DECODING TRAJECTORY FORGETTING: CONTINUAL REINFORCEMENT LEARNING FOR DIFFUSION LARGE LANGUAGE MODELS VIA TRAJECTORY ALIGNMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion large language models (dLLMs) introduce a parallel decoding paradigm that resolves multiple masked positions with a flexible order. Recent developments in post-training for dLLMs have substantially improved dLLM reasoning capabilities in single-domain training and inference scenarios with reinforcement learning (RL). Yet, continually adapting dLLMs to evolving real-world tasks remains challenging, as previously learnt capabilities can rapidly degrade after model post-training on new domains. To investigate the reason behind catastrophic forgetting in dLLMs, we examine how previously learnt decoding trajectories evolve during continual RL from two complementary dimensions: (1) token prediction distributions that characterise what to decode, where continual RL can alter prediction behaviour on previous domains, and (2) decoding order patterns that characterise which positions to decode, where continual RL can distort the relative decoding order between token positions. Beyond changes in token predictions, we find that forgotten tasks exhibit more pronounced order drift in the decoding trajectory than preserved ones, revealing decoding trajectory forgetting as an important dimension of catastrophic forgetting in dLLM continual RL. Motivated by these insights, we introduce **CLD**, a novel continual RL framework for dLLMs that enables continual adaptation while preserving previously learnt decoding trajectories. During continual RL, the native RL objective drives adaptation to the incoming domain, while decoding trajectory alignment provides stability by preserving previously learnt decoding behaviour. Moreover, CLD can be easily integrated with different dLLM RL algorithms in a plug-and-play style. Extensive experiments on 9 benchmarks demonstrate that CLD effectively mitigates catastrophic forgetting in dLLMs. Our code is provided at https://anonymous.4open.science/r/CLD-2026.

1 INTRODUCTION

Diffusion large language models (dLLMs) introduce an alternative parallel decoding paradigm, where multiple masked positions can be decoded in flexible orders (Arriola et al., 2025; Wu et al., 2026; Jiang et al., 2026a;b). Recent advances in reinforcement learning (RL) with verifiable rewards on outcome tokens have substantially improved dLLM reasoning capabilities in single-domain scenarios without explicitly controlling the decoding trajectory (Cheng et al., 2025; Zhao et al., 2025; Tang et al., 2026; Rojas et al., 2026; Zhong et al., 2026; Ni et al., 2026; He et al., 2025; Wang et al., 2026b;c). As dLLMs are increasingly adapted to evolving tasks and domains, continual learning is becoming important for acquiring new capabilities without forgetting previous ones. **To examine whether existing dLLM RL methods can effectively support continual scenarios**, we

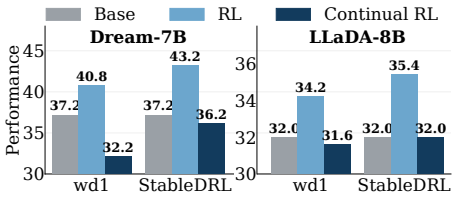
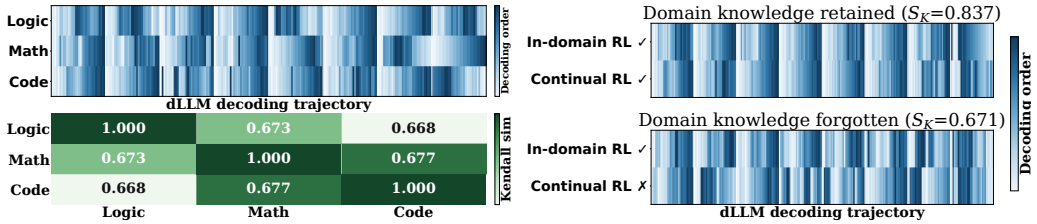


Figure 1: Catastrophic forgetting in continual RL for dLLMs with wd1 and StableDRL on Dream-7B and LLaDA-8B. After post-training on math data **(blue)**, model performance on math is improved over base models **(gray)**. Yet further continual RL on coding data **(navy)** harms the model performance on math.

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.



(a) Decoding order patterns across domains. **(b)** Decoding order pattern retention/forgetting.

Figure 2: **Decoding order patterns in dLLMs.** Heatmaps **(navy)** show decoding order patterns for 8-block generation on LLaDA-8B. A lighter colour denotes an earlier decoded order within a block. **(a) Top**: Different domains exhibit distinct decoding order patterns. **Bottom**: Heatmap (green) shows pairwise Kendall similarity score S_K (defined in Section 3) among domains. **(b) Top**: dLLM achieves a correct answer (✓) for a logic reasoning task after both in-domain RL and continual RL with similar decoding order patterns (S_K=0.837). **Bottom**: dLLM has a correct answer for another logic reasoning task after in-domain RL, but a wrong answer (×) after continual RL on the same code domain. A largely different decoding order pattern happens with reduced order similarity (S_K=0.671). More visualisation details are provided in Appendix F.1 and F.2.

conduct a preliminary study in Figure 1 using two recent dLLM RL methods, wd1 (Tang et al., 2026) and StableDRL (Zhong et al., 2026), together with two representative dLLM backbones, Dream (Ye et al., 2025) and LLaDA (Nie et al., 2025). Performance on mathematical reasoning tasks is compared among (1) base dLLMs (**gray**); (2) RL-trained with MATH training set (**blue**); and (3) continually RL-trained on a coding dataset, KodCode (**navy**). Direct math domain RL can improve model in-domain performance, while **straightforward continual RL training on the coding data decreases the model's capability** on math domain, indicating clear catastrophic forgetting.

To investigate the mechanism behind dLLM forgetting, we further examine how their underlying decoding trajectories evolve during continual RL. Existing analyses of continual learning mainly characterise forgetting through performance degradation or changes in model outputs. dLLMs, however, introduce an additional dimension of decoding behaviour: the order in which token positions are progressively decoded during denoising. We characterise this decoding order through the relative order between token positions and adopt Kendall similarity (Kendall, 1938) in Figure 2 to quantify pairwise order agreement. Darker colours in the trajectory heatmaps indicate positions decoded later, while lighter colours indicate positions decoded earlier. Our analysis reveals two observations. **(a) Different domains exhibit distinct decoding order patterns.** More importantly, **(b) continual learning can disrupt previously learnt decoding orders in dLLMs, with forgotten tasks exhibiting substantially larger order drift than retained ones.** Together with the changes in token prediction distributions, this reveals a dLLM-specific form of catastrophic forgetting that we term **decoding trajectory forgetting**: continual RL alters not only **what** tokens the model predicts, but also **how** their decoding unfolds over the trajectory. This naturally raises the question: **Can preserving previous decoding trajectories enable effective continual RL for dLLMs?**

In light of these observations, we introduce **CLD**, a plug-and-play framework for continual RL of dLLMs. CLD characterises dLLM decoding trajectories from two complementary aspects: token prediction distributions, which capture what tokens are decoded, and decoding order patterns, which capture which positions are decoded. For a previous domain, CLD stores a compact trajectory memory consisting of selected intermediate diffusion states. During subsequent continual RL stages, the native RL objective learns the incoming domain, while trajectory alignment preserves previously learnt decoding behaviours on previous domains. Specifically, a token-level KL objective aligns token prediction distributions, while a Kendall-inspired pairwise ranking objective preserves decoding order patterns. Together, these objectives mitigate trajectory drift while still allowing the model to acquire new decoding behaviours. As a plug-and-play framework, CLD can be directly integrated with existing dLLM RL algorithms and different RL objectives. Our main contributions are:

- We uncover **decoding trajectory forgetting** in dLLMs. Beyond degradation in task performance and token predictions, continual RL can distort previously learnt decoding order patterns.
- We reveal decoding order as a distinctive dimension of forgetting in dLLMs: domain tasks exhibit different order patterns and forgotten tasks show larger decoding order drift than preserved ones.
- We propose **CLD**, a plug-and-play method for effective continual RL in dLLMs that enables existing RL algorithms to continually acquire new-domain capabilities while preserving decoding trajectories through token distribution and decoding order alignment.

2 PRELIMINARIES

2.1 DIFFUSION LARGE LANGUAGE MODELS

For dLLMs, given an input prompt x, let θ denote the model parameters and $\mathbf{s}_t = (s_t^1, \ldots, s_t^L)$ denote the partially denoised response state of length L at decoding step t. The masked positions are denoted by $\mathcal{M}_t = \{i \in \{1, \ldots, L\} : s_t^i = \texttt{mask}\}$. For each $i \in \mathcal{M}_t$, the model predicts a token distribution $p_\theta(\cdot \mid x, \mathbf{s}_t)_i$ over the vocabulary $\mathcal{V}$. Most existing dLLMs adopt block-based parallel decoding (Arriola et al., 2025; Jiang et al., 2026b), where the response is divided into B sequential blocks and positions within each block are denoised in parallel with flexible decoding orders. Let $\mathcal{B}_b \subseteq \{1, \ldots, L\}$ denote the token positions in block b, and let r_i denote the decoding step at which position i is first committed from `mask` to a token. The decoding order is defined as

$$\pi_b = \mathrm{argsort}_{i \in \mathcal{B}_b}(r_i, i), \qquad \Pi_\theta(x) = (\pi_1, \ldots, \pi_B), \tag{1}$$

where π_b orders the positions in block b by their decoding steps, with ties deterministically broken by position indices, and $\Pi_\theta(x)$ collects the decoding orders of all B blocks. Therefore, a dLLM decoding trajectory is characterised by two dimensions: token prediction distributions, which describe what is decoded, and decoding order patterns, which describe which positions are decoded earlier.

2.2 CONTINUAL REINFORCEMENT LEARNING FOR DIFFUSION LARGE LANGUAGE MODELS

Recent dLLM post-training methods commonly adopt GRPO-style RL to optimise dLLM reasoning capabilities according to outcome rewards (Zhao et al., 2025; Tang et al., 2026; Rojas et al., 2026; Ou et al., 2026; Wang et al., 2026a; Zhong et al., 2026; Wang et al., 2026b). We use $\mathcal{L}_{\mathrm{RL}}(\theta, \mathcal{D})$ to denote the native RL objective applied to domain $\mathcal{D}$, with further details provided in Appendix C. Beyond single-domain training scenarios, continual RL aims to continuously adapt a model to evolving domains while preserving capabilities learnt from previous domains $\{\mathcal{D}_j\}_{j<k}$, without access to future domains. We formulate continual RL for dLLMs over a sequence of T domain distributions $\{\mathcal{D}_k\}_{k=1}^T$ arriving sequentially, where T denotes the total number of domains. At stage k, let θ denote the trainable model parameters at stage k. The model is initialised from the previous stage checkpoint and optimised on the current domain $\mathcal{D}_k$ with the native dLLM RL objective:

$$\mathcal{L}_{\mathrm{RL}}^{(k)}(\theta) = \mathcal{L}_{\mathrm{RL}}(\theta, \mathcal{D}_k). \tag{2}$$

3 DECODING TRAJECTORY FORGETTING IN DLLM CONTINUAL RL

3.1 DECOMPOSING DECODING TRAJECTORY FORGETTING IN DLLMS

We quantify decoding trajectory forgetting from token predictions and decoding order discrepancy:

Decoding token discrepancy. For the same task, we use diffusion states from the previously learnt decoding trajectory as reference states and evaluate both model on these states. Let $p_{t,i}$ and $p'_{t,i}$ denote token distributions of the previous and later model at masked position $i \in \mathcal{M}_t$. We measure token discrepancy by averaging the Kullback Leibler (KL) divergence (Kullback & Leibler, 1951):

$$D_{\mathrm{tok}} = \mathbb{E}_t \left[\frac{1}{|\mathcal{M}_t|} \sum_{i \in \mathcal{M}_t} D_{\mathrm{KL}} \left(p_{t,i} \| p'_{t,i} \right) \right], \tag{3}$$

where the expectation is taken over the reference diffusion states from the stored decoding trajectory. A smaller D_{tok} indicates stronger agreement in token prediction behaviour.

Decoding order discrepancy. To measure agreement in **which positions** are decoded earlier, we adopt normalised Kendall similarity (Kendall, 1938). Consider two decoding order sequences π and π' over the same n token positions and let $N_{\mathrm{dis}}(\pi, \pi')$ denote the number of position pairs whose relative order is reversed, the normalised Kendall similarity is defined as:

$$S_K(\pi, \pi') = 1 - \frac{N_{\mathrm{dis}}(\pi, \pi')}{\binom{n}{2}}. \tag{4}$$

Under the deterministic tie-breaking rule in Eq. 1, each block induces a strict order. S_K ranges from 0 to 1, where a larger value indicates stronger decoding order agreement. For dLLM block decoding,

the decoding order similarity is averaged across blocks. The decoding order discrepancy is defined:

$$D_{\text{ord}} = 1 - S_K(\Pi, \Pi'), \quad \text{where } S_K(\Pi, \Pi') = \frac{1}{B}\sum_{b=1}^{B} S_K(\pi_b, \pi'_b). \tag{5}$$

A smaller D_{ord} indicates stronger agreement between the decoding order patterns. Building on the two components, we quantify decoding trajectory forgetting via a unified trajectory discrepancy:

$$D_{\text{traj}} = \alpha D_{\text{tok}} + (1-\alpha) D_{\text{ord}}, \tag{6}$$

where $\alpha \in [0, 1]$ is a weighting coefficient between token prediction drift and decoding order drift. A smaller D_{traj} indicates stronger consistency between two decoding trajectories. During continual RL, we define **decoding trajectory forgetting** as increased trajectory discrepancy between a previously learnt decoding trajectory and its counterpart after continual learning on subsequent domains.

