# Reward-Wise Value Estimation for Multi-Reward Optimization in Large Language Models

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

Value functions play a central role in critic-based reinforcement learning (RL) algorithms such as proximal policy optimization (PPO) and in inference-time scaling for large language models (LLMs). In complex LLM tasks, multiple reward signals are often leveraged to guide RL training and inference. However, existing value function estimation methods either aggregate all rewards into a single scalar, limiting flexibility across reward components, or rely on separate critic models for each reward, which is computationally inefficient. To address these limitations, we propose an efficient and flexible value function estimation method for multi-reward optimization in LLMs. Specifically, we introduce a multi-head critic model architecture that jointly estimates reward-wise value functions within a single model. This design enables fine-grained credit assignment across reward components while maintaining computational efficiency. We further prove that the proposed multi-head critic loss yields lower irreducible error compared to the vanilla single-head critic loss. Experimental results on multi-turn agentic search and mathematical reasoning tasks demonstrate that our reward-wise value estimation approach consistently improves both policy optimization and critic learning in PPO training.

## 1 Introduction

Reinforcement learning (RL) has become a standard paradigm for aligning large language models (LLMs) on complex reasoning [1, 2] and agentic tasks [3, 4]. Existing RL algorithms for LLM training can be broadly categorized into critic-free and critic-based methods. Critic-free algorithms, such as Group Relative Policy Optimization (GRPO) [5] and REINFORCE Leave-One-Out (RLOO) [6, 7], remove the value function and estimate advantages in a group-relative manner. While this avoids the difficulty of critic training, the resulting advantage signals are typically derived from final rewards and can be coarse-grained, leading to less precise credit assignment [8, 9, 10]. Critic-based algorithms, such as proximal policy optimization (PPO) [11], leverage value functions to compute generalized advantage estimation (GAE) [12], providing fine-grained advantage signals for policy optimization [13]. Moreover, a well-trained critic can be used to guide inference-time scaling and search [14, 15, 16, 17], further improving policy performance.

Recently, multiple reward signals have been widely used to guide RL training and inference in LLMs. Beyond answer rewards, auxiliary rewards such as format rewards and process rewards are often incorporated for reward shaping in search [18, 19], tool-calling agentic tasks [20, 21], preference learning [22] and complex multi-agent intelligence [23]. However, these methods typically aggregate all reward components into a single scalar, which limits flexibility across heterogeneous rewards and may lead to unstable training or even training collapse [10, 24]. This naive aggregation can lead to poor credit assignment, since the model loses access to reward-specific learning signals and

Submitted to 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Do not distribute.

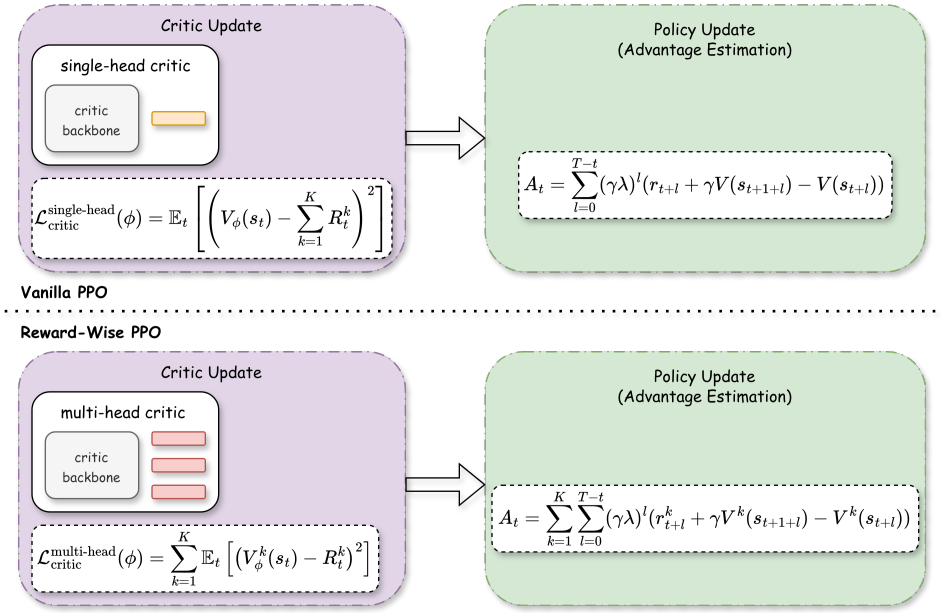


Figure 1: Comparison between vanilla PPO and reward-wise PPO with a multi-head critic. Vanilla PPO aggregates multiple reward components into a single scalar target and learns a single-head critic, while reward-wise PPO estimates reward-wise value functions with multiple value heads. The reward-wise advantages are computed separately for each reward component and then aggregated for policy optimization, enabling fine-grained value estimation across heterogeneous rewards with minimal additional critic overhead.

cannot identify which reward components contribute most during optimization. In addition, a critic trained on aggregated rewards can only estimate an overall value function, rather than providing reward-specific value signals for individual components. This limits its flexibility in inference-time scaling, where different reward components may need to be weighted or queried separately.

In this paper, we focus on critic-based RL algorithms for multi-reward settings. On the one hand, optimizing with aggregated rewards can lead to poor credit assignment and results in an inflexible critic that only provides overall value estimates, without reward-specific value signals for individual components. On the other hand, training a separate critic model for each reward component is computationally expensive and memory inefficient. Therefore, this raises the following question:

*How can we design critic-based RL algorithms that improve policy optimization while maintaining an efficient and flexible critic model for multi-reward settings?*

To address these limitations, we propose an efficient and flexible value function estimation method for multi-reward optimization in LLMs. Our contributions are summarized as follows:

- We introduce a multi-head critic architecture that jointly estimates reward-wise value functions within a single model. This design enables fine-grained value estimation across reward components while maintaining computational efficiency. We prove that the reward-wise critic loss yields lower irreducible error compared to the vanilla single-head critic loss.
- Building on the multi-head critic, we propose reward-wise PPO, which separately estimates GAE for each reward component and aggregates reward-wise advantages for policy optimization, enabling more fine-grained credit assignment across heterogeneous rewards. Figure 1 compares vanilla PPO with a single-head critic and reward-wise PPO with a multi-head critic.
- Experimental results demonstrate that our reward-wise value estimation approach consistently improves both policy optimization and critic learning in PPO training across different model sizes and datasets on both multi-turn agentic search and math reasoning tasks.

## 2 Related Work

**RL Algorithms for LLMs.** For LLM training, existing RL algorithms can be categorized into critic-free methods, such as GPRO [5] and RLOO [6, 7], and critic-based methods, such as PPO [11]. GRPO-type methods estimate advantages from group-level rollout samples and avoid critic training, resulting in lower memory and computational overhead. However, although they bypass the difficulty of training a critic, their advantage signals are typically derived from final rewards and can lead to coarse-grained trajectory-level advantage estimation, resulting in less precise credit assignment [8, 9, 10]. Recent studies further suggest that such coarse-grained advantage estimation can lead to unstable training and even training collapse [10, 25, 26]. In contrast, PPO-type methods leverage value functions to compute GAE [12], providing fine-grained credit assignment for policy optimization [13]. Moreover, a well-trained critic can be used to guide or evaluate inference-time scaling and search [14, 15, 16, 17], further improving performance.

