d3LLM-v2: Ultra-Fast Agentic Diffusion LLM via Reward Gating and Online Distillation

Anonymous authors
Paper under double-blind review

Abstract

Diffusion large language models (dLLMs) have emerged as a promising alternative to autoregressive (AR) LLMs, offering potential advantages such as parallel decoding and global context modeling. While many efforts have been devoted to improving dLLMs and enhancing their reasoning capabilities, extending dLLMs to *agentic settings* that involve multiple interaction turns and tackling the inherent *accuracy–parallelism trade-off* in such long-horizon agentic tasks, remain underexplored. In this work, we study on-policy training for agentic dLLMs and propose d3LLM-v2 (*Adaptive-Distilled Diffusion Large Language Model*) framework, striking a balance between accuracy and parallelism in long-horizon agentic tasks. *(i)* We introduce *reward-gated agentic RL* for dLLMs: due to bidirectional attention and parallel decoding, dLLMs are more likely to generate invalid actions or failed tool calls; moreover, ELBO-based KL regularization provides only a coarse approximation, which becomes unreliable across multi-turn interactions. To address this, we propose *validity-based reward gating* that penalizes invalid actions, paired with implicit policy regularization for agentic dLLMs. *(ii)* We further introduce *trajectory-based online self-distillation* to improve the efficiency of dLLMs in an online manner, where the model serves as its own teacher and learns to reproduce its decoding outputs in fewer diffusion steps along its own trajectories, guided by an easy-to-hard curriculum mechanism. Experiments demonstrate that d3LLM-v2 achieves up to $4.4\times$ speedup while maintaining competitive task accuracy.

1 Introduction

Diffusion large language models (dLLMs) have emerged as a promising alternative to autoregressive (AR) LLMs. A key advantage of dLLMs is their use of *bidirectional attention*, which enables capabilities such as parallel decoding and global context modeling, i.e., features that are not feasible with AR models. Several closed-source diffusion models, including Mercury (Khanna et al., 2025), Gemini Diffusion (Google DeepMind, 2025), and Seed Diffusion (Song et al., 2025), have demonstrated impressive efficiency, sometimes exceeding 1000 tokens per second. Open-source dLLMs such as LLaDA (Nie et al., 2025) and Dream (Ye et al., 2025) have also attracted growing attention from the community. However, despite these promising advances, the performance of current dLLMs still lags behind that of similarly sized AR models.

With the growing interest of the community, many efforts have been devoted to closing this gap from various perspectives. One line of research improves dLLMs through distillation (Chen et al., 2026a; Zhu et al., 2026b; Dat et al., 2026). Another line adapts AR-style mechanisms to enhance the quality and efficiency of dLLMs (Fu et al., 2025; Wu et al., 2025; Cheng et al., 2026; Li et al., 2026). Recently, reinforcement learning (RL) has been introduced to improve the reasoning ability of dLLMs, with methods such as d1 (Zhao et al., 2025), VRPO (Zhu et al., 2026a), and GDPO (Rojas et al., 2026), demonstrating the effectiveness of RL post-training on single-round reasoning tasks.

More recently, several studies have explored dLLMs in agentic settings. DLLM-Searcher (Zhao et al., 2026) designs a two-stage post-training pipeline for search agents: agentic SFT followed by agentic VRPO, a preference optimization method that constructs winner/loser trajectory pairs from the model's rollouts. Zhen et al. (2026) provide an empirical study of dLLM backbones in multi-step agent workflows and identify challenges such as tool-call failures in agentic dLLMs. Very recently, LLaDA2.2 (Bie et al., 2026) applies agentic RL with environmental feedback, demonstrating the

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.

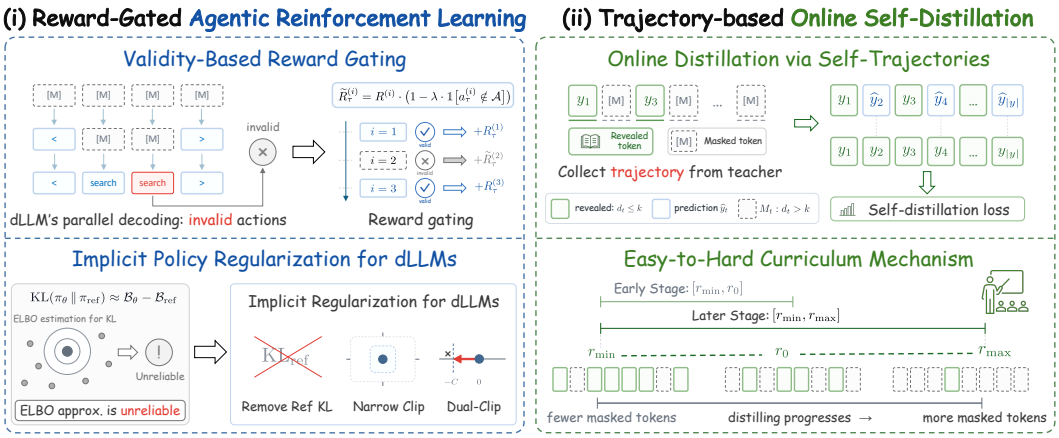


Figure 1: Overview of our *d3LLM-v2* framework. *(i) Reward-gated agentic RL* gates the rewards by action validity and replaces explicit KL regularization with implicit regularization for stable policy updates. *(ii) Trajectory-based online self-distillation* trains the model to reproduce its decoding outputs in fewer steps along its own trajectory, guided by an easy-to-hard curriculum to improve the efficiency of agentic dLLMs.

promising performance of agentic dLLMs. However, dLLMs inherently face an *accuracy–parallelism trade-off* (Qian et al., 2026), which is amplified in long-horizon agentic tasks, where errors at any step can compound across the trajectory. Existing agentic approaches primarily focus on improving task accuracy without explicitly tackling this trade-off.

In this work, we push the accuracy–parallelism frontier of long-horizon agentic dLLMs through *on-policy* training, which is essential for both sides of the trade-off: the agentic actions that fail and the tokens that can be safely decoded in parallel are both determined by the evolving policy through environment interaction, and thus require training signals that evolve with the policy itself. To this end, we propose d3LLM-v2 framework, which comprises two components: *(i) reward-gated agentic RL*: due to bidirectional attention and parallel decoding, dLLMs are more likely to generate invalid actions or failed tool calls; moreover, ELBO-based KL regularization provides only a coarse approximation, which becomes unreliable across multi-turn interactions; we address these issues with *validity-based reward gating* that penalizes invalid actions, paired with implicit policy regularization to ensure stable updates. *(ii)* We further propose *trajectory-based online self-distillation* to improve the efficiency of agentic dLLMs, where distillation is driven by the model's own on-policy rollouts: the dLLM generates and records its decoding trajectories through environment interaction, and learns to reproduce its outputs in fewer diffusion steps. We further introduce an easy-to-hard curriculum that progressively increases the parallelism of agentic dLLMs.

We validate d3LLM-v2 on three agentic benchmarks across five foundation models based on SDAR (Cheng et al., 2026) and d3LLM-LLaDA (Qian et al., 2026), using AUP (*Accuracy Under Parallelism*) (Qian et al., 2026), a metric that jointly measures accuracy and parallelism, as the primary evaluation metric. Experiments show that d3LLM-v2 consistently achieves the highest AUP on different agentic tasks, achieving up to $4.4\times$ speedup compared to the best existing method, with up to $9.1\times$ speedup compared to the AR baseline, while maintaining competitive task accuracy.

Organization. Section 2 introduces the preliminaries. Section 3 presents our d3LLM-v2 framework. Section 4 reports experimental results, and Section 5 concludes the paper. Due to page limits, related work is deferred to Appendix A and additional experiments are deferred to Appendix B.

2 Preliminaries

In this section, we introduce the preliminaries. Let π_θ denote a language model policy parameterized by θ. Given an input prompt x, the model generates a response $y = (y_1, y_2, \ldots, y_{|y|})$.

Policy Optimization: From AR to dLLM. For AR LLMs, the sequence log-likelihood admits a chain-rule decomposition $\log \pi_\theta(y \mid x) = \sum_{t=1}^{|y|} \log \pi_\theta(y_t \mid x, y_{<t})$, which can be computed exactly in a single forward pass. This enables standard policy optimization methods such as PPO (Schulman et al., 2017) and GRPO (Shao et al., 2024). In particular, GRPO samples a group of responses

$\{y^{(i)}\}_{i=1}^{G}$ for each prompt x, assigns each a reward $R^{(i)}$, and maximizes the following objective

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|y^{(i)}|}\sum_{t=1}^{|y^{(i)}|}\left(\min\left(\rho_t^{(i)}\,\widehat{A}^{(i)},\ \text{clip}\left(\rho_t^{(i)}, 1-\varepsilon, 1+\varepsilon\right)\widehat{A}^{(i)}\right) - \beta\,\text{KL}\left(\pi_\theta \| \pi_{\text{ref}}\right)\right)\right],$$

where $\rho_t^{(i)} \triangleq \pi_\theta(y_t^{(i)}|x, y_{<t}^{(i)})/\pi_{\theta_{\text{old}}}(y_t^{(i)}|x, y_{<t}^{(i)})$ is the token-level importance ratio, ε is the clipping threshold, π_{ref} is a frozen reference policy, and β controls the strength of KL regularization. The advantage is computed via group-wise normalization $\widehat{A}^{(i)} \triangleq (R^{(i)} - \mu_g)/\sigma_g$, where μ_g and σ_g denote the mean and standard deviation of rewards within the group, respectively.

However, extending RL to dLLMs is non-trivial: the iterative denoising process does not admit such a factorization, making the exact computation of $\log \pi_\theta(y \mid x)$ intractable (Zhu et al., 2026a; Zhao et al., 2025). Instead, existing methods typically rely on the evidence lower bound (ELBO) (Lou et al., 2024; Shi et al., 2024; Ou et al., 2025) as a surrogate for the log-likelihood:

$$\mathcal{B}_\theta(y \mid x) \triangleq \mathbb{E}_{s\sim\mathcal{U}[0,1]}\,\mathbb{E}_{y_s\sim q(y_s|s,y,x)}\,\ell_\theta(y_s,\ s,\ y \mid x) \le \log \pi_\theta(y \mid x),$$

where s denotes the diffusion timestep, y_s is the corresponding masked sequence with $y_{s,t}$ being its t-th token, $\ell_\theta(y_s, s, y \mid x) \triangleq \frac{1}{s}\sum_{t=1}^{|y|}\mathbb{1}[y_{s,t}=\texttt{[M]}]\log p_\theta(y_t \mid y_s, x)$, $q(y_s \mid s, y, x)$ is the forward masking process that independently replaces each response token with the mask token `[M]` with probability s, and p_θ is the learned mask predictor. How to estimate this bound accurately and efficiently constitutes the central challenge for RL in dLLMs.

Single-round RL for dLLMs. Several efforts have been made to address this challenge, differing primarily in how they estimate the log-likelihood. VRPO (Zhu et al., 2026a) adopts a DPO-style (Rafailov et al., 2023) preference optimization framework and estimates the ELBO via Monte Carlo (MC) sampling with n_s timestep samples:

$$\widehat{\mathcal{B}}_\theta^{\text{VRPO}}(y \mid x) = \frac{1}{n_s}\sum_{m=1}^{n_s}\ell_\theta\left(y_{s^{(m)}},\ s^{(m)},\ y \mid x\right),$$

where each $s^{(m)} \sim \mathcal{U}[0,1]$ and $y_{s^{(m)}} \sim q(\cdot \mid s^{(m)}, y, x)$. VRPO further introduces variance reduction strategies, including optimal budget allocation across timesteps and antithetic sampling between the policy and reference models. ESPO (Ou et al., 2026) also adopts the sequence-level ELBO framework and further introduces per-token normalization of importance ratios and robust KL estimation for stable large-scale training. These methods have demonstrated the effectiveness of RL post-training for dLLMs, but focus exclusively on single-round reasoning tasks.

