OFF-POLICY VALUE-BASED REINFORCEMENT LEARNING FOR LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Improving data utilization efficiency is critical for scaling reinforcement learning (RL) for long-horizon tasks where generating trajectories is expensive. However, the dominant RL methods for LLMs are largely *on-policy*: they update each batch of data only once, discard it, and then collect fresh samples, resulting in poor sample efficiency. In this work, we explore an alternative *value-based* RL framework for LLMs that naturally enables *off-policy* learning. We propose ReVal, a Bellman-update-based method that combines stepwise signals capturing internal consistency with trajectory-level signals derived from outcome verification. ReVal naturally supports replay-buffer-based training, allowing efficient reuse of past trajectories. Experiments on standard mathematical reasoning benchmarks show that ReVal not only converges faster but also outperforms GRPO in final performance. On DeepSeek-R1-Distill-1.5B, ReVal improves training efficiency and achieves improvement of **2.7%** in AIME24 and **4.5%** in out-of-domain benchmark GPQA over GRPO. These results suggest that value-based RL is a practical alternative to policy-based methods for LLM training.

1 INTRODUCTION

Since the advent of reinforcement learning from human feedback (RLHF), reinforcement learning (RL) has become a central component of large language model (LLM) post-training (Guo et al., 2025; Team et al., 2025; Li et al., 2025b; Wang et al., 2026). In particular, reinforcement learning with verifiable rewards (RLVR) has proven highly effective for improving the reasoning ability of LLMs by training them from correctness signals on complete responses (Lambert et al., 2024; OpenAI, 2024; Guo et al., 2025). Because autoregressive LLMs are naturally parameterized as token-

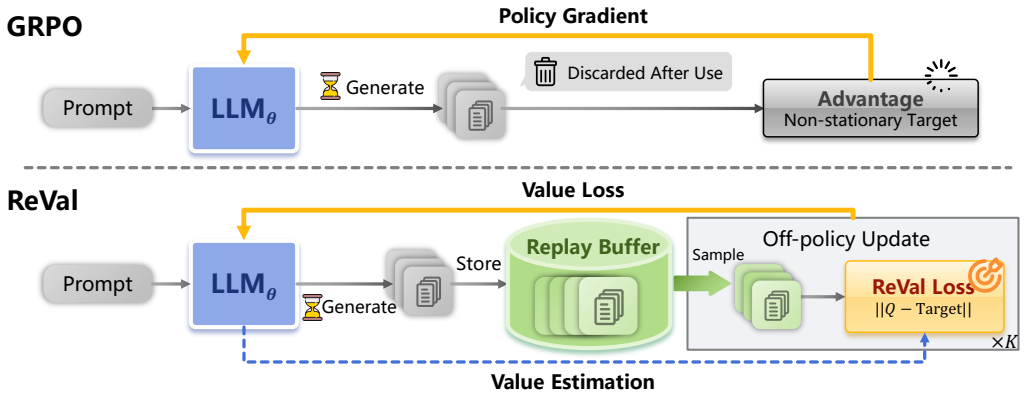


Figure 1: Framework of ReVal. By interpreting LLM logits as Q-values, ReVal unifies policy and value within a single model and enables replay-based off-policy updates.

level policies, actor-critic policy optimization algorithms such as PPO (Schulman et al., 2017) initially became the dominant approach for RL-based post-training. They offered a stable and conceptually straightforward framework for optimizing pretrained language models. As the field matured,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

however, it became clear that at LLM scale, an RL algorithm is only practical if it is computationally efficient. ReMax (Li et al., 2024) was the first to move from actor-critic to actor-only RL, significantly reducing memory usage and training time for LLM post-training. Following ReMax, a series of methods, including GRPO (Shao et al., 2024) and DAPO (Yu et al., 2025), further advanced this low-cost policy optimization paradigm.

However, these actor-only methods remain fundamentally *on-policy*. Updates must be computed from data sampled from the current policy, so collected trajectories quickly become stale and can only be reused to a limited extent. For short-horizon tasks, this inefficiency may be acceptable. But LLM development is increasingly shifting toward *agentic* settings with long and highly variable horizons, where trajectory collection is expensive and often dominates the total training cost (Gao et al., 2025; Team et al., 2026). In this regime, reducing per-update overhead is no longer sufficient. We discuss the limitations of on-policy methods in detail in Appendix E. The next fundamental requirement is the ability to *reuse experience*, that is, RL for LLMs must become off-policy.

In classical RL, this naturally points to value-based methods, whose efficiency comes from Bellman learning and whose replay-buffer-based training readily supports off-policy data reuse (Watkins & Dayan, 1992; Mnih et al., 2013; 2015). Yet despite these advantages, standard value-based formulations are not directly compatible with LLM post-training. Conventional value-based RL typically relies on a value model that explicitly predicts values, which is apparently incompatible with LLMs. More importantly, introducing such an additional value model would undermine the very low-cost property that makes actor-only methods attractive at LLM scale by increasing both memory and computation overhead. A recent work by Li et al. (2025a) offers a way around this obstacle. It shows that the logits of a pretrained LLM can be interpreted as parameterizing action values of an endogenous reward, up to a state-dependent transformation. If logits can serve as Q-values, then policy and value no longer need to be represented by separate models. Once a pretrained LLM is viewed as an endogenous value model, a single-model, low-cost, off-policy RL algorithm becomes possible. This perspective leads directly to our method.

In this paper, we propose ReVal, a value-based RL framework for LLM post-training that preserves the efficiency advantages of ReMax while introducing the off-policy capability required for agentic and long-horizon learning. Our method is built on two key principles. First, on the *objective* side, effective value learning for LLMs should combine supervision at different temporal scales: stepwise signals provide dense feedback by encouraging internal consistency, while trajectory-level signals convey outcome-level correctness from verification. Either signal alone is insufficient: stepwise feedback by itself is inefficient at reflecting trajectory-level outcomes, whereas trajectory-level Bellman learning alone, as in Yuan et al. (2025), can suffer from mis-calibration at initialization. We therefore introduce a reward-shaping formulation that naturally integrates both stepwise and trajectory-level signals, yielding substantially more stable optimization. Second, on the *data* side, value-based RL should fully exploit replay. We therefore introduce a replay-buffer training mechanism that reuses historical trajectories, converting what would otherwise be discarded rollouts into useful supervision. Together, these designs yield a practical single-model value-learning algorithm that is both computationally efficient and genuinely off-policy.

Empirically, we first show that increased off-policy reuse directly accelerates learning. With more frequent replay updates, ReVal reaches comparable performance substantially faster, achieving an average $4.3\times$ speedup over GRPO. We then evaluate ReVal on standard mathematical reasoning benchmarks and find that it consistently outperforms strong policy-based baselines in both convergence speed and final accuracy. On DeepSeek-R1-Distill-1.5B, ReVal improves over GRPO by **2.7%** on AIME24 and **4.5%** on the out-of-domain benchmark GPQA. On Qwen2.5-Math-7B, it further surpasses GRPO by **4.3%** on GPQA. The advantage is pronounced in the limited-rollout setting ($N = 1$), where fresh trajectories are scarce and off-policy reuse is especially valuable: in this setting, ReVal exceeds GRPO by **4.8%** on AIME and **4.6%** on GPQA. We further provide ablations on KL regularization, the hyperparameter β, and reward design, yielding practical guidance for stable value-based RL in LLMs. Overall, ReVal shows that it is possible to make RL *off-policy* without making it more expensive, by unifying value and policy within the pretrained LLM itself.

2 PRELIMINARIES

LLM and its MDP Formulation An LLM π generates tokens from a finite vocabulary $\mathcal{V} = \{1, 2, \ldots, |\mathcal{V}|\}$ and generates a sequence in an autoregressive manner. At step h, given a context sequence $(a_1, \ldots, a_{h-1})$, an LLM produces the next token according to the conditional distribution, namely, $a_h \sim \pi(\cdot|a_1, \ldots, a_{h-1})$. This process continues until a designated end-of-sequence (EOS) token is generated or a predefined maximum length H is reached. We adopt the Markov decision process (MDP) formulation of LLMs from (Li et al., 2024), defined by the tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{V}, r, P, \rho, H \rangle$. The state space $\mathcal{S}$ is the set of all finite-length strings formed by the concatenation of elements in $\mathcal{V}$ and the action space is the vocabulary set $\mathcal{V}$. When generating a response, the initial state (prompt) $s_1 = (x_1, x_2, \cdots, x_m)$ is sampled from the initial state distribution ρ, with $m \in \mathbb{N}$ and $\forall i \in [m], x_i \in \mathcal{V}$. At each step $h \in [H]$, the LLM selects an action (or equivalently, a token) $a_h \in \mathcal{V}$ according to $\pi(\cdot|s_h)$. The environment then transits to the next state $s_{h+1} = (x, a_1, \cdots, a_h)$, rewarding the LLM with $r(s_h, a_h) \in [0, 1]$. That is, the transition model $P : \mathcal{S} \times \mathcal{V} \to \Delta(\mathcal{S})$ is usually deterministic. $P(s_{h+1}|s_h, a_h) = 1$ if and only if $s_{h+1} = s_h \oplus a_h$, where $\oplus$ means concatenation. The trajectory ends after a total of H steps. In the context of RL, we also call π as a policy. Throughout this paper, the terms "policy" and "LLM" will be used interchangeably.