3.2 Decoding Trajectory Forgetting and Catastrophic Forgetting in dLLMs

To examine whether decoding trajectory forgetting accompanies catastrophic forgetting, we conduct a correlation analysis with Vanilla Continual RL on Dream-7B and LLaDA-8B. Each model is first optimised with RL on Math and then continually trained on Code. We use backward transfer (BWT) to denote performance change, $\text{BWT} = P_{\text{Math}} - P_{\text{Math,Math}}$, where $P_{\text{Math,Math}}$ is the Math performance immediately after Math RL. D_{traj} measures decoding trajectory discrepancy with a default $\alpha = 0.5$, and S_K measures decoding order similarity relative to the model after Math RL.

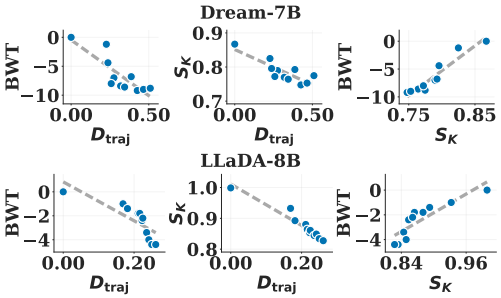


Figure 3: **Decoding trajectory forgetting is associated with catastrophic forgetting in dLLMs.** Vanilla Continual RL is shown from Math to Code on Dream-7B and LLaDA-8B. A smaller BWT denotes a larger forgetting.

As shown in Figure 3, consistent trends are observed across both dLLM backbones. **(i) (left) Larger decoding trajectory forgetting is associated with more severe catastrophic forgetting**, where increasing D_{traj} corresponds to lower BWT. (ii) (middle) Larger D_{traj} generally corresponds to a lower S_K, although the token distribution discrepancy still persists. **(iii) (right) Higher decoding order similarity is associated with higher BWT with a better capability retention**. These observations suggest that decoding trajectory forgetting is a consistent indicator of catastrophic forgetting across different dLLM backbones.

4 Continual RL for dLLMs with Decoding Trajectory Alignment

CLD aims to preserve token predictions and decoding order patterns while remaining compatible with most existing dLLM RL methods. The pipeline is in Figure 4 and Algorithm 1.

4.1 Decoding Trajectory Memory Construction

To preserve dLLM decoding behaviour, CLD constructs a compact memory of decoding trajectories learnt from previous domains. Specifically, after RL on a newly arriving domain $\mathcal{D}_k$, CLD filters out zero-reward rollouts and preserves reward-positive trajectories for alignment in subsequent continual RL stages. Detailed memory filtering and reward configurations are provided in Appendix E.4. Yet, preserving complete decoding trajectories would require storing full vocabulary distributions, resulting in substantial memory overhead. Therefore, for each selected state, CLD preserves the top-K token predictions at each position together with the residual probability mass and the decoding order. Let the overbar denote quantities stored from the decoding trajectories in previous domains. A stored trajectory record m is represented by the tuple

$$m = (x, \bar{\mathbf{s}}_t, \mathcal{M}_t, \{\bar{q}_{t,i}\}_{i \in \mathcal{M}_t}, \bar{\pi}) \,. \tag{7}$$

Here, m is a structured record rather than a numerical vector, x is the prompt, $\bar{\mathbf{s}}_t$ is a stored response state of length L, and $\mathcal{M}_t \subseteq \{1, \ldots, L\}$ contains the masked positions for dLLMs to predict. For each $i \in \mathcal{M}_t$, $\bar{q}_{t,i}$ denotes the compact token distribution consisting of the top-K predicted tokens, their probabilities, and the residual probability mass over the remaining vocabulary tokens.

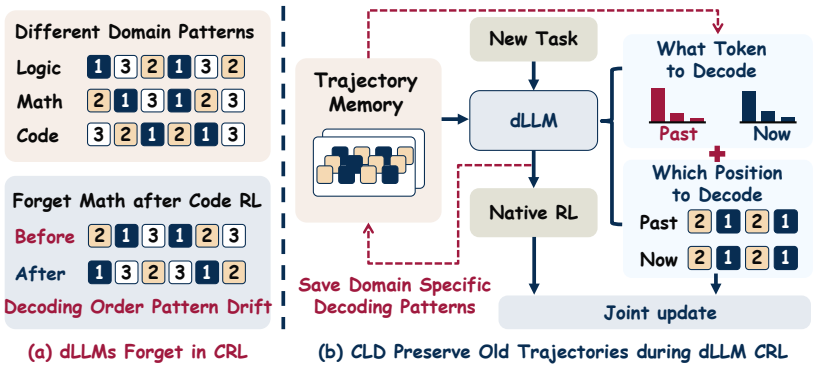


Figure 4: **Overview of CLD. (a)** Different domains exhibit distinct decoding patterns, while learning new domains can distort previous decoding trajectories. **(b)** CLD preserves both token prediction and decoding order patterns to mitigate decoding trajectory forgetting in dLLM continual RL.

$\bar{\pi}$ denotes the stored previous decoding order. Let $\mathcal{H}_j$ denote the set of trajectory records stored after learning domain $\mathcal{D}_j$. The complete memory before continual RL stage k can then be defined as $\mathcal{H}_{<k} = \bigcup_{j<k} \mathcal{H}_j$. CLD therefore preserves both token decoding behaviour through $\{\bar{q}_{t,i}\}$ and decoding order behaviour through $\bar{\pi}$, without storing separate checkpoints for each previous domain.

4.2 Decoding Trajectory Alignment in dLLMs

During continual RL at stage k, optimising dLLMs on the incoming domain $\mathcal{D}_k$ can induce decoding trajectory forgetting on previous domains by distorting previously learnt token prediction distributions and decoding order patterns. CLD therefore preserves the same two trajectory dimensions identified in our analysis: token prediction and decoding order. Token prediction can be directly aligned through distributional divergence, whereas realised decoding order is discrete and cannot provide gradients through Kendall similarity. CLD therefore constructs a differentiable pairwise ranking surrogate for the decoding order discrepancy and combines it with the token discrepancy into a unified trajectory alignment objective from two aspects:

(i) What to decode: Token prediction alignment. Aligning token distributions allows the current dLLM to recover prediction behaviour learnt from previous domains. For a stored decoding state $\bar{\mathbf{s}}_t$, let $\bar{q}_{t,i}$ denote the compact token distribution stored at masked position $i \in \mathcal{M}_t$, as defined in Section 4.1. The current distribution $q^{\theta}_{t,i}$ is constructed over the same top-K tokens with the residual probability mass. For a trajectory record m, the token discrepancy is instantiated as:

$$D_{\mathrm{tok}}(\theta; m) = \frac{1}{|\mathcal{M}_t|} \sum\nolimits_{i \in \mathcal{M}_t} D_{\mathrm{KL}}\left(\bar{q}_{t,i} \| q^{\theta}_{t,i}\right). \tag{8}$$

Minimising D_{tok} encourages the current model to preserve what was predicted at the stored diffusion states, while a larger discrepancy indicates stronger deviation from the token behaviour.

(ii) Which positions to decode: Decoding order alignment. Aligning decoding orders allows the currently trained dLLM to preserve decoding order patterns learnt from previous domains. The decoding order discrepancy D_{ord} in Section 3.1 measures whether previously learnt pairwise order is preserved, yet cannot be directly optimised since the realised decoding order is discrete. CLD thus constructs a differentiable ranking surrogate based on the same pairwise order structure. Let $\mathrm{rank}_{\bar{\pi}}(i)$ denote the previous decoding rank of position i, and define the previous decoding order pairs as $\bar{\mathcal{P}} = \{(i,j) : \mathrm{rank}_{\bar{\pi}}(i) < \mathrm{rank}_{\bar{\pi}}(j)\}$. For each masked position $i \in \mathcal{M}_t$, the model outputs the confidence score $c^{\theta}_{t,i}$. For each previous position pair $(i,j) \in \bar{\mathcal{P}}$, preserving the order relation requires the current trained dLLMs to assign a higher decoding confidence to i than to j, that is, $c^{\theta}_{t,i} > c^{\theta}_{t,j}$. Therefore, the differentiable decoding order discrepancy is formulated as:

$$\widetilde{D}_{\mathrm{ord}}(\theta; m) = \frac{1}{|\bar{\mathcal{P}}|} \sum\nolimits_{(i,j) \in \bar{\mathcal{P}}} \log\left(1 + \exp\left(-(c^{\theta}_{t,i} - c^{\theta}_{t,j})/\gamma\right)\right), \tag{9}$$

where $\gamma > 0$ is the ranking temperature. This surrogate replaces the discrete comparison of realised decoding orders with continuous confidence differences while retaining the same pairwise order structure used by Kendall similarity. Intuitively, minimising $\widetilde{D}_{\mathrm{ord}}$ preserves the relative decoding order between positions that determines which positions to decode first.

Unified decoding trajectory alignment. The token and decoding order discrepancies characterise two complementary aspects of the decoding trajectory in dLLMs. Thus, to preserve dLLM decoding behaviour, CLD combines them into a unified differentiable trajectory discrepancy, defined as:

$$\widetilde{D}_{\mathrm{traj}}(\theta; m) = \alpha D_{\mathrm{tok}}(\theta; m) + (1-\alpha)\widetilde{D}_{\mathrm{ord}}(\theta; m), \tag{10}$$

where $\alpha \in [0, 1]$ balances token behaviour and decoding order preservation. The trajectory alignment loss in CLD is then defined over the memory $\mathcal{H}_{<k}$ as:

$$\mathcal{L}_{\mathrm{traj}}(\theta) = \mathbb{E}_{m \sim \mathcal{H}_{<k}} \left[\widetilde{D}_{\mathrm{traj}}(\theta; m) \right]. \tag{11}$$

Minimising $\mathcal{L}_{\mathrm{traj}}$ preserves both token predictions and decoding order patterns, thereby mitigating decoding trajectory forgetting in dLLMs during continual RL.

4.3 Continual RL for dLLMs with Decoding Trajectory Alignment

CLD performs continual RL by jointly optimising the incoming domain RL and the decoding trajectory alignment objective, enabling dLLMs to acquire decoding behaviours on new domains while preserving the learnt decoding patterns from previous domains. At each continual RL stage k, the incoming domain $\mathcal{D}_k$ can be optimised with various dLLM RL objectives, while the decoding trajectory memory $\mathcal{H}_{<k}$ provides the trajectory alignment signal. The complete CLD objective is:

$$\mathcal{L}^{(k)}_{\mathrm{CLD}}(\theta) = \mathcal{L}^{(k)}_{\mathrm{RL}}(\theta) + \lambda_{\mathrm{traj}} \mathcal{L}_{\mathrm{traj}}(\theta), \tag{12}$$

where $\lambda_{\mathrm{traj}} \geq 0$ controls the strength of decoding trajectory preservation. Intuitively, the native RL objective provides the reward-based learning signal for acquiring capabilities from the incoming domain, while $\mathcal{L}_{\mathrm{traj}}$ provides the decoding trajectory alignment signal for preserving previously learnt decoding behaviour. When $\lambda_{\mathrm{traj}} = 0$, $\mathcal{L}^{(k)}_{\mathrm{CLD}}(\theta) = \mathcal{L}^{(k)}_{\mathrm{RL}}(\theta)$ and CLD reduces to vanilla continual RL. CLD can therefore be directly integrated with various dLLM RL algorithms.

4.4 Complexity Analysis

Let M denote the number of stored decoding trajectory records per domain, R the average number of masked positions per record, and K the number of preserved token probabilities per position. At continual RL stage k, the memory contains $(k-1)M$ records and therefore requires $\mathcal{O}((k-1)MRK)$ storage for token information and $\mathcal{O}((k-1)MR)$ storage for decoding order information. During training, CLD requires additional computation only on the sampled stored states used to optimise $\mathcal{L}_{\mathrm{traj}}$. CLD introduces no additional inference parameters or decoding cost.

4.5 Theoretical Insights

In this section, we show that aligning decoding trajectories provides a principled way to constrain final response distribution drift in dLLMs. Let θ^- denote a reference dLLM from a previous domain, and let ξ denote a complete decoding trajectory with final response y. We use $\mathbb{P}_{\theta^-}(\xi \mid x)$ and $\mathbb{P}_\theta(\xi \mid x)$ for the trajectory distributions induced by the reference and current dLLMs.