On the other hand, d1 (Zhao et al., 2025) bypasses the ELBO and directly estimates the log-likelihood with a single forward pass $\sum_{t=1}^{|y|}\log p_\theta\left(y_t \mid x' \oplus \texttt{[M]}\cdots\texttt{[M]}\right)$, where x' is a randomly masked version of the prompt and the entire response is masked. This estimator enables token-level importance ratios and is computationally efficient, but introduces bias due to the token-level independence assumption.

Agentic RL for dLLMs. In agentic tasks, the model no longer produces a single response, but interacts with an environment over multiple turns. For each prompt x, a group of G trajectories $\{y^{(i)}\}_{i=1}^{G}$ is sampled, and each trajectory $y^{(i)} = (a_1^{(i)}, o_1^{(i)}, \ldots, a_{\mathcal{T}}^{(i)}, o_{\mathcal{T}}^{(i)})$ consists of $\mathcal{T}$ turns: at turn τ, the model generates an action $a_\tau^{(i)}$ and the environment returns an observation $o_\tau^{(i)}$. The outcome reward $R^{(i)}$ is obtained only at the end of the trajectory, and is thus shared by all $\mathcal{T}$ turns, which makes credit assignment across turns a central issue in this long-horizon agentic setting.

Recently, several advances have explored agentic settings for dLLMs. DLLM-Searcher (Zhao et al., 2026) and Zhen et al. (2026) have applied SFT and preference optimization to dLLMs in multi-turn agentic tasks, and LLaDA2.2 (Bie et al., 2026) further applies agentic RL with environment feedback. However, these methods mainly focus on improving task accuracy, leaving the accuracy–parallelism trade-off underexplored in agentic settings. In this paper, we present d3LLM-v2, an on-policy training framework for long-horizon agentic dLLMs that consists of reward-gated agentic RL with validity-based reward gating, and implicit policy regularization with trajectory-based online self-distillation. Consequently, d3LLM-v2 jointly improves the accuracy and parallelism of agentic dLLMs.

3 OUR APPROACH

In this section, we introduce our method, d3LLM-v2, for building an ultra-fast agentic dLLM.

Motivation. dLLMs inherently face an *accuracy–parallelism trade-off* (Qian et al., 2026), and this trade-off becomes even more critical in long-horizon agentic tasks, where errors can compound across the trajectory. In d3LLM-v2, as illustrated in Figure 1, we adopt *on-policy* training, in which dLLMs learn from rollouts generated by the current policy itself, to strike a balance between accuracy and parallelism. Specifically, d3LLM-v2 comprises two components:

(i) *Reward-gated agentic RL.* We perform on-policy RL for agentic dLLMs with validity-based reward gating that penalizes invalid actions, paired with implicit policy regularization equipped with narrow clipping and dual-clip, thereby improving accuracy through stable policy updates.

(ii) *Trajectory-based online self-distillation.* After the RL stage, the dLLM further learns to reproduce its own outputs in fewer steps along its own trajectories, equipped with an easy-to-hard curriculum mechanism, thereby improving efficiency while preserving accuracy.

3.1 REWARD-GATED AGENTIC REINFORCEMENT LEARNING

In this part, we introduce our RL method for agentic dLLMs. Two key challenges arise in this setting: *(i)* dLLMs decode multiple tokens in parallel within each block, and the lack of step-by-step causal conditioning makes them more likely to produce invalid actions or failed tool calls; and *(ii)* KL regularization against a reference model, commonly adopted in AR-based RL to stabilize updates, can only be coarsely approximated via the ELBO and becomes unreliable over long multi-turn trajectories. We address these challenges with two complementary techniques.

Validity-based Reward Gating. Due to bidirectional attention and parallel decoding, dLLMs lack the step-by-step causal conditioning available in AR models, and are more likely to generate structurally invalid actions. In long-horizon agentic tasks, this problem is amplified: a single invalid action wastes an interaction turn and can undermine the entire decision-making chain. To this end, we gate the reward of each turn by the validity of its action

$$\widetilde{R}_\tau^{(i)} = R^{(i)} \cdot \big(1 - \lambda \cdot \mathbb{1}\big[a_\tau^{(i)} \notin \mathcal{A}\big]\big),$$

where $a_\tau^{(i)}$ is the action at turn τ of the i-th trajectory, $\mathcal{A}$ is the set of valid actions, $R^{(i)}$ is the trajectory reward, $\mathbb{1}[\cdot]$ is the indicator function, and $\lambda \in [0, 1]$ controls the gating strength. Intuitively, gating discounts the reward for turns containing invalid actions, assigning them lower credit relative to valid turns within the same group. Since this gating is applied *before* the group-level advantage normalization in Section 2, the advantage of each turn is computed over the *gated* rewards. As a result, invalid turns tend to receive negative advantages, while valid turns from high-reward trajectories retain positive advantages. Note that without gating, the outcome reward is shared equally across all turns within a trajectory, so an invalid turn in a successful trajectory would still receive positive reinforcement. By discounting the reward before normalization, gating shifts invalid turns below the group mean, explicitly pushing the dLLM agent away from generating invalid actions.

Implicit Policy Regularization for dLLMs. As discussed in Section 2, dLLMs cannot compute the exact log-likelihood $\log \pi_\theta(y \mid x)$; only the ELBO $\mathcal{B}_\theta(y \mid x) \le \log \pi_\theta(y \mid x)$ is available as a surrogate. Inspired by VRPO (Zhu et al., 2026a) and ESPO (Ou et al., 2026), we adopt a sequence-level ELBO estimate for policy optimization. Standard RL methods use a KL penalty $\mathrm{KL}(\pi_\theta \| \pi_{\text{ref}})$ against a reference model to stabilize training. For dLLMs, however, this KL can only be approximated via the ELBO difference, which provides only a coarse estimation. For long-horizon agentic tasks, where the ELBO approximation error accumulates across multiple turns, KL regularization becomes unreliable and may even mislead optimization in the wrong direction.

To this end, we introduce *implicit policy regularization* for agentic dLLMs, replacing the explicit KL penalty. Specifically, inspired by GSPO (Zheng et al., 2025) and Ye et al. (2020), we employ narrow asymmetric clipping $(\varepsilon_l, \varepsilon_h)$ together with a dual-clip lower bound c to implicitly regularize updates without a reference model. Concretely, we maximize the following objective:

$$\mathcal{J}_{\text{RL}}(\theta) = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{\mathcal{T}}\sum_{\tau=1}^{\mathcal{T}} \max\Big(\min\big(\rho_\tau^{(i)}\widehat{A}_\tau^{(i)},\ \text{clip}(\rho_\tau^{(i)}, 1-\varepsilon_l, 1+\varepsilon_h)\,\widehat{A}_\tau^{(i)}\big),\ c\widehat{A}_\tau^{(i)}\,\mathbb{1}[\widehat{A}_\tau^{(i)}<0]\Big)\right],$$

where $\widehat{A}_\tau^{(i)} \triangleq (\widetilde{R}_\tau^{(i)} - \mu_g)/\sigma_g$ is the turn-level advantage, obtained by applying the group normalization in Section 2 to the gated rewards, and $\rho_\tau^{(i)}(\theta) \triangleq \exp\big(\big[\widehat{\mathcal{B}}_\theta\big(a_\tau^{(i)} \mid x, y_{<\tau}^{(i)}\big) - \widehat{\mathcal{B}}_{\theta_{\text{old}}}\big(a_\tau^{(i)} \mid x, y_{<\tau}^{(i)}\big)\big]/|a_\tau^{(i)}|\big)$ is the turn-level importance ratio. Here, $y_{<\tau}^{(i)} \triangleq (a_1^{(i)}, o_1^{(i)}, \ldots, o_{\tau-1}^{(i)})$ denotes the interaction before turn τ. The importance ratio is narrowly clamped, which constrains each update step. For turns with negative advantage, the outer $\max$ imposes a dual-clip lower bound, preventing unbounded negative updates from large importance ratios. Together with the gated advantages that suppress invalid actions, the objective improves long-horizon accuracy through stable policy updates.

Remark 1 (ELBO Estimation in the Importance Ratio)**.** Note that the importance ratio $\rho_\tau^{(i)}(\theta)$ also relies on the ELBO estimation. However, this ratio enters the objective multiplicatively and is bounded by the clipping mechanism, so its estimation error can affect only the magnitude of each update, within a bounded range. KL regularization, in contrast, enters the loss additively *without any clipping constraint*: its estimation error directly shifts the optimization target away, potentially causing training instability or collapse.

3.2 Trajectory-based Online Self-Distillation

In this part, we further improve the efficiency of dLLMs via *trajectory-based online self-distillation*, where the model learns to decode more tokens per forward pass along its own decoding trajectories.

Online Distillation via Self-Trajectories. The efficiency of dLLMs comes from decoding multiple tokens per forward pass. During multi-step denoising, the model denoises tokens in a certain order: confident tokens are unmasked early, while uncertain ones are determined later. We refer to this order as the *decoding trajectory*, formally represented by the *decode order* $d_t \in \{1, \ldots, |y|\}$ for each position t, the order in which token y_t is determined. This trajectory provides a natural supervision signal for distillation: by training the model to reproduce its output from an intermediate state along this trajectory, we teach it to denoise more tokens in fewer denoising steps.

Concretely, the current model π_θ serves as both teacher and student. During rollout, the model generates a response y via multi-step denoising and records the decode order $\{d_t\}_{t=1}^{|y|}$. During training, for each block, we sample an intermediate step $k \in \{1, \ldots, |y|-1\}$ and mask all positions with $d_t > k$, forming the masked set $M_k = \{t : d_t > k\}$ and constructing a noisy response $\widetilde{y} \in \mathbb{R}^{|y|}$:

$$[\widetilde{y}]_t = \begin{cases} y_t & \text{if } d_t \le k, \\ \texttt{[M]} & \text{if } d_t > k. \end{cases}$$

The model is then trained to predict all masked tokens from $\widetilde{y}$ using the following self-distillation loss

$$\mathcal{L}_{\text{distill}}(\theta) = -\,\mathbb{E}_{y\sim\pi_\theta}\,\mathbb{E}_k\left[\frac{1}{|M_k|}\sum_{t\in M_k}\log p_\theta\big(y_t \mid \widetilde{y},\, x\big)\right].$$

Unlike previous dLLM distillation methods that rely on trajectories from a fixed teacher or external ground-truth pairs, our distillation is driven entirely by on-policy rollouts: the decoding trajectories are generated by the current policy through environment interaction and continuously refreshed, so the distillation targets naturally track the evolving policy. Environment success feedback serves as a quality filter—only trajectories that successfully complete the task are distilled, preventing the reinforcement of failed action sequences. This design directly addresses the accuracy–parallelism trade-off: the model learns to decode more tokens per step on its own environment-verified trajectories, rather than fitting external targets that may mismatch its own decoding distribution.

Easy-to-Hard Curriculum Mechanism. During distillation, the noise level (mask ratio) controls how many tokens the model must decode in a single forward pass: a higher mask ratio means more positions are re-masked, requiring the model to predict more tokens simultaneously.

However, starting with high mask ratios from the beginning leads to unstable training, as the gradient is dominated by difficult high-noise samples. We therefore introduce an easy-to-hard curriculum that progressively increases the mask ratio, enabling the model to gradually learn to decode more tokens per step. Specifically, at distillation step s, the mask ratio is sampled as $r \sim \mathcal{U}\big(r_{\min},\, \bar{r}(s)\big)$, and the intermediate step k is chosen such that $|M_k| = \lceil r \cdot |y| \rceil$, where

$$\bar{r}(s) = r_0 + \min(s/S,\, 1) \cdot (r_{\max} - r_0),$$

which increases from r_0 to $r_{\max}$ over S steps. Early in distillation, the model only needs to decode a few additional tokens per step (low noise); as distillation progresses, it is gradually required to decode more tokens in parallel (high noise), resulting in stable and progressive improvement in parallelism.