Reinforcement Learning with Verifiable Reward Reinforcement Learning with Verifiable Reward (RLVR) has become a widely adopted paradigm in LLM reasoning following recent breakthroughs such as OpenAI-o1 (OpenAI, 2024) and DeepSeek-R1 (Guo et al., 2025). Unlike RLHF, which relies on learned reward models, RLVR trains LLMs by maximizing a *rule-based outcome reward* with KL-regularization:

$$\max_{\theta} \mathbb{E}_{x\sim\rho}\Big[\mathbb{E}_{a_{1:H}\sim\pi_\theta(\cdot|x)}\big[r_{\text{rule}}(x, a_{1:H})\big] - \beta\, D_{\text{KL}}\big(\pi_\theta(\cdot \mid x), \pi_{\text{ref}}(\cdot \mid x)\big)\Big]. \tag{1}$$

This rule-based reward r_{rule} evaluates the correctness of the final answer based on deterministic verification procedures. For mathematical tasks, one can directly compare the final answer in the response with the ground-truth answer, checking for mathematical equivalence. Besides, $D_{\text{KL}}(\pi_\theta(\cdot|x), \pi_{\text{ref}}(\cdot|x)) = \sum_{a_{1:H}} \pi_\theta(a_{1:H}|x) \log(\pi_\theta(a_{1:H}|x)/\pi_{\text{ref}}(a_{1:H}|x))$ denotes the KL divergence, which prevents the learning model from deviating too far from the reference model and $\beta > 0$ controls the regularization strength.

3 PROPOSED METHOD

3.1 TOWARDS VALUE-BASED RL FOR LLMS

Q-Function Parameterization in LLMs. A key challenge in applying value-based RL to LLMs is how to represent or initialize the Q-function. In standard Q-learning, the Q-function is parameterized as a mapping from a state to a vector of Q-values over all actions (Watkins & Dayan, 1992; Mnih et al., 2015), i.e., $f(s_h) \to \mathbb{R}^{|\mathcal{A}|}$, enabling greedy action selection by taking the $\arg\max$ over the output. Unlike standard RL settings where such a Q-function can be learned from a randomly initialized network, this approach is infeasible for LLMs for two reasons. First, the token vocabulary constitutes an enormous action space and rewards are typically sparse, making it difficult to learn a reliable Q-function from outcome signals alone. Second, learning a Q-function from scratch requires a large amount of data, whereas the amount of data available in RL fine-tuning is far from sufficient.

Li et al. (2025a) established a principled solution to this challenge. They showed that a language model trained via next-token prediction implicitly learns a soft Q-function: given a language model $\hat{\pi}$ parameterized as $\hat{\pi}(\cdot \mid s_h) = \text{softmax}(\hat{f}(s_h, \cdot))$, the logits $\hat{f}(s_h, a_h)$ directly correspond to the soft Q-values of the data-generating policy. Q-values can be learned from the implicit rewards in pre-training data through an inverse RL formulation. This reveals that LLM logits are not arbitrary scores but encode value-relevant information about token-level decisions, providing a well-initialized Q-function for free.

TBRM. Similar to Li et al. (2025a), TBRM (Yuan et al., 2025) adopts the logit-as-Q:

$$Q_\theta(s_h, a_h) := \text{logit}_\theta(s_h, a_h),$$

where the LLM's own logits serve as the Q-function without requiring a separate Q-value network. Given this Q-function parameterization, TBRM learns by minimizing the trajectory-level Bellman residual in KL-regularized RL framework. TBRM minimizes the trajectory-level Bellman residual:

$$\mathcal{L}_{\text{TBRM}}(\theta) = \frac{1}{|\hat{\mathcal{D}}|} \sum_{\tau \in \hat{\mathcal{D}}} \left(\log \pi_\theta(\tau) - \log \pi_{\text{ref}}(\tau) - \frac{r_{\text{rule}}(\tau)}{\beta} + V_\theta(s_1) \right)^2,$$

where $\hat{\mathcal{D}}$ denotes the on-policy data which is collected from current policy. $\pi(\tau) = \prod_{h=1}^{H} \pi(a_h|s_h)$ denotes the probability of trajectory τ and $V_\theta(s_1)$ is the induced V-function, $V_\theta(s_1) = \log \sum_{a \in \mathcal{A}} \exp Q(s_1, a)$.

TBRM showed empirical success with on-policy data. However, we have identified that TBRM does not satisfy *Calibrated Initialization*, which leads to spurious policy drift in the absence of reward signals. In the next section, we propose our training objective to address these limitations.

3.2 Off-Policy Value-Based Reinforcement Learning with Replay Buffer (ReVal)

We begin by examining the limitations of the TBRM training objective. First, we show that a desirable property for the training objective is defined as follows.

Definition 1 (Calibrated Initialization)**.** *A training objective satisfies* Calibrated Initialization *if, when $r_{rule} = 0$, the optimal policy under the KL-regularized RL objective reduces to the reference policy, i.e., $\pi^* = \pi_{ref}$.*

At the beginning of training, when no reward signal is available (i.e., $r_{\text{rule}} = 0$), the desired behavior is to leave the policy unchanged, i.e., $\pi^* = \pi_{\text{ref}}$. If this property is not satisfied, the model will still produce parameter updates, leading to spurious policy drift.

Proposition 1. *TBRM does not satisfy* Calibrated Initialization*. Specifically, setting $r(\tau) = 0$ in the TBRM objective does not yield $\pi^* = \pi_{ref}$ as the optimal solution.*

However, TBRM does not satisfy this property, as stated in Proposition 1. Setting $r(\tau) = 0$ in the TBRM objective yields:

$$\mathcal{L}_{\text{TBRM}}(\theta) = \left(V_\theta(s_1) + \sum_{h=1}^{H} \log \frac{\pi_\theta(a_h \mid s_h)}{\pi_{\text{ref}}(a_h \mid s_h)} \right)^2.$$

Minimizing this drives the squared term toward zero, which requires the log-likelihood ratio to cancel $V_\theta(s_1)$ rather than merely minimizing the KL divergence between π_θ and π_{ref}. As a result, the optimal solution does not correspond to matching the reference policy. We provide a detailed discussion and empirical verification in Appendix F.1.

To address this issue, we introduce reward shaping to redefine the Bellman objective. Specifically, we define a *modified reward* function:

$$R_\beta(s_h, a_h) := \frac{r_{\text{rule}}(s_h, a_h)}{\beta} + \log \pi_{\text{ref}}(a_h \mid s_h) + \underbrace{V_\theta(s_h) - V_{\text{ref}}(s_h)}_{\text{reward shaping term}},$$

where the first term is the scaled environment reward, and the remaining terms form an endogenous reward (Li et al., 2025a) guided by the reference policy. Intuitively, the endogenous reward incorporates the reference policy to guide the model, and the reward shaping term introduces a state-dependent offset that does not affect the optimal solution (Ng et al., 1999).

Based on this modified reward, we define the Bellman operator as:

$$(\mathcal{T}_\beta Q)(s_h, a_h) = \underbrace{\frac{r_{\text{rule}}(s_h, a_h)}{\beta}}_{\text{task reward}} + \log \pi_{\text{ref}}(a_h \mid s_h) + \underbrace{V_{\text{ref}}(s_h) - V_{\text{ref}}(s_{h+1})}_{\text{reward shaping term}}$$
$$+ \mathbb{E}_{s_{h+1} \sim \mathcal{P}(\cdot|s_h, a_h)} \left[\log \sum_{a \in \mathcal{A}} \exp Q(s_{h+1}, a) \right],$$

where $V_{\text{ref}}(s_h) = \log \sum_{a \in \mathcal{A}} \exp Q_{\text{ref}}(s_h, a)$ and $\pi_{\text{ref}}(\cdot \mid s_h) = \text{softmax}(Q_{\text{ref}}(s_h, \cdot))$. The trajectory-level Bellman residual loss is then:

$$\mathcal{L}_{\text{ReVal}}(\theta) = \frac{1}{|\mathcal{D}|} \sum_{\tau \in \mathcal{D}} \left(\sum_{h=1}^{H} Q_\theta(s_h, a_h) - (\mathcal{T}_\beta Q_\theta)(s_h, a_h) \right)^2$$
$$= \frac{1}{|\mathcal{D}|} \sum_{\tau \in \mathcal{D}} \left(V_\theta(s_1) - V_{\text{ref}}(s_1) + \log \pi_\theta(\tau) - \frac{r_{\text{rule}}(\tau)}{\beta} - \log \pi_{\text{ref}}(\tau) \right)^2. \quad (2)$$

Where D denotes the off-policy data. This formulation ensures *Calibrated Initialization*, as stated in the following proposition (the proof is shown in Appendix C). The Eq. 2 is used for optimizing as shown in Algorithm 1.