**Multi-Reward RL.** Recently, RL training and inference-time scaling have moved beyond single-reward optimization. Beyond answer correctness rewards, auxiliary rewards such as format rewards and process rewards are often incorporated for reward shaping in search [18, 19], tool-calling agentic tasks [20, 21], and complex multi-agent intelligence [23]. Another important application of multi-reward RL is improving the efficiency of reasoning models while maintaining task performance, primarily by introducing length-based rewards alongside outcome-based rewards [27, 28]. In addition, preference learning methods can use different preference rubrics to adapt multiple reward modeling to diverse tasks [22]. However, these methods typically aggregate all reward components into a single scalar, which limits flexibility across heterogeneous rewards and may lead to unstable training or even training collapse [10, 24]. The most closely related works are concurrent studies that propose estimating advantages separately for each reward component in a GRPO-style manner [24, 29]. The key difference is that these methods are critic-free, whereas our work focuses on critic-based RL. Moreover, these GRPO-type methods do not learn critic models, and thus cannot directly provide value functions for inference-time scaling or search.

## 3 Preliminaries

### 3.1 Modeling Language Generation as Token-Level MDP

Let $x \sim \mathcal{D}$ denote the input prompt from the dataset $\mathcal{D}$, and $y = [y_1, \ldots, y_T]$ the generated response. Text generation by a language model can be formulated as sampling from a policy in a Markov decision process (MDP), defined as the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ the action space, $P$ the transition dynamics, $r$ the reward function, and $\gamma$ the discount factor. At each time step $t$, the state is $s_t = (x, y_{<t}) \in \mathcal{S}$, consisting of the prompt $x$ and the previously generated tokens, and the action $a_t = y_t \in \mathcal{A}$ corresponds to selecting the next token from the vocabulary. The transition dynamics $P$ are deterministic: given $s_t$ and $a_t$, the next state is $s_{t+1} = (x, y_{\leq t})$. The reward function $r_t = r(s_t, a_t)$ assigns a scalar value at each time step. Since it is a common practice to use the discount factor $\gamma = 1$ in RL for LLM training, we omit $\gamma$ in the subsequent notation for simplicity, unless otherwise specified. The value function is defined as $V(s_t) = \mathbb{E}\left[\sum_{t'=t}^{T} r_{t'}\right]$, representing the expected return from state $s_t$. RL aims to maximize the expected cumulative reward

$$J(\theta) = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)}\left[\sum_{t=1}^{T} r_t\right]. \tag{1}$$

In this work, we consider a terminal reward provided only at the end of the sequence, denoted as $\mathcal{R}(x, y)$, i.e., $r_t = 0$ for $t < T$ and $r_T = \mathcal{R}(x, y)$.

### 3.2 Proximal Policy Optimization

PPO [11] is a popular actor-critic RL algorithm commonly used for LLM training [13]. It optimizes the policy by maximizing the clipped surrogate objective:

$$\mathcal{J}_{\text{PPO}}(\theta) = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_{\theta_{\text{old}}}(\cdot|x)}\left[\sum_{t=1}^{T} \min(\rho_t A_t,\ \text{clip}(\rho_t,\ 1-\epsilon,\ 1+\epsilon)A_t)\right], \tag{2}$$

where $\rho_t = \frac{\pi_\theta(y_t|y_{<t},x)}{\pi_{\theta_{\text{old}}}(y_t|y_{<t},x)}$ is the importance sampling ratio between the current policy $\pi_\theta$ and the old policy $\pi_{\theta_{\text{old}}}$, $A_t$ is the advantage term, and $\epsilon$ is the clipping parameter. The PPO algorithm mainly consists of two components: advantage estimation and value estimation.

- **Advantage Estimation.** The advantage $A_t$ measures the relative quality of action $a_t$ at state $s_t$. In PPO, it is typically estimated using GAE [12]: $A_t = \sum_{l=0}^{T-t}(\gamma\lambda)^l\delta_{t+l}$, where $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$ is the temporal-difference (TD) error, and $V(s_t)$ denotes the value of state $s_t$.
- **Value Estimation.** The value function, often referred to as the critic model, estimates the expected return for each state. The objective of the critic is to minimize the discrepancy between its predicted values and the target returns, typically via a mean squared error (MSE) loss:

$$\mathcal{L}_{\text{critic}}(\phi) = \mathbb{E}_t\left[\left(V_\phi(s_t) - R_t\right)^2\right], \tag{3}$$

  where $V_\phi(s_t)$ denotes the predicted value of state $s_t$ parameterized by $\phi$, and $R_t$ denotes the target return, which can be estimated using Monte Carlo returns: $R_t = \sum_{t'=t}^{T} r_{t'}$.

### 3.3 Multi-Reward Modeling

In multi-reward modeling, the reward function in the MDP formulation outputs a $K$-dimensional vector, i.e., $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^K$, where $r^k$ denotes the $k$-th reward component. The reward components are aggregated into a scalar reward, i.e., $r_t = \sum_{k=1}^{K} r_t^k$, for policy optimization in Eq. (1). When $K = 1$, this formulation reduces to the standard single-reward setting. In the terminal reward setting, each data pair $(x, y)$ is associated with a set of $K$ terminal reward components $\{\mathcal{R}^k(x, y)\}_{k=1}^K$. For multi-reward settings, learning an effective critic for policy optimization and inference-time scaling poses significant challenges. On the one hand, aggregating all reward components into a single scalar limits flexibility and obscures the contribution of individual rewards. On the other hand, learning separate critic models for each reward component incurs substantial computational and memory overhead.

## 4 Reward-Wise Value Estimation and Reward-Wise PPO

### 4.1 Reward-Wise Value Estimation

To address these limitations, we propose an efficient and flexible value function estimation method for multi-reward optimization in LLMs. Specifically, we introduce a multi-head critic architecture that jointly estimates reward-wise value functions within a single model. Compared to a traditional single-head critic, our multi-head critic shares the same backbone while incorporating multiple value heads, each learning a distinct reward component independently.

Given $K$ Monte Carlo returns $R_t^k$ for $k = 1, 2, \ldots, K$, we define the reward-wise critic loss as

$$\mathcal{L}_{\text{critic}}^{\text{rw}}(\phi) = \sum_{k=1}^{K} \mathbb{E}_t\left[\left(V_\phi^k(s_t) - R_t^k\right)^2\right], \tag{4}$$

where $V_\phi^k(s_t)$ denotes the predicted value of state $s_t$ from the $k$-th value head corresponding to the $k$-th reward component. Compared to the vanilla critic loss

$$\mathcal{L}_{\text{critic}}(\phi) = \mathbb{E}_t\left[\left(V_\phi(s_t) - \sum_{k=1}^{K} R_t^k\right)^2\right], \tag{5}$$

which estimates the aggregated return across all reward components, the proposed reward-wise critic loss decomposes value estimation into per-reward components, enabling more fine-grained modeling of heterogeneous rewards.