Theorem 1 (Decoding Trajectory KL Upper Bound in dLLMs)**.** *Let $p_{\theta^-}(y \mid x)$ and $p_\theta(y \mid x)$ denote the final response distributions induced by the reference and current dLLMs. The KL divergence between the final response distributions is upper-bounded by the KL divergence between their decoding trajectory distributions:*

$$D_{\mathrm{KL}}(p_{\theta^-}(y \mid x) \| p_\theta(y \mid x)) \leq D_{\mathrm{KL}}(\mathbb{P}_{\theta^-}(\xi \mid x) \| \mathbb{P}_\theta(\xi \mid x)). \tag{13}$$

Detailed proof is in Appendix D. **Intuitively**, Theorem 1 shows that controlling decoding trajectory divergence provides a sufficient way to constrain final response distribution drift in dLLMs. Thus, CLD aligns token predictions through $D_{\mathrm{tok}}(\theta; m)$ and decoding order through $\widetilde{D}_{\mathrm{ord}}(\theta; m)$. For each order pair $(i, j) \in \bar{\mathcal{P}}$, define $\eta^\theta_{t,ij} = \sigma((c^\theta_{t,i} - c^\theta_{t,j})/\gamma)$ as the soft pairwise probability that i is decoded before j, where σ is sigmoid. Let $\mathrm{Bern}(q)$ denote a Bernoulli distribution with probability q. Since $(i, j) \in \bar{\mathcal{P}}$ indicates that i precedes j, the pairwise objective admits the KL form:

$$\log\big(1 + \exp\big(-(c^\theta_{t,i} - c^\theta_{t,j})/\gamma\big)\big) = D_{\mathrm{KL}}\big(\mathrm{Bern}(1) \| \mathrm{Bern}(\eta^\theta_{t,ij})\big). \tag{14}$$

Thus, $\mathcal{L}_{\mathrm{traj}}$ in Eq. 11 combines token-level KL and soft pairwise-order KL alignment, providing a tractable surrogate motivated by the trajectory KL upper bound in Eq. 13.

5 EXPERIMENTS

Dataset. We evaluate CLD on 9 benchmarks covering code generation, mathematical reasoning, logical reasoning, and general capabilities. Our main continual RL sequence is Logic → Math → Code, using Knights-and-Knaves (KK) (Xie et al., 2025), MATH (Lightman et al., 2024) (evaluated on MATH500), and KodCode (Xu et al., 2025), following prior dLLM RL setups (Zhao et al., 2025; Jiang et al., 2026b;a; Rojas et al., 2026). For benchmarks without training sets, e.g., HumanEval, we train on KodCode following prior work (Rojas et al., 2026). Single-domain RL trains on one domain independently, mix-domain RL jointly trains on all available domains, and continual RL sequentially trains on each domain without future-domain access. More details are in Appendix E.1.

Baselines and Metrics. (i) Single-domain RL. We train each domain by dLLM RL methods, including *Diffu*-GRPO (Zhao et al., 2025), *d1* (Zhao et al., 2025), GDPO (Rojas et al., 2026), ESPO (Ou et al., 2026), SPG (Wang et al., 2026a), *wd1* (Tang et al., 2026), and StableDRL (Zhong et al., 2026). *d1* builds upon *Diffu*-GRPO with additional SFT before RL. **(ii) Mix-domain RL.** We jointly optimise on all training domains following prior work (Jiang et al., 2026a) and include block decoding method AdaBlock-dLLM (Lu et al., 2026) and *b1* (Jiang et al., 2026b). **(iii) Continual RL.** Vanilla Continual RL sequentially optimises dLLMs on each domain, while Continual Prompt Replay (Luo et al., 2026a) rehearses historical prompts under the same memory budget as CLD for a fair comparison. Average performance (ACC), backward transfer (BWT), and average forgetting (AF) are used with detailed definitions in Table 7. BWT measures the change from the performance immediately after learning each domain to its final performance, while AF measures the degradation from the historical best performance to the final performance.

Implementation Details. All experiments are conducted on one AMD MI355X GPU. Zero-shot evaluation protocol is used following existing work. Unless otherwise specified, LLaDA-8B-Instruct (Nie et al., 2025) is used by default with the completion length of 256 and the block size of 32, following the most common settings in prior dLLMs (Zhao et al., 2025; Tang et al., 2026; Rojas et al., 2026; Zhong et al., 2026) for fairness and consistency. More details are in Appendix E.

Table 1: **Overall performance for different dLLM training settings.** Single-domain RL trains independently, while Mix-domain RL jointly trains on all domains. Zero-shot accuracy is highlighted by **best** and runner-up. **RL** is performance after each in-domain stage. **Continual RL** is final-stage performance. All continual RL start from the same Stage 1 StableDRL checkpoint for fair comparison. *ACC* is average accuracy (%). BWT and AF measure backward transfer and forgetting.

Method	Logic	Math	Code	ACC↑
	knights and knaves	MATH500	KodCode	
LLaDA-8B-Instruct	30.00	32.00	22.00	28.00
+ SFT	32.00	32.20	19.40	27.87
Single-domain RL				
+ *Diffu*-GRPO	36.86	33.60	24.60	31.69
+ *d1*	37.00	33.40	25.20	31.87
+ GDPO	36.86	32.20	26.80	31.95
+ ESPO	36.14	35.00	25.80	32.31
+ SPG	36.57	34.80	26.40	32.59
+ *wd1*	38.00	34.20	25.60	32.60
+ StableDRL	51.91	35.40	28.20	38.50
Mix-domain RL				
Vanilla Mix RL	30.14	28.20	22.60	26.98
+ AdaBlock-dLLM	31.43	30.40	23.40	28.41
+ *b1*	33.86	33.00	25.20	30.69
Continual RL				

Continual RL Methods	Stage 1: Logic		Stage 2: Math		Stage 3: Code	Continual RL Metrics		
	RL	Continual RL	RL	Continual RL	Continual RL	ACC↑	BWT↑	AF↓
Vanilla Continual RL	51.91	49.17	35.40	32.00	26.80	35.99	−3.07	4.36
Continual Prompt Replay	51.91	50.59	36.00	33.20	26.60	36.80	−2.06	2.58
CLD	51.91	**52.69**	36.20	**36.00**	**28.40**	**39.03**	**+0.29**	**0.10**

5.1 OVERALL PERFORMANCE

The overall performance is in Table 1. **(i) CLD achieves the strongest overall performance among all compared training settings.** Compared with single and mix-domain RL methods, continual

optimisation with decoding trajectory preservation can effectively preserve previously learnt capabilities while acquiring new-domain capabilities. **(ii) Direct continual RL suffers from substantial backward forgetting, while previous-domain preservation alleviates this degradation.** Sequentially optimising only on incoming domains in Vanilla Continual RL can substantially degrade previously learnt capabilities. Preserving historical information in Continual Prompt Replay can alleviate catastrophic forgetting. **(iii) CLD further mitigates dLLM forgetting and facilitates positive backward transfer with decoding trajectory alignment.** Additional analyses on different domains and domain orders are provided in Sections 5.3 and 5.4. Sensitivity analysis, more dLLMs, RL algorithms and block sizes are provided in Appendices F.6, F.8, F.9 and F.10, respectively.

5.2 Ablation Study

We examine the effect of token prediction and decoding order alignment in Table 2. **Both alignment components in CLD consistently improve over Vanilla Continual RL across the two dLLM backbones.** In particular, preserving decoding order patterns plays a more prominent role in mitigating forgetting. Yet, token prediction alignment provides additional improvements, and combining both components achieves the highest average accuracy on both dLLM backbones.

Table 2: **Ablation study.** Token and Order denote token prediction and decoding order alignment. Average accuracy over continual RL domains is adopted.

Method	Token	Order	LLaDA	Dream
Base	×	×	28.00	24.75
Vanilla Continual RL	×	×	35.99	37.35
Continual Prompt Replay	×	×	36.80	35.92
CLD w/o Token	×	✓	38.40	41.30
CLD w/o Order	✓	×	36.65	40.28
CLD	✓	✓	**39.03**	**43.96**

5.3 Cross-domain Generalisation Beyond Continual RL Domains

CLD enhances the cross-domain generalisability of dLLMs after continual RL. As shown in Table 3, CLD consistently improves zero-shot performance on additional benchmarks not included during training. These results suggest that decoding trajectory preservation not only mitigates forgetting on previous domains, but also improves cross-domain generalisability of the trained dLLMs.

Table 3: **Cross-domain generalisation to additional benchmarks.** StableDRL is used as the default RL algorithm. All additional domains are evaluated zero-shot and are not used during training.

dLLM Model	Method	MBPP	HumanEval	GSM8K	Countdown	ARC-E	ARC-C	Avg.
LLaDA-8B-Instruct	Base	33.60	28.66	76.19	16.80	76.98	73.21	50.91
	Vanilla Continual RL	33.80	27.44	78.24	18.75	78.16	75.09	51.91
	CLD	**34.40**	**29.88**	**80.29**	**26.56**	**93.06**	**84.90**	**58.18**
Dream-v0-Instruct-7B	Base	33.40	24.39	77.33	19.14	80.81	76.02	51.85
	Vanilla Continual RL	35.20	31.10	77.56	21.48	82.32	78.67	54.39
	CLD	**39.60**	**40.85**	**80.59**	**32.42**	**93.94**	**87.20**	**62.43**

5.4 Stability against Continual Domain Orders

In Table 4, for each domain order, Vanilla Continual RL and CLD start from the same first-domain RL checkpoint for a fair comparison. **CLD consistently achieves higher performance and BWT with lower forgetting** than the Vanilla Continual RL method without decoding trajectory alignment, demonstrating its stability under different continual RL domain sequences.

Table 4: **CLD under different continual RL domain orders.** *Vanilla* denotes the baseline continual RL method without trajectory alignment. *RL* and *Continual RL* denote performance after learning each stage and after the final stage, respectively.

Domain Order	Method	Stage 1		Stage 2		Stage 3	Continual RL Metrics		
		RL	Continual RL	RL	Continual RL	Continual RL	ACC↑	BWT↑	AF↓
Logic, Math, Code	Vanilla Continual RL	51.91	49.17	35.40	32.00	26.80	35.99	−3.07	4.36
	CLD	51.91	52.69	36.20	36.00	28.40	39.03	+0.29	0.10
Math, Code, Logic	Vanilla Continual RL	35.40	33.20	27.00	26.20	48.17	35.86	−1.50	2.24
	CLD	35.40	36.80	27.20	27.40	53.79	39.33	+0.80	0.00
Code, Math, Logic	Vanilla Continual RL	28.20	26.60	33.60	30.20	47.26	34.69	−2.50	2.50
	CLD	28.20	30.40	35.80	36.20	51.79	39.46	+1.30	0.00

5.5 Decoding Order Preservation

We visualise the evolution of decoding orders in CLD and the Vanilla Continual RL baseline *Before* and *After* continual RL in Figure 5, where each cell indicates the decoding order of the corresponding token position. Block size is 32 and dLLM decodes at least two tokens at each diffusion step following (Zhao et al., 2025; Tang et al., 2026). **The decoding trajectory under CLD remains highly similar before and after continual RL** on the Code domain. In contrast, Vanilla Continual RL exhibits substantially larger changes across token positions after learning the Code domain.

As dLLMs decode by blocks, we further provide a block-averaged visualisation in Figure 6 to show decoding order changes at the block level. Each cell represents the average decoding order shift between a pair of positions within blocks, with darker colours indicating larger shifts. Vanilla Continual RL exhibits widespread and substantial decoding order drift on the Math task after learning the Code domain. **After adding decoding trajectory alignment in CLD, block-level shifts are substantially reduced**. These results provide direct evidence that CLD mitigates decoding trajectory forgetting by mitigating shifts in decoding order patterns. Further analysis on token preservation, more blocks and other domains are in Appendix F.3, F.4 and F.5 respectively.

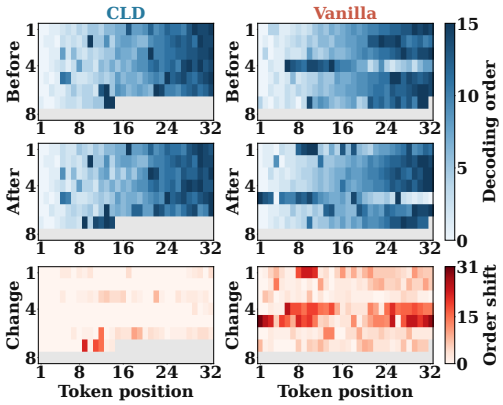


Figure 5: **Decoding trajectory changes *Before* and *After* continual RL.** Blue indicates a decoding order index. Red indicates order changes by *After*−*Before* before EOS token.

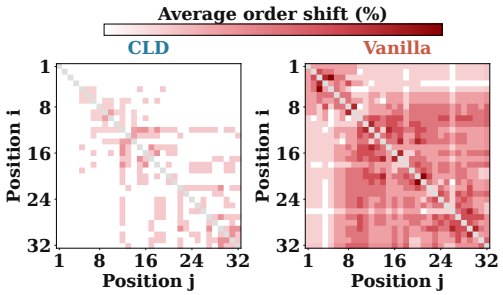


Figure 6: **Mean decoding order shift on the Math domain.** Left: CLD. Right: Vanilla. Darker colours indicate larger average decoding order shifts for a pair of positions in blocks.