4 EXPERIMENTS

This section provides the experimental results of our approach. To comprehensively evaluate the effectiveness and efficiency of our d3LLM-v2 framework,[1] we conduct experiments across three benchmark datasets and five foundation models, aiming to answer the following research questions:

Q1. Does d3LLM-v2 improve the accuracy–parallelism trade-off across tasks and model backbones?

Q2. Is each component of d3LLM-v2 effective, and how much does it contribute to the overall gain?

Q3. How sensitive is d3LLM-v2 to hyperparameters, and does it stay stable across different settings?

Experimental setup. We evaluate on three long-horizon agentic tasks: *WebShop* (Yao et al., 2022) (shopping navigation, 15 turns, success rate), *Search-R1* (Jin et al., 2025) (retrieval-augmented QA, 4 turns, exact-match accuracy), and *ALFWorld* (Shridhar et al., 2021) (text-based embodied household tasks, 50 turns, success rate). We evaluate five foundation models based on SDAR (Cheng et al., 2026) and d3LLM-LLaDA (Qian et al., 2026). The main table reports SDAR-8B ($b \in \{4, 8\}$), SDAR-4B (b=4), and fully bidirectional d3LLM-LLaDA (b=16), where b denotes the decoding block size. Contenders include autoregressive baselines (Qwen3-8B/4B (Yang et al., 2025a) with GRPO training), vanilla dLLMs with prompts, and four dLLM RL methods: d1 (Zhao et al., 2025), BGPO (Lin et al., 2026b), GDPO (Rojas et al., 2026), and ESPO (Ou et al., 2026). We report Acc, TPF, and AUP (Qian et al., 2026) as the metrics. Due to page limits, details of all five models and additional Search-R1 and ALFWorld results are deferred to Appendix B.

Implementation details. All experiments are conducted on four NVIDIA B200 GPUs, using FSDP with bfloat16 precision for both training and rollout. We optimize with AdamW (Loshchilov & Hutter, 2019) (weight decay 0.01, gradient clipping 1.0), and set the learning rate to 5×10^{-7} for 8B models and 1×10^{-6} for 4B models. On WebShop, the rollout group size is 8 and the maximum response length is 768 tokens; on Search-R1, they are 5 and 512 tokens, respectively. For agentic RL, we estimate the sequence-level ELBO of each turn with three Monte Carlo samples, where the mask ratio of each sample is drawn uniformly from $[10^{-3}, 1]$. The reward gating coefficient is λ=1, the narrow asymmetric clip range is $(\varepsilon_l, \varepsilon_h)$=$(3\times10^{-4}, 4\times10^{-4})$, and the dual-clip lower bound is c=3. For trajectory-based online self-distillation, we sample four intermediate states per successful trajectory and train with a cross-entropy loss. The easy-to-hard curriculum linearly increases the mask-ratio upper bound from 0.2 to 1.0 over 300 steps, while the lower bound stays fixed at 0.1.

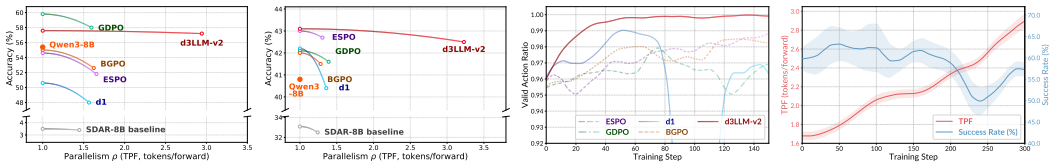
Open-source agentic dLLM training framework. To the best of our knowledge, our d3LLM-v2 is the *first open-source framework for on-policy agentic RL training* of dLLMs. Existing dLLM RL methods are developed and evaluated exclusively on single-round tasks. Recently, DLLM-Searcher (Zhao et al., 2026) extends dLLMs to agentic settings, but adopts offline preference optimization. LLaDA2.2 (Bie et al., 2026) performs on-policy agentic RL for dLLMs, but its training code has not been publicly released. Concretely, we build a dLLM actor/worker stack on top of veRL-agent (Feng et al., 2025b), together with an adapted SGLang (Zheng et al., 2024) rollout interface, enabling existing dLLM RL algorithms to operate in the multi-turn agentic setting. The key adaptation is replacing the AR chain-rule log-probability computation with a unified log-likelihood estimation interface for dLLMs, which supports both the one-step token-level estimator of d1 (Zhao et al., 2025) and the sequence-level ELBO estimators of BGPO (Lin et al., 2026b), GDPO (Rojas et al., 2026), ESPO (Ou et al., 2026), and our d3LLM-v2, and computes importance ratios accordingly.

Besides, we identified a data leakage issue in the original veRL-agent code repository, where evaluation goals overlap with the training set, and fixed it by a corrected train/evaluation split. As a result, some reported numbers may appear lower than those in prior work that uses the original (leaked) split. For example, GiGPO (Feng et al., 2025a) reports a WebShop success rate of $66.1 \pm 3.7\%$ for GRPO with Qwen2.5-7B, whereas we obtain 55.2% for GRPO with Qwen2.5-7B under the corrected split. All contenders in our experiments are evaluated under the same fixed split for a fair comparison.

[1] Code is at anonymous GitHub: https://anonymous.4open.science/r/d3LLM-v2-Code

Table 1: Main results on WebShop and Search-R1 for SDAR-8B with block sizes $b \in \{4, 8\}$, SDAR-4B (b=4), and d3LLM-LLaDA (b=16). ACC is the WebShop success rate or the overall Search-R1 exact-match accuracy, TPF is the number of generated response tokens per network forward evaluation, and AUP is Accuracy Under Parallelism; higher is better for all metrics.

Method	SDAR-8B (b=4)			SDAR-8B (b=8)			SDAR-4B (b=4)			d3LLM-LLaDA (b=16)		
	ACC↑	TPF↑	AUP↑	ACC↑	TPF↑	AUP↑	ACC↑	TPF↑	AUP↑	ACC↑	TPF↑	AUP↑
Agentic task: WebShop (Yao et al., 2022)												
Qwen3 (+GRPO)	55.40	1.000	55.40	55.40	1.000	55.40	50.60	1.000	50.60	55.40	1.000	55.40
Vanilla dLLM (+prompt)	3.400	1.447	4.559	1.200	1.552	1.499	4.000	1.267	5.068	0.200	3.961	1.058
d1	50.20	1.550	76.52	46.20	2.188	101.0	45.20	1.678	76.31	26.40	2.643	64.95
BGPO	52.60	1.624	86.56	43.00	1.718	74.30	56.40	1.713	96.80	41.00	2.525	103.3
GDPO	58.00	1.596	93.41	52.40	1.789	87.86	55.40	2.049	111.6	43.80	2.768	121.2
ESPO	53.00	1.494	78.78	48.60	1.724	82.34	47.40	2.002	82.57	40.60	2.863	115.0
d3LLM-v2 (ours)	**57.20**	**2.944**	**168.1**	**49.60**	**3.142**	**141.2**	**54.00**	**2.873**	**132.8**	**40.40**	**7.611**	**304.7**
Agentic task: Search-R1 (Jin et al., 2025)												
Qwen3 (+GRPO)	40.80	1.000	40.80	40.80	1.000	40.80	41.82	1.000	41.82	40.80	1.000	40.80
Vanilla dLLM (+prompt)	32.53	1.242	40.83	17.00	1.304	22.39	26.98	1.213	32.62	19.30	2.494	47.97
d1	40.49	1.358	56.12	34.97	2.442	43.09	37.87	2.125	80.55	29.12	2.629	73.40
BGPO	41.54	1.281	53.53	43.32	1.336	57.97	40.32	1.326	53.65	36.66	2.728	99.73
GDPO	41.69	1.390	58.14	41.36	1.438	59.77	38.62	1.300	50.83	37.83	3.214	120.8
ESPO	42.65	1.306	55.98	42.99	1.391	59.94	40.55	1.354	50.90	37.30	2.944	109.0
d3LLM-v2 (ours)	**42.50**	**3.229**	**136.6**	**41.23**	**4.212**	**172.1**	**42.01**	**3.051**	**127.1**	**36.35**	**9.081**	**322.5**



(a) (b) (c) (d)

Figure 2: Results with SDAR-8B (b=4). (a) & (b): Accuracy–parallelism curves on WebShop and Search-R1 tasks, respectively; each point corresponds to a decoding confidence threshold, and d3LLM-v2 achieves the best accuracy–parallelism trade-off. (c): Valid action ratio during the RL stage on WebShop, where d3LLM-v2 with reward gating achieves a higher ratio than all baselines. (d): TPF (left axis) and success rate (right axis) during online self-distillation on WebShop, showing that parallelism grows steadily while task accuracy is preserved.

4.1 MAIN RESULTS

Table 1 presents the main WebShop and Search-R1 results. Figure 2(a)–(b) shows the corresponding accuracy–parallelism curves for SDAR-8B (b=4). We highlight three key observations. First, the base dLLMs with task-specific prompts alone perform poorly on both agentic tasks, confirming that pre-trained dLLMs cannot effectively handle multi-turn decision making without further training. Second, existing dLLM RL methods (d1, BGPO, GDPO, ESPO) substantially improve task accuracy, reaching levels competitive with or even exceeding the size-matched AR baselines in several configurations, but their TPF remains substantially below that of d3LLM-v2, indicating that improved task accuracy alone does not fully exploit parallel decoding. Third, our d3LLM-v2 achieves both competitive accuracy and significantly higher TPF across model–task configurations. By combining reward-gated agentic RL with trajectory-based online self-distillation, d3LLM-v2 attains the highest AUP on both tasks, pushing the accuracy–parallelism frontier beyond all baselines. As shown in Figure 2(a)–(b), the accuracy–parallelism curve of d3LLM-v2 consistently dominates those of all contenders.

Larger block sizes provide more room for parallelism improvement through self-distillation. For SDAR-8B on Search-R1, increasing b from 4 to 8 raises the TPF of d3LLM-v2 from 3.229 to 4.212, while accuracy remains similar (42.50% vs. 41.23%). Our framework also works with fully bidirectional dLLMs: with d3LLM-LLaDA (b=16), d3LLM-v2 reaches TPF values of 7.611 and 9.081 on WebShop and Search-R1, respectively, and achieves the highest AUP in both settings while maintaining accuracy competitive with the dLLM RL baselines. Together, these results affirmatively answer **Q1**: d3LLM-v2 effectively improves the accuracy–parallelism trade-off across different tasks, model backbones, and block-size configurations. Due to page limits, more detailed results are deferred to Appendix B.

Table 2: Ablation study on WebShop with SDAR-8B (b=4). We report Acc, TPF, and AUP for each component combination. All variants share the same SFT initialization and training budget.

Agentic RL		Online Distillation		WebShop		
Reward Gating	Implicit Regularization	Self-Trajectory	Easy-to-hard Curriculum	ACC ↑	TPF ↑	AUP ↑
				53.00	1.494	78.78
✓				49.60	1.601	79.41
	✓			51.80	1.656	87.08
✓	✓			**59.80**	**1.683**	**96.96**
✓	✓			59.80	1.683	96.96
✓	✓	✓		48.80	2.478	120.8
✓	✓		✓	48.80	2.862	138.9
✓	✓	✓	✓	**57.20**	**2.944**	**168.1**

4.2 Ablation Study

To answer **Q2**, we conduct an ablation study on SDAR-8B (b=4) using WebShop, progressively adding each component of d3LLM-v2. Table 2 reports the results: the upper half examines the two components of reward-gated agentic RL, while the lower half examines the two components of trajectory-based online self-distillation.