**Overhead Analysis.** We compare our multi-head critic architecture with the vanilla single-head critic and a multi-critic architecture in terms of parameter count and forward computational cost (FLOPs) in Table 1. For parameter count, all three critic architectures share the same backbone with $12lh^2 + vh$

Table 1: Parameter and forward FLOPs comparison of different critic architectures. Here, $l$ denotes the number of Transformer layers, $h$ the hidden dimension, $v$ the vocabulary size, $s$ the sequence length, $b$ the batch size, and $K$ the number of reward components. Detailed derivations of the parameter count and forward FLOPs can be found in Appendix A.

| **Architecture** | **Parameter Count** | **Forward FLOPs** |
|---|---|---|
| single-head critic | $12lh^2 + vh + h$ | $b\big[l(12sh^2 + 2s^2h) + sh\big]$ |
| multi-critic | $K(12lh^2 + vh + h)$ | $Kb\big[l(12sh^2 + 2s^2h) + sh\big]$ |
| multi-head critic | $12lh^2 + vh + Kh$ | $b\big[l(12sh^2 + 2s^2h) + Ksh\big]$ |

parameters, which dominates the overall parameter size. The difference arises from the output head: the single-head critic has a single value head with $h$ parameters, the multi-critic architecture replicates the entire backbone $K$ times, and the multi-head critic introduces only $K$ lightweight value heads with $Kh$ parameters. In particular, compared to the single-head critic, the multi-head critic introduces only an additional $(K-1)h$ parameters, which corresponds to a negligible overhead relative to the backbone $12lh^2 + vh$. For forward FLOPs, all three critic architectures share the same backbone cost $l(12sh^2 + 2s^2h)$, which dominates the overall computation. The difference lies in the output head: the single-head critic incurs an additional $sh$ cost, the multi-critic architecture requires $K$ independent forward passes, and the multi-head critic introduces only $Ksh$ additional cost from the value heads. Therefore, the multi-head critic achieves comparable parameter efficiency and computational cost to the single-head critic, while avoiding the $K$-fold increase in both parameters and FLOPs incurred by the multi-critic architecture.

We provide a theoretical analysis comparing the two critic losses in Eq. (4) and Eq. (5).

**Proposition 1.** *Let each reward-wise return be decomposed as*

$$R_t^k = \mu_t^k + \epsilon_t^k,$$

*where $\mu_t^k = \mathbb{E}[R_t^k \mid s_t]$ and $\epsilon_t^k$ is zero-mean noise. Let $V_\phi^k(s_t)$ be the reward-wise value estimate and $V_\phi(s_t)$ be the vanilla value estimate for the aggregated return. Define the reward-wise prediction error as $e_t^k = V_\phi^k(s_t) - \mu_t^k$, and the vanilla prediction error as $e_t = V_\phi(s_t) - \sum_{k=1}^K \mu_t^k$. Then the expected reward-wise critic loss decomposes as*

$$\mathbb{E}[\mathcal{L}_{\text{critic}}^{\text{rw}}] = \sum_{k=1}^K \mathbb{E}\big[(e_t^k)^2\big] + \sum_{k=1}^K \mathbb{E}\big[(\epsilon_t^k)^2\big]\,,$$

*whereas the expected vanilla critic loss decomposes as*

$$\mathbb{E}[\mathcal{L}_{\text{critic}}] = \mathbb{E}\big[e_t^2\big] + \mathbb{E}\left[\left(\sum_{k=1}^K \epsilon_t^k\right)^2\right].$$

*Moreover,*

$$\mathbb{E}\left[\left(\sum_{k=1}^K \epsilon_t^k\right)^2\right] = \sum_{k=1}^K \mathbb{E}\big[(\epsilon_t^k)^2\big] + 2\sum_{i<j} \mathbb{E}\Big[\epsilon_t^i \epsilon_t^j\Big]\,.$$

*Thus, the vanilla critic loss contains cross-component noise terms, while the reward-wise critic loss decomposes the regression problem across individual reward components. In particular, when the reward noise terms are positively correlated, i.e., $\mathbb{E}[\epsilon_t^i \epsilon_t^j] > 0$ for $i \neq j$, the cross-component terms further increase the noise level of the vanilla critic target.*

The proof is provided in Appendix B. Here, the noise refers to the randomness in the Monte Carlo return targets induced by policy sampling, environment stochasticity, and reward estimation, formally defined as $\epsilon_t^k = R_t^k - \mathbb{E}[R_t^k \mid s_t]$. This corresponds to the irreducible noise in the critic regression target, which cannot be eliminated by improving the quality of value function estimation. When rewards are aggregated, these component-wise noise terms are also summed, introducing additional cross-component noise through covariance terms. In particular, when reward components are positively correlated, aggregation further amplifies the noise level in the vanilla critic target.

**Algorithm 1:** Reward-Wise PPO with Multi-Head Critic Learning

**Input:** Policy model $\pi_\theta$, old policy $\pi_{\theta_{\text{old}}}$, multi-head critic $V_\phi = \{V_\phi^k\}_{k=1}^K$, reward components $\{r_t^k\}_{k=1}^K$

**Output:** Updated policy parameters $\theta$ and critic parameters $\phi$

**for** *each training iteration* **do**
- `// 1. Rollout stage`
- Collect trajectories using $\pi_{\theta_{\text{old}}}$ and obtain reward components $\{r_t^k\}_{k=1}^K$;
- `// 2. Training stage`
- Compute reward-wise value estimates $\{V_\phi^k(s_t)\}_{k=1}^K$ using the multi-head critic;
- **for** *each reward component* $k = 1, \ldots, K$ **do**
  - Compute reward-wise advantages $A_t^k$ using reward-wise GAE in Eq. (7);
  - Compute reward-wise returns $R_t^k$;
- Aggregate reward-wise advantages $A_t = \sum_{k=1}^K A_t^k$ as in Eq. (6);
- Update the policy using the PPO objective in Eq. (2) with the aggregated advantage $A_t$;
- Update the multi-head critic using the reward-wise critic loss in Eq. (4);
- Set $\theta_{\text{old}} \leftarrow \theta$;

In contrast, the reward-wise critic keeps the regression targets separated and therefore avoids such aggregation-induced cross-component noise. As a result, it provides a cleaner supervision signal for critic learning and mitigates interference across heterogeneous reward components.

### 4.2 Reward-Wise PPO

Building upon our reward-wise value estimation with a multi-head critic, we introduce reward-wise PPO to further improve policy optimization. Specifically, we decompose the advantage function into reward-wise components and aggregate them for policy updates.

We define the reward-wise advantage as

$$A_t = \sum_{k=1}^{K} A_t^k, \tag{6}$$

where each reward-wise advantage is estimated using GAE in a component-wise manner:

$$A_t^k = \sum_{l=0}^{T-t} (\gamma\lambda)^l \delta_{t+l}^k, \quad \delta_t^k = r_t^k + \gamma V_\phi^k(s_{t+1}) - V_\phi^k(s_t), \tag{7}$$

and $V_\phi^k(s_t)$ denotes the value estimate from the $k$-th value head corresponding to the $k$-th reward component. The aggregated advantage $A_t$ is then used in the PPO objective in Eq. (2).

Compared to vanilla PPO, which computes advantages from aggregated rewards, reward-wise PPO estimates advantages separately for each reward component and aggregates them at the policy update stage, enabling more fine-grained credit assignment across heterogeneous rewards. Please refer to the algorithm pseudo-code of reward-wise PPO with multi-head critic learning in Algorithm 1 for details.

## 5 Experiments

We conduct experiments on multi-turn agentic search and mathematical reasoning tasks to demonstrate the effectiveness of our method. We keep all hyperparameters and reward design the same for reward-wise PPO (RW-PPO) and vanilla PPO. All experiments are conducted on 8×H100 GPUs. Due to space limitations, experiments on mathematical reasoning tasks are provided in Appendix C.