6 Related Work

Our work connects three research domains. **(1) Diffusion Large Language Models.** dLLMs generate sequences through iterative denoising rather than fixed left-to-right generation (Sohl-Dickstein et al., 2015; Campbell et al., 2022; Meng et al., 2022; Austin et al., 2021a; Lou et al., 2024; Shi et al., 2024; Sahoo et al., 2024). Recent dLLMs, including LLaDA, Dream, LLaDA2.0, and LLaDA2.1, exhibit strong reasoning capabilities (Zhu et al., 2026; Nie et al., 2025; Ye et al., 2025; Bie et al., 2025; 2026), with block-based generation enabling flexible parallel decoding (Arriola et al., 2025). **(2) Reinforcement Learning for dLLMs.** Recent RL methods improve dLLM reasoning (Zhao et al., 2025; Wang et al., 2026b; Tang et al., 2026; Rojas et al., 2026; He et al., 2025; Ou et al., 2026; Wang et al., 2026a; Zhong et al., 2026; Zhu et al., 2025; Yang et al., 2026; Ni et al., 2026), yet mainly focus on single-domain scenarios, leaving continual RL largely unexplored. **(3) Continual Reinforcement Learning.** Existing methods mitigate forgetting through regularisation, replay, and related continual learning techniques (Kaplanis et al., 2019; Rolnick et al., 2019; Chung et al., 2024; Huang et al., 2024; He et al., 2024; Feng et al., 2025; Wang et al., 2023; Liang et al., 2025; Gutiérrez et al., 2025; Luo et al., 2026a;b). Unlike prior approaches developed mainly for AR LLMs, CLD targets the dLLM-specific decoding process. Additional details are provided in Appendix B.

7 Conclusion

In this work, we study catastrophic forgetting in continual RL for dLLMs from the perspective of decoding trajectories. We identify decoding trajectory forgetting in both token predictions and decoding order patterns, and propose CLD to preserve these behaviours through trajectory alignment. Extensive experiments show that CLD consistently mitigates forgetting while maintaining strong cross-domain generalisability. CLD paves the way for extending conventional dLLM methods from single-domain post-training to practical continual adaptation across real-world evolving domains.

AI USE STATEMENT

AI tools were used only for grammar checking. All content was independently prepared, reviewed, and verified by the authors, who take full responsibility for the final manuscript.

ETHICS STATEMENT

This work does not involve human participants, private or sensitive personal information, or newly collected user data. All experiments are based on publicly available benchmark datasets and established models and methods. Further details on the experimental setup and reproducibility are provided in Appendix E.

REFERENCES

Marianne Arriola, Aaron Gokaslan, Justin T. Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block Diffusion: Interpolating Between Autoregressive and Diffusion Language Models. In *ICLR*, 2025.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured Denoising Diffusion Models in Discrete State-Spaces. *arXiv preprint arXiv:2107.03006*, 2021a.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program Synthesis with Large Language Models. *arXiv preprint arXiv:2108.07732*, 2021b.

Tiwei Bie, Maosong Cao, Kun Chen, Lun Du, Mingliang Gong, Zhuochen Gong, Yanmei Gu, Jiaqi Hu, Zenan Huang, Zhenzhong Lan, Chengxi Li, Chongxuan Li, Jianguo Li, Zehuan Li, Huabin Liu, Lin Liu, Guoshan Lu, Xiaocheng Lu, Yuxin Ma, Jianfeng Tan, Lanning Wei, Ji-Rong Wen, Yipeng Xing, Xiaolu Zhang, Junbo Zhao, Da Zheng, Jun Zhou, Junlin Zhou, Zhanchao Zhou, Liwang Zhu, and Yihong Zhuang. LLaDA2.0: Scaling Up Diffusion Language Models to 100B. *arXiv preprint arXiv:2512.15745*, 2025.

Tiwei Bie, Maosong Cao, Xiang Cao, Bingsen Chen, Fuyuan Chen, Kun Chen, Lun Du, Daozhuo Feng, Haibo Feng, Mingliang Gong, Zhuocheng Gong, Yanmei Gu, Jian Guan, Kaiyuan Guan, Hongliang He, Zenan Huang, Juyong Jiang, Zhonghui Jiang, Zhenzhong Lan, Chengxi Li, Jianguo Li, Zehuan Li, Huabin Liu, Lin Liu, Guoshan Lu, Yuan Lu, Yuxin Ma, Xingyu Mou, Zhenxuan Pan, Kaida Qiu, Yuji Ren, Jianfeng Tan, Yiding Tian, Zian Wang, Lanning Wei, Tao Wu, Yipeng Xing, Wentao Ye, Liangyu Zha, Tianze Zhang, Xiaolu Zhang, Junbo Zhao, Da Zheng, Hao Zhong, Wanli Zhong, Jun Zhou, Junlin Zhou, Liwang Zhu, Muzhi Zhu, and Yihong Zhuang. LLaDA2.1: Speeding Up Text Diffusion via Token Editing. *arXiv preprint arXiv:2602.08676*, 2026.

Andrew Campbell, Joe Benton, Valentin De Bortoli, Thomas Rainforth, George Deligiannidis, and Arnaud Doucet. A Continuous Time Framework for Discrete Denoising Models. In *NeurIPS*, 2022.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating Large Language Models Trained on Code. *arXiv preprint arXiv:2107.03374*, 2021.

Zhoujun Cheng, Shibo Hao, Tianyang Liu, Fan Zhou, Yutao Xie, Feng Yao, Yuexin Bian, Nilabjo Dey, Yonghao Zhuang, Yuheng Zha, Yi Gu, Kun Zhou, Yuqi Wang, Yuan Li, Richard Fan, Jianshu She, Chengqian Gao, Abulhair Saparov, Taylor W. Killian, Haonan Li, Mikhail Yurochkin, Eric P.

Xing, Zhengzhong Liu, and Zhiting Hu. Revisiting Reinforcement Learning for LLM Reasoning from A Cross-Domain Perspective. In *NeurIPS*, 2025.

Wesley Chung, Lynn Cherif, David Meger, and Doina Precup. Parseval Regularization for Continual Reinforcement Learning. In *NeurIPS*, 2024.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Yuhao Du, Zhuo Li, Pengyu Cheng, Zhihong Chen, Yuejiao Xie, Xiang Wan, and Anningzhe Gao. RLHF in an SFT Way: From Optimal Solution to Reward-Weighted Alignment. *arXiv preprint arXiv:2502.11026*, 2025.

Yujie Feng, Xujia Wang, Zexin Lu, Shenghong Fu, Guangyuan Shi, Yongxin Xu, Yasha Wang, Philip S. Yu, Xu Chu, and Xiao-Ming Wu. Recurrent Knowledge Identification and Fusion for Language Model Continual Learning. In *ACL*, 2025.

Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. From RAG to Memory: Non-Parametric Continual Learning for Large Language Models. In *ICML*, 2025.

Haoyu He, Katrin Renz, Yong Cao, and Andreas Geiger. Mdpo: Overcoming the training-inference divide of masked diffusion language models. *arXiv preprint arXiv:2508.13148*, 2025.

Jinghan He, Haiyun Guo, Kuan Zhu, Zihan Zhao, Ming Tang, and Jinqiao Wang. SEEKR: Selective attention-guided knowledge retention for continual learning of large language models. In *EMNLP*, 2024.

Jianheng Huang, Leyang Cui, Ante Wang, Chengyi Yang, Xinting Liao, Linfeng Song, Junfeng Yao, and Jinsong Su. Mitigating Catastrophic Forgetting in Large Language Models with Self-Synthesized Rehearsal. In *ACL*, 2024.

Yan Jiang, Ruihong Qiu, and Zi Huang. Block-R1: Rethinking the Role of Block Size in Multi-domain Reinforcement Learning for Diffusion Large Language Models. *arXiv preprint arXiv:2605.11726*, 2026a.

Yan Jiang, Ruihong Qiu, and Zi Huang. Break the Block: Dynamic-size Reasoning Blocks for Diffusion Large Language Models via Monotonic Entropy Descent with Reinforcement Learning. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026b.

Christos Kaplanis, Murray Shanahan, and Claudia Clopath. Policy Consolidation for Continual Reinforcement Learning. In *ICML*, 2019.

M. G. Kendall. A New Measure of Rank Correlation. *Biometrika*, 30(1–2):81–93, 1938. doi: 10.1093/biomet/30.1-2.81.

Solomon Kullback and Richard A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 1951.

Yan-Shuo Liang, Jia-Rui Chen, and Wu-Jun Li. Gated Integration of Low-Rank Adaptation for Continual Learning of Large Language Models. In *NeurIPS*, 2025.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *ICLR*, 2024.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete Diffusion Modeling by Estimating the Ratios of the Data Distribution. In *ICML*, 2024.

Guanxi Lu, Hao Mark Chen, Yuto Karashima, Zhican Wang, Daichi Fujiki, and Hongxiang Fan. AdaBlock-dLLM: Semantic-Aware Diffusion LLM Inference via Adaptive Block Size. In *ICLR*, 2026.

Lirui Luo, Guoxi Zhang, Hongming Xu, Rongqing Li, Cong Fang, and Lifeng Fan. Continual Reasoning Gym: Diagnosing and Harnessing Shared Reasoning in Continual RLVR. *arXiv preprint arXiv:2608.18574*, 2026a.

Mao-Lin Luo, Zhe-Xu Wang, Zi-Hao Zhou, Bo Ye, Jian Zhao, Min-Ling Zhang, and Tong Wei. RL Forgets! Towards Continual Policy Optimization. *arXiv preprint arXiv:2607.04364*, 2026b.

Chenlin Meng, Kristy Choi, Jiaming Song, and Stefano Ermon. Concrete Score Matching: Generalized Score Matching for Discrete Data. *NeurIPS*, 2022.

Youssef Mroueh. Reinforcement Learning with Verifiable Rewards: GRPO's Effective Loss, Dynamics, and Success Amplification. *arXiv preprint arXiv:2503.06639*, 2025.

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 *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. *arXiv preprint arXiv:2502.09992*, 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 *ICLR*, 2026.

Jiayi Pan, Junjie Zhang, Xingyao Wang, Lifan Yuan, Hao Peng, and Alane Suhr. Tinyzero. *GitHub repository https://github.com/Jiayi-Pan/TinyZero*, 2025.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-Weighted Regression: Simple and Scalable Off-Policy Reinforcement Learning. *arXiv preprint arXiv:1910.00177*, 2019.

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 *NeurIPS*, 2023.

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 *ICLR*, 2026.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience Replay for Continual Learning. In *NeurIPS*, 2019.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and Effective Masked Diffusion Language Models. *arXiv preprint arXiv:2406.07524*, 2024.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust Region Policy Optimization. In *ICML*, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. 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 K. Titsias. Simplified and Generalized Masked Diffusion for Discrete Data. In *NeurIPS*, 2024.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep Unsupervised Learning Using Nonequilibrium Thermodynamics. In *ICML*, 2015.

Xiaohang Tang, Rares Dolga, Sangwoong Yoon, and Ilija Bogunovic. Wd1: Weighted policy optimization for reasoning in diffusion language models. In *ICLR*, 2026.

Chenyu Wang, Paria Rashidinejad, DiJia Su, Song Jiang, Sid Wang, Siyan Zhao, Cai Zhou, Shannon Zejiang Shen, Feiyu Chen, Tommi Jaakkola, Yuandong Tian, and Bo Liu. SPG: Sandwiched Policy Gradient for Masked Diffusion Language Models. In *ICLR*, 2026a.

Guanghan Wang, Gilad Turok, Yair Schiff, Marianne Arriola, and Volodymyr Kuleshov. d2: Improving Reasoning in Diffusion Language Models via Trajectory Likelihood Estimation. In *ICML*, 2026b.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuanjing Huang. Orthogonal Subspace Learning for Language Model Continual Learning. In *Findings of EMNLP*, 2023.

Yinjie Wang, Ling Yang, Bowen Li, Ye Tian, Ke Shen, and Mengdi Wang. Revolutionizing reinforcement learning framework for diffusion large language models. In *ICLR*, 2026c.

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 *ICLR*, 2026.

Chulin Xie, Yangsibo Huang, Chiyuan Zhang, Da Yu, Xinyun Chen, Bill Yuchen Lin, Bo Li, Badih Ghazi, and Ravi Kumar. On Memorization of Large Language Models in Logical Reasoning. In *IJCNLP-AACL*, 2025.

Wei Xiong, Jiarui Yao, Yuhui Xu, Bo Pang, Lei Wang, Doyen Sahoo, Junnan Li, Nan Jiang, Tong Zhang, Caiming Xiong, and Hanze Dong. A Minimalist Approach to LLM Reasoning: From Rejection Sampling to Reinforce. *arXiv preprint arXiv:2504.11343*, 2025.

Zhangchen Xu, Yang Liu, Yueqin Yin, Mingyuan Zhou, and Radha Poovendran. KodCode: A Diverse, Challenging, and Verifiable Synthetic Dataset for Coding. In *Findings of ACL*, 2025.

Jingyi Yang, Yuxian Jiang, Xuhao Hu, Shuang Cheng, Biqing Qi, and Jing Shao. Dare: Diffusion large language models alignment and reinforcement executor. *arXiv preprint arXiv:2604.04215*, 2026.

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.