Proposition 2. *ReVal satisfies Calibrated Initialization. Consider the objective in Eq. 2. When* $r = 0$ *and the policy is initialized as* $\pi_\theta = \pi_{ref}$*, we have* $V_\theta(s_1) = V_{ref}(s_1)$ *and* $\log \pi_\theta(\tau) = \log \pi_{ref}(\tau)$*, which yields* $\mathcal{L}_{ReVal}(\theta) = 0$.

3.3 Replay Buffer for Off-Policy Learning

A key advantage of value-based RL over policy gradient methods is its natural compatibility with off-policy data. TBRM operates in an on-policy manner, discarding each batch of trajectories after a single update. ReVal introduces a replay buffer $\mathcal{D}_{\text{replay}}$ that stores historical trajectories and enables off-policy learning, which satisfy desirable properties of value-based RL. At each iteration, newly collected trajectories are added to the buffer, and a batch is sampled from the full buffer for the gradient update, allowing efficient reuse of past experience.

We adopt a first-in-first-out (FIFO) replay buffer of size M. At each iteration, B new trajectories are collected and stored in the buffer. We then perform K updates per iteration, each sampling a batch of size B uniformly from the buffer. A trajectory remains in the buffer for $\lfloor M/B \rfloor$ iterations before being evicted. Since each trajectory is sampled with probability B/M per update step, the expected total number of gradient updates a single trajectory contributes to is:

$$\mathbb{E}[\text{updates per trajectory}] = \left\lfloor \frac{M}{B} \right\rfloor \cdot \frac{B}{M} \cdot K \approx K. \quad (3)$$

In practice, we use $B = 1024$, $M = 5120$, and $K = 2$, yielding an expected reuse of $K \approx 2$ gradient updates per trajectory, compared to the one-shot usage in on-policy methods. We leave the exploration of more efficient sampling strategies, such as prioritized experience replay (Schaul et al., 2015), to future work.

4 Experiments

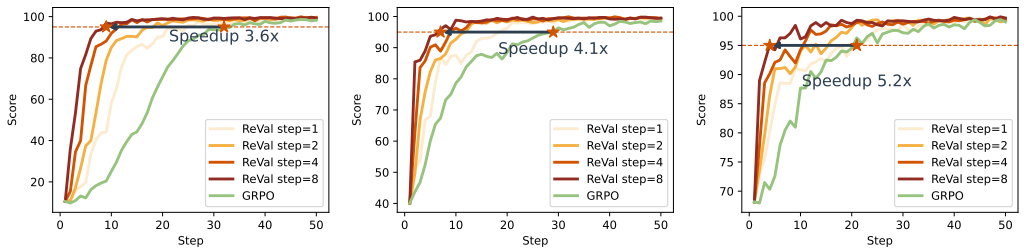
4.1 Experimental Setup

Experiment Setting. We implement ReVal and baseline methods using the large-scale RL training framework Verl (Sheng et al., 2024). Our primary focus is on GRPO (Shao et al., 2024), which are widely examined policy optimization methods in LLM training. We also adopt a value-based baseline, namely TBRM (Yuan et al., 2025). We train our models using the DeepScaleR dataset (Luo et al., 2025). All methods are trained for 650 iterations.

We conduct experiments with DeepSeek-R1-DistillQwen-1.5B (Guo et al., 2025) (abbreviated as DPSK-R1-Distill-1.5B) and Qwen2.5-Math-7B (Yang et al., 2024). In each iteration, we employ a batch size of M = 128 prompts and generate N = 8 rollouts. All responses are sampled with a temperature of 1.0. More details of our implementation can be found in Appendix A. For evaluation, we follow Luo et al. (2025) and assess our method on several mathematical reasoning benchmarks: AIME, AIME25, AMC, MATH, MINERVA, Olympiad Bench (Olympiad for short) and GPQA. All reported performance metrics are averaged over 16 generated responses.

4.2 The Importance of Off-Policy Data for Optimization

We verify the importance of off-policy data reuse across tasks of varying difficulty, demonstrating that frequent data reuse can drastically reduce the number of optimization steps. We use the same



(a) Hard task (b) Medium task (c) Easy task

Figure 2: Performance under different data reuse frequencies on tasks with varying difficulty levels.

tasks as in Section E, using a one-shot task learning setting, where training is conducted on a single prompt. We construct three task difficulty levels, quantified by the average success rate at 1024 samples (`avg@1024`): a hard task with `avg@1024` = 0.10, a medium task with `avg@1024` = 0.40, and an easy task with `avg@1024` = 0.68. We evaluate our proposed ReVal under different data reuse frequencies (`step=1`, `step=2`, `step=4`, `step=8`), where `step=K` indicates that data is sampled from the buffer and optimized in each of K_{update} before new data collection, and compare against the GRPO which collects fresh data at every optimization step.

As shown in Figure 2, ReVal, by reusing data more frequently, achieves an average 4.3Œ speedup in convergence to high performance across tasks of varying difficulty: for the hard task (Figure 2a), ReVal with step=9 reaches a score of 95.0 while GRPO requires 33 steps (3.6x speedup); for the medium task (Figure 2b) and easy task (Figure 2c), ReVal yield 4.1x and 5.2x speedups. These results confirm that off-policy data reuse directly enhances model performance. Furthermore, the advantage of multiple optimization updates becomes increasingly prominent as task difficulty rises. As observed in the Figure 2, harder tasks require more optimization steps to reach 95.0. ReVal achieves the same performance with substantially fewer steps than GRPO, with the gap widening as task difficulty increases. For hard tasks, valid samples are inherently scarce, making intensive and repeated utilization of off-policy data particularly critical for effective policy optimization. These results show the importance of off-policy data for model optimization.

4.3 Main Results

Table 1: Evaluation performance (`avg@16`) comparison across different models and benchmarks.

	AIME24	AIME25	AMC	MATH	MINERVA	Olympiad	GPQA	Avg
DPSK-R1-Distill-1.5B	19.8	20.0	50.7	76.3	22.9	37.5	15.8	34.7
+ GRPO	29.4	24.4	65.0	82.6	27.6	46.3	28.8	43.4
+ TBRM	26.9	22.1	65.1	81.3	28.8	44.6	27.3	42.3
+ ReVal	**32.1**	23.8	**68.6**	**84.6**	**30.3**	**46.6**	**33.3**	**45.6**
Qwen2.5-Math-7B	19.0	6.9	43.6	60.1	10.9	26.3	12.8	25.7
+ GRPO	**34.8**	12.3	59.4	74.0	**30.8**	37.0	20.4	38.4
+ TBRM	30.8	10.0	58.3	74.4	27.3	37.8	19.9	36.9
+ ReVal	34.0	**13.3**	**60.2**	**75.2**	**30.7**	**39.2**	**24.8**	**39.6**

We conduct experiments on both DPSK-R1-Distill-1.5B and Qwen2.5-Math-7B. The training curves and final evaluation results are presented in Table 1 and Figure 10, respectively. Table 1 reports the final evaluation results, while Figure 10 presents the evaluation performance curves of different algorithms throughout training. From Table 1, we observe that ReVal outperforms the baselines on DPSK-R1-Distill-1.5B on almost all benchmarks, achieving state-of-the-art performance. Beyond the in-domain benchmarks, ReVal further surpasses GRPO by 4.3% on the out-of-domain benchmark GPQA, demonstrating the strongest generalization ability. As shown in Figure 10a, ReVal consistently maintains superior performance compared to on-policy methods throughout the entire training process. This trend highlights the importance of incorporating off-policy data, which enables more efficient and stable policy improvement.

We further conduct the same set of experiments on Qwen2.5-Math-7B. In experiments on Qwen2.5-Math-7B, we observed that the model converges rapidly. We suspect that, for non-reasoning models, outputs tend to be shorter and easier to learn, leading to faster convergence (see Section 4.5.1 for detailed discussion). To mitigate this, we introduce reward normalization and periodic reference policy reset, which enable the utilization of negative samples while gradually relaxing the KL constraint. With these enhancements, as shown in Table 1, ReVal achieves performance comparable to or exceeding that of the baseline, with an overall improvement of 1.2%. On out-of-domain benchmarks, ReVal similarly achieves a 4.2% improvement, demonstrating strong generalization performance.