### 5.1 Experimental Settings

**Models and Datasets.** We compare our proposed RW-PPO with vanilla PPO on multi-turn agentic search tasks. We train Qwen2.5-3B-Instruct and Qwen2.5-7B-Instruct [30] on a mixed training

Table 2: Parameter and memory comparison of base models and critics. Memory is estimated in BF16 (2 bytes per parameter). Values in parentheses denote the difference compared to the 1-head critic. For Qwen2.5-3B, the LM head is tied with the input embedding, resulting in negligible parameter difference between the full model and the critic. In contrast, Qwen2.5-7B does not tie word embeddings, so removing the LM head leads to a significant reduction in parameters.

| Model | Hidden Size | #Params | Memory (GB) |
|---|---|---|---|
| Qwen2.5-3B-Instruct (full) | 2048 | 3,085,938,688 | 6.17 |
| Critic (1-head) | 2048 | 3,085,940,736 | 6.17 |
| Critic (3-head) | 2048 | 3,085,944,832 (+4,096) | 6.17 (+8 KB) |
| Qwen2.5-7B-Instruct (full) | 3584 | 7,615,616,512 | 15.23 |
| Critic (1-head) | 3584 | 7,070,622,720 | 14.14 |
| Critic (3-head) | 3584 | 7,070,629,888 (+7,168) | 14.14 (+14 KB) |

dataset constructed from NQ [31] and HotpotQA [32], and evaluate the trained models on the test sets of NQ [31], TriviaQA [33], PopQA [34], HotpotQA [32], 2Wiki [35], MuSiQue [36], and Bamboogle [37], following the Search-R1 setting [3]. We use E5 [38] as the retriever, and 2018 Wikipedia dump [31] as the corpus. We observe training instability and occasional collapse. For fair comparison, we report the maximum total reward achieved across all training steps.

**Reward Design.** We design three terminal reward components: (1) *Format correctness.* This reward ensures structural validity by verifying that the outputs in both the final turn and all intermediate turns comply with the specified format. (2) *Retrieval correctness.* This reward evaluates whether any accepted answer appears in at least one search result (from the `<information>` tag), using case-insensitive string matching. (3) *Answer correctness.* This reward evaluates whether the extracted answer (from the `<answer>` tag) exactly matches any accepted ground-truth answer after normalization (e.g., lowercasing and whitespace removal). We assign reward values as $\mathcal{R}_{\text{format}} \in \{0, 0.2\}$, $\mathcal{R}_{\text{ret}} \in \{0, 0.3\}$, and $\mathcal{R}_{\text{ans}} \in \{0, 1.0\}$.

**PPO Training.** We enable gradient checkpointing and use FSDP2 with CPU offloading. The learning rates of the policy and critic models are set to $1.0 \times 10^{-6}$ and $1.0 \times 10^{-5}$, respectively. Training is performed for one epoch with one update epoch per batch. The total batch size is 512, with policy and critic mini-batch sizes of 256 and a micro-batch size of 4 per GPU. The maximum input length is set to 4,096 tokens, and the maximum generation length is set to 500 tokens. KL loss is disabled. We apply off-policy correction for token-level importance sampling, with the importance ratio clipped at 2.0. Specifically, we use RW-PPO with three value heads and vanilla PPO with a single value head for the critic.

**Rollout Generation.** We use vLLM with one inference engine per GPU, tensor parallel size 1, and a GPU memory utilization ratio of 0.5. We sample one response per prompt with a temperature of 1.0 and a top-$p$ value of 1.0. We allow up to 4 search turns and retrieve the top 3 documents for each search query.

### 5.2 Overhead Analysis

As shown in Table 2, we compare the parameter and memory overhead of the multi-head critic with the single-head critic across different model scales. The multi-head critic introduces negligible overhead compared to the single-head critic. Specifically, adding multiple value heads increases the parameter count by only a few thousand parameters (e.g., +4,096 for 3B and +7,168 for 7B), corresponding to only several kilobytes of additional memory. This overhead is negligible relative to the total model size (on the order of billions of parameters and tens of gigabytes).

### 5.3 Performance Comparison

As shown in Table 3, RW-PPO consistently outperforms vanilla PPO across both model scales and most datasets. Compared to PPO, RW-PPO achieves higher total rewards on all datasets for the 3B model and on nearly all datasets for the 7B model, demonstrating the effectiveness of reward-wise optimization. These gains are primarily driven by improvements in answer reward, which show larger relative increases than other reward components. The improvements are particularly

Table 3: Performance comparison between RW-PPO and PPO on multi-turn agentic search tasks. We report Answer, Format, Retrieval, and Total rewards. Values in parentheses indicate relative improvements over PPO. Avg denotes the average across all datasets. † and * denote in-domain and out-of-domain datasets, respectively.

| Dataset | Method | Answer (0–1) | Format (0–0.2) | Retrieval (0–0.3) | Total (0–1.5) |
|---|---|---|---|---|---|
| | | *Qwen2.5-3B-Instruct* | | | |
| NQ† | PPO | 0.3825 | 0.1914 | 0.2171 | 0.7911 |
| | RW-PPO | **0.4371** (+14.3%) | **0.1995** | 0.2171 | **0.8537** (+7.9%) |
| TriviaQA* | PPO | 0.5717 | 0.1922 | 0.2229 | 0.9868 |
| | RW-PPO | **0.5985** (+4.7%) | **0.1991** | **0.2246** | **1.0222** (+3.6%) |
| PopQA* | PPO | 0.4046 | 0.1946 | **0.1976** | 0.7968 |
| | RW-PPO | **0.4281** (+5.8%) | **0.1995** | 0.1972 | **0.8247** (+3.5%) |
| HotpotQA† | PPO | 0.3618 | 0.1924 | 0.1828 | 0.7369 |
| | RW-PPO | **0.3904** (+7.9%) | **0.1975** | **0.1873** | **0.7752** (+5.2%) |
| 2Wiki* | PPO | 0.3497 | 0.1757 | 0.1523 | 0.6777 |
| | RW-PPO | **0.3885** (+11.1%) | **0.1889** | **0.1566** | **0.7340** (+8.3%) |
| MuSiQue* | PPO | 0.1415 | 0.1764 | 0.1089 | 0.4268 |
| | RW-PPO | **0.1700** (+20.1%) | **0.1820** | **0.1143** | **0.4663** (+9.3%) |
| Bamboogle* | PPO | 0.3520 | 0.2000 | 0.1464 | 0.6984 |
| | RW-PPO | **0.4000** (+13.6%) | 0.2000 | **0.1560** | **0.7560** (+8.2%) |
| Avg | PPO | 0.3663 | 0.1889 | 0.1754 | 0.7306 |
| | RW-PPO | **0.4018** (+9.7%) | **0.1952** | **0.1790** | **0.7760** (+6.2%) |
| | | *Qwen2.5-7B-Instruct* | | | |
| NQ† | PPO | 0.4723 | 0.1987 | 0.2267 | 0.8977 |
| | RW-PPO | **0.4820** (+2.1%) | **0.2000** | **0.2344** | **0.9164** (+2.1%) |
| TriviaQA* | PPO | 0.6491 | 0.1979 | 0.2332 | 1.0802 |
| | RW-PPO | **0.6606** (+1.8%) | **0.1999** | **0.2358** | **1.0962** (+1.5%) |
| PopQA* | PPO | 0.4466 | 0.1992 | 0.1945 | 0.8403 |
| | RW-PPO | **0.4683** (+4.9%) | **0.2000** | **0.2030** | **0.8713** (+3.7%) |
| HotpotQA† | PPO | 0.4451 | 0.1965 | 0.2000 | 0.8416 |
| | RW-PPO | **0.4528** (+1.7%) | **0.1998** | **0.2040** | **0.8566** (+1.8%) |
| 2Wiki* | PPO | 0.3985 | 0.1778 | **0.1654** | 0.7417 |
| | RW-PPO | **0.4369** (+9.6%) | **0.1986** | 0.1606 | **0.7960** (+7.3%) |
| MuSiQue* | PPO | **0.2156** | 0.1809 | **0.1393** | 0.5357 |
| | RW-PPO | 0.2069 (-4.0%) | **0.1945** | 0.1343 | 0.5357 (+0.0%) |
| Bamboogle* | PPO | 0.4480 | **0.2000** | **0.1608** | 0.8088 |
| | RW-PPO | **0.4640** (+3.6%) | 0.1968 | 0.1536 | **0.8144** (+0.7%) |
| Avg | PPO | 0.4393 | 0.1930 | 0.1886 | 0.8209 |
| | RW-PPO | **0.4531** (+3.1%) | **0.1985** | **0.1894** | **0.8410** (+2.4%) |