Siyan Zhao, Devaansh Gupta, Qinqing Zheng, and Aditya Grover. d1: Scaling Reasoning in Diffusion Large Language Models via Reinforcement Learning. In *NeurIPS*, 2025.

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*, 2026.

Banghua Zhu, Hiteshi Sharma, Felipe Vieira Frujeri, Shi Dong, Chenguang Zhu, Michael I Jordan, and Jiantao Jiao. Fine-Tuning Language Models with Advantage-Induced Policy Alignment. *arXiv preprint arXiv:2306.02231*, 2023.

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 *ACL*, 2026.

Ying Zhu, Jiaxin Wan, Xiaoran Liu, Siyang He, Qiqi Wang, Xu Guo, Tianyi Liang, Zengfeng Huang, Ziwei He, and Xipeng Qiu. Dirl: An efficient post-training framework for diffusion language models. *arXiv preprint arXiv:2512.22234*, 2025.

A Appendix Overview

This appendix provides supplementary literature, theoretical analysis, implementation details, and additional experimental results. The structure is as follows:

- **Appendix B** provides detailed related work on:
  - Diffusion large language models;
  - Reinforcement learning for LLMs;
  - Reinforcement learning for dLLMs;
  - Continual reinforcement learning;
  - Continual reinforcement learning for LLMs.
- **Appendix C** reviews the reinforcement learning preliminaries for dLLMs, including the GRPO-style objective used in recent dLLM post-training methods.
- **Appendix D** provides the detailed theoretical analysis, including:
  - The proof of Theorem 1, which establishes the decoding trajectory KL upper bound on final response distribution drift;
  - The connection between the theoretical upper bound and the tractable token prediction and soft pairwise-order alignment objectives in CLD.
- **Appendix E** provides implementation and evaluation details, including:
  - Benchmark configuration and dataset statistics in Appendix E.1;
  - Detailed backbone and baseline implementations in Appendix E.2;
  - The default training configuration in Table 6;
  - The complete CLD pipeline in Algorithm 1;
  - Continual RL training protocol in Appendix E.3;
  - Task reward configuration and trajectory memory filtering in Appendix E.4;
  - Evaluation protocol and continual RL metrics in Appendix E.5.
- **Appendix F** provides additional experimental analyses, including:
  - Decoding order shifts across domains in Appendix F.1;
  - Detailed decoding trajectory visualisations in Appendix F.2;
  - Token prediction distribution preservation in Appendix F.3;
  - Decoding order shifts across individual blocks in Appendix F.4;
  - Additional cross-domain visualisation in Appendix F.5;
  - Hyperparameter sensitivity analysis in Appendix F.6;
  - Robustness under multiple random seeds in Appendix F.7;
  - Generalisation to other dLLM backbones and generation lengths in Appendix F.8;
  - Generalisation to different dLLM RL algorithms in Appendix F.9;
  - Robustness to different decoding block sizes in Appendix F.10.

B Detailed Related Work

B.1 Diffusion Large Language Models

Diffusion large language models (dLLMs) have recently emerged as a parallel generation alternative to autoregressive LLMs. Rather than committing to a fixed left-to-right factorisation, dLLMs iteratively recover masked tokens from partially observed sequences through discrete denoising (Sohl-Dickstein et al., 2015; Campbell et al., 2022; Meng et al., 2022; Austin et al., 2021a; Lou et al., 2024; Shi et al., 2024; Sahoo et al., 2024). Recent dLLMs, including LLaDA (Zhu et al., 2026; Nie et al., 2025), Dream (Ye et al., 2025), LLaDA2.0 (Bie et al., 2025), and LLaDA2.1 (Bie et al., 2026), have shown competitive language understanding and reasoning capabilities at increasing model scales. Many dLLMs further employ block-based generation, allowing several masked positions within a block to be resolved in parallel (Arriola et al., 2025). Consequently, their generation behaviour is characterised not only by the predicted tokens but also by the order in which masked positions are revealed. This additional flexibility distinguishes dLLM decoding trajectories from the predetermined generation order of autoregressive models and motivates studying how such trajectories change during continual model updates.

B.2 Reinforcement Learning for LLMs

Reinforcement learning has become an important post-training paradigm for improving LLM reasoning through rewards derived from correctness, execution outcomes, or other task-specific criteria. General policy optimisation algorithms such as TRPO (Schulman et al., 2015) and PPO (Schulman et al., 2017) laid the foundations for learning from reward signals, while Group Relative Policy Optimisation (GRPO) (Shao et al., 2024) simplifies this process by estimating advantages from groups of sampled responses without requiring an additional value model. Recent studies have also related reasoning RL to weighted regression and selective optimisation (Mroueh, 2025; Peng et al., 2019; Zhu et al., 2023; Du et al., 2025). Rejection Sampling Fine Tuning (Xiong et al., 2025) and preference-based approaches such as DPO (Rafailov et al., 2023) provide related mechanisms for increasing the probability of preferred responses. While these approaches have substantially advanced reasoning post-training, they predominantly optimise models under a stationary task distribution rather than a sequence of changing domains.

B.3 Reinforcement Learning for dLLMs

Recently, a growing body of work adapts RL techniques to the distinctive probability modelling and denoising process of dLLMs. Representative approaches include *d1* (Zhao et al., 2025), *d2* (Wang et al., 2026b), *wd1* (Tang et al., 2026), GDPO (Rojas et al., 2026), MDPO (He et al., 2025), ESPO (Ou et al., 2026), SPG (Wang et al., 2026a), StableDRL (Zhong et al., 2026), DiRL (Zhu et al., 2025) and DARE (Yang et al., 2026). Recent JustGRPO (Ni et al., 2026) shows that constraining arbitrary order exploration can stabilise single-domain RL. This is complementary to CLD, which does not encourage arbitrary decoding orders but preserves previously learnt within-block decoding order under sequential domain shifts. All these existing RL for dLLM formulations differ in how the policy objective is constructed from diffusion generation. Some methods apply optimisation through token-level likelihood terms, whereas others formulate the update at the sequence or denoising trajectory level. Despite their progress, most existing dLLM RL research largely focuses on learning one domain at a time or optimising a fixed mixture of domains. Sequentially acquiring new domains while preserving previously learnt dLLM capabilities has received substantially less attention. Moreover, continual optimisation can alter both token distributions and parallel decoding orders, creating a form of trajectory change that is specific to diffusion-based generation.

B.4 Continual Reinforcement Learning

Continual RL aims to acquire new behaviours from sequential tasks or changing environments while preserving previously learnt capabilities. Existing studies have explored different mechanisms for mitigating catastrophic forgetting during continual RL. Policy Consolidation (Kaplanis et al., 2019) regularises the current policy with a cascade of historical policies operating at different timescales. CLEAR (Rolnick et al., 2019) combines on-policy learning from current experience with off-policy

replay and behavioural cloning from historical experience to balance learning new behaviours and preserving previous ones. Parseval Regularization (Chung et al., 2024) introduces an additional regularisation objective that maintains orthogonality of network weight matrices to preserve trainability during continual RL. These studies mitigate forgetting by extending RL optimisation with mechanisms for preserving historical behaviours across sequential tasks.

B.5 Continual Reinforcement Learning for LLMs

Continual learning techniques have been increasingly adopted for LLMs to support sequential adaptation to evolving tasks while preserving previously learnt capabilities. Earlier continual learning studies for LLMs have explored rehearsal and distillation (Huang et al., 2024; He et al., 2024), parameter-level adaptation (Feng et al., 2025; Wang et al., 2023; Liang et al., 2025), and external memory (Gutiérrez et al., 2025) to mitigate catastrophic forgetting.

Recently, Continual Reasoning Gym (Luo et al., 2026a) studies continual RL with verifiable rewards over sequential reasoning tasks and introduces Continual Prompt Replay, which revisits historical prompts while regenerating responses using the current policy. Continual Policy Optimization (CPO) (Luo et al., 2026b) further studies catastrophic forgetting during continual RL post-training for multimodal reasoning models and limits policy drift on previous tasks through prior-task behavioural regularisation.

Despite these advances in conventional AR LLM domains, continual learning for dLLMs remains underexplored. In particular, sequential RL for dLLMs can greatly distort not only token prediction distributions but also decoding order patterns. Token distribution distillation can constrain prediction probabilities at individual positions but does not directly preserve their relative confidence ordering across positions, which determines decoding order in dLLMs. CLD therefore enables continual RL for dLLMs through previous domain decoding trajectory alignment, preserving both token prediction and decoding order patterns in dLLMs while acquiring capabilities from new domains.

C Preliminaries on Reinforcement Learning for dLLMs

This section reviews the common RL formulation used in recent dLLM post-training methods. Recent approaches commonly adopt GRPO-style RL to optimise dLLM reasoning capabilities according to outcome rewards (Zhao et al., 2025; Tang et al., 2026; Rojas et al., 2026; Ou et al., 2026; Wang et al., 2026a; Zhong et al., 2026; Wang et al., 2026b). Given an input x, a group of G responses $\{y_g\}_{g=1}^{G}$ is sampled from the rollout policy and assigned group-normalised advantages $\{\hat{A}_g\}_{g=1}^{G}$. A clipped GRPO objective for dLLMs (Zhao et al., 2025) is formulated as:

$$\mathcal{L}_{\text{GRPO}}(\theta) = -\mathbb{E}\left[\frac{1}{G}\sum_{g=1}^{G}\min\left(\rho_g(\theta)\hat{A}_g, \text{clip}\left(\rho_g(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_g\right)\right], \tag{15}$$

where $\rho_g(\theta)$ denotes the likelihood ratio between the current and rollout policies for response y_g, and ϵ is the clipping coefficient. Different dLLM RL methods modify the policy objective, likelihood computation, or optimisation strategy while following this general post-training paradigm. We use $\mathcal{L}_{\text{RL}}(\theta, \mathcal{D})$ to denote the native RL objective applied to domain $\mathcal{D}$ during each continual RL stage.

D Proof for Insights into Decoding Trajectory Alignment

We provide a detailed proof for Theorem 1 and its connection to the trajectory alignment objective in CLD.

Proof. For a fixed prompt x, let ξ denote a complete decoding trajectory and $y = f(\xi)$ its final response. The final response distribution is obtained by marginalising trajectories that produce y:

$$p_\theta(y \mid x) = \sum_{\xi : f(\xi) = y} \mathbb{P}_\theta(\xi \mid x). \tag{16}$$

Since y is a deterministic function of ξ, the data-processing inequality for KL divergence gives

$$D_{\mathrm{KL}}(p_{\theta^-}(y \mid x) \| p_\theta(y \mid x)) \le D_{\mathrm{KL}}(\mathbb{P}_{\theta^-}(\xi \mid x) \| \mathbb{P}_\theta(\xi \mid x)), \tag{17}$$

which proves Eq. 13. □

Connection to CLD. Theorem 1 motivates preserving decoding trajectories as a way to constrain final response distribution drift. CLD implements this principle through tractable token prediction and decoding order alignment objectives. The token alignment term $D_{\mathrm{tok}}(\theta; m)$ in Eq. 8 directly minimises KL divergence between the stored and current token distributions.

For decoding order, let $\eta^\theta_{t,ij} = \sigma((c^\theta_{t,i} - c^\theta_{t,j})/\gamma)$ denote the soft pairwise probability that i is decoded before j. For each stored order pair $(i, j) \in \bar{\mathcal{P}}$,

$$D_{\mathrm{KL}}\big(\mathrm{Bern}(1) \| \mathrm{Bern}(\eta^\theta_{t,ij})\big) = -\log \eta^\theta_{t,ij} \tag{18}$$

$$= \log\left(1 + \exp\left(-\frac{c^\theta_{t,i} - c^\theta_{t,j}}{\gamma}\right)\right), \tag{19}$$

which exactly recovers each pairwise term in Eq. 9. Therefore,

$$\widetilde{D}_{\mathrm{ord}}(\theta; m) = \frac{1}{|\bar{\mathcal{P}}|} \sum_{(i,j) \in \bar{\mathcal{P}}} D_{\mathrm{KL}}\big(\mathrm{Bern}(1) \| \mathrm{Bern}(\eta^\theta_{t,ij})\big). \tag{20}$$

Accordingly, $\widetilde{D}_{\mathrm{traj}}$ in Eq. 10 combines token-level KL alignment with soft pairwise-order KL alignment. Minimising $\mathcal{L}_{\mathrm{traj}}$ therefore provides a tractable trajectory alignment surrogate motivated by the trajectory KL upper bound in Theorem 1.

Table 5: **Benchmarks used in the CLD evaluation.** The evaluation suite contains 9 benchmarks covering code generation, mathematical reasoning, logical reasoning, general capabilities, and advanced reasoning. For the three training domains, 10% of the training split is held out for checkpoint selection. The **Training** column reports the original training split size before this holdout, while the remaining benchmarks provide broader cross-domain evaluation.

Domain	Benchmark	Training	Evaluation
Code Generation	MBPP	374	500
	HumanEval	–	164
	KodCode	9,285	500
Mathematical Reasoning	GSM8K	7,473	1,319
	MATH / MATH500	7,500	500
	Countdown	240,632	256
Logical Reasoning	Knights-and-Knaves	6,200	700
General Capabilities	ARC-E	2,251	2,376
Advanced Reasoning	ARC-C	1,119	1,172

E Implementation and Evaluation Details

This section provides additional details of the experimental configuration, continual RL protocol, trajectory memory, task rewards, and evaluation criteria used for CLD. The detailed pseudocode is demonstrated in Algorithm 1.