4.4 Performance under Limited Rollouts

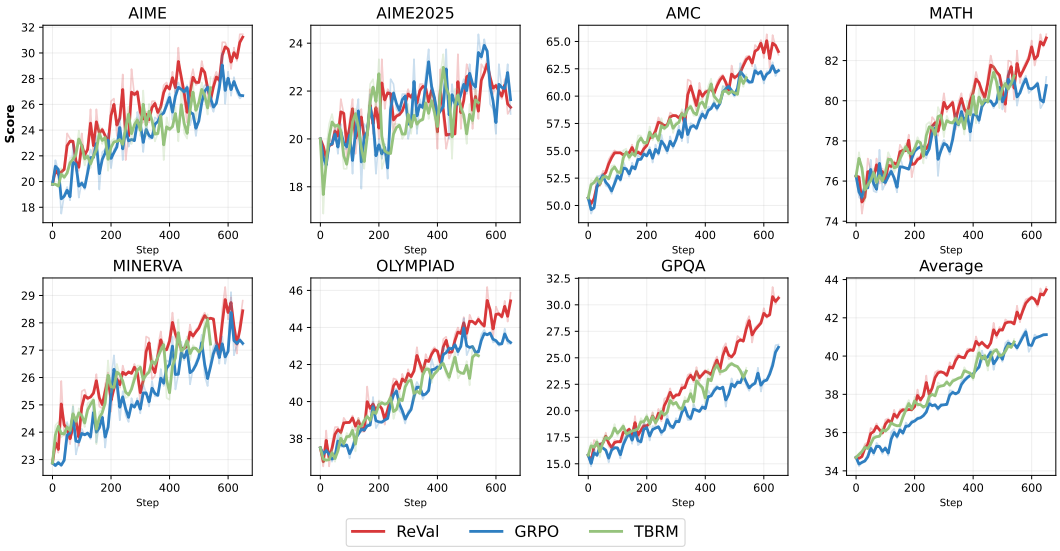


Figure 3: Training Curves of DPSK-R1-Distill-1.5B with N=1

In many real-world applications, generating on-policy trajectories at every training iteration is prohibitively expensive, as each rollout incurs substantial computational or monetary costs. Under such constraints, improving sample efficiency becomes particularly critical with limited rollouts. To evaluate this aspect, we further investigate whether ReVal can still achieve strong performance under the extreme setting of $n = 1$. We conduct experiments using DPSK-R1-Distill-1.5B with the same set of baselines. The training curves are shown in Figure 3. On challenging benchmarks such as AIME and GPQA, the advantage of ReVal is especially pronounced, indicating that off-policy reuse is particularly valuable when rollouts are both expensive and informative. On the Average metric, ReVal achieves a final score higher than GRPO. These results demonstrate that off-policy learning substantially improves sample utilization efficiency, making ReVal especially advantageous in low-rollout regimes.

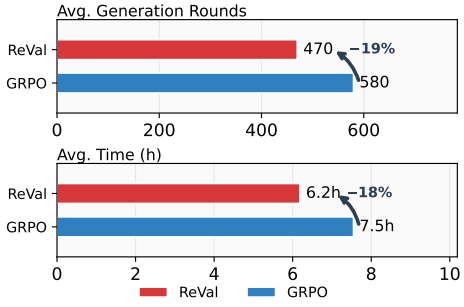


Figure 4: The Average Generation Rounds and Average Total Wall-clock Time (h).

Furthermore, under this setting, we report the average number of generation rounds and total wall-clock time required to reach SOTA performance across seven tasks, as shown in Figure 4. ReVal with more update steps per generation round consistently requires fewer generation rounds, with ReVal (step=8) reducing the number of generations from 580 (GRPO) to 470. In terms of total training time, ReVal also achieves the lower time cost at 6.3h, compared to 7.5h for GRPO, a reduction of 1.3h seconds, corresponding to a 18% decrease in total training time. The experiments show that, when parameter updates are much cheaper than generation (namely, 2.8s per update vs. 36.8s per

trajectory), the benefit of off-policy methods lies not only in improved sample efficiency, but also in reduced training cost through more effective reuse of generated trajectories.

4.5 Key Factors Shaping ReVal

In this section, we empirically investigate the key factors that influence the performance of ReVal, with analysis deferred to Appendix H. We study the effects of the reference policy, the hyperparameter β, and the utilization of negative samples. All experimental datasets are kept consistent with the main experiments on Qwen2.5-Math-7B, and results are reported as average over seven benchmarks.

4.5.1 Relaxing KL Regularization via Reference Policy Updates

ReVal incorporates KL regularization toward a reference policy, which constrains policy updates but causes the KL term $\log \frac{\pi_\theta}{\pi_{\text{ref}}}$ to grow monotonically during training, progressively weakening the gradient signal. The direct way to mitigate the influence of KL regularization is to periodically reset the reference model to the current policy. By updating π_{ref} in this manner, the effective KL constraint is relaxed, allowing the policy to continue improving without being overly restricted by the accumulated divergence from the initial reference.

In principle, the reference policy could be updated based on the residual error (i.e., the term inside the square in Eq. 2). In practice, we find that simple periodic updates work well. We experiment with periodically resetting the reference model every 50/200/400 training steps, as well as a no-reset baseline. We keep all other experimental settings consistent with the main experiments and vary only the update frequency of the reference policy. The results are shown in Figure 5. From the Figure 5, we can find that without updating the reference policy, the model performance saturates at around 200 training steps and remains unchanged thereafter. Second, updating the reference policy every 200 steps yields the best performance, suggesting that a moderate update frequency provides the most effective balance for training. When the reference policy is updated every 400 steps, a noticeable improvement occurs around the 400th step. These results suggest that periodically resetting the reference policy enables the model to escape the shrinking region induced by reference model, resulting in continued performance improvements during training.

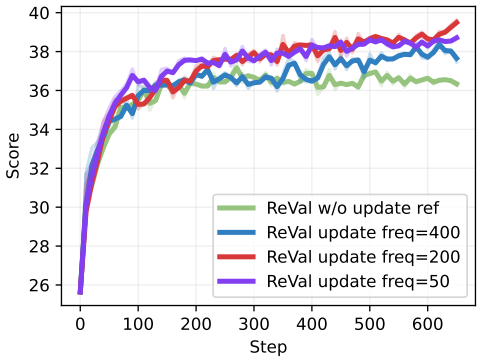
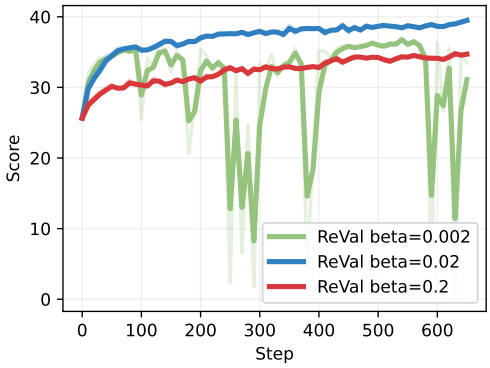


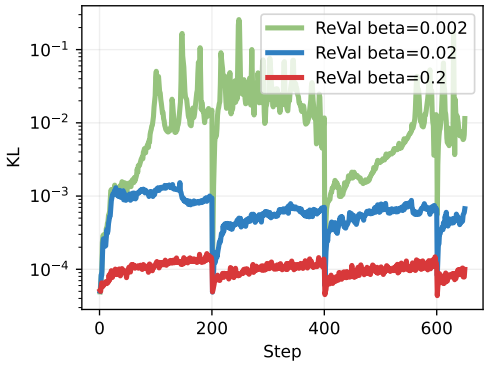
Figure 5: Comparison of different update frequency of reference model.

4.5.2 Hyperparameter β

The hyperparameter β controls the strength of the reward. A larger β imposes a weaker signal and keeps the policy closer to the reference model, while a smaller β allows more freedom during optimization. The choice of β is correlated with the response length. Since the log-ratio term in Eq. 2 is summed over tokens, longer responses produce larger accumulated values and thus require a smaller β to maintain comparable regularization. In experiments, DPSK-Distill-R1-1.5B generates responses of about 5K tokens and we set $\beta = 0.002$, while Qwen2.5-Math-7B produces responses of around 600 tokens and we use $\beta = 0.02$. We experiment with different values of β (0.2, 0.02, and 0.002), and the results are shown in Figure 6. Specifically, we report the average benchmark performance of the model in Figure 6a and the corresponding KL divergence in Figure 6b. For KL divergence, the reference policy is updated every 200 training steps, which results in a periodic increase. As shown in Figure 6, the value of β significantly affects both the KL divergence and the training dynamics. When $\beta = 0.2$, the KL penalty is strong, forcing the policy to stay close to the reference model. In contrast, when $\beta = 0.002$, the KL constraint becomes weaker, allowing the policy to deviate further from the reference model. As a result, the KL divergence increases and the policy explores more aggressively. However, overly large deviations can destabilize optimization and eventually lead to training collapse.



(a) Performance under different β.



(b) KL divergence under different β.

Figure 6: Effect of the hyperparameter β. A smaller β leads to a larger KL divergence between the policy and the reference model.

4.5.3 The Utilization of Negative Samples

In reasoning tasks, 0/1 reward is commonly used (i.e., a reward of 1 for correct answers and 0 for incorrect ones). From Equation 2, it can be seen that under this reward scheme the model increases the logits of correct responses. In contrast, when the answer is incorrect, the policy is only nudged toward the reference policy, instead of explicitly suppressing the probability of incorrect samples. However, negative samples contain informative signals about incorrect behaviors and are crucial for effective learning (Zhu et al., 2025). To better exploit this information, we explore alternative reward formulations that incorporate signals from negative samples. Specifically, we consider two variants. The first variant uses the normalized advantage $\hat{r}_{\text{norm}} = r(x, y_i) - \text{mean}\left(\{r(x, y_i)\}_{i=1}^{G}\right)$ as the reward, following the same formulation as in GRPO. The second variant adopts a ± 1 reward scheme, where correct answers receive a reward of $+1$ and incorrect answers receive a reward of -1. As shown in the Figure 7, using the normalized advantage yields the best performance, whereas the ± 1 reward scheme can even result in performance degradation.