notable on out-of-domain datasets such as 2Wiki, MuSiQue, and Bamboogle, where RW-PPO yields gains of up to 9.3% in total reward and over 20% in answer reward for the 3B model, indicating better generalization ability. On in-domain datasets (NQ and HotpotQA), RW-PPO also brings consistent improvements, though the gains are relatively smaller, suggesting that the benefits are more pronounced under distribution shift. Overall, RW-PPO improves the average total reward by 6.2% on the 3B model and 2.4% on the 7B model, while achieving larger gains in answer reward (+9.7% and +3.1%, respectively), showing stronger advantages on smaller models.

As shown in Figure 2, RW-PPO achieves more stable reward trajectories than vanilla PPO on multi-turn agentic search tasks. For the 3B model, PPO collapses in answer, format, and retrieval rewards, while RW-PPO keeps these reward components stable. The total return also remains higher under RW-PPO, suggesting that the multi-head critic learns better value estimates for advantage estimation. For the 7B model, training is more stable for both methods, but RW-PPO still yields higher answer reward, retrieval reward, and total return.

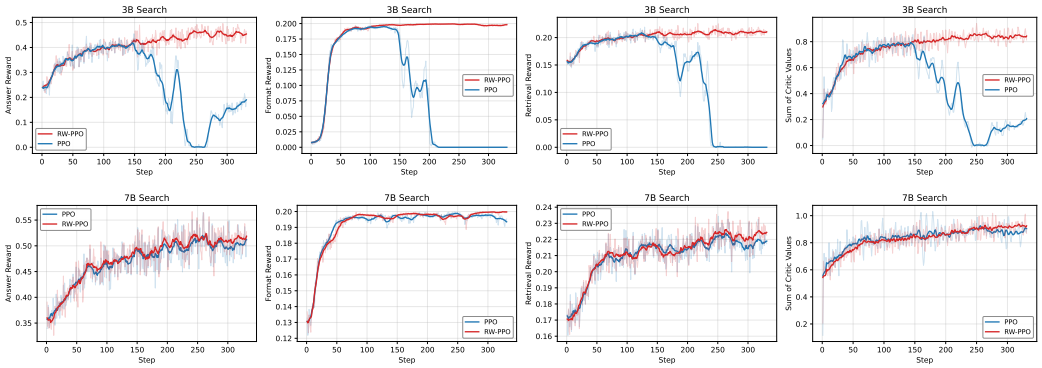


Figure 2: Training dynamics on multi-turn agentic search tasks. The top row shows results for Qwen2.5-3B-Instruct, and the bottom row shows results for Qwen2.5-7B-Instruct. From left to right, we report answer reward, format reward, retrieval reward, and total critic values during training. Compared with vanilla PPO, RW-PPO yields more stable reward trajectories and consistently improves the final total reward, especially for the 3B model.

Table 4: Answer reward comparison across methods. We compare GRPO and PPO with a single answer reward in Search-R1, PPO with multiple rewards, and RW-PPO with multiple rewards on multi-turn agentic search benchmarks. $^{\dagger}$ and $^{*}$ denote in-domain and out-of-domain datasets, respectively.

| **Method** | **NQ**$^{\dagger}$ | **TriviaQA**$^{*}$ | **PopQA**$^{*}$ | **HotpotQA**$^{\dagger}$ | **2Wiki**$^{*}$ | **MuSiQue**$^{*}$ | **Bamboogle**$^{*}$ | **Avg** |
|---|---|---|---|---|---|---|---|---|
| *Qwen2.5-3B-Instruct* | | | | | | | | |
| GRPO (single-reward) | 0.397 | 0.565 | 0.391 | 0.331 | 0.310 | 0.124 | 0.232 | 0.336 |
| PPO (single-reward) | 0.341 | 0.545 | 0.378 | 0.324 | 0.319 | 0.103 | 0.264 | 0.325 |
| PPO (multi-reward) | 0.383 | 0.572 | 0.405 | 0.362 | 0.350 | 0.142 | 0.352 | 0.366 |
| RW-PPO (multi-reward) | **0.437** | **0.599** | **0.428** | **0.390** | **0.389** | **0.170** | **0.400** | **0.402** |
| *Qwen2.5-7B-Instruct* | | | | | | | | |
| GRPO (single-reward) | 0.429 | 0.623 | 0.427 | 0.386 | 0.346 | 0.162 | 0.400 | 0.396 |
| PPO (single-reward) | 0.393 | 0.610 | 0.397 | 0.370 | 0.414 | 0.146 | 0.368 | 0.385 |
| PPO (multi-reward) | 0.472 | 0.649 | 0.447 | 0.445 | 0.399 | **0.216** | 0.448 | 0.439 |
| RW-PPO (multi-reward) | **0.482** | **0.661** | **0.468** | **0.453** | **0.437** | 0.207 | **0.464** | **0.453** |

### 5.4 Benchmark with Other RL Methods

We further benchmark against GRPO and PPO with a single answer reward under the Search-R1 setting [18, 3]. As shown in Table 4, using multiple rewards consistently improves over single-reward training, showing the benefit of reward shaping. RW-PPO further achieves the best average performance on both 3B and 7B models, demonstrating that reward-wise value estimation can better leverage heterogeneous reward signals.

## 6 Conclusion

In this paper, we propose reward-wise PPO with a multi-head critic for multi-reward RL in LLMs, enabling reward-wise value estimation and advantage computation while sharing a single efficient critic backbone. Our analysis shows that reward-wise critic learning avoids aggregation-induced cross-component noise, and experiments on multi-turn agentic search and mathematical reasoning tasks demonstrate consistent improvements in both answer reward and total reward over vanilla PPO. These results highlight reward-wise value estimation as an efficient and flexible approach for critic-based multi-reward optimization. One limitation of our work is that our experiments consider a relatively small number of reward components. An interesting future direction is to extend our method to large-scale multi-reward RL scenarios with hundreds of reward components.

# References


[1] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[2] Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. *arXiv preprint arXiv:2503.18892*, 2025.

[3] Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

[4] Zhenghai Xue, Longtao Zheng, Qian Liu, Yingru Li, Xiaosen Zheng, Zejun Ma, and Bo An. Simpletir: End-to-end reinforcement learning for multi-turn tool-integrated reasoning. *arXiv preprint arXiv:2509.02479*, 2025.

[5] 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.

[6] Wouter Kool, Herke van Hoof, and Max Welling. Buy 4 reinforce samples, get a baseline for free! 2019.

[7] Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce-style optimization for learning from human feedback in llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12248–12267, 2024.

[8] Yiran Guo, Lijie Xu, Jie Liu, Dan Ye, and Shuang Qiu. Segment policy optimization: Effective segment-level credit assignment in rl for large language models. *arXiv preprint arXiv:2505.23564*, 2025.