E.1 Benchmark Configuration

Our evaluation covers 9 benchmarks across five reasoning domains. Code generation includes MBPP (Austin et al., 2021b), HumanEval (Chen et al., 2021), and KodCode (Xu et al., 2025); mathematical reasoning includes GSM8K (Cobbe et al., 2021), MATH500 (Lightman et al., 2024), and Countdown (Pan et al., 2025); logical constraint reasoning uses Knights-and-Knaves (Xie et al., 2025); and general and advanced reasoning are evaluated with ARC-E and ARC-C (Clark et al., 2018). The default continual RL sequence is Knights-and-Knaves $\rightarrow$ MATH $\rightarrow$ KodCode. MATH training examples are used for optimisation and MATH500 is used for evaluation. The remaining benchmarks provide broader evaluation beyond the three sequential continual RL domains. For Knights-and-Knaves, we report character-role partial accuracy following prior work (Jiang et al., 2026a) rather than exact-puzzle accuracy, so the reported scores are not constrained by the 700 evaluation instances. The detailed dataset statistics are summarised in Table 5.

E.2 Detailed Backbones and Baseline Implementations

LLaDA-8B-Instruct (Nie et al., 2025) serves as the default backbone in the main experiments. Additional experiments with other dLLMs are reported in Section F.8. StableDRL (Zhong et al., 2026) is used as the default RL algorithm because of its competitive performance across the three continual RL domains, while results with alternative dLLM RL algorithms are provided in Appendix F.9. While reproducing StableDRL, we observe early training instability issues on the logic reasoning domain, where the policy can collapse to near-empty outputs and lose meaningful reward variation. We therefore use a reduced learning rate throughout continual training and improve numerical stability in the likelihood computation by accumulating sequence likelihoods in chunked FP32. To further prevent numerical errors from propagating into optimisation, isolated non-finite vocabulary entries are masked during the softmax computation, while severely corrupted rows are discarded. These modifications stabilise and improve StableDRL training without changing its underlying RL objective, resulting in stronger baseline performance. Detailed default hyperparameters are provided in Table 6, with sensitivity analysis provided in Appendix F.6.

For reference, single-domain experiments include *Diffu*-GRPO (Zhao et al., 2025), *d1* (Zhao et al., 2025), GDPO (Rojas et al., 2026), ESPO (Ou et al., 2026), SPG (Wang et al., 2026a), *wd1* (Tang

Table 6: **Default training configuration of CLD.** Main experiments are run on one AMD Instinct MI355X GPU with 288GB of memory. The general experimental setup, optimisation schedule, and dLLM decoding hyperparameters follow common settings in existing dLLM RL work (Zhao et al., 2025; Tang et al., 2026; Zhong et al., 2026) for a fair comparison under a consistent setting.

Configuration	Value
Experimental Setup	
Default backbone	LLaDA-8B-Instruct
Default RL algorithm	StableDRL
Policy training	Full parameter
Precision	bfloat16
Random seed	42
Continual RL Schedule	
Learning rate	3×10^{-7}
Learning-rate schedule	Constant with warm-up
Warm-up ratio	0.0001
Adam β_1	0.9
Adam β_2	0.99
Weight decay	0.1
Gradient clipping threshold	0.2
Policy KL coefficient β	0.04
Clipping coefficient ϵ	0.5
Per-device training batch size	12
Gradient accumulation steps	1
Maximum training steps per domain	1,000
Checkpoint evaluation interval	100
Rollout-group size G	6
dLLM Decoding	
Default response length	256
Diffusion steps	128
Block size	32
Trajectory Alignment	
Token–order balance α	0.5
Alignment strength λ_{traj}	1.0
Ranking temperature γ	0.1
Preserved top-K tokens	32
Memory budget M for each domain	100

Algorithm 1: Continual Reinforcement Learning for dLLMs via CLD

Input: Sequential domains $\{\mathcal{D}_k\}_{k=1}^{T}$, initial dLLM θ_0, trajectory weight λ_{traj}, token weight α, ranking temperature γ, top-K size K, memory budget per domain M.
Output: Trained dLLM θ_T.
Initialise decoding trajectory memory $\mathcal{H}_{<1} \leftarrow \emptyset$;
for $k = 1, \ldots, T$ **do**
  Initialise $\theta \leftarrow \theta_{k-1}$;
  Stage 1: Continual RL for dLLMs with Trajectory Alignment;
  while *training on* $\mathcal{D}_k$ **do**
    Sample an incoming batch $\mathcal{B}_k \sim \mathcal{D}_k$ and compute the native RL loss $\mathcal{L}_{\text{RL}}^{(k)}$;
    if $\mathcal{H}_{<k} \neq \emptyset$ **then**
      Sample historical trajectories $\mathcal{B}_{\mathcal{H}} \subset \mathcal{H}_{<k}$;
      Evaluate the current dLLM on their stored historical diffusion states;
      Compute the token discrepancy D_{tok} by Eq. 8;
      Compute the decoding order discrepancy $\widetilde{D}_{\text{ord}}$ by Eq. 9;
      Compute the unified trajectory alignment loss $\mathcal{L}_{\text{traj}}$ by Eqs. 10 and 11;
      Compute $\mathcal{L}_{\text{CLD}}^{(k)}$ by Eq. 12;
    else
      $\mathcal{L}_{\text{CLD}}^{(k)} \leftarrow \mathcal{L}_{\text{RL}}^{(k)}$;
    Update θ by minimising $\mathcal{L}_{\text{CLD}}^{(k)}$;
  Select the stage-k checkpoint θ_k according to validation performance;
  Stage 2: Decoding Trajectory Memory Update;
  Collect decoding trajectories from $\mathcal{D}_k$ using θ_k;
  Select historical diffusion states and preserve their masked positions, top-K token probabilities, residual probability mass, and decoding orders;
  Construct $\mathcal{H}_k$ with M trajectory records from $\mathcal{D}_k$;
  Update $\mathcal{H}_{<k+1} \leftarrow \mathcal{H}_{<k} \cup \mathcal{H}_k$;
return θ_T;

et al., 2026), and StableDRL (Zhong et al., 2026). We additionally compare against Mix-domain RL settings and two continual baselines. Vanilla performs sequential RL without preserving information from earlier domains. Continual Prompt Replay stores historical training prompts and reuses them during later continual RL stages. Continual Prompt Replay and CLD use the same memory budget so that improvements can be attributed to trajectory alignment rather than additional memory storage.

E.3 Continual RL Training Protocol in CLD

The main experiments are run on one AMD Instinct MI355X GPU with 288GB of memory. All continual RL methods follow the same domain order, training data, underlying RL objective, and checkpoint evaluation procedure. During CLD training after the first domain, each optimisation step uniformly samples 12 decoding trajectory records from $\mathcal{H}_{<k}$ for trajectory alignment, matching the incoming-domain training batch size. Unless specified otherwise, generation uses a maximum length of 256 and a block size of 32, following the decoding configuration in most existing dLLM methods. Checkpoints are evaluated every 100 global training steps during each domain stage, and the selected checkpoint is used to initialise optimisation on the next domain. Additional results on longer generation length, different dLLMs, different RL algorithms, and different block sizes are provided in Appendix F.8, F.9 and F.10, respectively.

To control ties arising from dLLM parallel decoding, when multiple positions are decoded at the same step, their position indices are used to impose a deterministic order following Eq. 1. The resulting historical order defines the pairwise order relations used for decoding order alignment, while the current model is optimised through the differentiable confidence-based surrogate in Eq. 9. This alignment relies only on the stored decoding trajectory records and does not require retaining previous model checkpoints.

Notably, CLD leaves the native RL procedure on the incoming domain unchanged, including rollout generation, reward computation, advantage estimation, and policy optimisation. Therefore, it can be freely plugged in with existing RL for dLLM methods and support different RL algorithms during continual RL. The complete CLD pipeline is illustrated in Algorithm 1.

E.4 Task Reward Configuration and Trajectory Memory Filtering

The reward functions and reward weights are kept consistent across all methods in our experiments, following the reward evaluation protocols used in existing RL methods for dLLMs (Zhao et al., 2025; Tang et al., 2026; Jiang et al., 2026a) for a fair comparison. Vanilla, Rehearsal, and CLD therefore differ only in how historical information is preserved. For MATH, generated answers are evaluated based on output structure and mathematical equivalence between the predicted and reference solutions. KodCode evaluates Python generations through format and syntax validation followed by execution against the provided test cases. Knights-and-Knaves evaluates the predicted character roles against the reference assignments and includes a small formatting signal for structured responses. CLD follows these existing reward implementations and does not introduce any additional task-specific reward.

During each domain RL stage, CLD uses the same task reward to construct the decoding trajectory memory. Specifically, zero-reward rollouts are filtered out, and the memory is filled with the remaining reward-positive rollouts to avoid preserving low-quality generations. From each preserved rollout, CLD stores one midpoint diffusion state from each semi-autoregressive block. Each preserved record compactly stores a diffusion state, masked positions, top-K token probabilities, the residual probability mass over the remaining vocabulary and the associated decoding order.

E.5 Evaluation Protocol and Continual RL Metrics

All task results are evaluated under zero-shot pass@1. For most benchmarks, each example is scored as correct or incorrect according to its task-specific evaluation criterion. For Knights-and-Knaves, we follow existing work (Jiang et al., 2026a) to report the fraction of correctly predicted character labels over all puzzles. This partial credit metric is adopted because exact instance matching is near-degenerate for this task. Our checkpoint selection follows the same practice in prior dLLM studies (Zhao et al., 2025; Tang et al., 2026; Rojas et al., 2026; Jiang et al., 2026a) based on periodic evaluation during training to ensure a consistent setting and fair comparison with existing baselines. By default, each domain is trained for 1,000 global steps with a batch size of 12, and checkpoints are evaluated every 100 steps following existing dLLM post-training benchmarks (Jiang et al., 2026a).

For each continual RL domain, *RL* performance is measured immediately after learning that domain, whereas *Continual RL* performance is measured after completing the full continual RL sequence. BWT measures the change from the *RL* performance $P_{i,i}$ to the final performance $P_{T,i}$ on each historical domain. In contrast, AF measures the degradation from the best checkpoint performance $P_i^\star$ observed on domain i after that domain is learnt to its final performance after continual RL. For example, in Table 1, for Continual Prompt Replay, the best historical accuracies are 52.95 on Logic and 36.00 on Math, while their final accuracies after continual RL are 50.59 and 33.20, respectively. Therefore, its averaged AF score is 2.58. We further report average accuracy (ACC) to quantify final performance across domains. Decoding order preservation is analysed using normalised Kendall similarity S_K. For each record, the historical and final trajectories are compared block by block under the same decoding configuration, and S_K is averaged over the matched blocks. Formal continual RL metric definitions are provided in Table 7.

Table 7: **Evaluation metrics for continual RL on dLLMs.** T is the number of domains and $P_{k,i}$ is the performance on domain i after completing stage k. $P_i^\star$ denotes the highest checkpoint performance observed on domain i from the point at which domain i is learnt until the end of continual RL. For decoding order preservation, N_i denotes the number of historical prompts from domain i, while $\Pi_i^{(j)}$ and $\Pi_T^{(j)}$ denote the realised blockwise decoding orders of the j-th prompt after learning domain i and after the final continual RL stage, respectively. Higher ACC, BWT, and S_K indicate better performance or preservation, whereas lower AF indicates less forgetting.

Metric	Definition	Direction
Avg Performance (ACC)	$\mathrm{ACC} = \frac{1}{T}\sum_{i=1}^{T} P_{T,i}$	↑
Backward Transfer (BWT)	$\mathrm{BWT} = \frac{1}{T-1}\sum_{i=1}^{T-1} (P_{T,i} - P_{i,i})$	↑
Average Forgetting (AF)	$\mathrm{AF} = \frac{1}{T-1}\sum_{i=1}^{T-1} (P_i^\star - P_{T,i})$	↓
Decoding Order Retention (S_K)	$S_K = \frac{1}{T-1}\sum_{i=1}^{T-1}\frac{1}{N_i}\sum_{j=1}^{N_i} S_K\left(\Pi_i^{(j)}, \Pi_T^{(j)}\right)$	↑

F Additional Experimental Analysis

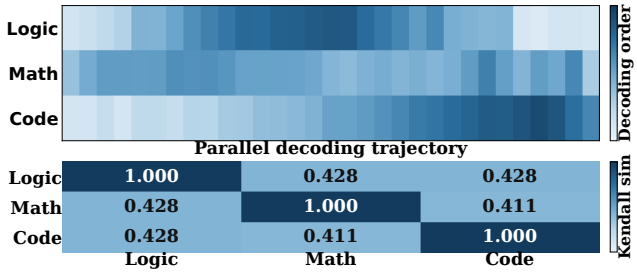


Figure 7: **Detailed block-wise visualisation of decoding order drift across domains.** The figure shows the averaged block decoding trajectories of Logic, Math, and Code tasks produced by LLaDA-8B-Instruct, together with their pairwise Kendall similarities. Different domains exhibit distinct decoding order patterns, resulting in relatively low cross-domain Kendall similarities. This observation is consistent with Figure 2(a) and suggests that domain-specific decoding order patterns are not limited to individual examples, but also arise consistently at the domain level.