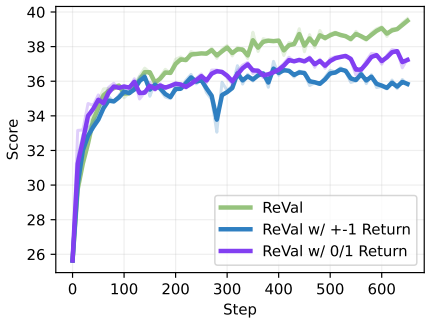


Figure 7: Comparison of different reward designs.

5 Conclusion

In this paper, we investigated the role of off-policy data in LLM RL and proposed ReVal, a value-based algorithm designed to efficiently leverage historical trajectories. We propose ReVal, a Bellman-update-based method that combines stepwise signals capturing internal consistency with trajectory-level signals derived from outcome verification. ReVal naturally supports replay-buffer-based training, allowing efficient reuse of past trajectories. Extensive experiments on standard mathematical reasoning benchmarks demonstrate that ReVal achieves faster convergence, improves sample efficiency, and outperforms strong baselines such as GRPO, with a **2.7%** improvement in AIME24 and **4.5%** in out-of-domain benchmark GPQA on the DPSK-R1-Distill-1.5B model. Ablation studies further reveal the impact of key components, including the reference policy, the hyperparameter β, and different reward and objective designs, providing insights into how each design choice contributes to performance. Currently, our method adopts a standard FIFO replay buffer, which is not the sample-efficient design. In future work, we plan to explore more advanced buffer sampling strategies, such as prioritized experience replay. Furthermore, we will investigate the underlying mechanisms governing the varying update requirements of different data samples in value-based RL for LLM training.

AI USE STATEMENT

Large language model are used to assist with language editing, manuscript organization, and LaTeX formatting. The authors reviewed and revised all AI-assisted text for technical accuracy and consistency with the research. The authors take full responsibility for the methodology, analyses, experimental results, and conclusions in the final manuscript.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we provide detailed descriptions of our method in Section 3.1 and list all hyperparameters used in our experiments in Appendix A. All experiments are conducted on publicly available base models and benchmarks.

REFERENCES

Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *In Proceedings of 19th International Conference on Computational StatisticsParis France*, pp. 177–186, 2010.

Wei Fu, Jiaxuan Gao, Xujie Shen, Chen Zhu, Zhiyu Mei, Chuyi He, Shusheng Xu, Guo Wei, Jun Mei, Jiashu Wang, et al. Areal: A large-scale asynchronous reinforcement learning system for language reasoning. *arXiv preprint arXiv:2505.24298*, 2025.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In Jennifer G. Dy and Andreas Krause (eds.), *In Proceedings of the 35th International Conference on Machine Learning*, 2018.

Wei Gao, Yuheng Zhao, Dakai An, Tianyuan Wu, Lunxi Cao, Shaopan Xiong, Ju Huang, Weixun Wang, Siran Yang, Wenbo Su, et al. Rollpacker: Mitigating long-tail rollouts for fast, synchronous rl post-training. *arXiv preprint arXiv:2509.21009*, 2025.

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

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In Doina Precup and Yee Whye Teh (eds.), *In Proceedings of the 34th International Conference on Machine Learning*, 2017.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer G. Dy and Andreas Krause (eds.), *In Proceedings of the 35th International Conference on Machine Learning*, 2018.

Haoran He, Yuxiao Ye, Qingpeng Cai, Chen Hu, Binxing Jiao, Daxin Jiang, and Ling Pan. Random policy valuation is enough for LLM reasoning with verifiable rewards. *arXiv preprint arXiv:2510.01161*, 2025.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding, 2023. *URL https://arxiv. org/abs/2211.17192*, 1(2), 2022.

Yi-Chen Li, Tian Xu, Yang Yu, Xuqin Zhang, Xiong-Hui Chen, Zhongxiang Ling, Ningjing Chao, Lei Yuan, and Zhi-Hua Zhou. Generalist reward models: Found inside large language models. *arXiv preprint arXiv:2506.23235*, 2025a.

Ziniu Li, Tian Xu, Yushun Zhang, Zhihang Lin, Yang Yu, Ruoyu Sun, and Zhi-Quan Luo. Remax: A simple, effective, and efficient reinforcement learning method for aligning large language models. In *In Proceedings of the 41st International Conference on Machine Learning*, pp. 29128–29163, 2024.

Ziniu Li, Pengyuan Wang, Tian Xu, Tian Ding, Ruoyu Sun, and Yang Yu. Review of reinforcement learning for large language models: Formulations, algorithms, and opportunities, 2025b.

Tian-Shuo Liu, Xu-Hui Liu, Ruifeng Chen, Lixuan Jin, Pengyuan Wang, Zhilong Zhang, and Yang Yu. Semantic temporal abstraction via vision-language model guidance for efficient reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=zY37C8d6bS.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Erran Li, Raluca Ada Popa, and Ion Stoica. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl. https://pretty-radio-b75.notion.site/DeepScaleR-Surpassing-O1-Preview-with-a-1-5B-Model-by-Scaling-RL-19681902c1468005b 2025. Notion Blog.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518:529–533, 2015.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pp. 278–287, 1999.

OpenAI. Learning to reason with LLMs. OpenAI Blog, Sep 2024. https://openai.com/index/learning-to-reason-with-llms/.

Jing-Cheng Pang, Pengyuan Wang, Kaiyuan Li, Xiong-Hui Chen, Jiacheng Xu, Zongzhang Zhang, and Yang Yu. Language model self-improvement by reinforcement learning contemplation. In *In Proceedings of the 12th International Conference on Learning Representations*, 2024.

Ruoyu Qin, Weiran He, Weixiao Huang, Yangkun Zhang, Yikai Zhao, Bo Pang, Xinran Xu, Yingdi Shan, Yongwei Wu, and Mingxing Zhang. Seer: Online context learning for fast synchronous llm reinforcement learning. *arXiv preprint arXiv:2511.14617*, 2025.

Daniel Ritter, Owen Oertell, Bradley Guo, Jonathan Chang, Kianté Brantley, and Wen Sun. Llms can learn to reason via off-policy rl. *arXiv preprint arXiv:2602.19362*, 2026.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

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

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.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

Richard Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3: 9–44, 1988.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, SH Cai, Yuan Cao, Y Charles, HS Che, Cheng Chen, Guanduo Chen, et al. Kimi k2. 5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*, 2026.

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. *arXiv preprint arXiv:1509.06461*, 2015.

Peng-Yuan Wang, Tian-Shuo Liu, Chenyang Wang, Ziniu Li, Yidi Wang, Shu Yan, Chengxing Jia, Xu-Hui Liu, Xinwei Chen, Jiacheng Xu, et al. A survey on large language models for mathematical reasoning. *ACM Computing Surveys*, 58:1–35, 2026.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.

Xue Yan, Yan Song, Xidong Feng, Mengyue Yang, Haifeng Zhang, Haitham Bou Ammar, and Jun Wang. Efficient reinforcement learning with large language model priors. *arXiv preprint arXiv:2410.07927*, 2024.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

Feng Yao, Liyuan Liu, Dinghuai Zhang, Chengyu Dong, Jingbo Shang, and Jianfeng Gao. Your efficient rl framework secretly brings you off-policy rl training, August 2025. URL `https://fengyao.notion.site/off-policy-rl`.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Weinan Dai, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. DAPO: an open-source LLM reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Yang Yu. Towards sample efficient reinforcement learning. In *IJCAI*, pp. 5739–5743, 2018.

Yurun Yuan, Fan Chen, Zeyu Jia, Alexander Rakhlin, and Tengyang Xie. Trajectory bellman residual minimization: A simple value-based method for llm reasoning. *arXiv preprint arXiv:2505.15311*, 2025.

Hongzhi Zhang, Jia Fu, Jingyuan Zhang, Kai Fu, Qi Wang, Fuzheng Zhang, and Guorui Zhou. RLEP: reinforcement learning with experience replay for LLM reasoning. *arXiv preprint arXiv:2507.07451*, 2025.

Yaxiang Zhang, Yingru Li, Jiacai Liu, Jiawei Xu, Ziniu Li, Qian Liu, and Haoyuan Li. Beyond precision: Training-inference mismatch is an optimization problem and simple lr scheduling fixes it. *arXiv preprint arXiv:2602.01826*, 2026.

Haizhong Zheng, Jiawei Zhao, and Beidi Chen. Prosperity before collapse: How far can off-policy RL reach with stale data on llms? *arXiv https://arxiv.org/abs/2510.01161*, 2025.

Xinyu Zhu, Mengzhou Xia, Zhepei Wei, Wei-Lin Chen, Danqi Chen, and Yu Meng. The surprising effectiveness of negative reinforcement in llm reasoning. *arXiv preprint arXiv:2506.01347*, 2025.

Brian D. Ziebart, Andrew L. Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In Dieter Fox and Carla P. Gomes (eds.), *In Proceedings of the 23rd Conference on Artificial Intelligence*, 2008.