Reward-gated agentic RL. As shown in Table 2 (upper), adding implicit policy regularization alone, which replaces explicit KL with narrow asymmetric clipping, improves AUP by approximately 10% over the baseline without either component. This confirms the motivation in Section 3.1: ELBO-based KL regularization is unreliable in multi-turn agentic settings, and removing it in favor of implicit clipping-based regularization leads to more stable updates. Adding reward gating on top further improves both Acc and AUP, as invalid actions are turned into negative-advantage examples rather than being reinforced alongside successful trajectories. The combination of both components yields the best RL result, improving AUP by over 23% relative to the baseline. Figure 2(c) further supports this finding: the variant with reward gating maintains a consistently higher valid action ratio throughout training, confirming that the gating mechanism actively suppresses malformed outputs as described in Section 3.1.

Trajectory-based online self-distillation. As shown in Table 2 (lower), starting from the full agentic RL checkpoint, adding either self-trajectory masking or easy-to-hard curriculum alone substantially improves TPF and AUP, demonstrating that distillation effectively teaches the model to decode more tokens per forward pass. However, using either component alone incurs a noticeable accuracy drop, because the model encounters difficult high-noise states before it is ready. When both components are combined, the accuracy is largely recovered while TPF remains high, achieving the best AUP with approximately 70% improvement over the RL-only variant. This confirms the design in Section 3.2: the self-trajectory provides a natural and faithful supervision signal aligned with the model's own denoising behavior, while the easy-to-hard curriculum ensures training stability by exposing the model to progressively harder intermediate states. Figure 2(d) illustrates this process: during distillation, TPF grows steadily while task accuracy is preserved. Together, these results answer **Q2**: each component of d3LLM-v2 contributes meaningfully to the final performance, with agentic RL improving accuracy and self-distillation improving parallelism, jointly striking a balance in the accuracy–parallelism trade-off.

4.3 Hyperparameter Analysis

To answer **Q3**, Table 3 examines the sensitivity of two key hyperparameters on SDAR-8B (b=4, WebShop), varying one at a time. We first sweep the reward-gating coefficient λ from 0 to 1. As λ increases, Acc improves monotonically, confirming that stronger gating more effectively suppresses invalid actions. The AUP remains stable across intermediate values and reaches its best at λ=1 (full gating), indicating that the method is robust to the choice of λ.

Table 3: Hyperparameter sensitivity on SDAR-8B (b=4, WebShop). (i) Reward-gating coefficient λ; (ii) clip scale κ for the asymmetric clipping interval.

	(i) Reward-gating λ				*(ii) Clip scale κ*			
	0	0.25	0.50	1.00	0.5	1.0	2.0	500
ACC ↑	51.80	53.40	55.60	**59.80**	52.20	**59.80**	54.20	53.80
TPF ↑	1.656	1.633	1.597	**1.683**	1.674	**1.683**	1.736	1.598
AUP ↑	87.08	87.70	86.21	**96.96**	87.67	**96.96**	95.96	87.27

We then vary the clip scale $\kappa \cdot (3 \times 10^{-4}, 4 \times 10^{-4})$, which controls the width of the asymmetric clipping interval. As shown in Table 3 (ii), the best AUP is achieved at κ=1.0, which we use as the default across all configurations. Performance remains stable across nearby values, confirming that the method is not sensitive to the exact clipping width as long as the constraint remains narrow. However, setting κ to a very large value (e.g., 500) removes the proximity constraint and leads to clear performance degradation, validating the necessity of implicit regularization through narrow clipping. These results answer **Q3**: d3LLM-v2 is robust to its key hyperparameters.

Variance across runs. Table 4 summarizes run-to-run variation for BGPO, GDPO, ESPO, and d3LLM-v2 on WebShop with SDAR-8B (b=8), reporting the mean and sample standard deviation across three independent runs. All methods exhibit low to moderate variance. Notably, d3LLM-v2 maintains a small accuracy standard deviation, consistently achieving the highest AUP by a large margin over other contenders. This confirms that the accuracy–parallelism improvement of d3LLM-v2 is a stable and reproducible outcome, further supporting **Q3**.

Table 4: Three-run results on WebShop with SDAR-8B (b=8), with standard deviation.

Method	ACC↑	TPF↑	AUP↑
BGPO	45.20±3.47	1.74±0.02	84.39±10.5
GDPO	**52.73±0.42**	1.81±0.02	93.57±4.94
ESPO	50.33±1.51	1.76±0.06	87.62±5.23
d3LLM-v2 (ours)	49.73±0.90	**2.93±0.20**	**142.11±7.88**

Wall-clock speed comparison. In addition to AUP, we measure the wall-clock throughput of all methods in terms of tokens per second (TPS) on WebShop with SDAR-8B (b=4). All methods are evaluated on a single NVIDIA B200 GPU with batch size 1. As shown in Table 5, d3LLM-v2 achieves 276.71 TPS, outperforming the best existing dLLM RL method by 1.49× and the AR baseline (Qwen3-8B) by 1.25×, while maintaining competitive accuracy. Note that all dLLM RL baselines achieve similar TPS to vanilla SDAR. In contrast, d3LLM-v2 achieves higher TPF and a wall-clock advantage, confirming that the parallelism improvement from online distillation leads to TPS speedup and pushes the accuracy–parallelism frontier of agentic dLLMs.

Table 5: Wall-clock throughput on WebShop with SDAR-8B (b=4) on a single NVIDIA B200 GPU. We report TPS, TPF, and ACC.

Method	TPS ↑	TPF ↑	ACC ↑
Qwen3-8B (AR)	221.26	1.000	55.40
SDAR (+prompt)	181.22	1.447	3.400
d1	181.30	1.550	50.20
BGPO	186.24	1.624	52.60
GDPO	180.12	1.596	58.00
ESPO	172.97	1.494	53.00
d3LLM-v2 (ours)	**276.71**	**2.944**	57.20

5 CONCLUSION

In this paper, we present d3LLM-v2, a two-stage on-policy training framework that balances accuracy and parallelism for dLLMs in long-horizon agentic tasks. While recent advances have improved dLLMs for single-round tasks, extending them to long-horizon agentic settings remains underexplored; moreover, the accuracy–parallelism trade-off inherent in dLLMs becomes even more critical when the model interacts with environments across multiple decision-making turns. *(i)* We introduce *reward-gated agentic RL*, which suppresses invalid actions via validity-based reward gating and stabilizes training through implicit policy regularization that avoids the unreliable ELBO-based KL approximation over multi-turn trajectories. *(ii)* We propose *trajectory-based online self-distillation*, which teaches the model to decode more tokens per forward pass along its own decoding trajectory, with an easy-to-hard curriculum that progressively increases parallelism. Experiments on three agentic tasks across five foundation models demonstrate that d3LLM-v2 consistently achieves the highest AUP scores, delivering competitive task accuracy with significantly improved decoding parallelism, pushing the accuracy–parallelism frontier of agentic dLLMs.

AI USE STATEMENT

In this work, we used generative AI tools for editing and polishing the manuscript text, and for assisting with writing and debugging experiment code. These tools were primarily used to improve the clarity and fluency of our writing, as well as to accelerate the implementation of training pipelines and evaluation scripts. We have not used generative AI tools for formulating research ideas, designing the core methodology, or generating scientific claims; the problem formulation, algorithmic design, and all experimental analyses are entirely human contributions. We have reviewed all AI-assisted work. Specifically, all polished text was manually reviewed and revised by the authors to ensure accuracy and faithfulness to the intended meaning, and all LLM-generated code was verified and tested for correctness by 2 authors before integration into the codebase. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work focuses on foundational reinforcement learning and distillation techniques for agentic diffusion large language models. Our research is methodological in nature and does not involve human subjects, private data, or direct deployment in sensitive applications. The benchmark environments used in our experiments are publicly available and well-established in the community. We do not foresee immediate negative societal impacts arising from this work.

REPRODUCIBILITY STATEMENT

To ensure reproducibility, we provide detailed descriptions of our training procedure, hyperparameters, and evaluation protocols. Our source code is publicly available at an anonymous GitHub repository. We believe these resources are sufficient for reproducing the main results reported in this paper.

REFERENCES

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems 34 (NeurIPS)*, pp. 17981–17993, 2021.

Tiwei Bie, Maosong Cao, Kun Chen, Lun Du, Mingliang Gong, Zhuochen Gong, Yanmei Gu, Jiaqi Hu, Zenan Huang, Zhenzhong Lan, et al. LLaDA2.0: Scaling up diffusion language models to 100B. *ArXiv preprint*, arXiv:2512.15745, 2025.

Tiwei Bie, Jiansheng Cai, Maosong Cao, Xiang Cao, Binbin Chen, Fuyuan Chen, Kun Chen, Yuqi Ding, Lun Du, Zhuocheng Gong, Yanmei Gu, Kaiyuan Guan, Xiaoxu Guo, et al. LLaDA2.2: Enabling agentic diffusion language models via levenshtein editing. Technical report, Inclusion AI, July 2026. URL https://github.com/inclusionAI/LLaDA2.X/blob/main/LLaDA2_2_tech_report.pdf.

Zigeng Chen, Gongfan Fang, Xinyin Ma, Ruonan Yu, and Xinchao Wang. dParallel: Learnable parallel decoding for dLLMs. In *Proceedings of the 14th International Conference on Learning Representations (ICLR)*, pp. 53193–53211, 2026a.

Zigeng Chen, Gongfan Fang, Xinyin Ma, Ruonan Yu, and Xinchao Wang. DMax: Aggressive parallel decoding for dLLMs. *ArXiv preprint*, arXiv:2604.08302, 2026b.

Shuang Cheng, Yihan Bian, Dawei Liu, Yuhua Jiang, Yihao Liu, Linfeng Zhang, Qian Yao, Zhongbo Tian, Wenhai Wang, Qipeng Guo, Kai Chen, Biqing Qi, and Bowen Zhou. SDAR: A synergistic diffusion-autoregression paradigm for scalable sequence generation. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 22058–22075, 2026.

Phuong Tuan Dat, Qi Li, and Xinchao Wang. dOPSD: On-policy self-distillation for diffusion language models. *ArXiv preprint*, arXiv:2607.04428, 2026.

Justin Deschenaux and Caglar Gulcehre. Beyond autoregression: Fast LLMs via self-distillation through time. In *Proceedings of the 13th International Conference on Learning Representations (ICLR)*, pp. 5007–5045, 2025.

Yijia Fan, Jing Yang, Mingyu Liu, Kaitong Cai, Jian Wang, Keze Wang, and Jusheng Zhang. Reinforcement learning for diffusion LLMs via energy-based Gibbs alignment. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 45938–45948, 2026.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. In *Advances in Neural Information Processing Systems 38 (NeurIPS)*, pp. 46375–46408, 2025a.

Lang Feng et al. veRL-agent. https://github.com/langfengq/verl-agent, 2025b.

Yonggan Fu, Lexington Whalen, Zhifan Ye, Xin Dong, Shizhe Diao, Jingyu Liu, Chengyue Wu, Hao Zhang, Enze Xie, Song Han, Maksim Khadkevich, Jan Kautz, Yingyan Celine Lin, and Pavlo Molchanov. Efficient-dlm: From autoregressive to diffusion language models, and beyond in speed. *ArXiv preprint*, arXiv:2512.14067, 2025.

Yifeng Gao, Ziang Ji, Yuxuan Wang, Biqing Qi, Hanlin Xu, and Linfeng Zhang. Self speculative decoding for diffusion large language models. *ArXiv preprint*, arXiv:2510.04147, 2025.