[9] Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training. *arXiv preprint arXiv:2505.10978*, 2025.

[10] Quan Wei, Siliang Zeng, Chenliang Li, William Brown, Oana Frunza, Wei Deng, Anderson Schneider, Yuriy Nevmyvaka, Yang Katie Zhao, Alfredo Garcia, and Mingyi Hong. Reinforcing multi-turn reasoning in llm agents via turn-level reward design, 2025.

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

[12] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[13] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[14] Zhixuan Liu, Zhanhui Zhou, Yuanfu Wang, Chao Yang, and Yu Qiao. Inference-time language model alignment via integrated value guidance. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 4181–4195, 2024.

[15] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

[16] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The twelfth international conference on learning representations*, 2023.

[17] Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for llm reasoning. *arXiv preprint arXiv:2410.08146*, 2024.

[18] Bowen Jin, Jinsung Yoon, Priyanka Kargupta, Sercan O Arik, and Jiawei Han. An empirical study on reinforcement learning for reasoning-search interleaved llm agents. *arXiv preprint arXiv:2505.15117*, 2025.

[19] Shu Zhao, Tan Yu, Anbang Xu, Japinder Singh, Aaditya Shukla, and Rama Akkiraju. Parallelsearch: Train your llms to decompose query and search sub-queries in parallel with reinforcement learning. *arXiv preprint arXiv:2508.09303*, 2025.

[20] Xuefeng Li, Haoyang Zou, and Pengfei Liu. Torl: Scaling tool-integrated rl. *arXiv preprint arXiv:2503.23383*, 2025.

[21] Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. Toolrl: Reward is all tool learning needs. *arXiv preprint arXiv:2504.13958*, 2025.

[22] Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean Hendryx. Rubrics as rewards: Reinforcement learning beyond verifiable domains. *arXiv preprint arXiv:2507.17746*, 2025.

[23] Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, S. H. Cai, Yuan Cao, Y. Charles, H. S. Che, Cheng Chen, Guanduo Chen, Huarong Chen, Jia Chen, Jiahao Chen, Jianlong Chen, Jun Chen, Kefan Chen, Liang Chen, Ruijue Chen, Xinhao Chen, Yanru Chen, Yanxu Chen, Yicun Chen, Yimin Chen, Yingjiang Chen, Yuankun Chen, Yujie Chen, Yutian Chen, Zhirong Chen, Ziwei Chen, Dazhi Cheng, Minghan Chu, Jialei Cui, Jiaqi Deng, Muxi Diao, Hao Ding, Mengfan Dong, Mengnan Dong, Yuxin Dong, Yuhao Dong, Angang Du, Chenzhuang Du, Dikang Du, Lingxiao Du, Yulun Du, Yu Fan, Shengjun Fang, Qiulin Feng, Yichen Feng, Garimugai Fu, Kelin Fu, Hongcheng Gao, Tong Gao, Yuyao Ge, Shangyi Geng, Chengyang Gong, Xiaochen Gong, Zhuoma Gongque, Qizheng Gu, Xinran Gu, Yicheng Gu, Longyu Guan, Yuanying Guo, Xiaoru Hao, Weiran He, Wenyang He, Yunjia He, Chao Hong, Hao Hu, Jiaxi Hu, Yangyang Hu, Zhenxing Hu, Ke Huang, Ruiyuan Huang, Weixiao Huang, Zhiqi Huang, Tao Jiang, Zhejun Jiang, Xinyi Jin, Yu Jing, Guokun Lai, Aidi Li, C. Li, Cheng Li, Fang Li, Guanghe Li, Guanyu Li, Haitao Li, Haoyang Li, Jia Li, Jingwei Li, Junxiong Li, Lincan Li, Mo Li, Weihong Li, Wentao Li, Xinhang Li, Xinhao Li, Yang Li, Yanhao Li, Yiwei Li, Yuxiao Li, Zhaowei Li, Zheming Li, Weilong Liao, Jiawei Lin, Xiaohan Lin, Zhishan Lin, Zichao Lin, Cheng Liu, Chenyu Liu, Hongzhang Liu, Liang Liu, Shaowei Liu, Shudong Liu, Shuran Liu, Tianwei Liu, Tianyu Liu, Weizhou Liu, Xiangyan Liu, Yangyang Liu, Yanming Liu, Yibo Liu, Yuanxin Liu, Yue Liu, Zhengying Liu, Zhongnuo Liu, Enzhe Lu, Haoyu Lu, Zhiyuan Lu, Junyu Luo, Tongxu Luo, Yashuo Luo, Long Ma, Yingwei Ma, Shaoguang Mao, Yuan Mei, Xin Men, Fanqing Meng, Zhiyong Meng, Yibo Miao, Minqing Ni, Kun Ouyang, Siyuan Pan, Bo Pang, Yuchao Qian, Ruoyu Qin, Zeyu Qin, Jiezhong Qiu, Bowen Qu, Zeyu Shang, Youbo Shao, Tianxiao Shen, Zhennan Shen, Juanfeng Shi, Lidong Shi, Shengyuan Shi, Feifan Song, Pengwei Song, Tianhui Song, Xiaoxi Song, Hongjin Su, Jianlin Su, Zhaochen Su, Lin Sui, Jinsong Sun, Junyao Sun, Tongyu Sun, Flood Sung, Yunpeng Tai, Chuning Tang, Heyi Tang, Xiaojuan Tang, Zhengyang Tang, Jiawen Tao, Shiyuan Teng, Chaoran Tian, Pengfei Tian, Ao Wang, Bowen Wang, Chensi Wang, Chuang Wang, Congcong Wang, Dingkun Wang, Dinglu Wang, Dongliang Wang, Feng Wang, Hailong Wang, Haiming Wang, Hengzhi Wang, Huaqing Wang, Hui Wang, Jiahao Wang, Jinhong Wang, Jiuzheng Wang, Kaixin Wang, Linian Wang, Qibin Wang, Shengjie Wang, Shuyi Wang, Si Wang, Wei Wang, Xiaochen Wang, Xinyuan Wang, Yao Wang, Yejie Wang, Yipu Wang, Yiqin Wang, Yucheng Wang, Yuzhi Wang, Zhaoji Wang, Zhaowei Wang, Zhengtao Wang, Zhexu Wang, Zihan Wang, Zizhe Wang, Chu Wei, Ming Wei, Chuan Wen, Zichen Wen, Chengjie Wu, Haoning Wu, Junyan Wu, Rucong Wu, Wenhao Wu, Yuefeng Wu, Yuhao Wu, Yuxin Wu, Zijian Wu, Chenjun Xiao, Jin Xie, Xiaotong Xie, Yuchong Xie, Yifei Xin, Bowei Xing, Boyu Xu, Jianfan Xu, Jing Xu, Jinjing Xu, L. H. Xu, Lin Xu, Suting Xu, Weixin Xu, Xinbo Xu, Xinran Xu, Yangchuan Xu, Yichang Xu, Yuemeng Xu, Zelai Xu, Ziyao Xu, Junjie Yan, Yuzi Yan, Guangyao Yang, Hao Yang, Junwei Yang, Kai Yang, Ningyuan Yang, Ruihan Yang, Xiaofei Yang, Xinlong Yang, Ying Yang, Yi Yang, Yi Yang, Zhen Yang, Zhilin Yang, Zonghan Yang, Haotian Yao, Dan Ye, Wenjie Ye, Zhuorui Ye, Bohong Yin, Chengzhen Yu,