A Detailed Experimental Setup

All experiments were implemented using the large-scale reinforcement learning framework Verl (v0.5.0). The default training and inference pipelines were preserved without modification. For optimization, we adopted a learning rate of $1e-6$, following prior recommendations (Yu et al., 2025).

For ReVal, the β value was set to 0.002 for DPSK-R1-Distill-1.5B and 0.02 for Qwen-2.5-Math-7B. The buffer size was set to 5,120. For each sampled batch, we perform two updates. First, an on-policy update, followed by an off-policy update using a sample from the replay buffer. For GRPO, the importance sampling clipping thresholds were set asymmetrically to 0.28 (upper) and 0.2 (lower), and a compensation term was applied to account for inconsistencies between vLLM and FSDP. No additional KL or entropy regularization terms were used in any experiments.

DPSK-R1-Distill-1.5B was trained with a maximum sequence length of 8K tokens due to its longer CoT reasoning patterns, which require extended context windows. Qwen2.5-Math-7B was trained with an 8K token limit.

During training, evaluation was conducted every 10 iterations. For each evaluation phase, 16 responses were generated per prompt. To ensure manageable evaluation time, we capped the evaluation set size at 100 samples by randomly sub-sampling benchmarks exceeding this number.

B Maximum Entropy Reinforcement Learning

The original KL-regularized RL objective in eq. (1) can be transformed into the following maximum entropy RL objective (Haarnoja et al., 2018) with a modified reward.

$$\mathbb{E}_{x\sim\rho}\Big[\mathbb{E}_{a_{1:H}\sim\pi_\theta(\cdot|x)}\big[r_{\text{rule}}(x,a_{1:H})\big]-\beta\,D_{\text{KL}}\big(\pi_\theta(\cdot\mid x),\pi_{\text{ref}}(\cdot\mid x)\big)\Big]$$
$$=\beta\cdot\mathbb{E}_{\tau\sim\pi}\left[\sum_{h=1}^{H}\Bigg(\underbrace{\frac{r(s_h,a_h)}{\beta}+\log\pi_{\text{ref}}(a_h|s_h)}_{:=r_\beta(s_h,a_h)}+\mathcal{H}(\pi(\cdot|s_h))\Bigg)\right].$$

Here $r(s_h,a_h)=0$ if $h\neq H$ and $r(s_h,a_h)=r_{\text{rule}}(x,a_{1:H})$ otherwise denotes the token-level reward and $r_\beta(s_h,a_h):=r(s_h,a_h)/\beta+\log\pi_{\text{ref}}(a_h|s_h)$ is the modified reward. Besides, $\mathcal{H}(\pi(\cdot|s_h))=\mathbb{E}_{a_h\sim\pi(\cdot|s_h)}[\log(1/\pi(a_h|s_h))]$ denotes the entropy. In maximum entropy RL, the soft optimal Q-function satisfies the Bellman equation.

$$\mathbb{E}_{s_{h+1}\sim P(\cdot|s_h,a_h)}[]$$

Here $V_Q(s):=\log(\sum_{a\in\mathcal{A}}\exp(Q(s,a)))$ denotes the V-function induced by Q. By defining the Bellman operator $(\mathcal{T}_\beta Q)(s_h,a_h):=r_\beta(s_h,a_h)+\mathbb{E}_{s_{h+1}\sim P(\cdot|s_h,a_h)}\left[V_Q(s_{h+1})\right]$, we have that $Q^\star_\beta$ is the fixed point w.r.t the Bellman operator, i.e., $Q^\star_\beta=\mathcal{T}_\beta Q^\star_\beta$. Given the soft optimal Q-function, we can derive the soft optimal policy through a softmax transformation.

$$\pi^\star_\beta(a_h|s_h)=\frac{\exp(Q^\star_\beta(s_h,a_h))}{\sum_{a\in\mathcal{A}}\exp(Q^\star_\beta(s_h,a))}=\exp\left(Q^\star_\beta(s_h,a_h)-V_{Q^\star_\beta}(s_h)\right). \tag{4}$$

C Proof

Proposition 3. *Consider the objective in Eq. 2. When $r=0$ and the policy is initialized as $\pi_\theta=\pi_{ref}$, we have $V_\theta(s_1)=V_{ref}(s_1)$ and $\log\pi_\theta(\tau)=\log\pi_{ref}(\tau)$, which yields $\mathcal{L}_{ReVal}(\theta)=0$.*

Proof. When $\pi_\theta=\pi_{\text{ref}}$, the Q-function satisfies $Q_\theta=Q_{\text{ref}}$, which directly implies $V_\theta(s_1)=\log\sum_a\exp Q_\theta(s_1,a)=\log\sum_a\exp Q_{\text{ref}}(s_1,a)=V_{\text{ref}}(s_1)$. Furthermore, $\log\pi_\theta(\tau)=$

$\sum_{h=1}^{H} \log \pi_\theta(a_h \mid s_h) = \sum_{h=1}^{H} \log \pi_{\text{ref}}(a_h \mid s_h) = \log \pi_{\text{ref}}(\tau)$. Substituting into Eq. 2 with $r = 0$:

$$\mathcal{L}_{\text{ReVal}}(\theta) = \frac{1}{|\mathcal{D}|} \sum_{\tau \in \mathcal{D}} \left(V_\theta(s_1) - V_{\text{ref}}(s_1) + \log \pi_\theta(\tau) - \frac{r_{\text{rule}}(\tau)}{\beta} - \log \pi_{\text{ref}}(\tau) \right)^2$$
$$= \frac{1}{|\mathcal{D}|} \sum_{\tau \in \mathcal{D}} (0 + 0 - 0)^2 = 0.$$

□

D Algorithm Description

Algorithm 1 Off-Policy Value-Based Reinforcement Learning with Replay Buffer (ReVal)

Require: Task prompt dataset $\mathcal{D}_{\text{task}}$, first-in-first-out (FIFO) replay buffer $\mathcal{D}_{\text{replay}} = \emptyset$, task reward r, reward scaling coefficient β, reference policy π_{ref} with parameter θ_{ref}, number of iterations T.

1: Initialize.
2: **for** $t = 1, 2, \ldots, T$ **do**
3: For each question $q \in \mathcal{D}^t_{\text{task}}$, sample trajectories from policy π_θ , and collect these trajectories into batch $\mathcal{D}^t$.
4: Augment buffer with the batch $\mathcal{D}_{\text{replay}} \leftarrow \mathcal{D}_{\text{replay}} \cup \mathcal{D}^t$ and evict oldest samples if capacity is exceeded.
5: Sample an off-policy batch $\mathcal{D}^t_{\text{replay}} \subset \mathcal{D}_{\text{replay}}$.
6: Update θ via gradient descent using Eq. 2 based on off-policy batch $\mathcal{D}^t_{\text{replay}}$.
7: Sample a prompt batch $\mathcal{D}^t_{\text{task}} \subset \mathcal{D}_{\text{task}}$.
8: **end for**

E Discussion on the Limitations of On-Policy Methods

We revisit the foundations of policy gradient methods and examine their implications for practical training cost. By construction, policy gradient methods (Sutton, 1988) update the policy at iteration k using gradient estimates computed from trajectories sampled under the current policy π_k. *After the update, those trajectories are no longer on-policy and therefore cannot be reused directly in subsequent iterations; fresh samples must be collected again.* Because each policy update is typically small and local, a single gradient step cannot fully exploit the information contained in one batch of trajectories (Bottou, 2010). As a result, convergence generally requires many iterations of alternating data collection and optimization.

We empirically demonstrate this limitation using a one-shot task learning setting, where training is conducted on a single prompt (Figure 8). GRPO (Shao et al., 2024), as an on-policy method, collects fresh trajectories at every iteration and therefore directly reflects the sampling difficulty of the task. On hard task (`avg@1024` = 0.10), GRPO requires substantially more optimization steps than on medium (`avg@1024` = 0.40) or easy (`avg@1024` = 0.68) task to reach the same performance threshold. These findings indicate that: (1) even in a highly simplified one-shot setting, learning still depends on repeated alternation between data collection and parameter updates; and (2) harder tasks may require many more such iterations before they can be solved.

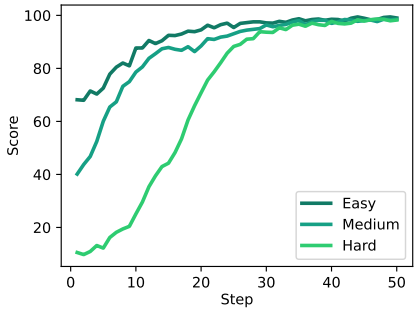


Figure 8: Performance of GRPO across different difficulty levels.