Shansan Gong, Ruixiang Zhang, Huangjie Zheng, Jiatao Gu, Navdeep Jaitly, Lingpeng Kong, and Yizhe Zhang. DiffuCoder: Understanding and improving masked diffusion models for code generation. In *Proceedings of the 14th International Conference on Learning Representations (ICLR)*, pp. 15723–15750, 2026.

Google DeepMind. Gemini diffusion, 2025. https://deepmind.google/models/gemini-diffusion/.

Yanzhe Hu, Yijie Jin, Pengfei Liu, Kai Yu, and Zhijie Deng. Lightningrl: Breaking the accuracy-parallelism trade-off of block-wise dllms via reinforcement learning. *ArXiv preprint*, arXiv:2603.13319, 2026.

Zemin Huang, Zhiyang Chen, Zijun Wang, Tiancheng Li, and Guo-Jun Qi. Reinforcing the diffusion chain of lateral thought with diffusion language models. In *Advances in Neural Information Processing Systems 38 (NeurIPS)*, pp. 152677–152710, 2025.

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.

Wonjun Kang, Kevin Galim, Seunghyuk Oh, Minjae Lee, Yuchen Zeng, Shuibai Zhang, Coleman Hooper, Yuezhou Hu, Hyung Koo, Nam Ik Cho, and Kangwook Lee. ParallelBench: Understanding the trade-offs of parallel decoding in diffusion LLMs. In *Proceedings of the 14th International Conference on Learning Representations (ICLR)*, pp. 89900–89938, 2026.

Samar Khanna, Siddhant Kharbanda, Shufan Li, Harshit Varma, Eric Wang, Sawyer Birnbaum, Ziyang Luo, Yanis Miraoui, Akash Palrecha, Stefano Ermon, et al. Mercury: Ultra-fast language models based on diffusion. *ArXiv preprint*, arXiv:2506.17298, 2025.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019.

Jia-Nan Li, Jian Guan, Wei Wu, and Chongxuan Li. ReFusion: A diffusion large language model with parallel autoregressive decoding. In *Proceedings of the 14th International Conference on Learning Representations (ICLR)*, pp. 53846–53869, 2026.

Haokun Lin, Xinle Jia, Shaozhen Liu, Shujun Xia, Weitao Huang, Haobo Xu, Junyang Li, Yicheng Xiao, Xingrun Xing, Ziyu Guo, et al. Efficient diffusion language models: A comprehensive survey. *Authorea Preprints*, 2026a. doi: 10.22541/au.176918713.36402137/v1.

Nianyi Lin, Jiajie Zhang, Lei Hou, and Juanzi Li. Boundary-guided policy optimization for memory-efficient RL of diffusion large language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7539–7550, 2026b.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, 2019.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, pp. 32819–32848, 2024.

Xiaocheng Lu, Hualei Zhang, Shuhan Guo, Jie Zhang, Xiaoyi Pang, Jian Liu, Haoxi Li, Bohai Gu, Haoxuan Che, Jingcai Guo, and Song Guo. OPTD: On-policy transition distillation with consistency-guided adaptive compression for few-step diffusion language models. *ArXiv preprint*, arXiv:2608.02942, 2026.

Yifu Luo, Zeyu Chen, Haoyu Wang, Xinhao Hu, Yuxuan Zhang, Zhizhou Sha, and Shiwei Liu. Learning from the self-future: On-policy self-distillation for dLLMs. *ArXiv preprint*, arXiv:2606.18195, 2026.

Xinyin Ma, Runpeng Yu, Gongfan Fang, and Xinchao Wang. dKV-Cache: The cache for diffusion language models. In *Advances in Neural Information Processing Systems 38 (NeurIPS)*, pp. 149009–149033, 2025a.

Yuxin Ma, Lun Du, Lanning Wei, Kun Chen, Qian Xu, Kangyu Wang, Guofeng Feng, Guoshan Lu, Lin Liu, Xiaojing Qi, et al. dInfer: An efficient inference framework for diffusion language models. *ArXiv preprint*, arXiv:2510.08666, 2025b.

Zanlin Ni, Shenzhi Wang, Yang Yue, Tianyu Yu, Weilin Zhao, Yeguo Hua, Tianyi Chen, Jun Song, Cheng Yu, Bo Zheng, and Gao Huang. The flexibility trap: Rethinking the value of arbitrary order in diffusion language models. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. In *Advances in Neural Information Processing Systems 38 (NeurIPS)*, pp. 50608–50646, 2025.

Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. Your absorbing discrete diffusion secretly models the conditional distributions of clean data. In *Proceedings of the 13th International Conference on Learning Representations (ICLR)*, pp. 64972–65009, 2025.

Jingyang Ou, Jiaqi Han, Minkai Xu, Shaoxuan Xu, Jianwen Xie, Stefano Ermon, Yi Wu, and Chongxuan Li. Principled RL for diffusion LLMs emerges from a sequence-level perspective. In *Proceedings of the 14th International Conference on Learning Representations (ICLR)*, pp. 4941–4966, 2026.

Yu-Yang Qian, Junda Su, Lanxiang Hu, Peiyuan Zhang, Zhijie Deng, Peng Zhao, and Hao Zhang. d3LLM: Ultra-fast diffusion LLM using pseudo-trajectory distillation. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, pp. to appear, 2026.

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, et al. Qwen2.5 technical report. *ArXiv preprint*, arXiv:2412.15115, 2025.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems 36 (NeurIPS)*, pp. 53728–53741, 2023.

Haolin Ren, Ziyang Huang, Chenhao Yuan, Jun Zhao, and Kang Liu. Trace-based on-policy distillation for masked diffusion language models. *ArXiv preprint*, arXiv:2607.16872, 2026.

Kevin Rojas, Jiahe Lin, Kashif Rasul, Anderson Schneider, Yuriy Nevmyvaka, Molei Tao, and Wei Deng. Improving reasoning for diffusion language models via group diffusion policy optimization. In *Proceedings of the 14th International Conference on Learning Representations (ICLR)*, pp. 1938–1973, 2026.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Mariano Marroquin, Justin T. Chiu, Alexander M. Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *Advances in Neural Information Processing Systems 37 (NeurIPS)*, pp. 130136–130184, 2024.

Subham Sekhar Sahoo, Justin Deschenaux, Aaron Gokaslan, Guanghan Wang, Justin T. Chiu, and Volodymyr Kuleshov. The diffusion duality. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, pp. 52584–52619, 2025.

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, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *ArXiv preprint*, arXiv:2402.03300, 2024.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. Simplified and generalized masked diffusion for discrete data. In *Advances in Neural Information Processing Systems 37 (NeurIPS)*, pp. 103131–103167, 2024.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning Text and Embodied Environments for Interactive Learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

Yuxuan Song, Zheng Zhang, Cheng Luo, Pengyang Gao, Fan Xia, Hao Luo, Zheng Li, Yuehang Yang, Hongli Yu, Xingwei Qu, et al. Seed diffusion: A large-scale diffusion language model with high-speed inference. *ArXiv preprint*, arXiv:2508.02193, 2025.

Xiaohang Tang, Rares Dolga, Sangwoong Yoon, and Ilija Bogunovic. wd1: Weighted policy optimization for reasoning in diffusion language models. In *Proceedings of the 14th International Conference on Learning Representations (ICLR)*, pp. 150357–150385, 2026a.

Xiaohang Tang, Keyue Jiang, Che Liu, Qifang Zhao, Xiaoxiao Xu, Sangwoong Yoon, and Ilija Bogunovic. GDSD: Reinforcement learning as guided denoiser self-distillation for diffusion language models. *ArXiv preprint*, arXiv:2605.29398, 2026b.

Chenyu Wang, Paria Rashidinejad, Andy (DiJia) Su, Song Jiang, Sid Wang, Siyan Zhao, Cai Zhou, Shannon Shen, Feiyu Chen, Tommi Jaakkola, Yuandong Tian, and Bo Liu. SPG: Sandwiched policy gradient for masked diffusion language models. In *Proceedings of the 14th International Conference on Learning Representations (ICLR)*, pp. 24623–24660, 2026a.

Guanghan Wang, Gilad Turok, Yair Schiff, Marianne Arriola, and Volodymyr Kuleshov. d2: Improving reasoning in diffusion language models via trajectory likelihood estimation. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026b.

Yinjie Wang, Ling Yang, Bowen Li, Ye Tian, Ke Shen, and Mengdi Wang. Revolutionizing reinforcement learning framework for diffusion large language models. In *Proceedings of the 14th International Conference on Learning Representations (ICLR)*, pp. 101476–101501, 2026c.

Chenxing Wei, Jiazhen Kang, Hong Wang, Jianqing Zhang, Hao Jiang, Xiaolong Xu, Ningyuan Sun, Ying He, F. Richard Yu, Yao Shu, and Bo Jiang. LFPO: Likelihood-free policy optimization for masked diffusion models. *ArXiv preprint*, arXiv:2603.01563, 2026.

Chengyue Wu, Hao Zhang, Shuchen Xue, Shizhe Diao, Yonggan Fu, Zhijian Liu, Pavlo Molchanov, Ping Luo, Song Han, and Enze Xie. Fast-dLLM v2: Efficient block-diffusion LLM. *ArXiv preprint*, arXiv:2509.26328, 2025.

Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. Fast-dLLM: Training-free acceleration of diffusion LLM by enabling KV cache and parallel decoding. In *Proceedings of the 14th International Conference on Learning Representations (ICLR)*, pp. 57027–57051, 2026.

Shutong Wu and Jiawei Zhang. Free draft-and-verification: Toward lossless parallel decoding for diffusion large language models. In *NeurIPS Workshop on Efficient Reasoning*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, et al. Qwen3 technical report. *ArXiv preprint*, arXiv:2505.09388, 2025a.

Jingyi Yang, Guanxu Chen, Xuhao Hu, and Jing Shao. Taming masked diffusion language models via consistency trajectory reinforcement learning with fewer decoding step. *ArXiv preprint*, arXiv:2509.23924, 2025b.

Ling Yang, Ye Tian, Bowen Li, Xinchen Zhang, Ke Shen, Yunhai Tong, and Mengdi Wang. MMaDA: Multimodal large diffusion language models. In *Advances in Neural Information Processing Systems 38 (NeurIPS)*, pp. 138867–138907, 2025c.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2369–2380, 2018.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems 35 (NeurIPS)*, pp. 20744–20757, 2022.

Deheng Ye, Zhao Liu, Mingfei Sun, Bei Shi, Peilin Zhao, Hao Wu, Hongsheng Yu, Shaojie Yang, Xipeng Wu, Qingwei Guo, et al. Mastering complex control in MOBA games with deep reinforcement learning. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI)*, pp. 6672–6679, 2020.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7B: Diffusion large language models. *ArXiv preprint*, arXiv:2508.15487, 2025.

Anthony Zhan. Simple policy gradients for reasoning with diffusion language models. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026.

Jiahao Zhao, Shaoxuan Xu, Zhongxiang Sun, Fengqi Zhu, Jingyang Ou, Yuling Shi, Chongxuan Li, Xiao Zhang, and Jun Xu. DLLM-Searcher: Adapting diffusion language model for efficient search agents. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 2498–2508, 2026.

Siyan Zhao, Devaansh Gupta, Qinqing Zheng, and Aditya Grover. d1: Scaling reasoning in diffusion large language models via reinforcement learning. In *Advances in Neural Information Processing Systems 38 (NeurIPS)*, pp. 56729–56762, 2025.

Huiling Zhen, Weizhe Lin, Renxi Liu, Kai Han, Yiming Li, Yuchuan Tian, Hanting Chen, Xiaoguang Li, Xiaosong Li, Chen Chen, et al. DLLM Agent: See farther, run faster. *ArXiv preprint*, arXiv:2602.07451, 2026.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization. *ArXiv preprint*, arXiv:2507.18071, 2025.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. SGLang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems 37 (NeurIPS)*, pp. 62557–62583, 2024.