Longhui Yu, Tao Yu, Tianxiang Yu, Enming Yuan, Mengjie Yuan, Xiaokun Yuan, Yang Yue, Weihao Zeng, Dunyuan Zha, Haobing Zhan, Dehao Zhang, Hao Zhang, Jin Zhang, Puqi Zhang, Qiao Zhang, Rui Zhang, Xiaobin Zhang, Y. Zhang, Yadong Zhang, Yangkun Zhang, Yichi Zhang, Yizhi Zhang, Yongting Zhang, Yu Zhang, Yushun Zhang, Yutao Zhang, Yutong Zhang, Zheng Zhang, Chenguang Zhao, Feifan Zhao, Jinxiang Zhao, Shuai Zhao, Xiangyu Zhao, Yikai Zhao, Zijia Zhao, Huabin Zheng, Ruihan Zheng, Shaojie Zheng, Tengyang Zheng, Junfeng Zhong, Longguang Zhong, Weiming Zhong, M. Zhou, Runjie Zhou, Xinyu Zhou, Zaida Zhou, Jinguo Zhu, Liya Zhu, Xinhao Zhu, Yuxuan Zhu, Zhen Zhu, Jingze Zhuang, Weiyu Zhuang, Ying Zou, and Xinxing Zu. Kimi k2.5: Visual agentic intelligence, 2026.

[24] 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.

[25] Yingru Li, Jiawei Xu, Ziniu Li, Jiacai Liu, Wei Liu, Yuxuan Tong, Longtao Zheng, Zhenghai Xue, Yaxiang Zhang, Tianle Cai, et al. The optimal token baseline: Variance reduction for long-horizon llm-rl. *arXiv preprint arXiv:2602.07078*, 2026.

[26] Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, et al. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025.

[27] Wei Liu, Ruochen Zhou, Yiyun Deng, Yuzhen Huang, Junteng Liu, Yuntian Deng, Yizhe Zhang, and Junxian He. Learn to reason efficiently with adaptive length-based reward shaping. *arXiv preprint arXiv:2505.15612*, 2025.

[28] Pranjal Aggarwal and Sean Welleck. L1: Controlling how long a reasoning model thinks with reinforcement learning. *arXiv preprint arXiv:2503.04697*, 2025.

[29] Jie Zhu, Yiyang Su, and Xiaoming Liu. Can textual reasoning improve the performance of mllms on fine-grained visual classification? *arXiv preprint arXiv:2601.06993*, 2026.

[30] Qwen Team. Qwen2.5: A party of foundation models, September 2024.

[31] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick SH Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, pages 6769–6781, 2020.

[32] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*, 2018.

[33] Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In Regina Barzilay and Min-Yen Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[34] Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. *arXiv preprint arXiv:2212.10511*, 2022.

[35] Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. *arXiv preprint arXiv:2011.01060*, 2020.

[36] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

[37] Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711, 2023.

[38] Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022.

[39] DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.

[40] 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. *arXiv preprint arXiv:2503.14476*, 2025.

# A Detailed Derivations of Parameter Count and Forward FLOPs

In this section, we provide detailed derivations of the parameter count and forward computational cost (FLOPs) for a Transformer-based critic model.

## A.1 Parameter Count

We consider a Transformer-based critic with hidden dimension $h$, vocabulary size $v$, and $l$ Transformer layers. The total parameter count consists of three parts: input embedding, Transformer backbone, and output head.

Table 5: Parameter count of each component in a Transformer-based critic. Here, $h$ denotes the hidden dimension, $v$ the vocabulary size, and $l$ the number of layers.

| **Component** | **Parameter Count** |
|---|---|
| Input Embedding | $vh$ |
| Q, K, V projections | $3h^2$ |
| Output projection | $h^2$ |
| Feed-Forward Network | $8h^2$ |
| Per-layer total | $12h^2$ |
| Backbone ($l$ layers) | $12lh^2$ |
| Output Head | $h$ |
| **Total** | $12lh^2 + vh + h$ |

The attention module consists of four projection matrices (query, key, value, and output), contributing $4h^2$ parameters per layer. The feed-forward network (FFN) expands the hidden dimension from $h$ to $4h$ and projects it back, contributing $8h^2$ parameters. Therefore, each layer has $12h^2$ parameters, and the total backbone parameter count is $12lh^2$. The embedding layer contributes $vh$ parameters, and the value head contributes $h$ parameters.

## A.2 Forward FLOPs

We now derive the forward computational cost (FLOPs). Let $b$ denote the batch size and $s$ the sequence length.

Table 6: Forward FLOPs of each component in a Transformer-based critic. Here, $b$ denotes the batch size, $s$ the sequence length, $h$ the hidden dimension, and $l$ the number of layers.

| **Component** | **Forward FLOPs** |
|---|---|
| Input Embedding | negligible |
| Q, K, V projections | $3bsh^2$ |
| Output projection | $bsh^2$ |
| Attention computation | $2bs^2h$ |
| Feed-Forward Network | $8bsh^2$ |
| Per-layer total | $12bsh^2 + 2bs^2h$ |
| Backbone ($l$ layers) | $bl(12sh^2 + 2s^2h)$ |
| Output Head | $bsh$ |
| **Total** | $b\left[l(12sh^2 + 2s^2h) + sh\right]$ |

Although the embedding layer can be written as a matrix multiplication with one-hot inputs, it is implemented as a lookup operation in practice and thus incurs negligible FLOPs. For each Transformer layer, the main computational cost comes from linear projections, attention computation, and the feed-forward network, resulting in $12bsh^2 + 2bs^2h$ FLOPs per layer. Therefore, the total backbone cost is $bl(12sh^2 + 2s^2h)$. The output head introduces an additional $bsh$ FLOPs.

### A.3 Discussion

From Table 5 and Table 6, we observe that both the parameter count and computational cost are dominated by the Transformer backbone, particularly the $O(lh^2)$ parameters and $O(blsh^2)$ FLOPs. In contrast, the output head contributes negligible overhead. This observation forms the basis for the efficiency of the proposed multi-head critic design.

## B Proof of Proposition 1

*Proof.* For the reward-wise critic loss, we have

$$\mathcal{L}_{\text{critic}}^{\text{rw}} = \sum_{k=1}^{K} \left(V_\phi^k(s_t) - R_t^k\right)^2.$$

Substituting $R_t^k = \mu_t^k + \epsilon_t^k$ yields

$$\mathcal{L}_{\text{critic}}^{\text{rw}} = \sum_{k=1}^{K} \left(V_\phi^k(s_t) - \mu_t^k - \epsilon_t^k\right)^2 = \sum_{k=1}^{K} \left(e_t^k - \epsilon_t^k\right)^2.$$

Taking expectation and using $\mathbb{E}[\epsilon_t^k \mid s_t] = 0$, we obtain

$$\mathbb{E}[\mathcal{L}_{\text{critic}}^{\text{rw}}] = \sum_{k=1}^{K} \mathbb{E}\left[(e_t^k)^2\right] + \sum_{k=1}^{K} \mathbb{E}\left[(\epsilon_t^k)^2\right].$$

For the vanilla critic loss, the aggregated return is

$$R_t = \sum_{k=1}^{K} R_t^k = \sum_{k=1}^{K} \mu_t^k + \sum_{k=1}^{K} \epsilon_t^k.$$

Thus,

$$\mathcal{L}_{\text{critic}} = \left(V_\phi(s_t) - \sum_{k=1}^{K} R_t^k\right)^2 = \left(e_t - \sum_{k=1}^{K} \epsilon_t^k\right)^2.$$