These inefficiencies translate directly into practical bottlenecks. Let $K_{\text{generation}}$ denote the number of generation rounds and K_{update} the number of parameter updates. In standard on-policy training, the two

are tightly coupled, so typically $K_{\text{generation}} = K_{\text{update}}$, since each update requires newly sampled trajectories. If the time cost of one generation round is $T_{\text{generation}}$ and that of one update is T_{update}, then the total training time can be approximated as

$$T_{\text{total}} \approx K_{\text{generation}} T_{\text{generation}} + K_{\text{update}} T_{\text{update}}.$$

For RL with LLMs, $T_{\text{generation}}$ is often much larger than T_{update} because of the cost of autoregressive sequence generation (Qin et al., 2025). Consequently, reducing total training time requires either decreasing $T_{\text{generation}}$ itself, for example through faster generation systems (Leviathan et al., 2022), or decreasing $K_{\text{generation}}$, that is, learning more efficiently from each round of collected data (Yu, 2018). In other words, the central challenge is to extract more useful learning signal from the same set of trajectories.

This observation motivates value-based and off-policy methods, which decouple data collection from policy updates and allow historical trajectories to be reused across multiple optimization steps (Watkins & Dayan, 1992; Mnih et al., 2015; Fujimoto et al., 2018). By performing more parameter updates on the same batch of trajectories, the algorithm can extract more learning signal from each generation round. This improved reuse can accelerate convergence, because the model makes greater progress before new data need to be collected. As a result, the total number of generation rounds required during training may decrease. Although this strategy introduces additional update cost, parameter updates are cheaper than autoregressive generation in LLM-based RL. Therefore, when we reduce generation rounds, the overall wall-clock training time can also decrease. From this perspective, the benefit of off-policy methods lies not only in improved sample efficiency, but also in lower practical training cost through more effective reuse of generated trajectories. In the next section, we explore value-based and off-policy methods from this perspective.

F Additional Experimental Results

F.1 The Issue of TBRM

When the reward is zero, the TBRM objective does not reduce to a KL minimization objective. Specifically, setting $r(\tau) = 0$ yields

$$\mathcal{L}(\theta) = \left(V_\theta(s_1) + \sum_{h=1}^{H} \log \frac{\pi_\theta(a_h \mid s_h)}{\pi_{\text{ref}}(a_h \mid s_h)} \right)^2 .$$

Minimizing this objective drives the squared term toward zero, which requires the log-likelihood ratio to cancel the value term $V_\theta(s_1)$ rather than directly minimizing the KL divergence between π_θ and π_{ref}. As a result, the optimal solution does not correspond to matching the reference policy. To empirically verify this behavior, we additionally conduct experiments where the reward is fixed to 0. As shown in Figure 9, the KL divergence does not converge to zero when the reward is fixed to 0, empirically confirming that the TBRM objective does not reduce to KL minimization in this case.

F.2 Training curves of DPSK-R1-Distill-1.5B and Qwen2.5-Math-7B

The training curves of DPSK-R1-Distill-1.5B and Qwen2.5-Math-7B is shown in Figure 10.

G Gradient Dynamics Analysis.

To understand the factors that influence training, we analyze the gradient of $\mathcal{L}_{\text{ReVal}}$ with respect to θ:

$$\nabla_\theta \mathcal{L}_{\text{ReVal}} = -2\, \mathbb{E}_{(x,y)} \left[\delta(x,y) \cdot \nabla_\theta \log \pi_\theta(y|x) \right], \tag{5}$$

where the residual error $\delta(x,y) = \frac{r(x,y)}{\beta} - \left(V_\theta(x) - V_{\text{ref}}(x) + \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)} \right)$ determines both the magnitude and direction of the gradient update. We identify three key factors that affect the gradient dynamics.

KL Regularization and Periodic Reset. The term $\log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}$ grows monotonically as the policy diverges from the reference model during training. As this term increases, it progressively reduces δ,

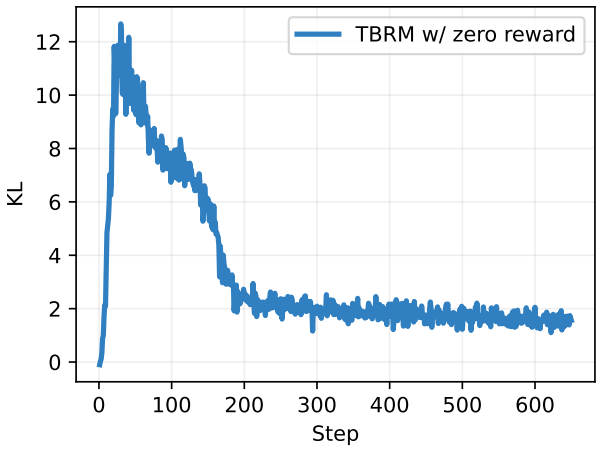


Figure 9: Training behavior of TBRM when the reward is fixed to 0. The KL divergence between the current policy and the reference policy does not converge to zero.

weakening the gradient signal and slowing learning. To mitigate this, we periodically reset the reference model to the current policy, which resets the KL term back to zero and restores the magnitude of the gradient signal.

Hyperparameter β. The parameter β directly scales the reward signal via $\frac{r(x,y)}{\beta}$, controlling its relative weight in δ. When β is too large, the reward signal is suppressed, causing δ to remain small throughout training and the gradient to vanish, leading to slow or stalled convergence. Conversely, when β is too small, the reward dominates and may cause excessively large gradient updates, destabilizing training.

Negative Samples. When $r(x, y) = 0$, the TD error reduces to:

$$\delta(x, y) = -\left(V_\theta(x) - V_{\text{ref}}(x) + \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}\right), \tag{6}$$

and the gradient becomes:

$$\nabla_\theta \mathcal{L}_{\text{ReVal}} \propto \left(V_\theta(x) - V_{\text{ref}}(x) + \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}\right) \cdot \nabla_\theta \log \pi_\theta. \tag{7}$$

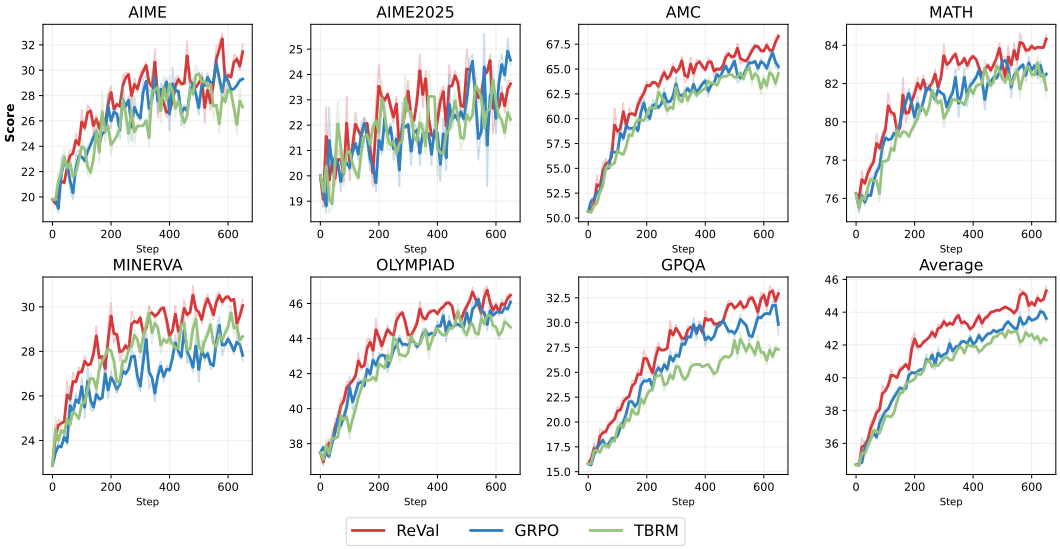
In this case, the optimization drives $\log \frac{\pi_\theta}{\pi_{\text{ref}}} \to 0$ and $V_\theta - V_{\text{ref}} \to 0$, pulling the policy back toward the reference model rather than decreasing $\log \pi_\theta$. Intuitively, negative samples should penalize incorrect responses by decreasing $\log \pi_\theta(y|x)$, rather than moving to the reference policy.