Jianyuan Zhong, Kaibo Wang, Ding Ding, Zijin Feng, Haoli Bai, Yang Xiang, Jiacheng Sun, and Qiang Xu. Stabilizing reinforcement learning for diffusion language models. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026.

Fengqi Zhu, Rongzhen Wang, Shen Nie, Xiaolu Zhang, Chunwei Wu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. LLaDA 1.5: Variance-reduced preference optimization for large language diffusion models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11425–11460, 2026a.

Huaisheng Zhu, MingYu Liu, Junze Liu, Zhen Ge, Tian Wang, Jiri Gesi, Dakuo Wang, Weiqi Zhang, Houyu Zhang, Yufan Guo, Xian Li, Bing Yin, and Sujay Sanghavi. On-policy self-distillation for efficient diffusion language models with early-stage calibration. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 26954–26965, 2026b.

A Related Work

In the following, we discuss the related topics.

Diffusion Language Models. Diffusion language models have attracted growing attention as an alternative to AR models. General discrete diffusion frameworks such as D3PM (Austin et al., 2021) and SEDD (Lou et al., 2024) include absorbing-state formulations, while masked diffusion models progressively replace tokens with masks and learn a denoiser to reverse the corruption process (Shi et al., 2024). MDLM introduces a simplified training objective (Sahoo et al., 2024), whereas RADD reparameterizes absorbing diffusion in terms of time-independent conditional distributions of clean data (Ou et al., 2025); both improve diffusion-LM's performance. To reduce inference steps, SDTT (Deschenaux & Gulcehre, 2025) distills multi-step teacher predictions into fewer student steps, while Duo (Sahoo et al., 2025) combines a Gaussian-process-guided training curriculum with Discrete Consistency Distillation, adapting consistency distillation to discrete diffusion and enabling competitive few-step generation.

The development of open-source foundation dLLMs has further advanced this field. LLaDA (Nie et al., 2025) is a native 8B dLLM trained from scratch, while Dream (Ye et al., 2025) initializes from Qwen2.5-7B (Qwen et al., 2025) and is subsequently pretrained as a 7B diffusion LM. More recently, the LLaDA2.0 (Bie et al., 2025) scales block-diffusion MoE models to 100B total parameters through AR-to-diffusion conversion, with the authors reporting competitive frontier-scale performance.

With the growing interest of the research community, an increasing number of works seek to improve dLLMs along two axes: efficiency and generation quality. On the efficiency side, recent methods explore caching, speculative decoding, trajectory-aware decoding, and learned parallel decoding (Wu et al., 2026; Ma et al., 2025a; Gao et al., 2025; Ma et al., 2025b; Yang et al., 2025b; Wu & Zhang, 2025; Chen et al., 2026a;b); a recent survey provides a broader taxonomy of this area (Lin et al., 2026a). Fast-dLLM (Wu et al., 2026) designs a training-free method based on block-wise approximate KV caching for dLLMs. dKV (Ma et al., 2025a) introduces delayed KV caching that stores key–value states using a delayed conditioning strategy to speed up inference. dInfer (Ma et al., 2025b) integrates iteration smoothing, hierarchical decoding, and system-level optimizations to accelerate dLLMs across multiple dimensions. dParallel (Chen et al., 2026a) proposes a learnable parallel decoding scheme trained via certainty-forcing distillation, achieving notable speedups. DMax (Chen et al., 2026b) reformulates decoding as progressive self-refinement from mask embeddings to token embeddings, enabling aggressive parallelism while mitigating error accumulation. ParallelBench (Kang et al., 2026) introduces a benchmark dedicated to dLLM decoding and shows that aggressive parallel decoding can substantially degrade quality, underscoring the need to navigate the accuracy–parallelism trade-off.

On the quality side, MMaDA (Yang et al., 2025c) introduces a unified policy-gradient RL method to improve dLLM performance across multiple modalities. ReFusion (Li et al., 2026) initializes from Qwen3-8B and combines inter-slot diffusion selection with intra-slot autoregressive infilling; the authors report a 34% average gain over prior masked diffusion models across seven benchmarks. TraceRL (Wang et al., 2026c) incorporates preferred inference trajectories into post-training, producing the TraDo model family, which demonstrates strong performance on math and coding tasks.

Recently, a growing body of work develops on-policy or trajectory-aware distillation for dLLMs. Luo et al. (2026) and Dat et al. (2026) construct supervision from the model's own denoising trajectory via self-future privileged information or later-to-earlier soft distribution matching, to improve single-turn reasoning quality. TOPD (Ren et al., 2026) distills a frozen external teacher on student-visited trace states for training compute efficiency. On the decoding efficiency side, COPSD (Zhu et al., 2026b) calibrates early-stage predictions from later diffusion states to enable more parallel token commits, and OPTD (Lu et al., 2026) learns compressed transitions on student-visited states verified by a frozen teacher, explicitly targeting the quality–parallelism trade-off for few-step generation. These methods are mainly developed for single-turn tasks; in contrast, our second stage (i.e., trajectory-based online self-distillation) distills environmental successful multi-turn trajectories, jointly navigating both accuracy and parallelism for long-horizon agentic dLLMs.

Unlike d3LLM, which constructs pseudo-trajectories for single-turn generation, our d3LLM-v2 framework targets the multi-turn agentic setting. Our contribution is a unified framework tailored to the central challenge of agentic dLLMs: learning from environment-interactive, policy-generated

multi-turn trajectories. The framework couples agentic RL for dLLMs, together with online self-distillation on successful agentic trajectories, jointly navigating the accuracy–parallelism trade-off as the policy evolves.

Reinforcement Learning for dLLMs. Recent studies have adapted post-training objectives to improve the reasoning capabilities of dLLMs. A central challenge is that evaluating the exact likelihood generally requires marginalizing over latent denoising paths, while the generation process does not admit a tractable autoregressive likelihood factorization (Zhu et al., 2026a; Zhao et al., 2025; Hu et al., 2026). Existing methods therefore rely on specific likelihood surrogates and optimization objectives.

VRPO (Zhu et al., 2026a) develops a variance-reduced, DPO-style (Rafailov et al., 2023) preference optimization framework based on Monte Carlo ELBO estimation. For reward-based policy optimization, d1 (Zhao et al., 2025) uses a mean-field factorization of sequence likelihood together with a one-step, prompt-masked estimator of per-token log probabilities. UniGRPO in MMaDA (Yang et al., 2025c) samples denoising stages and averages log probabilities at masked positions for unified policy optimization across text reasoning, multimodal understanding, and image generation, while coupled-GRPO in DiffuCoder (Gong et al., 2026) reduces the variance of token-level estimates using complementary masking. wd1 (Tang et al., 2026a) instead adopts a ratio-free loss that optimizes a weighted diffusion log-likelihood objective.

Sequence-level approaches use the ELBO as a proxy for sequence likelihood. GDPO (Rojas et al., 2026) reduces estimation variance through semi-deterministic integration schemes, including Gaussian quadrature, and performs policy optimization with sequence-level importance ratios. ESPO (Ou et al., 2026) introduces length-normalized sequence-level ratios and robust KL estimation, while BGPO (Lin et al., 2026b) constructs a memory-efficient lower-bound objective that permits gradient accumulation across Monte Carlo samples. Beyond one-sided ELBO surrogates, SPG (Wang et al., 2026a) maximizes a lower evidence bound for high-reward traces and minimizes an upper bound for low-reward traces to reduce policy-gradient bias. StableDRL (Zhong et al., 2026) targets the instability induced by noisy importance ratios through unconditional clipping and self-normalization.

Likelihood-free alternatives avoid constructing policy updates from approximate sequence likelihood ratios. DGA (Fan et al., 2026) reformulates dLLM RL as energy-based distribution matching without explicit sequence log-likelihoods. LFPO (Wei et al., 2026) directly optimizes denoising logits through contrastive updates derived from a discrete flow-matching view, while GDSD (Tang et al., 2026b) matches the denoiser to an advantage-guided self-teacher, reducing RL to likelihood-free distillation.

Another line of work exploits the diffusion trajectory itself. DCoLT (Huang et al., 2025) treats intermediate reverse-diffusion steps as latent thinking actions and optimizes the full trajectory; its LLaDA instantiation, LLaDOU, additionally learns a Plackett–Luce ranking-based unmasking policy. CJ-GRPO (Yang et al., 2025b) records intermediate rollout states and optimizes adjacent denoising transitions, avoiding one-step surrogates that mismatch the rollout path. To reduce the cost of trajectory-based optimization, TraceRL (Wang et al., 2026c) aggregates every s consecutive rollout steps into a shrunken trajectory for efficient trajectory-aware policy optimization. d2-StepMerge (Wang et al., 2026b) instead partitions the denoising trajectory into contiguous segments and approximates its likelihood with one model pass per segment, while AGRPO (Zhan, 2026) samples denoising timesteps to obtain an unbiased estimate of the full stepwise policy-gradient objective. In contrast, JustGRPO (Ni et al., 2026) fixes the rollout order to left-to-right so that standard GRPO can be applied, while retaining parallel diffusion decoding at inference.

Beyond single-turn tasks, several recent works study dLLMs in agentic settings. DLLM-Searcher (Zhao et al., 2026) combines agentic SFT with a preference optimization objective constructed from winner/loser trajectory pairs, and Zhen et al. (2026) empirically study dLLM backbones in multi-step agent workflows, identifying challenges such as tool-call failures. Very recently, LLaDA2.2 (Bie et al., 2026) applies reward-based agentic RL with environment feedback to multi-turn tool-use tasks, incorporating format-validity rewards and reporting efficiency on agent tasks.

Most of the methods above are developed and evaluated for single-turn reasoning or generation, and existing agentic efforts mainly focus on improving task accuracy. In contrast, we study on-policy optimization for multi-turn agentic tasks with an explicit focus on the accuracy–parallelism trade-off. We gate the per-turn rewards of structurally invalid actions before group-wise advantage

normalization, so that invalid turns are assigned negative advantages whenever the sampled group contains a successful trajectory. We further replace explicit KL penalties approximated by ELBO differences with implicit policy regularization through narrow asymmetric clipping and dual clipping, mitigating accumulated likelihood-estimation errors across multi-turn rollouts.

B Additional Experimental Results

In this section, we provide additional experimental results.

B.1 More Implementation Details

Datasets. We evaluate on three long-horizon agentic environments. *(i) WebShop* (Yao et al., 2022) is a simulated shopping environment where the agent navigates a website to purchase an item matching a natural-language goal through search, click, and option-selection actions, with at most 15 interaction turns per episode. The environment provides 500 training goals and 500 separate evaluation goals; an episode is successful only when its terminal task score equals 1, and we report the success rate. All methods are initialized from a task-specific SFT checkpoint; both the RL and self-distillation stages collect training data via on-policy rollouts in the environment. *(ii) Search-R1* (Jin et al., 2025) is a retrieval-augmented QA environment where the agent iteratively issues search queries, receives top-3 passages from an E5-base-v2 dense retriever over a 2018 Wikipedia corpus, and produces a final answer within at most 4 turns. The training prompts for both RL and self-distillation come from Natural Questions (NQ) (Kwiatkowski et al., 2019) and HotpotQA (Yang et al., 2018), with trajectories collected via on-policy rollouts; all methods start from the corresponding foundation model checkpoint without task-specific SFT. The evaluation set contains 51,713 questions from seven sources, including two seen during training (NQ and HotpotQA) and five unseen (TriviaQA, PopQA, 2WikiMultiHopQA, MuSiQue, and Bamboogle). We report the average exact-match accuracy over all questions. *(iii) ALFWorld* (Shridhar et al., 2021) is a text-based embodied environment designed to evaluate multi-step decision making. In each episode, the agent receives a natural-language household goal and must accomplish it through multi-turn interactions with the environment. The benchmark contains 3,827 task instances across six categories of household activities: Pick & Place, Examine in Light, Clean & Place, Heat & Place, Cool & Place, and Pick Two & Place. We allow at most 50 environment steps per episode, train on the standard training split, and evaluate on 128 episodes from the in-distribution evaluation split. An episode is successful only if the agent completes the specified task within the step limit, and we report the success rate. All dLLM RL methods and d3LLM-v2 start from the same task-specific SFT checkpoint, with trajectories collected via on-policy rollouts.