Taking expectation and using the zero-mean noise assumption, we obtain

$$\mathbb{E}[\mathcal{L}_{\text{critic}}] = \mathbb{E}\left[e_t^2\right] + \mathbb{E}\left[\left(\sum_{k=1}^{K} \epsilon_t^k\right)^2\right].$$

Expanding the squared sum yields

$$\mathbb{E}\left[\left(\sum_{k=1}^{K} \epsilon_t^k\right)^2\right] = \sum_{k=1}^{K} \mathbb{E}\left[(\epsilon_t^k)^2\right] + 2\sum_{i<j} \mathbb{E}\left[\epsilon_t^i \epsilon_t^j\right],$$

which completes the proof. □

## C Experiments on Math Reasoning

### C.1 Experimental Settings

**Experimental Setup.** We compare our proposed RW-PPO with vanilla PPO on mathematical reasoning tasks. We train DeepSeek-R1-Distill-Qwen-1.5B and DeepSeek-R1-Distill-Qwen-7B [39] on the DAPO-Math-17k [40] dataset, and evaluate them on AIME 2024, AIME 2025, AIME 2026, and AMC datasets. The maximum sequence length is set to 8k tokens.

**Reward Design.** We design three terminal reward components: (1) *Format correctness*: the model is required to produce outputs with the correct structure, i.e., using XML tags to separate the reasoning (`<think>`) and the final answer (`<answer>`). (2) *Integer correctness*: whether the predicted answer

is a valid integer. (3) *Answer correctness*: whether the final answer matches the ground-truth answer. We set the reward values as $\mathcal{R}_{\text{format}} \in \{0, 1.0\}$, $\mathcal{R}_{\text{int}} \in \{0, 1.0\}$, and $\mathcal{R}_{\text{ans}} \in \{0, 1.0\}$.

**PPO Training.** We enable gradient checkpointing and use FSDP with parameter and optimizer offloading. The learning rates of the policy and critic models are set to $1.0 \times 10^{-6}$ and $1.0 \times 10^{-5}$, respectively. Training is performed for one epoch. The total batch size is 1,024, with a mini-batch size of 256 and a micro-batch size of 4 per GPU. The maximum prompt length is set to 2,048 tokens, and the maximum response length is set to 8,192 tokens. KL loss is disabled. We apply token-level importance sampling for rollout correction, with the importance ratio clipped at 2.0. Specifically, we use RW-PPO with three value heads and vanilla PPO with a single value head for the critic.

**Rollout Generation.** We use vLLM for rollout generation with tensor parallel size 1 and a GPU memory utilization ratio of 0.5. During training, we sample one response per prompt with a temperature of 1.0 and a top-$p$ value of 1.0. During validation, we sample 16 responses per prompt with a temperature of 0.7 and a top-$p$ value of 0.95.

### C.2 Experiment Results

As shown in Table 7, RW-PPO consistently outperforms vanilla PPO across all datasets and model scales. RW-PPO achieves higher total rewards on every benchmark for both the 1.5B and 7B models, demonstrating the effectiveness of reward-wise optimization in mathematical reasoning tasks. These improvements are primarily driven by gains in answer reward, which show significantly larger relative increases compared to format and integer rewards that are already near saturation. The improvements are especially pronounced on more challenging benchmarks such as AIME 2025 and AIME 2026, where RW-PPO yields substantial gains in answer reward (e.g., up to 29.0% for the 1.5B model), leading to consistent improvements in total reward. In contrast, format and integer rewards remain close to their upper bounds for both methods, indicating that RW-PPO mainly improves reasoning accuracy rather than surface-level correctness. Overall, RW-PPO improves the average total reward by 2.3% on the 1.5B model and 1.6% on the 7B model, while achieving larger gains in answer reward (+11.3% and +5.5%, respectively), showing stronger benefits on smaller models.

Table 7: Performance comparison between RW-PPO and PPO on mathematical reasoning benchmarks. We report Answer, Format, Integer, and Total rewards averaged over 16 samples. Values in parentheses indicate relative improvements over PPO.

| Dataset | Method | Answer (0–1) | Format (0–1) | Integer (0–1) | Total (0–3) |
|---|---|---|---|---|---|
| *DeepSeek-R1-Distill-Qwen-1.5B* | | | | | |
| AIME 2024 | PPO | 0.1396 | 0.9563 | 0.9563 | 2.0521 |
| | RW-PPO | **0.1521** (+8.9%) | **0.9667** | **0.9667** | **2.0854** (+1.6%) |
| AIME 2025 | PPO | 0.1583 | 0.9854 | 0.9833 | 2.1271 |
| | RW-PPO | **0.2042** (+29.0%) | **0.9979** | **0.9979** | **2.2000** (+3.4%) |
| AIME 2026 | PPO | 0.1500 | 0.9729 | 0.9729 | 2.0958 |
| | RW-PPO | **0.1667** (+11.1%) | **0.9917** | **0.9896** | **2.1479** (+2.5%) |
| AMC | PPO | 0.6516 | 0.9922 | 0.9906 | 2.6344 |
| | RW-PPO | **0.7000** (+7.4%) | 0.9922 | **0.9922** | **2.6844** (+1.9%) |
| Avg | PPO | 0.2749 | 0.9767 | 0.9758 | 2.2274 |
| | RW-PPO | **0.3058** (+11.3%) | **0.9871** | **0.9866** | **2.2794** (+2.3%) |
| *DeepSeek-R1-Distill-Qwen-7B* | | | | | |
| AIME 2024 | PPO | 0.2313 | 0.9875 | 0.9875 | 2.2062 |
| | RW-PPO | **0.2500** (+8.1%) | **0.9958** | **0.9958** | **2.2417** (+1.6%) |
| AIME 2025 | PPO | 0.2792 | 0.9938 | 0.9938 | 2.2667 |
| | RW-PPO | **0.3083** (+10.4%) | **1.0000** | **1.0000** | **2.3083** (+1.8%) |
| AIME 2026 | PPO | 0.3229 | 0.9750 | 0.9750 | 2.2729 |
| | RW-PPO | **0.3583** (+11.0%) | **0.9917** | **0.9917** | **2.3417** (+3.0%) |
| AMC | PPO | 0.8281 | 0.9984 | 0.9984 | 2.8250 |
| | RW-PPO | **0.8359** (+0.9%) | **1.0000** | 0.9984 | **2.8344** (+0.3%) |
| Avg | PPO | 0.4154 | 0.9887 | 0.9887 | 2.3927 |
| | RW-PPO | **0.4381** (+5.5%) | **0.9969** | **0.9965** | **2.4315** (+1.6%) |

# NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the claims made

   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: The paper has limitations discussed in the conclusion part in Section 6

   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: [Yes]

Justification: The paper provides theoretical results with assumptions and proofs in Section 4

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: The experiments can be reproduced by following experimental setup in Section 5

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 data is open-sourced but the code is not open-sourced

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: The experimental setting are clearly presented

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: [No]

Justification: The paper follows most of related work to conduct experiments

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: 8x H100 GPUs

   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: The paper is conducted with 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: [N/A]

    Justification: There is no societal impact of the work performed.

    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: The paper poses no such risks.

    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: The assets are properly credited and the licenses are properly respected

    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: The paper does not release 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: The paper does not involve crowdsourcing nor research with human subjects

    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: The paper does not involve crowdsourcing nor research with human subjects.

    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: The core method development in this research does not involve LLMs as any important, original, or non-standard components

- 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.