H Analysis of Objective Variants

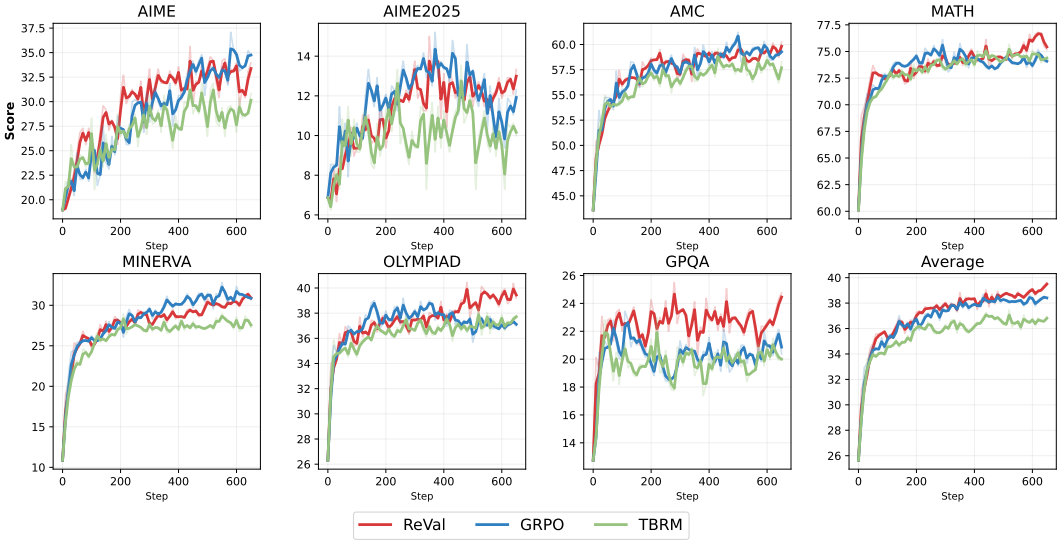
For TBRM, the additional term $V_\theta(s_1)$ in the objective undermines Calibrated Initialization. An alternative variant is to remove $V_\theta(s_1)$, since this term is independent of the action and does not affect the optimal solution. This yields:

$$\mathcal{L}_{\text{regression}}(\theta) = \frac{1}{|\mathcal{D}|} \sum_{\tau \in \mathcal{D}} \left(\log \pi_\theta(\tau) - \frac{r_{\text{rule}}(\tau)}{\beta} - \log \pi_{\text{ref}}(\tau)\right)^2. \tag{8}$$

This objective is closely related to the regression-based formulations in Team et al. (2025) and Ritter et al. (2026), with the key difference that these work introduce an additional $\log Z(x)$ term. Since $\log Z(x)$ is independent of the policy, it does not affect the optimal solution. In practice, $\log Z(x)$ can be replaced by reward normalization (Team et al., 2025). In regression-based methods, the target labels are fixed, which allows for multiple training passes over the same data stably. In Eq. 8, we follow the practice in Kimi K1.5 (Team et al., 2025) and introduce reward normalization to ensure



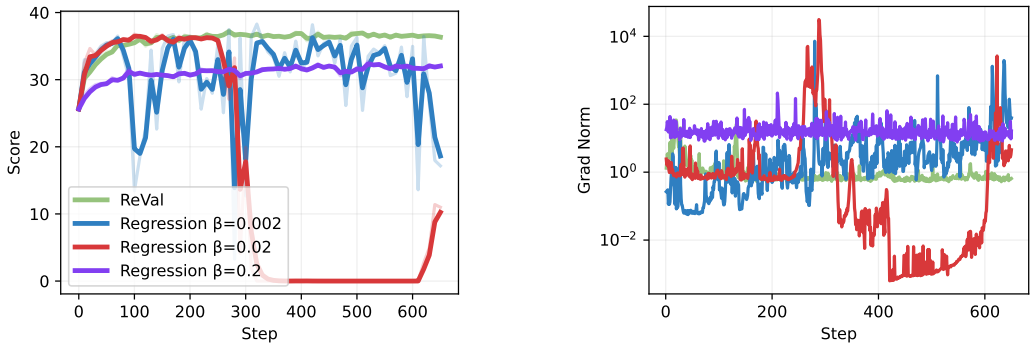
(a) DPSK-R1-Distill-1.5B



(b) Qwen2.5-Math-7B

Figure 10: Training curves of DPSK-R1-Distill-1.5B and Qwen2.5-Math-7B. Curves show the accuracy across seven benchmarks (AIME, AIME25, AMC, MATH, Minerva, Olympiad, and GPQA) as well as the average accuracy.

consistency with the regression-based method: $\hat{r}_{\text{norm}} = r_{\text{rule}}(x, y_i) - \text{mean}\left(\{r_{\text{rule}}(x, y_i)\}_{i=1}^{G}\right)$. However, we empirically find that training remains unstable with reward normalization applied.



(a) Average benchmark performance.

(b) Gradient norm.

Figure 11: Comparison of different objectives. (a) Average benchmark performance across training steps. (b) Corresponding gradient norm during training.

We explore different regression by comparing Eq. 8 and Eq. 2. Specifically, for Eq. 8, we follow the practice in Kimi K1.5 (Team et al., 2025) and introduce reward normalization: i.e., $\hat{r}_{\text{norm}} = r(x, y_i) - \text{mean}\left(\{r(x, y_i)\}_{i=1}^{G}\right)$. However, through experiments, we observe that optimizing the loss in Eq. 8 is not stable. We tried different values of $\beta = 0.2,\ 0.02,\ 0.002$ and observed that the model remained unstable. We then compared with ReVal under the same setting (using reward normalization, $\beta = 0.02$, and without periodic updates). The results are shown in Figure 11. It can be observed that the model remains stable only when $\beta = 0.2$, while instability appears at $\beta = 0.02$. Under the same conditions, ReVal exhibits better stability. Experimental observations show that the model can experience extremely large and instable gradient norms (around $1e4$) which may be related to some anomalous values of $\log \pi / \pi_{\text{REF}}$.

I Related Work

I.1 Q-function Representation in LLMs

Li et al. (2025a) established a theoretical cornerstone in this direction: they showed that a language model trained via standard next-token prediction implicitly learns a soft Q-function, where the model logits are a principled solution to the Q-function in an offline inverse reinforcement learning formulation. Formally, given a language model $\hat{\pi}$ parameterized as $\hat{\pi}(\cdot \mid s_h) = \text{softmax}(\hat{f}(s_h, \cdot); \alpha)$, the logits $\hat{f}$ directly correspond to the soft Q-values of the data-generating policy, revealing that LLM logits are not arbitrary scores but encode value information about token-level decisions.TBRM (Yuan et al., 2025) also utilized this connection between logits and Q-function, adopting a logit-as-Q parameterization within its Bellman update formulation, and empirically validated the effectiveness of this parameterization. These works suggest that the Q-function in LLMs can be naturally parameterized by the model logits, forming the theoretical foundation of our value-based RL framework.

I.2 Value-based Reinforcement Learning

Efficient RL is important for large-scale LLM training, particularly in asynchronous settings where policy updates and data collection are naturally decoupled (Yan et al., 2024; Liu et al., 2025; Ritter et al., 2026). Value-based reinforcement learning is a fundamental paradigm for this setting, as it focuses on learning action-value functions and naturally supports off-policy experience reuse. Deep Q-Networks (Mnih et al., 2013) and their numerous variants (van Hasselt et al., 2015) have demonstrated remarkable success in high-dimensional tasks, largely attributed to the effective use of experience replay buffers for stable off-policy learning. To encourage exploration and robustness,

a series of works have adopted the Maximum Entropy RL framework (Ziebart et al., 2008), where Soft Q-learning (SQL) (Haarnoja et al., 2017) introduces an energy-based formulation to satisfy the entropy-regularized objective. While advanced algorithms such as Soft Actor-Critic (Haarnoja et al., 2018) were further developed and achieved state-of-the-art performance in traditional RL. Inspired by the remarkable performance of value-based methods, we investigate the dynamics of off-policy buffer utilization within the SQL objective, an area that remains relatively under-explored in the context of scaling value-based RL for generative language tasks.

I.3 Reinforcement Learning in Large Language Models

RL has become a key component in the post-training stage of LLMs, with reward design and training algorithms being its central elements (Li et al., 2025b; Pang et al., 2024). As the field matured, however, it became clear that at LLM scale, an RL algorithm is only practical if it is computationally efficient. ReMax (Li et al., 2024) was the first to move from actor-critic to actor-only RL, significantly reducing memory usage and training time for LLM post-training. Following ReMax, a series of methods, including GRPO (Shao et al., 2024) and DAPO (Yu et al., 2025), further advanced this low-cost policy optimization paradigm. Policy gradient methods have dominated LLM alignment due to their simplicity, natural compatibility with pretrained language models, and relatively low computational overhead (Li et al., 2025b). But LLM development is increasingly shifting toward *agentic* settings with long and highly variable horizons. The high variance in trajectory lengths (Team et al., 2025; Fu et al., 2025), the training-inference mismatch (Zhang et al., 2026; Yao et al., 2025), and the difficulty of obtaining sufficient samples (Team et al., 2026; Gao et al., 2025) make these methods progressively brittle and inadequate for the demand of off-policy data. These limitations motivate us to turn towards off-policy value-based algorithms. Recently, value-based approaches such as TBRM (Yuan et al., 2025) and ROVER (He et al., 2025) have been proposed, which leverage the Q-function information implicitly encoded in the LLMs own logits for training but they are still trained in on-policy way. Meanwhile, some researchers have begun to train LLMs in an off-policy manner (Zhang et al., 2025; Zheng et al., 2025), but the algorithms they employ are still originally designed for on-policy settings. Our method explores combining value-based RL with off-policy training for LLMs.

J Prompt Templates

The prompt templates of all methods used for benchmarking reward models on Multifacted-Bench are shown below. As RM-Bench does not provide specific instructions for each sample, all methods use the default system prompt and the instructions in the **User** part of the prompt templates will also be removed.

User
{Question} Let's think step by step and output the final answer within \boxed{}.

Assistant
< think >

Figure 12: Prompt template of DeepSeek-R1-Distill-Qwen-1.5B.

System
Please reason step by step, and put your final answer within \boxed{}.

User
{Question} Let's think step by step and output the final answer within \boxed{}

Assistant

Figure 13: Prompt template of Qwen2.5-Math-7B.