Models and contenders. We evaluate five foundation models: SDAR-8B (b=4), SDAR-8B (b=8), SDAR-4B (b=4), SDAR-4B (b=16) (Cheng et al., 2026), and d3LLM-LLaDA (b=16) (Qian et al., 2026), where b denotes the decoding block size. SDAR uses block-wise attention, whereas d3LLM-LLaDA retains fully bidirectional attention, allowing us to evaluate our framework across different attention patterns. The SDAR-4B (b=16) configuration is evaluated on Search-R1 and reported in Appendix B.3; the other four models are evaluated on both tasks in Table 1. We compare d3LLM-v2 with the following methods:

- *Qwen3-8B and Qwen3-4B* (Yang et al., 2025a) are size-matched autoregressive baselines with GRPO (Shao et al., 2024) training.
- *Vanilla dLLM (+prompt)* evaluates the corresponding SDAR (Cheng et al., 2026) or d3LLM-LLaDA (Qian et al., 2026) checkpoint with task-specific prompts but without agentic training, serving as the dLLM baseline before agentic post-training.
- *d1* (Zhao et al., 2025) is a token-level RL method that masks the entire response and randomly masks prompt tokens with probability p=0.15 to estimate per-token log-probabilities in a single forward pass. It constructs token-level importance ratios for policy optimization.
- *BGPO* (Lin et al., 2026b) constructs a memory-efficient lower-bound objective using a first-order Taylor approximation of the importance ratio for positive-advantage tokens, enabling gradient accumulation across Monte Carlo samples.
- *GDPO* (Rojas et al., 2026) estimates the sequence-level ELBO via Gauss–Legendre quadrature, replacing random timestep sampling with deterministic integration points to reduce variance. We use three quadrature points (gauss-3) for all GDPO experiments.

- *ESPO* (Ou et al., 2026) adopts a sequence-level ELBO framework with per-token normalization of importance ratios and robust KL estimation for stable large-scale training.

All dLLM RL methods use the same training budget within each task. On WebShop, all RL methods and d3LLM-v2 start from the same task-specific SFT checkpoint for each foundation model; on Search-R1, they start from the corresponding foundation model checkpoint. Within each configuration, all methods share the same rollout budget and checkpoint-selection rule.

Metrics. We evaluate using three metrics: *(i) Acc* measures task performance: on WebShop, it is the success rate; on Search-R1, it is the overall exact-match accuracy across all evaluation questions. *(ii) Tokens per forward* (TPF) measures decoding parallelism, defined as the average number of response tokens generated per forward pass over complete trajectories; an AR model has TPF=1. *(iii) AUP* (Accuracy Under Parallelism) (Qian et al., 2026) captures the accuracy–parallelism trade-off as a single score: by sweeping the decoding confidence threshold, we obtain a curve of (TPF, Acc) pairs, and AUP computes the weighted area under this curve, where a discount function penalizes accuracy degradation so that a method scores high only when it maintains accuracy while decoding more tokens in parallel. We follow the setting of Qian et al. (2026) with α=3. For the point-wise Acc and TPF reported in the main table, all methods are evaluated at the same LowConfidence decoding threshold of 0.95; AUP is computed separately by sweeping across multiple thresholds (we select threshold from 0.95 to 1.00). Higher is better for all three metrics.

B.2 Effectiveness of Reward Gating and Implicit Policy Regularization

Reward gating. Figure 3 further illustrates why validity-based reward gating is important for agentic training. The trajectory-level outcome reward indicates whether the model eventually produces a correct answer, but it is shared by all turns and therefore provides potentially inaccurate process supervision. Without reward gating, malformed intermediate actions in a successful trajectory can receive the same positive credit as valid actions. Consequently, the model can learn to answer more questions correctly while its action format progressively collapses: the success rate increases in both runs, whereas the valid-action ratio without reward gating falls from 0.90 to 0.70. In contrast, reward gating maintains a valid-action ratio of approximately 0.95 by preventing invalid turns from being reinforced, without sacrificing the success rate.



Figure 3: Effectiveness of reward gating on Search-R1 with d3LLM-LLaDA.

Implicit policy regularization. Figure 4 compares narrow asymmetric clipping without an explicit KL penalty against the standard clipping range, both without KL and with two explicit KL coefficients. With the narrow interval $(3\times10^{-4},\ 4\times10^{-4})$, the valid-action ratio remains close to one throughout 150 training steps. In contrast, all variants using the standard $(0.2,\ 0.2)$ clipping range become unstable late in training: the run without KL collapses after roughly 105 steps, while explicit KL coefficients of 10^{-3} and 10^{-2} delay or reduce the degradation but do not consistently prevent it; these curves end at their final logged steps. This result provides behavioral evidence that an explicit ELBO-based KL penalty does not reliably preserve policy validity in this agentic setting. Narrow clipping instead imposes a direct proximity constraint on every policy update and yields substantially more stable training, supporting our use of clipping-based implicit policy regularization.

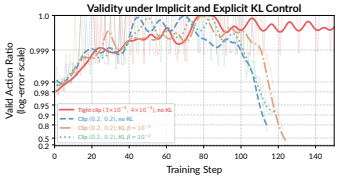


Figure 4: Effect of implicit and explicit policy regularization on WebShop with SDAR-8B. The log-error scale expands differences near one.

B.3 Search-R1 Results

Table 6 reports the additional Search-R1 tasks using SDAR-4B (b=16) as foundation model. Although the larger block allows more tokens to be decoded in parallel, existing dLLM RL methods achieve TPF values of only 1.405–1.493, indicating that a larger block alone does not ensure efficient decoding. In contrast, d3LLM-v2 reaches a TPF of 6.615 and an AUP of 254.1, while maintaining 39.29%

exact-match accuracy. Compared with ESPO, the strongest dLLM RL baseline in AUP, d3LLM-v2 achieves $4.5\times$ the TPF and $4.3\times$ the AUP, with accuracy 0.79 percentage points lower.

The comparison with SDAR-4B (b=4) further clarifies the effect of block size. With b=16, the TPF of d3LLM-v2 increases from 3.051 to 6.615. Thus, the larger block improves decoding parallelism by $2.17\times$ with a moderate accuracy cost, yielding a better overall accuracy–parallelism trade-off. These findings support the effectiveness of our framework in using the additional parallel decoding capacity offered by larger blocks. Notably, d1 collapses on the scenario, producing almost entirely invalid actions, which highlights the importance of training stability in long-horizon agentic tasks.

Table 6: Additional Search-R1 results for SDAR-4B (b=16). Metrics follow Table 1; higher is better.

	SDAR-4B (b=16)		
Method	ACC↑	TPF↑	AUP↑
Qwen3 (+GRPO)	41.82	1.000	41.82
SDAR (+prompt)	15.34	1.219	18.80
d1	–	–	–
BGPO	37.36	1.405	53.17
GDPO	35.14	1.493	54.47
ESPO	40.08	1.470	59.07
d3LLM-v2 (ours)	**39.29**	**6.615**	**254.1**

B.4 ALFWORLD RESULTS

Table 7 reports our results on ALFWorld (Shridhar et al., 2021) using SDAR-8B (b=4) as the foundation model. ALFWorld is a text-based embodied environment that requires agents to complete long-horizon household tasks through multi-turn interactions. The AUP is computed from all available pairs. The autoregressive Qwen3 (+GRPO) baseline achieves the highest success rate of 77.34% but is restricted to a TPF of 1.000. Among the dLLM RL baselines, GDPO obtains the highest success rate of 71.88% and the highest AUP of 118.9. In contrast, d3LLM-v2 reaches a TPF of 2.420 and an AUP of 164.3 while maintaining a competitive success rate of 69.53%, improving TPF by $1.46\times$ and AUP by 38.2% over the best existing dLLM RL baseline (GDPO).

Table 7: Results on ALFWorld with SDAR-8B (b=4). Metrics follow Table 1; higher is better.

Method	ACC↑	TPF↑	AUP↑
Qwen3 (+GRPO)	**77.34**	1.000	77.34
Vanilla dLLM (+prompt)	9.380	1.192	11.17
d1	57.03	1.203	62.28
BGPO	64.06	1.416	91.18
GDPO	71.88	1.655	118.9
ESPO	68.75	1.515	94.74
d3LLM-v2 (ours)	69.53	**2.420**	**164.3**

B.5 EXAMPLES OF INVALID ACTIONS IN WEBSHOP AND SEARCH-R1

As discussed in Section 3.1, dLLMs are more likely to produce invalid actions due to parallel decoding and the lack of causal conditioning. Below, we present representative examples of such failures in the WebShop and Search-R1 environments, with the invalid portion highlighted in red.

(i) WebShop Type I: Reasoning loop truncated without an action. The model enters a reasoning loop that exhausts the generation budget, so no action is emitted (Table 8).

Table 8: Type I invalid action: the reasoning span is truncated and no action is emitted.

User Prompt:
<|im_start|>user
You are an expert autonomous agent operating in the WebShop e-commerce environment. Your task is to: Find me machine wash, wash cold women's fashion hoodies & sweatshirts for dry clean, tumble dry with color: cotton candy, and size: medium, and price lower than 70.00 dollars.

Prior to this step, you have already taken 1 step(s). Below are the most recent 1 observations and the corresponding actions you took: [Observation 1: "Search", Action 1: 'search[women's fashion hoodies and sweatshirts color cotton candy medium under $70 dollars]']

You are now at step 2 and your current observation is: 'Back to Search' [SEP] 'Page 1 (Total results: 50)' [SEP] 'Next >' [SEP] 'B09S3BN15C' [SEP] 'Mens Linen Shirt, Men's Striped Shirts Casual Short Sleeve Button Down Shirts Regular Fit Hawaiian Shirts Beach Tees Tops' [SEP] '$3.78 to $11.38' [SEP] ... [SEP] 'B072PCHZC3' [SEP] 'Thin Red Line USA Flag Firefighter Men's Hoodie Sweatshirt' [SEP] '$44.95 to $47.95'.

Your admissible actions of the current situation are:
[
'click[back to search]',
'click[next >]',

. . .
’click[b072pchzc3]’,
].
Now it’s your turn to take one action for the current step. You should first reason step-by-step about the current situation, then think carefully which admissible action best advances the shopping goal. This reasoning process MUST be enclosed within <reason> </reason> tags. Once you’ve finished your reasoning, you should choose an admissible action for current step and present it within <action> </action> tags.
<|im_end|>
<|im_start|>assistant

Model Output:
<reason>
Okay, let’s break this down. The user is looking for machine wash, wash cold women’s fashion hoodies or sweatshirts that are dry clean, tumble dry, color cotton candy, size medium, and under $70. The first step was a search, and now the results are showing. The current observation has a list of products with their details.

Looking at the products, none of the titles mention “hoodies” or “sweshirts” specifically. The first item is a men’s shirt, which doesn’t fit. The next ones are Disney’s Winnie The Pooh, which is a tank top, not a hoodie. . . . The Expression Tees Pug Life is a hoodie, but the price range is $36.99 to $48.99, which
[Generation ends here: </reason> and <action>. . . </action> are missing.]