F.1 Detailed Visualisation of Decoding Order Shift across Domains

We further provide a detailed visualisation of decoding order patterns across domains in Figure 7. The figure shows the decoding trajectories of Logic, Math, and Code tasks produced by LLaDA-8B-Instruct averaged over blocks, together with their pairwise Kendall similarities. Consistent with the observation in Figure 2(a), different domains exhibit distinct decoding order patterns and relatively low cross-domain Kendall similarities, indicating that the difference in decoding order is not merely a sample-specific phenomenon but also consistently appears at the domain level. This further supports our motivation that decoding order patterns are domain-dependent in dLLMs, and that preserving decoding trajectories is important for effective continual RL in dLLMs. We further provide more block-level visualisations between CLD and vanilla continual RL in the detailed case study in Appendix F.4.

F.2 Detailed Visualisation of Decoding Trajectories in dLLMs

We further provide detailed token-level visualisations of dLLM decoding trajectories corresponding to the examples in Figure 2(b) and additional examples from another domain during continual RL.

Each cell shows a generated token together with its decoding step within the corresponding block, where darker colours indicate token positions decoded earlier and lighter colours indicate positions decoded later.

As shown in Figure 8, the preserved logic example largely preserves its previously learnt decoding order pattern after continual training and the answer remains correct. In contrast, the forgotten example exhibits substantially stronger decoding order changes and becomes incorrect. Figure 9 further shows the same qualitative behaviour on the Math domain, where the preserved example maintains a similar decoding order pattern with relatively high Kendall similarity and remains correct, while the forgotten example exhibits larger decoding order drift with reduced Kendall similarity and changes from a correct to an incorrect prediction. These fine-grained visualisations suggest that the association between the decoding order preservation and domain knowledge retention is not limited to a specific domain and is consistently observed across different domains during dLLM continual RL.

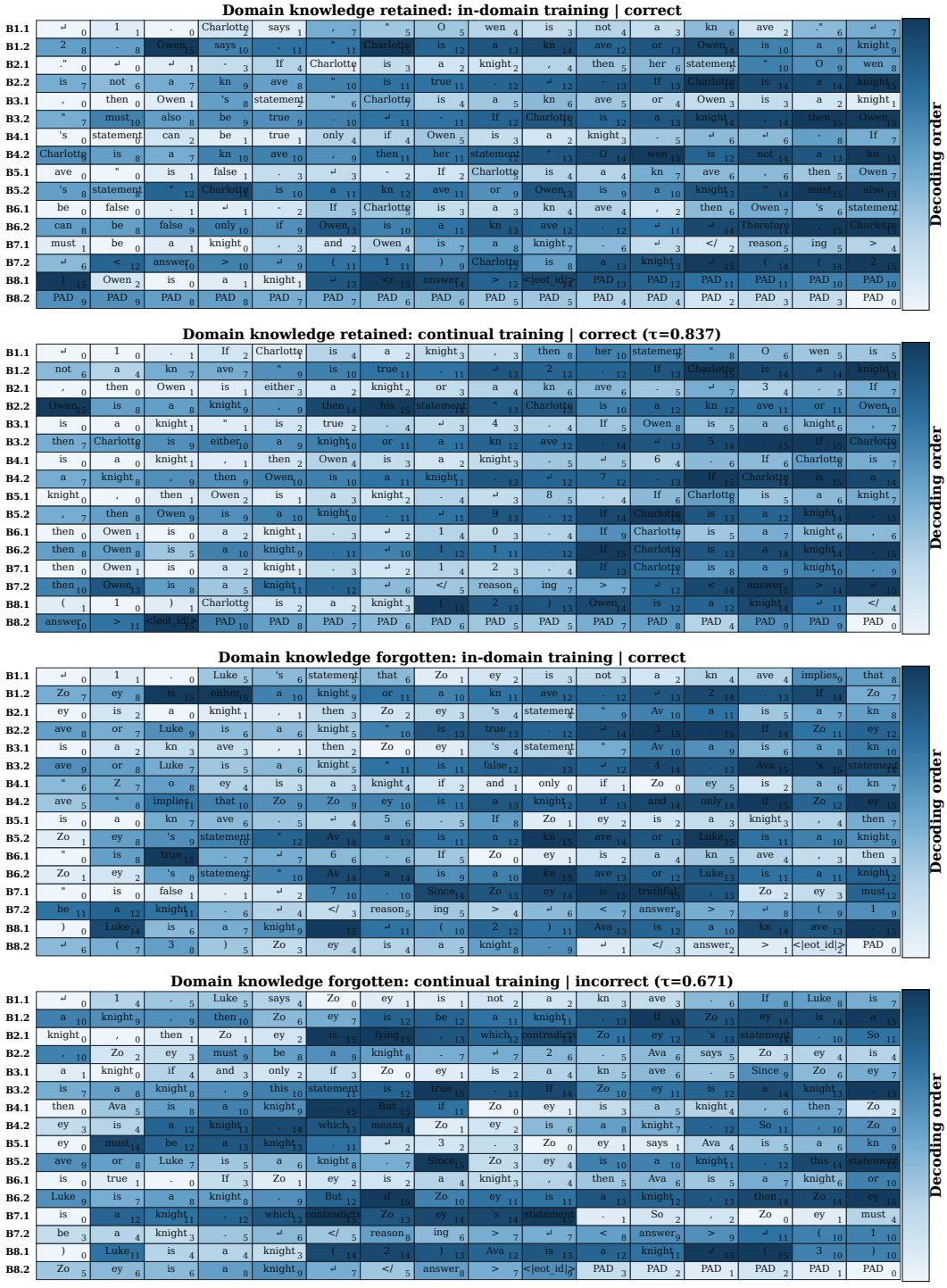


Figure 8: **Fine-grained token-level visualisation of dLLM decoding trajectories on the Logic domain.** The upper two panels show a preserved example before and after continual RL, while the lower two panels show a forgotten example under the same comparison. Each cell displays the generated token and its decoding step within the corresponding semi-autoregressive block. The preserved example maintains a similar decoding order pattern and remains correct after continual training. In contrast, the forgotten example undergoes stronger decoding order drift and changes from a correct to an incorrect prediction.

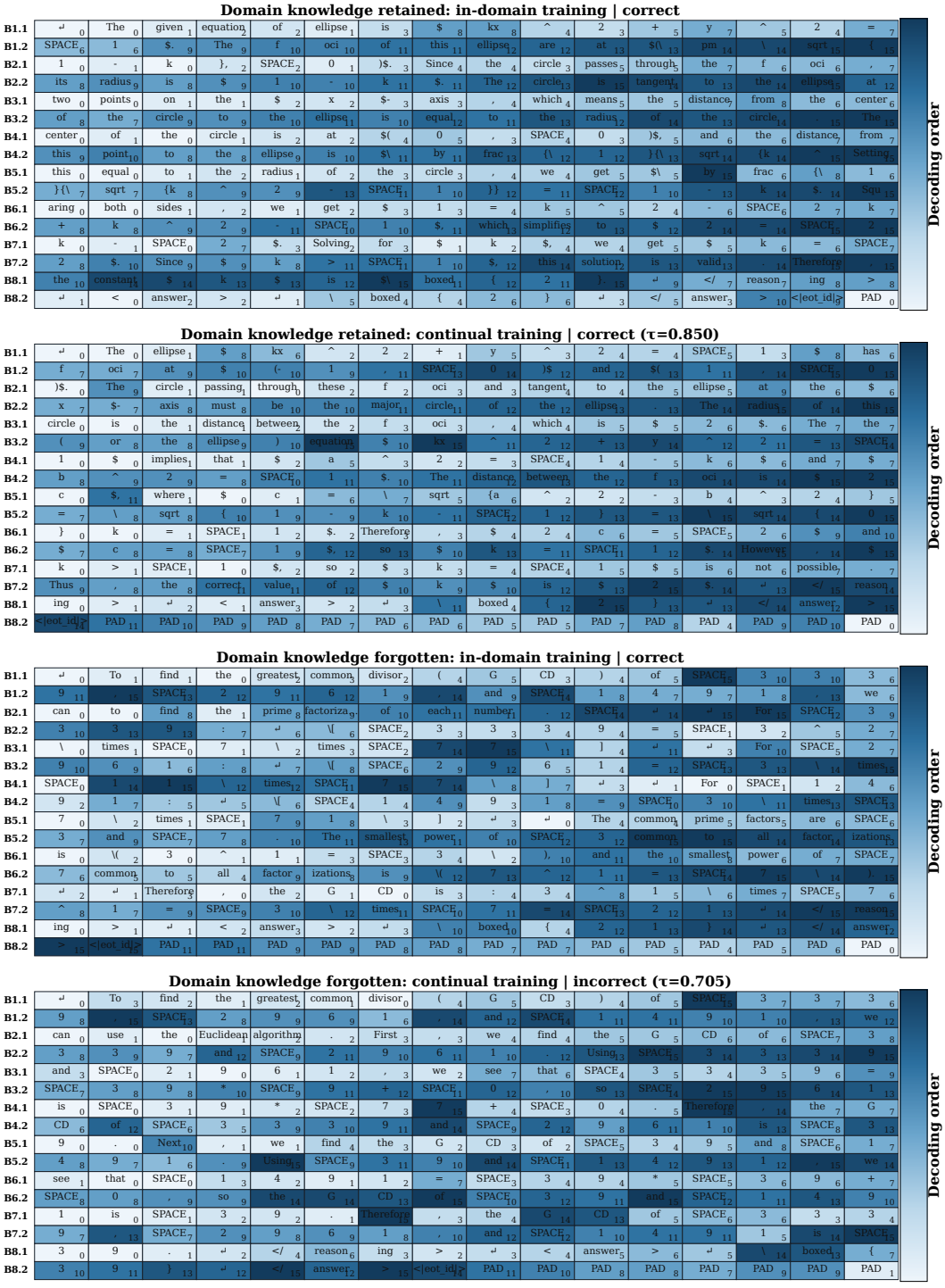


Figure 9: **Fine-grained token-level visualisation of dLLM decoding trajectories on the Math domain.** The upper two panels show a preserved example before and after continual RL, while the lower two panels show a forgotten example under the same comparison. Each cell displays the generated token and its decoding step within the corresponding semi-autoregressive block. Similar to the Logic examples, the preserved example largely preserves its decoding order pattern and remains correct, while the forgotten example exhibits stronger decoding order drift and changes from a correct to an incorrect prediction.

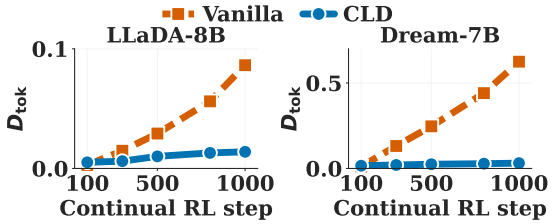


Figure 10: **Token prediction discrepancy during continual RL.** We measure D_{tok} on the previously learnt Math domain during continual RL on Code domain (KodCode). Vanilla Continual RL exhibits increasing token prediction discrepancy as training progresses, whereas CLD maintains substantially smaller discrepancy on both LLaDA-8B and Dream-7B.

F.3 How CLD Preserves Token Prediction Distributions

Following the same continual RL setting in Section 5.5, we further examine whether CLD preserves previously learnt token prediction distributions in dLLMs. Specifically, we measure the token prediction discrepancy D_{tok} on the Math domain throughout continual RL on Code. As shown in Figure 10, Vanilla Continual RL exhibits an increasing token prediction discrepancy as continual RL progresses, indicating substantial drift from the previously learnt token distributions. In contrast, CLD consistently maintains a small D_{tok} across continual RL steps on both dLLM backbones. This result shows that decoding trajectory alignment in CLD effectively preserves previously learnt token prediction distributions during continual RL. Together with the decoding order preservation observed in Section 5.5, these results demonstrate that CLD preserves both token predictions and decoding order patterns, thereby jointly maintaining the dLLM decoding trajectory for effective dLLM continual RL.

F.4 Detailed Visualisation of Decoding Order Shift across Blocks

To complement the block-averaged visualisation in Figure 6, we further provide the detailed decoding order shift for each individual block in Figure 11. We follow the same continual RL setting as in Section 5.5. For each block, every cell represents the relative decoding order between a pair of positions before and after continual RL. Red indicates that the pairwise decoding order is reversed, white indicates that the order is preserved, and grey denotes positions beyond the generated response, such as positions of the `EOS` token, that are not involved in the comparison. The percentage above each block reports the corresponding decoding order reversal ratio.

As shown in Figure 11, Vanilla Continual RL produces substantial pairwise decoding order shifts with a larger decoding order reversal ratio across multiple blocks, with particularly large shifts in several blocks. In contrast, CLD consistently exhibits fewer reversed position pairs across all valid blocks. This block-level further supports the observation in Figure 6 that decoding trajectory alignment in CLD preserves previously learnt decoding order patterns during continual RL.

