm-distillation (ot)	✓	FT	17.15	43.59	0.00	13.28	16.29
	INTUITOR	✗	RL	28.82	60.22	0.68	19.58	21.84
	CPC-GRPO	✓	RL	34.84	60.75	1.96	22.80	22.12
Qwen3-4B	pre-trained	✗	–	68.77	95.64	11.40	63.58	31.89
	self-refine	✗	–	13.40	35.49	0.35	9.17	12.62
	swarm-distillation (st)	✓	FT	16.60	53.11	0.04	14.72	19.48
	swarm-distillation (ot)	✓	FT	39.10	80.40	0.58	33.09	30.43
	INTUITOR	✗	RL	70.05	95.45	9.46	62.31	32.29
	CPC-GRPO	✓	RL	68.62	96.04	15.73	66.03	30.30

Table 10: RobustAlpacaEval (RAE) statistics aggregated over all prompt versions (original + paraphrases).