(ii) WebShop Type II: Malformed closing action tag. The model emits a corrupted closing action tag, preventing the environment from extracting the intended action (Table 9).

Table 9: Type II invalid action: the closing action tag is malformed.

User Prompt:
<|im_start|>user
You are an expert autonomous agent operating in the WebShop e-commerce environment. Your task is to: Find me machine wash women’s pants with elastic waist with color: orbit green, and size: x-small, and price lower than 70.00 dollars.

Prior to this step, you have already taken 2 step(s). Below are the most recent 2 observations and the corresponding actions you took: [Observation 1: ”Search”, Action 1: ’search[women’s pants elastic waist orbit green x-small under $70]’] [Observation 2: ”Back to Search’ [SEP] ’Page 1 (Total results: 50)’ [SEP] ’Next >’ [SEP] ’B09QCVCYVY’ [SEP] ’Burband Womens High Waist Bike Shorts Tummy Control Workout Yoga Pants Running Exercise Running Shorts with Side Pockets’ [SEP] ’$6.82’ [SEP] . . . , Action 2: ’click[b09qcvcyvy]’]

You are now at step 3 and your current observation is: ’Back to Search’ [SEP] ’< Prev’ [SEP] ’color’ [SEP] ’black’ [SEP] ’blue’ [SEP] . . . [SEP] ’size’ [SEP] ’small’ [SEP] ’medium’ [SEP] . . . [SEP] ’Burband Womens High Waist Bike Shorts Tummy Control Workout Yoga Pants Running Exercise Running Shorts with Side Pockets’ [SEP] ’Price: $6.82’ [SEP] ’Rating: N.A.’ [SEP] ’Description’ [SEP] ’Features’ [SEP] ’Reviews’ [SEP] ’Buy Now’.

Your admissible actions of the current situation are:
[
’click[back to search]’,
’click[< prev]’,
’click[description]’,
. . .
’click[xx-large]’,
].
Now it’s your turn to take one action for the current step. You should first reason step-by-step about the current situation, then think carefully which admissible action best advances the shopping goal. This reasoning process MUST be enclosed within <reason> </reason> tags. Once you’ve finished your reasoning, you should choose an admissible action for current step and present it

within <action> </action> tags.
<|im_end|>
<|im_start|>assistant

Model Output:
<reason>
Okay, let's break this down. The user is looking for machine wash women's pants with an elastic waist, specifically color orbit green, size x-small, under $70.

Looking at the current observation, the user is on a product page for the Burband Womens High Waist Bike Shorts. The admissible actions include clicking on color options like black, blue, etc., or size options like small, medium, etc. The current page shows the product details, including the color and size options.

...

Since the user hasn't found the color yet, the next step is to check the color options. So the best action should be to click on the color options to see if orbit green is listed. If it is, then the user can select it and check the size. If not, they might need to refine the search.
</reason>
<action>click[black]**</actionaction>**

(iii) WebShop Type III: Malformed closing reasoning tag. The model produces a malformed closing reasoning tag, breaking the boundary between reasoning and action spans (Table 10).

Table 10: Type III invalid action: the closing reasoning tag is malformed.

User Prompt:
<|im_start|>user
You are an expert autonomous agent operating in the WebShop e-commerce environment. Your task is to: Find me men's fashion sneakers with lace closure, rubber outsole, rubber sole with color: white | black, and size: 9.5, and price lower than 60.00 dollars.

Prior to this step, you have already taken 10 step(s). Below are the most recent 2 observations and the corresponding actions you took: [Observation 9: "Back to Search' [SEP] '< Prev' [SEP] 'size' [SEP] '6.5' [SEP] ... [SEP] 'color' [SEP] 'black' [SEP] ... [SEP] 'Buy Now", Action 9: 'click[9.5]'] [Observation 10: "Back to Search' [SEP] '< Prev' [SEP] 'size' [SEP] '6.5' [SEP] ... [SEP] 'color' [SEP] 'black' [SEP] ... [SEP] 'Buy Now", Action 10: 'click[white]']

You are now at step 11 and your current observation is: 'Back to Search' [SEP] '< Prev' [SEP] 'size' [SEP] '6.5' [SEP] '7' [SEP] ... [SEP] '9.5' [SEP] ... [SEP] 'color' [SEP] 'black' [SEP] ... [SEP] 'white' [SEP] ... [SEP] 'MCICI Mens Loafers Moccasin Driving Shoes Premium Genuine Leather Casual Slip On Flats Fashion Slipper Breathable Big Size' [SEP] 'Price: $15.99 to $32.99' [SEP] 'Rating: N.A.' [SEP] 'Description' [SEP] 'Features' [SEP] 'Reviews' [SEP] 'Buy Now'.

Your admissible actions of the current situation are:
[
'click[back to search]',
'click[< prev]',
'click[description]',
...
'click[yellow-a]',
].
Now it's your turn to take one action for the current step. You should first reason step-by-step about the current situation, then think carefully which admissible action best advances the shopping goal. This reasoning process MUST be enclosed within <reason> </reason> tags. Once you've finished your reasoning, you should choose an admissible action for current step and present it within <action> </action> tags.
<|im_end|>
<|im_start|>assistant

Model Output:
<reason>
Okay, let's break this down. The user is looking for men's fashion sneakers with a lace closure, rubber outsole, rubber sole, color white or black, size 9.5, and under $60.

Looking at the current observation, the user has already selected the size 9.5 and the color white. The next step would be to check if the product meets all the other criteria. The product listed here is the MCICI Mens Loafers. The description mentions it's a slipper, not a sneaker.

...

The admissible actions include clicking on the product details or features. The next logical step is to check the product's features to see if it's a sneaker. If it's not, then the user should go back and refine the search. ... However, the product's features don't mention a rubber outsole or lace closure. So, the best action is to click on the product to view details and then determine if it meets the criteria. If not, the user should adjust the search parameters.
reason>
<action>click[buy now]</action>

(iv) Search-R1 Type I: Both search and answer emitted. The model emits both a search query and a final answer, violating the mutual exclusivity constraint (Table 11).

Table 11: Invalid action: the model emits both a search query and a final answer.

User Prompt:
<|im_start|>user
You are an expert agent tasked with answering the given question step-by-step. Your question: who kills barry's mom in the flash?

Prior to this step, you have already taken 1 step(s). Below is the interaction history where <search> </search> wrapped your past search queries and <information> </information> wrapped the corresponding search results returned by the external search engine. History: Step 1:<search>The fate of Barry Allen's mother in The Flash</search> <information>{"result": "Doc 1: Flashpoint (comics). Barry Allen wakes up to discover everything and everyone around him has changed. His mother Nora (deceased in his own timeline) is alive. ... Doc 3: Eobard Thawne. Thawne reveals that he's responsible for every tragedy that has occurred in Barry's life, including the death of Nora Allen (Barry's mother)."}</information>

Now it's your turn to respond for the current step. You should first conduct reasoning process. This process MUST be enclosed within <reason> </reason> tags. After completing your reasoning, choose only one of the following actions (do not perform both): (1) If you find you lack some knowledge, you can call a search engine to get more external information using format: <search> your query </search>. (2) If you have enough knowledge to answer the question confidently, provide your final answer within <answer> </answer> tags, without detailed illustrations. For example, <answer>Beijing</answer>.
<|im_end|>
<|im_start|>assistant

Model Output:
<reason>The question asks for the identity of the villain who kills Barry Allen's mother in The Flash. Fortunately, information regarding this question is available in the search results provided.
</reason>
<search>The identity of the villain who killed Barry Allen's mother in The Flash</search>
<answer>Barry Allen's mother Nora is killed by Eobard Thawne in the storyline of The Flash.
</answer>

(v) Search-R1 Type II: Unclosed answer tag. The model opens an answer span but generation ends before the closing tag is produced (Table 12).

Table 12: Invalid action: the answer span is opened but never closed.

User Prompt:
<|im_start|>user
You are an expert agent tasked with answering the given question step-by-step. Your question: when does i'm a celebrity get me out of here start in the uk?

Now it's your turn to respond for the current step. You should first conduct reasoning process. This process MUST be enclosed within <reason> </reason> tags. After completing your reasoning, choose only one of the following actions (do not perform both): (1) If you find you lack some knowledge, you can call a search engine to get more external information using format: <search> your query </search>. (2) If you have enough knowledge to answer the question confidently, provide your final answer within <answer> </answer> tags, without detailed illustrations. For example, <answer>Beijing</answer>.
<|im_end|>
<|im_start|>assistant

Model Output:
<reason>
To determine when "I'm a Celebrity... Get Me Out of Here!" starts in the UK, we need to check the latest episode schedule or release date. UK viewers can typically access this information on the official website of the television network or through broadcast listings provided by the broadcasting company. </reason>
<answer>
I'm a Celebrity... Get Me Out of Here! starts in the UK on [insert date here].
[Generation ends here: </answer> is missing.]

(vi) Search-R1 Type III: Reasoning only without an action. The model completes its reasoning but emits neither a search query nor an answer (Table 13).

Table 13: Invalid action: the model closes its reasoning but emits neither a search query nor an answer.

User Prompt:
<|im_start|>user
You are an expert agent tasked with answering the given question step-by-step. Your question: when does the little couples new season start?

Prior to this step, you have already taken 1 step(s). Below is the interaction history where <search> </search> wrapped your past search queries and <information> </information> wrapped the corresponding search results returned by the external search engine. History: Step 1:<search>What is the release date of the new season of "Little Couples"?</search> <information>{"result": "Doc 1: Together With Me. A 2017 Thai BL drama aired on LINE TV. ... Doc 2: Younger (season 5). The fifth season premiered on June 5, 2018. ... Doc 3: Alone Together (TV series). An American comedy series that premiered on January 10, 2018 on Freeform."}</information>

Now it's your turn to respond for the current step. You should first conduct reasoning process. This process MUST be enclosed within <reason> </reason> tags. After completing your reasoning, choose only one of the following actions (do not perform both): (1) If you find you lack some knowledge, you can call a search engine to get more external information using format: <search> your query </search>. (2) If you have enough knowledge to answer the question confidently, provide your final answer within <answer> </answer> tags, without detailed illustrations. For example, <answer>Beijing</answer>.
<|im_end|>
<|im_start|>assistant

Model Output:
<reason>It appears there was a misunderstanding in the initial query. "Little Couples" does not match any known television series, movies, or any other forms of media. It's possible that the query might be referring to another title or there may be a typographical error in the name. If there is a specific series or show in mind, providing more information or the correct name would be helpful.</reason>
[Generation ends here: <search>...</search> and <answer>...</answer> are missing.]

(vii) Search-R1 Type IV: Reasoning loop truncated without an action. The model re-opens a reasoning tag instead of closing it (Table 14).

Table 14: Invalid action: the reasoning span is reopened and no search or answer is emitted.

User Prompt:
<|im_start|>user
You are an expert agent tasked with answering the given question step-by-step. Your question: how many episodes of touching evil are there?

Now it's your turn to respond for the current step. You should first conduct reasoning process. This process MUST be enclosed within <reason> </reason> tags. After completing your reasoning, choose only one of the following actions (do not perform both): (1) If you find you lack some knowledge, you can call a search engine to get more external information using format: <search> your query </search>. (2) If you have enough knowledge to answer the question confidently, provide your final answer within <answer> </answer> tags, without detailed illustrations. For example, <answer>Beijing</answer>.
<|im_end|>
<|im_start|>assistant

Model Output:
<reason>
To determine the number of episodes of "Touching Evil," we would need to refer to an official source or database published by the channel that airs the show or by the production company. However, as of my last update, I don't have specific information on the number of episodes of "Touching Evil". The show might have been discontinued or it might have been canceled at some point.
<reason>
[Generation ends here: </reason> and <search>...</search> or <answer>...</answer> are missing.]