F.5 Additional Cross-domain Visualisation

To further examine whether the trajectory preservation behaviour extends beyond the Math-to-Code continual RL setting studied in Section 5.5, we provide an additional cross-domain decoding trajectory visualisation in Figure 12. The model is first optimised on a logic constraint reasoning task from the Knights-and-Knaves (KK) dataset and then continually trained on the Math domain. Following Figure 5, *Before* denotes the decoding trajectory after the initial in-domain RL stage, while *After* denotes the trajectory after subsequent continual RL. Each cell indicates the decoding order of the corresponding token position, and the bottom row visualises the resulting decoding order shift.

As shown in Figure 12, the decoding trajectory under CLD remains substantially more similar before and after continual RL, with relatively limited changes across token positions. In contrast, Vanilla Continual RL exhibits substantially larger decoding order shifts after learning the subsequent task. This additional result provides qualitative evidence that the trajectory preservation behaviour of CLD

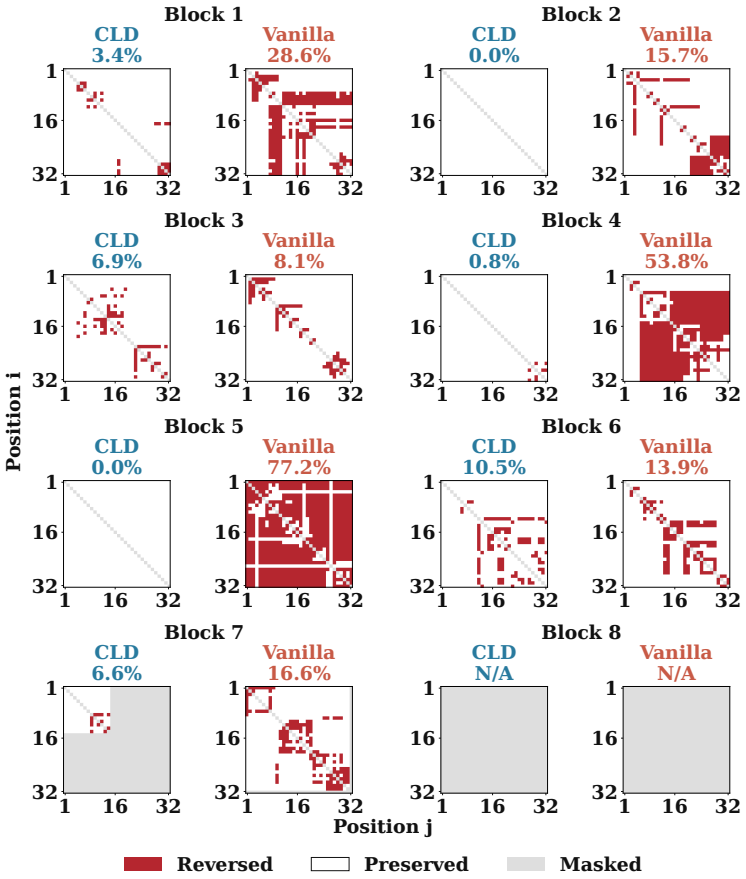


Figure 11: **Detailed decoding order shifts across individual blocks.** Left: CLD. Right: Vanilla. Red indicates reversed pairwise decoding orders, white indicates preserved orders, and grey indicates masked positions. The percentage above each block reports the decoding order reversal ratio.

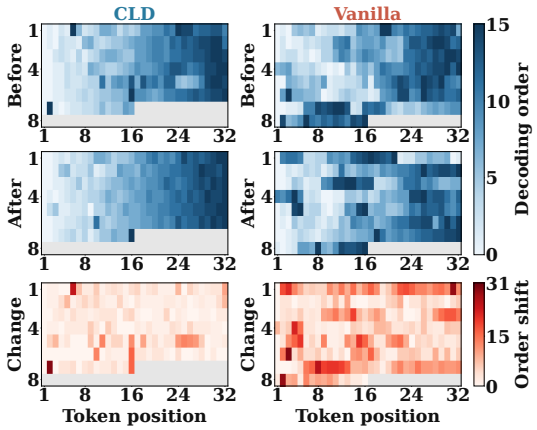


Figure 12: **Additional cross-domain visualisation of decoding trajectory changes before and after continual RL for the logic constraint domain on KK datasets.** Left: CLD. Right: Vanilla. Darker blue indicates a larger decoding-order index, while darker red indicates a larger decoding order shift during continual RL.

is also observed under a different cross-domain continual RL setting for other datasets, supporting its generalisation beyond the Math-to-Code continual RL setting analysed in the main text.

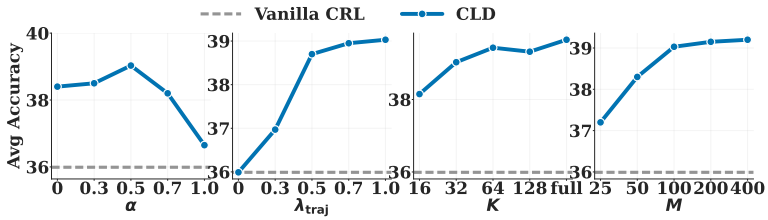


Figure 13: **Hyperparameters in CLD.** We vary the token-order balancing weight α, trajectory alignment weight λ_{traj}, preserved top-K size K, and memory budget M. For each sensitivity experiment, only one hyperparameter is varied while all others are fixed to their default values. The dashed line denotes Vanilla Continual RL without trajectory alignment. Performance is reported as the average accuracy over the continual RL domains in Table 1. CLD consistently outperforms Vanilla Continual RL across a broad range of hyperparameter settings.

F.6 Hyperparameter Sensitivity Analysis

CLD remains robust across a broad range of hyperparameter choices. As shown in Figure 13, we independently vary α, λ_{traj}, preserved top-K size K, and memory budget M, while strictly fixing all remaining hyperparameters to their default values in each experiment. Here, K controls the number of token prediction probabilities preserved at each token position, while M denotes the default number of preserved decoding trajectory records per domain. Performance is averaged over the continual RL domains used in the main setting in Table 1. For α, jointly preserving token distributions and decoding order patterns performs better than relying on either component alone. This further demonstrates the complementary benefits of aligning both token and order patterns for effective dLLM continual RL. Increasing K generally improves performance by preserving richer historical token information, while using the full vocabulary provides limited additional gains. Increasing M also improves the average performance by providing more decoding trajectory information for alignment and mitigating forgetting in previous domains, with the gains gradually becoming marginal under larger memory budgets. Overall, CLD is relatively insensitive to the choices of K and M and consistently outperforms Vanilla Continual RL without trajectory alignment. The results of α and λ_{traj} further support the effectiveness of jointly aligning token distributions and decoding order patterns in dLLM continual RL.

Table 8: **Detailed continual RL results under random seeds.** ACC, BWT and AF scores are averaged over all runs. Results over three random seeds show that CLD consistently improves dLLM capability retention during continual RL, achieving positive BWT and lower AF, while Vanilla Continual RL without decoding trajectory alignment exhibits negative BWT.

Method	Logic	Math	Code	ACC↑	BWT↑	AF↓
Vanilla Continual RL	47.02 ± 1.85	33.13 ± 1.70	26.33 ± 0.50	35.49	-4.42	4.85
CLD	$\mathbf{54.53 \pm 0.91}$	$\mathbf{35.80 \pm 0.28}$	$\mathbf{28.67 \pm 0.24}$	**39.67**	$+\mathbf{0.26}$	**0.04**

F.7 Robustness under Multiple Seeds

To further assess the robustness of CLD, we repeat the main continual RL experiments with three random seeds using the same setting as in Table 1. Vanilla Continual RL and CLD use the same training data, domain order, optimisation settings, and evaluation protocol across all runs. The only methodological difference between CLD and Vanilla Continual RL is that CLD introduces decoding trajectory alignment. As shown in Table 8, CLD maintains stronger final performance than Vanilla Continual RL and also achieves higher ACC and BWT together with lower AF, indicating more consistent retention of previously learnt capabilities during continual RL. This evaluation therefore provides further evidence that the effectiveness of CLD extends beyond the default single-seed setting in most existing dLLM methods (Tang et al., 2026; Zhao et al., 2025; Zhong et al., 2026; Jiang et al., 2026b).

Table 9: **Generalisation across different dLLMs and generation lengths.** The larger dLLMs LLaDA2.0 and LLaDA2.1 (16B) are evaluated with a generation length of 1024, as shorter lengths often truncate their reasoning before the final answer is produced. Final continual RL performance and ACC are highlighted by **best** and runner-up, while the best BWT, AF, and S_K are shown in bold. S_K is in percentage.

Backbone	Gen. Length	Stage 1: Logic		Stage 2: Math		Stage 3: Code	Continual RL Metrics			
		RL	Continual RL	RL	Continual RL	Continual RL	ACC↑	BWT↑	AF↓	S_K ↑
LLaDA-8B-Instruct	256	51.91	52.69	36.20	36.00	28.40	39.03	+0.29	0.10	86.33
LLaDA-8B-Instruct	512	53.26	54.27	38.80	36.60	34.80	41.89	−0.60	1.10	85.91
Dream-v0-Instruct-7B	256	54.29	57.67	46.60	44.40	29.80	43.96	+0.59	1.24	86.85
Dream-v0-Instruct-7B	512	55.80	53.69	38.20	41.20	**37.60**	44.16	+0.45	1.26	84.28
LLaDA-1.5 (8B)	256	50.16	53.36	36.40	37.00	24.40	38.25	+1.90	0.10	89.62
LLaDA-1.5 (8B)	512	54.22	55.51	38.00	38.60	30.00	41.37	+0.95	**0.00**	90.40
LLaDA2.0-mini (16B)	1024	36.41	40.05	66.40	**75.80**	30.20	48.68	**+6.52**	0.31	**96.35**
LLaDA2.1-mini (16B)	1024	52.57	**58.76**	74.60	73.60	31.80	**54.72**	+2.60	0.67	90.59

F.8 On Generalisation to Other dLLMs

We further evaluate CLD across different dLLM backbones and generation lengths in Table 9. Overall, CLD generalises across different model families and scales while maintaining strong final performance and low forgetting. We also vary the generation length for multiple backbones, showing that CLD remains effective under different decoding budgets. For larger dLLMs, we use a longer generation length to avoid truncating the reasoning process before the final answer and observe strong positive backward transfer in CLD under larger model scales. These results support the general applicability of CLD across different dLLM architectures, model scales, and generation settings.

Table 10: **Generalisation of CLD to different dLLM RL algorithms.** All experiments follow the Logic, Math, Code domain order. *RL* and *Continual RL* denote performance after learning each stage and after the final stage, respectively. Final performance and ACC are highlighted by **best** and runner-up, while the best BWT and AF are shown in bold.

Base RL Algorithm	Stage 1		Stage 2		Stage 3	Continual RL Metrics		
	RL	Continual RL	RL	Continual RL	Continual RL	ACC↑	BWT↑	AF↓
Diffu-GRPO	36.86	35.71	33.60	34.20	24.20	31.37	−0.28	0.58
d1	37.00	36.43	34.60	35.00	26.00	32.48	−0.09	0.29
wd1	38.00	37.64	35.00	35.00	25.40	32.68	−0.18	0.18
StableDRL	51.91	**52.69**	36.20	**36.00**	**28.40**	**39.03**	+0.29	**0.10**

F.9 On Generalisation to Different dLLM RL Algorithms

We further integrate CLD with different dLLM RL algorithms in Table 10. CLD is compatible with different underlying RL objectives and generally maintains low forgetting across all evaluated algorithms. These results further support the plug-and-play applicability of CLD for extending existing single-domain dLLM RL algorithms to continual RL settings.

Table 11: **Robustness to different decoding block sizes.** All experiments use LLaDA-8B-Instruct with StableDRL by default. *RL* and *Continual RL* denote performance after learning each stage and after the final stage following the continual learning setting in Table 1. The main experiments use a default block size of 32 following most existing dLLM methods. S_K is reported in percentage.

Block Size	Stage 1: Logic		Stage 2: Math		Stage 3: Code	Continual RL Metrics			
	RL	Continual RL	RL	Continual RL	Continual RL	ACC↑	BWT↑	AF↓	S_K ↑
32	51.91	52.69	36.20	36.00	28.40	39.03	+0.29	0.10	86.33
64	46.80	48.25	36.40	36.60	32.80	39.22	+0.82	0.00	86.10
128	48.05	48.96	33.00	33.20	35.60	39.25	+0.55	0.73	87.00

F.10 On Robustness to Different Decoding Block Sizes in dLLMs

We further evaluate CLD with different decoding block sizes in Table 11. While different block sizes lead to variations in individual domain performance due to changes in the dLLM parallel decoding configuration, CLD remains stable across all evaluated settings. In particular, block sizes of 32, 64, and 128 achieve highly consistent ACC, positive BWT, low AF, and similar decoding order similarity S_K. These results demonstrate that the effectiveness of CLD is robust to different decoding block sizes and is not dependent on the block size of 32, which is commonly adopted by default in existing dLLMs.